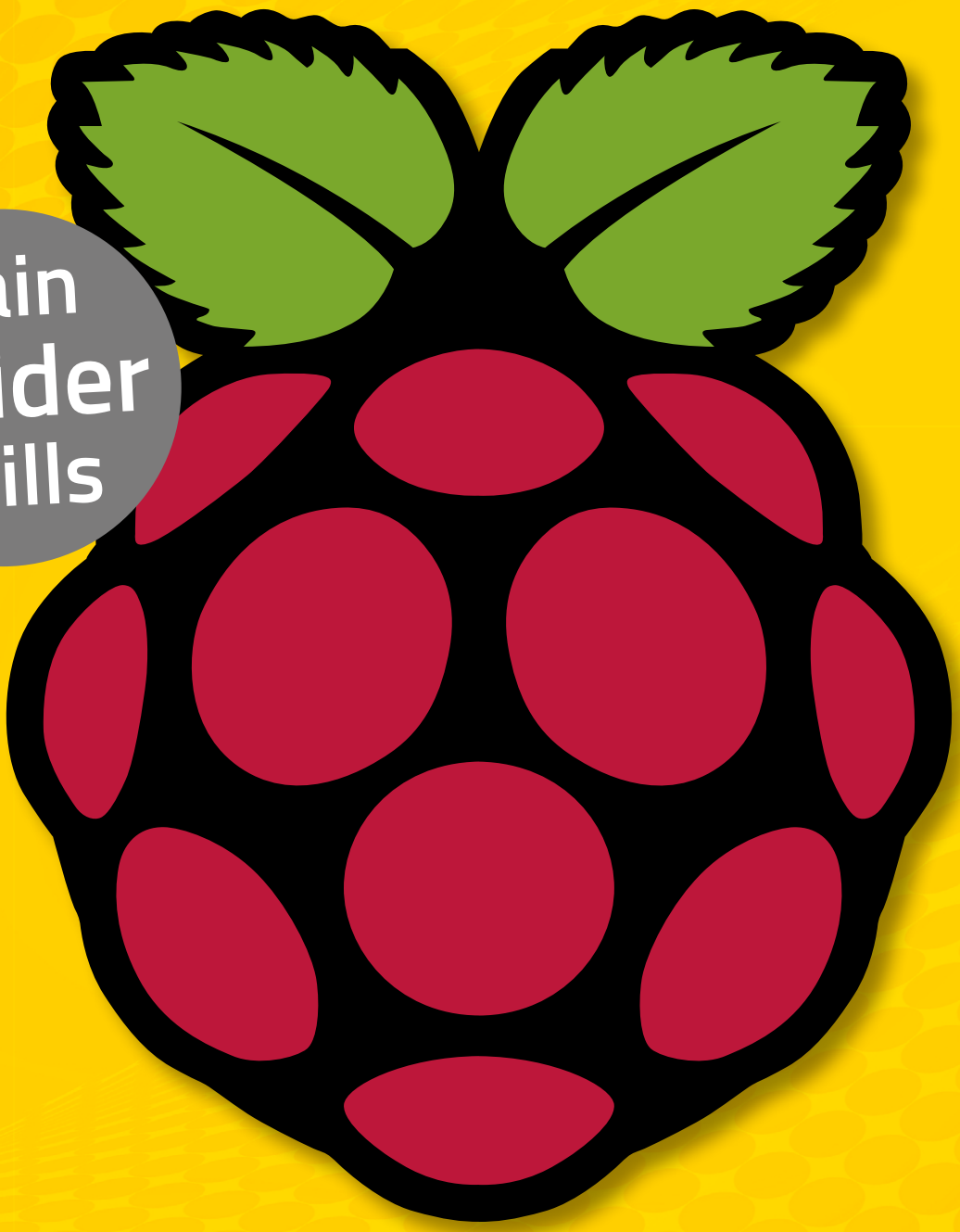
techgo

Raspberry Pi Tricks and Tips

Gain Insider Skills



Next level Secrets & Fixes

Advanced Guides & Tips

Rediscover Your Device

Raspberry Pi
Tricks and Tips

Raspberry Pi Tricks and Tips
ISBN: 978-1-912847-70-9

Foreword

Welcome back...

Having completed our exclusive For Beginners digital guidebook, we have taught you all you need to master the basics of your new device, software or hobby. Yet that's just the start!

Advancing your skill set is the goal of all users of consumer technology and our team of long term industry experts will help you achieve exactly that. Over this extensive series of titles we will be looking in greater depth at how you make the absolute most from the latest consumer electronics, software, hobbies and trends!

We will guide you step-by-step through using all the advanced aspects of the technology that you may have been previously apprehensive at attempting. Let our expert guide help you build your understanding of technology and gain the skills to take you from a confident user to an experienced expert.

Over the page our journey continues and we will be with you at every stage to advise, inform and ultimately inspire you to go further.

About the Publisher

From its humble beginnings in 2004, the BDM brand has quickly grown from a single publication produced by a team of just two to one of the biggest names in global tech print and digital publishing, for one simple reason. Our passion and commitment to deliver the very best product to the marketplace.

While the company has grown with a portfolio of over 500 publications crafted by our international staff of respected industry veterans, the foundation that it has been built upon remains the same. That being to create the best quality, fully independent, user friendly and, most essentially, 100% up-to-date content possible.

Delivering not only market leading publications but also piece of mind to our readers, so that they have the very best foundation to build their knowledge, confidence and understanding of their new software and hardware. Our regular readers trust BDM, as should you.

How to use this book

This book has been designed for you to progress through the coreconcepts and fundamentals of use, through to more advanced elements, projects, and ideas. There's something for every style of reader, and for every type of user; there's probably even a few terrible jokes dotted within the pages. So here's how to get the best from it.

Step 1

Don't skip - While it's fun to see what's coming up later in the book, it does make understanding what you're reading more difficult. After all, you wouldn't start reading a book on speaking French, then skip further in without first learning proper grammar, sentence structure and so on. The same applies here. Take your first read-through page by-page, once you've mastered the book, then you can return to key concepts whenever you need.

Step 2

Ever-Changing - While every effort has been made to ensure that this book is up to date, there's no knowing what updates may occur over time. While some companies do offer an accurate roadmap of their future development of a product, it's not always written in stone. For example, an app available for Windows 10 now may not be available with the next update of the operating system. It's up to Microsoft to decide whether they want to drop it for one reason or another. The same, to some extent, applies here. However, we continually update the content in this title, so it's as accurate as possible.

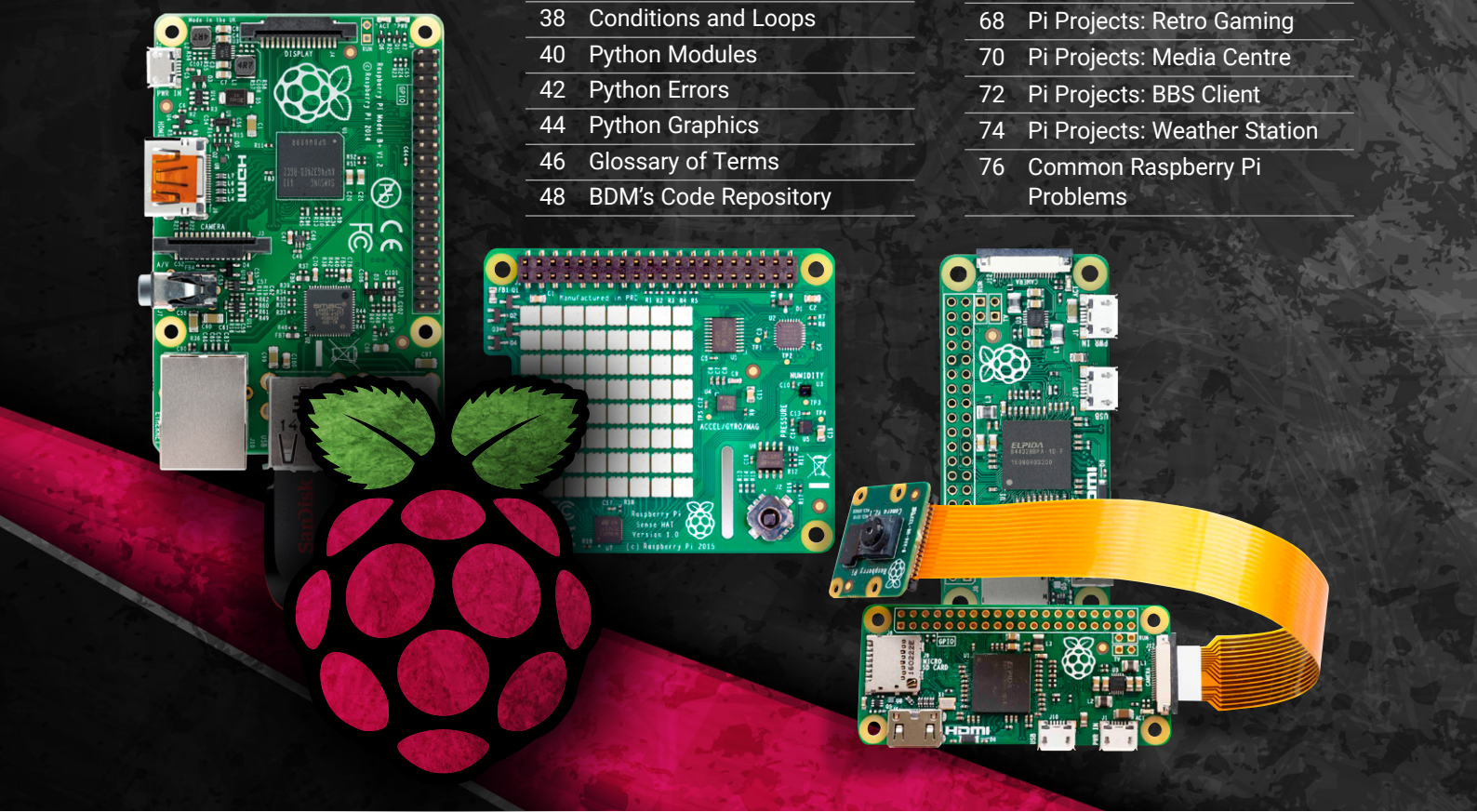
Step 3

Follow the Steps - An obvious one, this. Following the steps from one onwards, in most tutorials in this book, will ensure that you get the result that's intended. If you skip steps, then you may miss out on something important, and not understand how it works later in the book. The temptation to skip something you already know is often too much, but stick with the logical progression of the steps and you'll get the most from what's on offer.

Step 4

Have Fun - Learning a new skill is supposed to be fun. We had fun writing the book, and hopefully you'll have fun reading it and applying new skills. Everyone learns at a different pace, so take your time, digest the tutorials, and keep returning to key concepts if you feel the need to master any element within these pages. The content in this book isn't something we're going to be testing you on, so have fun and enjoy the art of learning something new. And if you create something amazing after reading this book, then let us know.

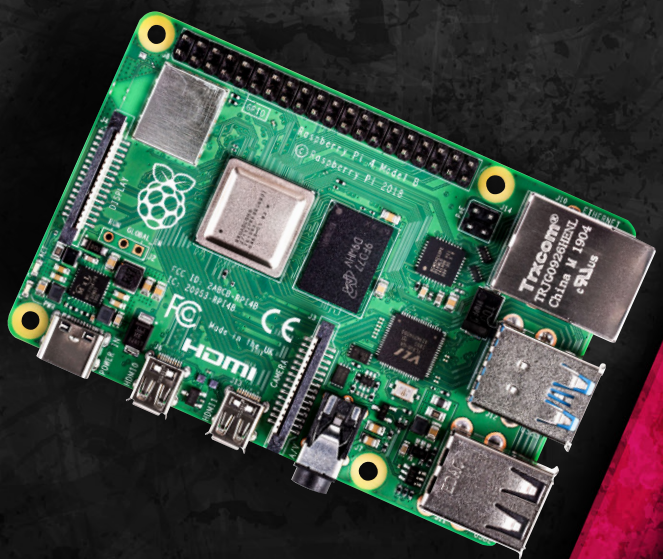
Contents



"...we've got projects, code, guides and tutorials that will help you unlock the power of the Pi..."

Learning Linux

In this section you will discover how the OS works, how the filesystem is built, and how you can list, move, create and delete files and folders. To truly be able to hack and master the Raspberry Pi, you will need to be familiar with the OS and its inner workings.



Fun Things to Do in the Terminal

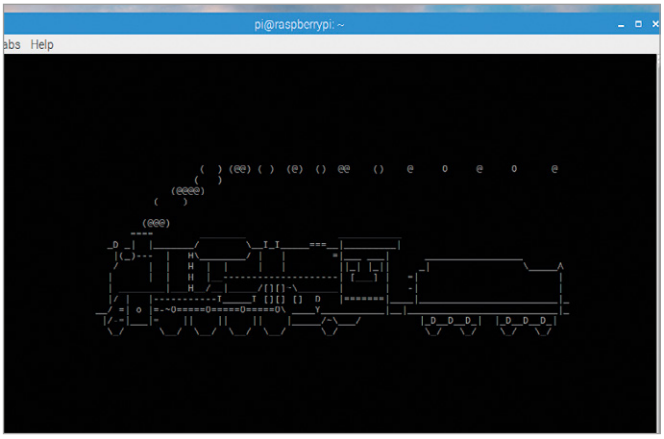
Despite the seriousness of an operating system, the Linux community are certainly no strangers to a bit of fun. Over the years, the developers have created and inserted all manner of quirky and entertaining elements into the Terminal.

TERMINAL FUN

All these commands are Linux-based, so not only can you use them on the Raspberry Pi but also on any of the Debian-based Linux distributions.

STEP 1

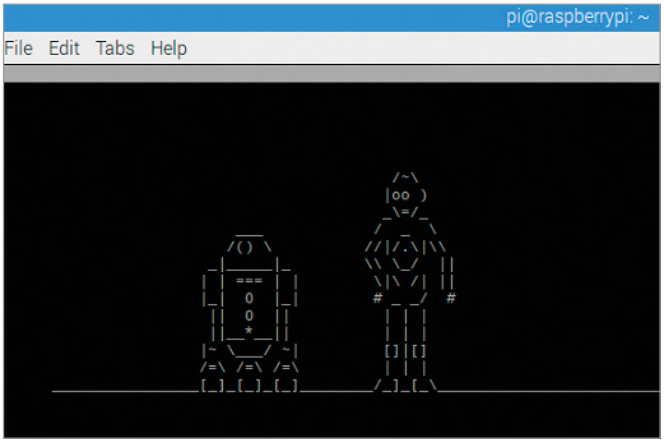
The first command we're going to use is `sl`, it's not installed by default so enter: `sudo apt-get install sl`. The command can be run with `sl` and when executed displays a Steam Locomotive travelling across the screen (hence 'sl'). Entering: `LS` (note the upper case) also works.



STEP 2

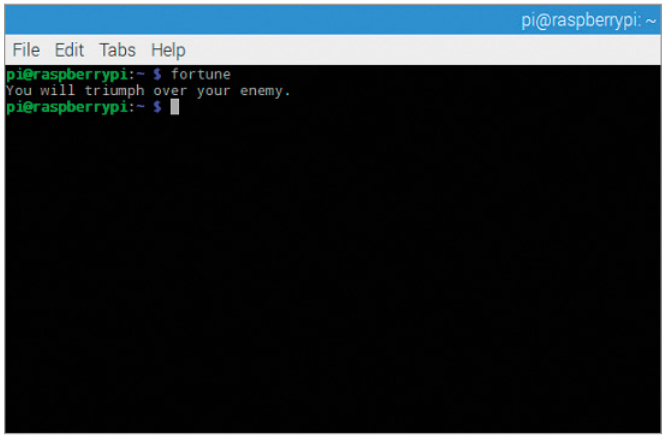
Fans of Star Wars even get a fix when it comes to the Terminal. By linking to a remote server via the telnet command, you can watch *Episode IV: A New Hope* being played out, albeit in ASCII. To view this spectacle, enter: `sudo apt-get install telnet`, followed by:

`telnet towel.blinkenlights.nl`



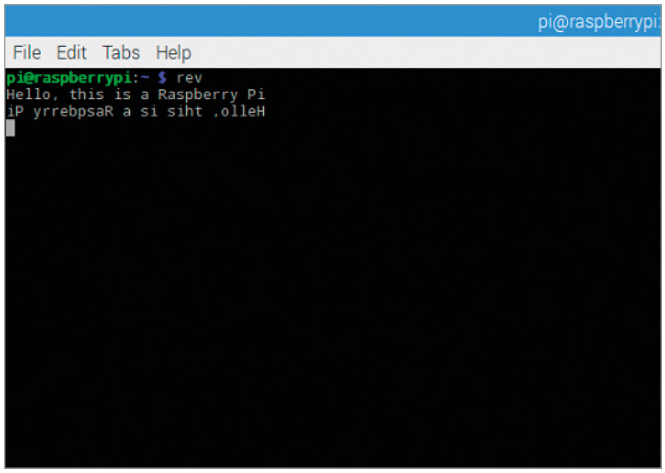
STEP 3

If you've ever fancied having the computer read a random fortune out to you then you're in luck. Raspbian requires you to install the Terminal app, Fortune, first. Enter: `sudo apt-get install fortune`, then simply enter: `fortune`, into the Terminal to see what comes up.



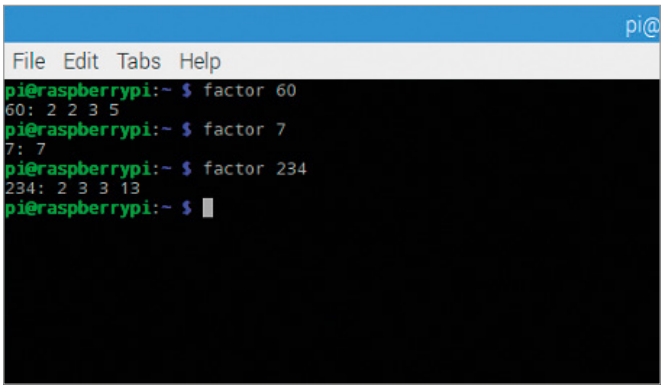
STEP 4

The rev command is certainly interesting, and at first what seems a quite useless addition to the OS can, however, be used to create some seemingly unbreakable passwords. Enter: `rev` and then type some text. Then press Enter and everything you typed in is reversed. Press Ctrl+C to exit.



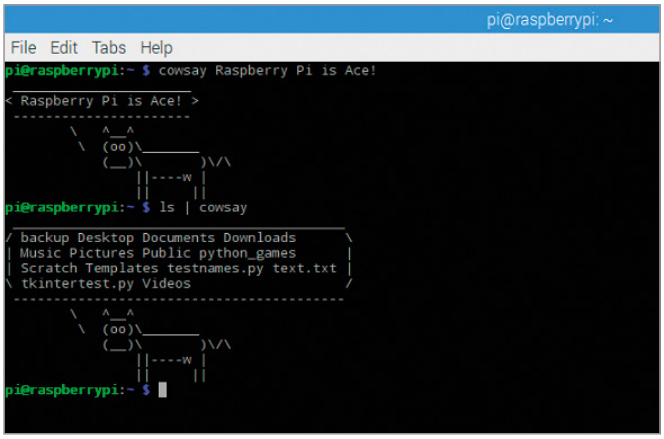
STEP 5

If you're stuck trying to work out all the possible factors for any particular number, simply enter: `factor` followed by the number. For example, factor 7 doesn't offer much output, whereas factor 60 displays more.



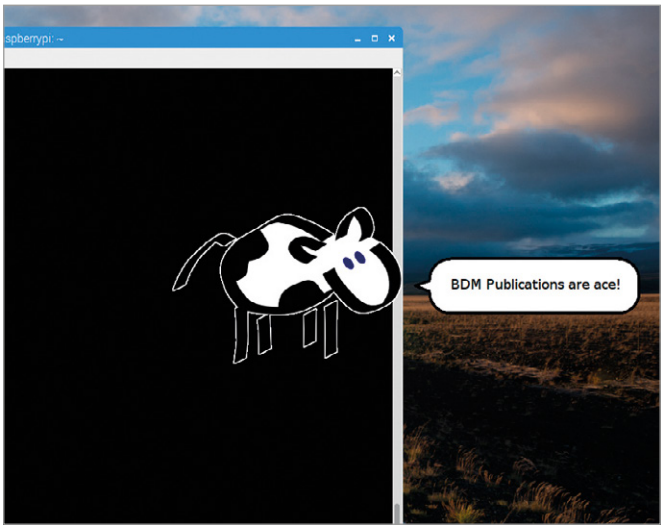
STEP 6

There's a fine line between the rather cool and the really quite weird. Having an ASCII cow repeat text to you could potentially fall into the latter. Start by installing cowsay: `sudo apt-get install cowsay`, then enter: `cowsay Raspberry Pi is Ace!`. In fact, you can even output the ls command through the cow, by entering: `ls | cowsay`.



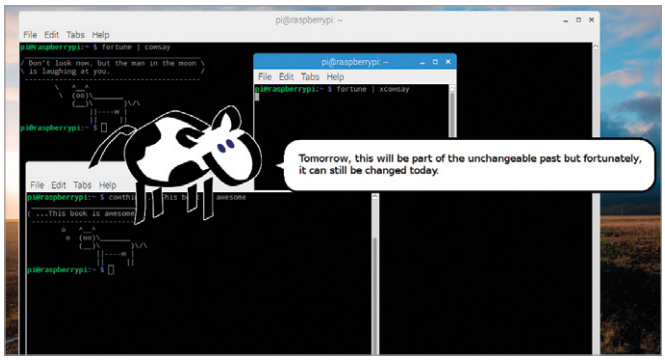
STEP 7

To further the cow element, there's even a graphical, i.e. non-Terminal, cow available. Install it with: `sudo apt-get install xcowsay`, then when it's installed enter something similar to cowsay, such as: `xcowsay BDM Publications are ace!`.



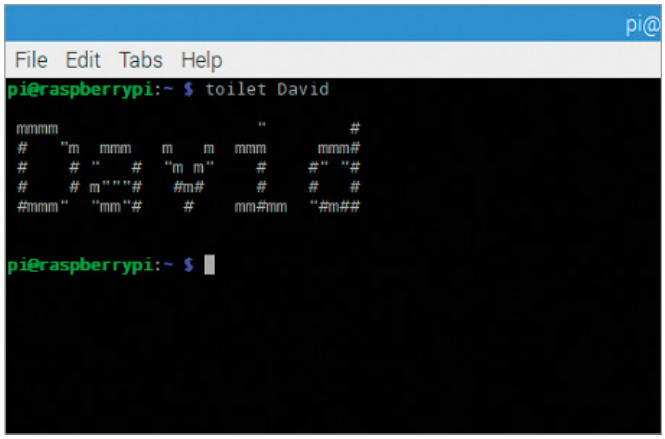
STEP 8

If you really want to expand the whole cow thing, for whatever reason, then pipe the fortune command through it, with: `fortune | cowsay`; and for the graphical cow equivalent: `fortune | xcowsay`. Plus, there's always cowthink. Try: `cowthink ...This book is awesome`.



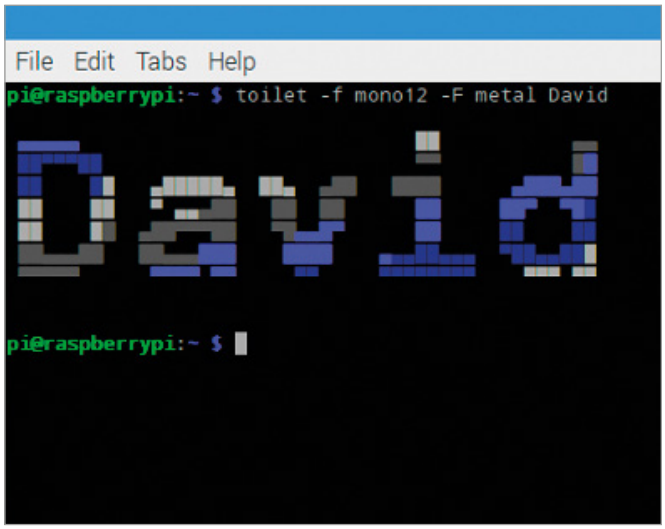
STEP 9

Admittedly, the command 'toilet' doesn't inspire much confidence. However, it's not as bad as it first sounds. Start by installing it with: `sudo apt-get install toilet`, then when it's installed, type something along the lines of: `toilet David`; or perhaps list the contents of the current folder through it, with: `ls | toilet`.



STEP 10

Expanding the toilet command, you can actually generate some decent looking graphics through it. For example, try this: `toilet -f mono12 -F metal David`. You can enter: `toilet --help`, for a list of the command line arguments to expand further.



More Fun Things to Do in the Terminal

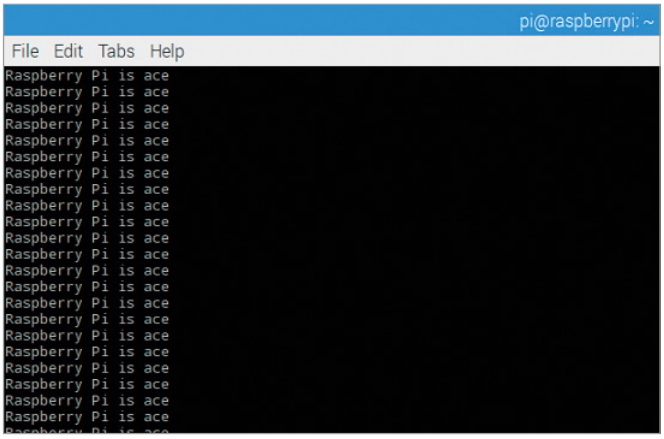
If the previous list of bizarre, and fun things to do in the Terminal has you wanting more, you're in luck. We've put together another batch of both useful, and not so useful, commands for you to try out.

MORE FUN, YAY

Since the Terminal session is already open, and your keyboard digits are nicely warmed up, here are another two pages of Terminal nonsense.

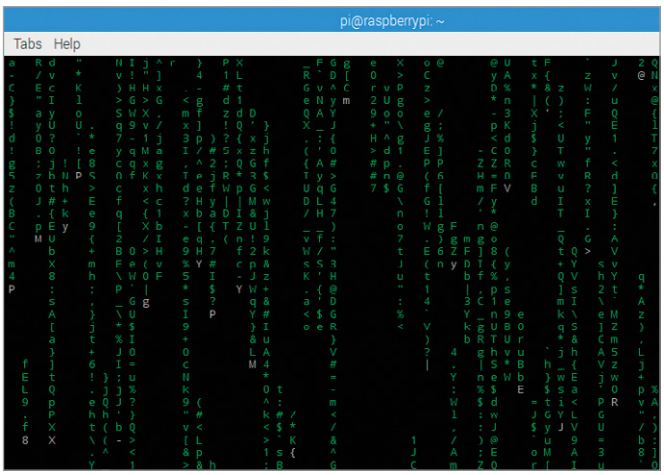
STEP 1

Remember the old ZX Spectrum days of computing, when you could type in 10 print "Hello", 20 goto 10 and Hello would list down the screen? Well, in Raspbian you can do the same. Simply enter: `yes` followed by some text, i.e. `yes Raspberry Pi` is ace. It keeps going until you press Ctrl+C.



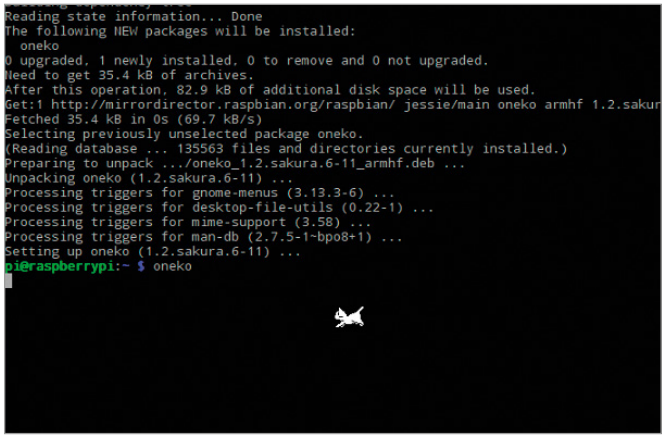
STEP 2

The Matrix was one of the most visually copied films ever released and there's even a version of the Matrix code available for Linux. Install it with: `sudo apt-get install cmatrix`. When it's done enter: `cmatrix` and follow the white rabbit, Neo. Unlike the real Matrix though, you can press Ctrl+C to exit.



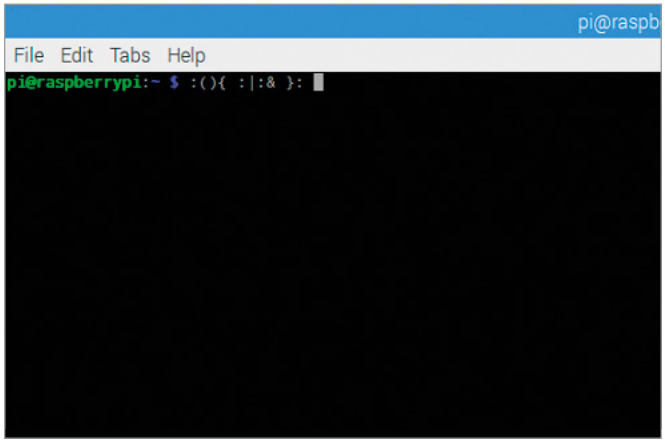
STEP 3

Having a little white cat chase your mouse pointer around the desktop may sound like a terrible waste of time. Oddly though, it isn't. Enter: `sudo apt-get install oneko`, then type: `oneko` to have the cat appear. Move your 'mouse' cursor around the screen and the cat chases it. Use Ctrl+C to exit the action.



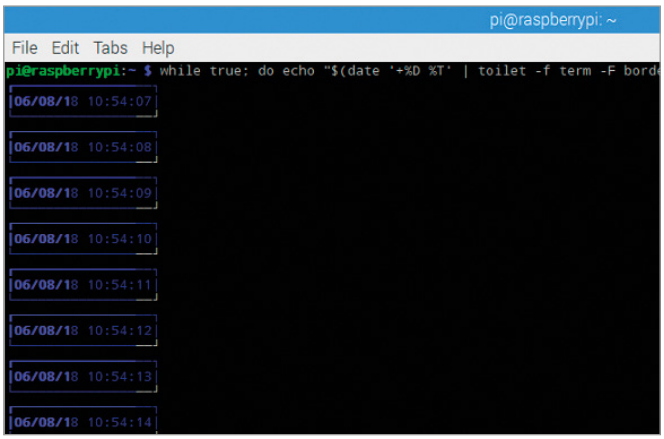
STEP 4

This entry is a little more serious than the previous. Fork Bomb is a command that continually replicates itself until it has used up all the available system resources, eventually causing your computer to crash. You don't have to try it, but it's interesting nonetheless. Simply enter: `:(){ :|:& }:` and be prepared to reboot.



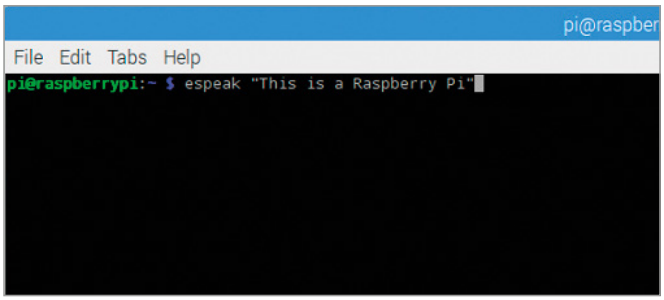
STEP 5

Stringing several commands and piping them through other commands is what makes scripting such a powerful element to an OS. For example, using the while command, together with toilet, can yield some impressive results. Enter: `while true; do echo "$(date '+%D %T' | toilet -f term -F border --metal)"; sleep 1; done`.



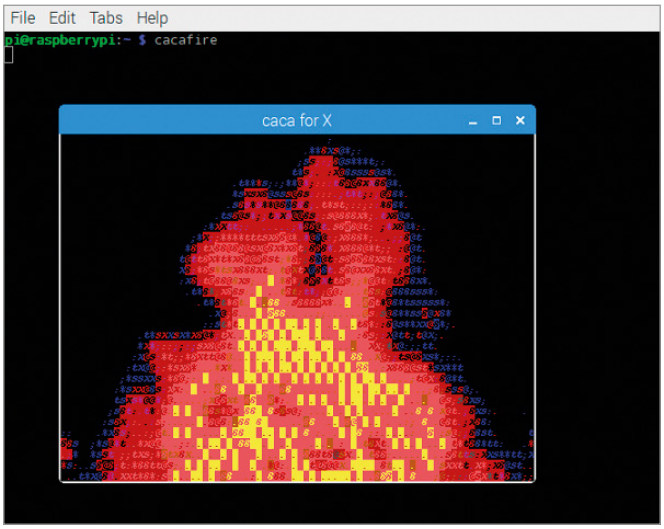
STEP 6

Talking computers were the craze of the '80s. To re-live the fun enter: `sudo apt-get install espeak`, then: `espeak "This is a Raspberry Pi"` to have the computer repeat the text inside the quotes to you. Make sure your volume is turned up and try the following: `ls > folders.txt && espeak -f folders.txt`. This gets Raspbian to read back the contents of the ls command.



STEP 7

A roaring ASCII fire isn't the most useful command to have at your disposal but it's fun. Install it with: `sudo apt-get install libaa-bin`, then use: `aafire`. It's not exactly warming but you get the idea. To expand the above, enter: `sudo apt-get install bb caca-utils`, then: `cacafire`.

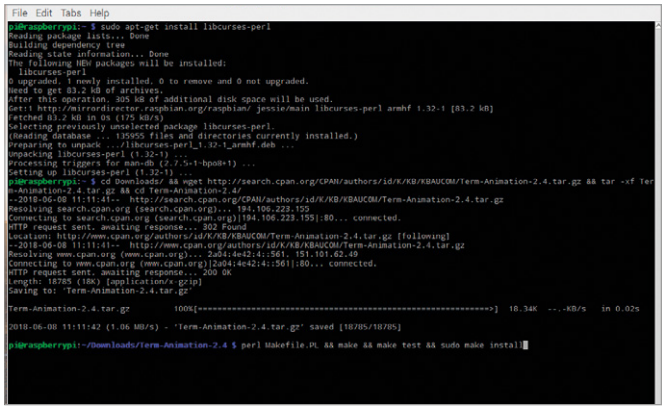


STEP 8

Used as a music demo from the old Amiga and DOS days, the bb command evokes memories of three and a half inch floppies crammed with all manner of demo scene goodies. You've already installed bb from the previous step, so just enter: `bb`. Follow the onscreen instructions and turn up your volume.

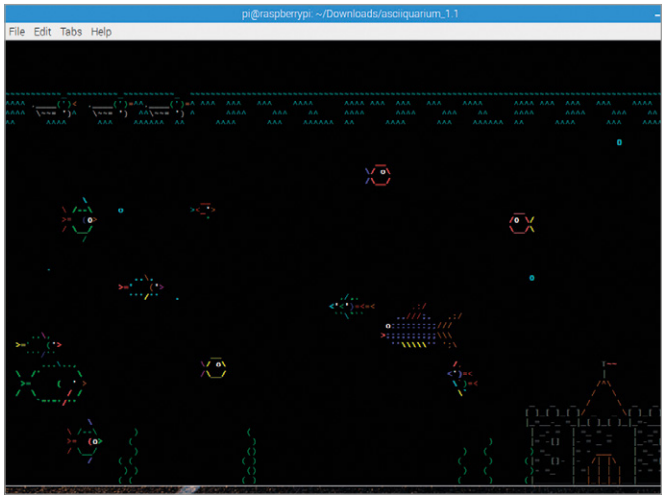
STEP 9

This entry is in two parts. First you need to get hold of the necessary packages: sudo apt-get install libcurses-perl. When that's done enter: `cd Downloads/ && wget http://search.cpan.org/CPAN/authors/id/K/KB/KBAUCOM/Term-Animation-2.4.tar.gz && tar -xf Term-Animation-2.4.tar.gz && cd Term-Animation-2.4/`. Followed by: `perl Makefile.PL && make && make test && sudo make install`.



STEP 10

With that little lot completed, onto the next part. Enter: `cd .. && wget http://www.robobunny.com/projects/asciiquarium/asciiquarium.tar.gz && tar -xf asciiquarium.tar.gz && cd asciiquarium_1.1/ && chmod +x asciiquarium`. Providing all went well, enter: `./asciiquarium` and enjoy your very own ASCII-based aquarium.



Linux Tips and Tricks

The Linux Terminal, you'll no doubt agree, is an exceptional environment and with a few extra apps installed along with a smidgen of command knowledge, incredible and often quite strange things can be accomplished.

TAKING COMMAND

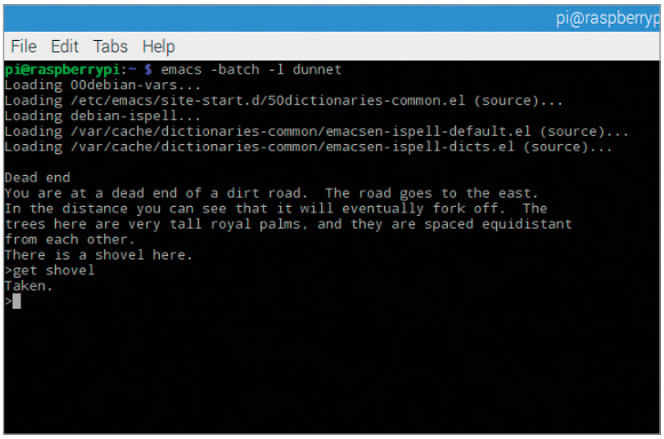
There are countless Linux tips, secrets, hacks and tricks out there. Some are very old, originating from Linux's Unix heritage, while others are recent additions to Linux lore. Here are our ten favourite tips and tricks.

EASTER EGGS

Emacs text editor, is a great piece of software but did you know it also contains a hidden Easter Egg? With Emacs installed (sudo apt-get install emacs24), drop to a Terminal session and enter:

```
emacs -batch -l dunnet
```

Dunnet is a text adventure written by Ron Schnell in 1982, and hidden in Emacs since 1994.



MOON BUGGY

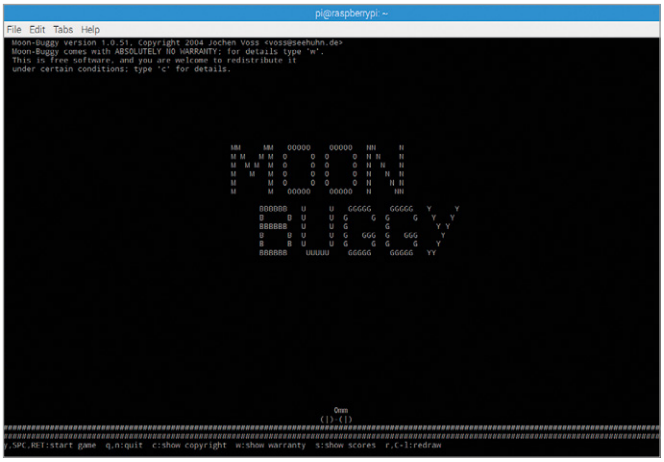
Based on the classic 1982 arcade game, Moon Patrol, Moon Buggy appeared on home computers in 1985 amid much praise. It's a cracking Atari game available in the Linux Terminal by entering:

```
sudo apt-get install moon-buggy
```

Then:

```
moon-buggy
```

Enjoy.



TERMINAL BROWSING

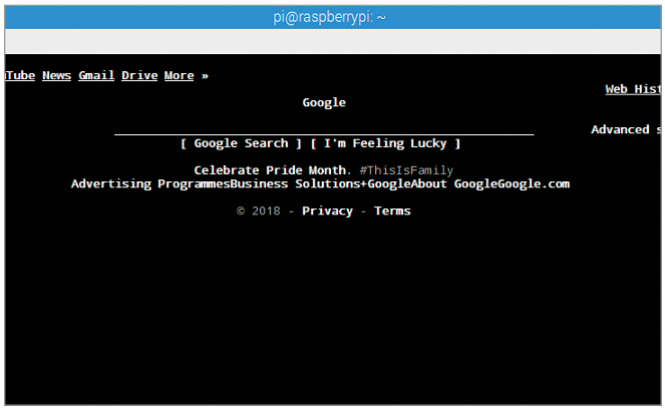
Ever fancied being able to browse the Internet from the Terminal? While not particularly useful, it is a fascinating thing to behold. To do so, enter:

```
sudo apt-get install elinks
```

Then:

```
elinks
```

Enter the website you want to visit.



LET IT SNOW

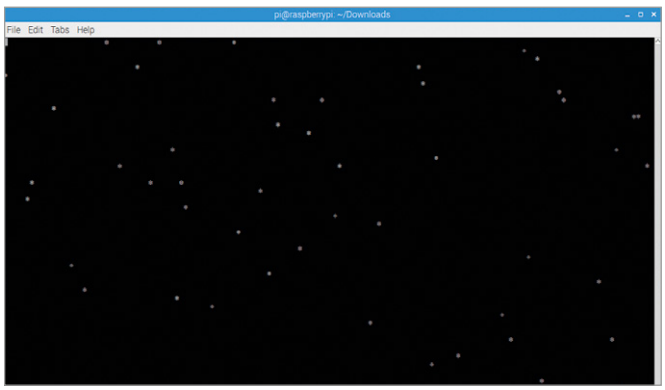
Snowing in the Terminal console isn't something you come across every day. If you're interested, however, enter:

```
wget
```

```
https://gist.githubusercontent.com/sontek/1505483/raw/7d024716ea57e69fb52632fee09f42753361c4a2/snowjob.sh
```

```
chmod +x snowjob.sh
```

```
./snowjob.sh
```

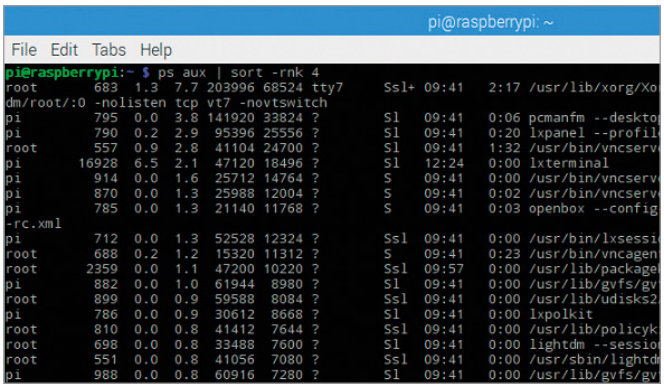


MEMORY HOGS

If you need to see which apps are consuming the most memory on your Raspberry Pi, simply enter:

```
ps aux | sort -rnk 4
```

This sorts the output by system memory use.

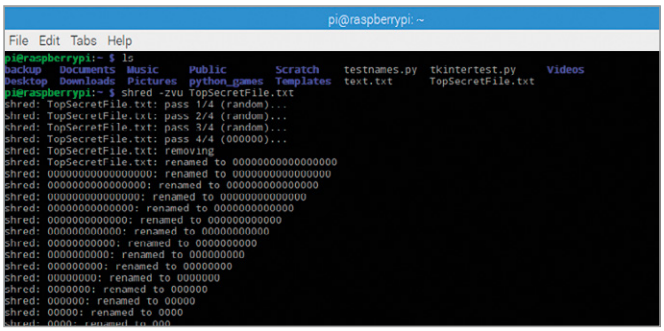


SHREDDER

When you delete a file, there's still a chance of someone with the right software being able to retrieve it. However, files can be securely and permanently deleted using Shred:

```
shred -zvu NAMEOFFILE.txt
```

Replace NAMEOFFILE with the name of the file to delete.



ASCII ART

ASCII art can be quite striking when applied to some images. However, it's often difficult to get just right. You can create some great ASCII art from the images you already have on the Raspberry Pi by using img2txt:

```
img2txt NAMEOFIMAGEFILE.png
```

Replace NAMEOFIMAGEFILE with the actual name of the image file on your system.

BBS

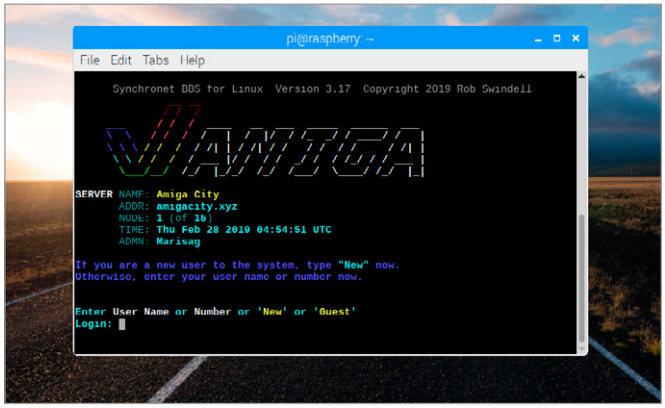
Back in the days of dial-up connections, the online world was made up of Bulletin Board Systems. These remote servers provided hangouts for users to chat, swap code, play games and more. Using telnet in Linux, we can still connect to some active BBSes. First, install Telnet with:

```
sudo apt-get install telnet
```

Then:

```
telnet amigacity.xyz
```

You will be connected to a BBS dedicated to the Commodore Amiga. There's plenty more, which you can find at www.telnetbbsguide.com.

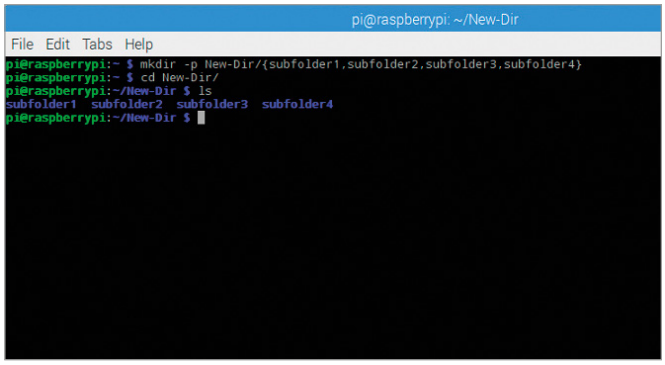


DIRECTORY TREES

If you want to create an entire directory (or folder) tree with a single command, you can use:

```
mkdir -p New-Dir/{subfolder1,subfolder2,subfolder3,subfolder4}
```

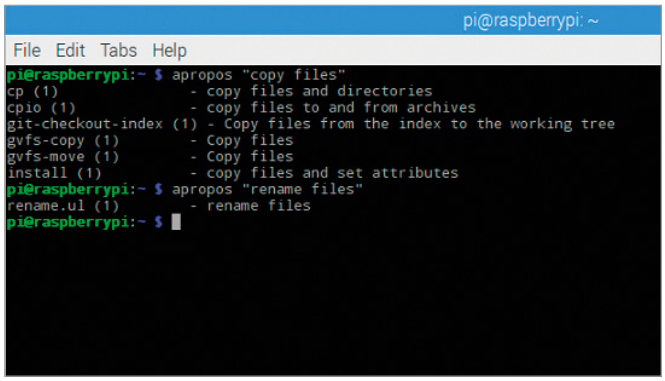
This creates a New-Dir with four sub folders within.

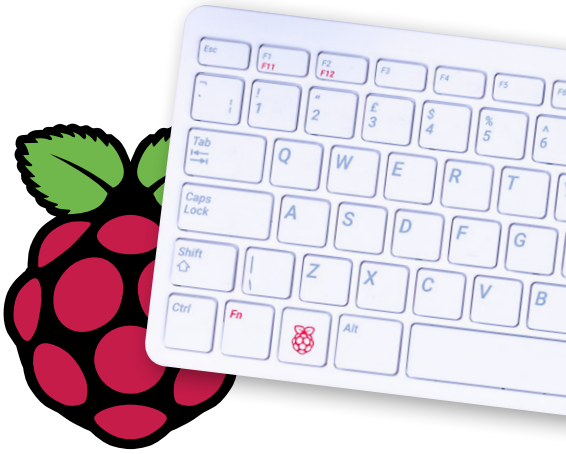


FORGOTTEN COMMANDS

It's not easy trying to remember all the available Linux commands. Thankfully, you can use apropos to help. Simply use it with a description of the command:

```
apropos "copy files"
apropos "rename files"
```





Command Line Quick Reference

When you start using Linux full time, you will quickly realise that the graphical interfaces of Ubuntu, Mint, etc. are great for many tasks but not great for all tasks. Understanding how to use the command line not only builds your understanding of Linux but also improves your knowledge of coding and programming in general. Our command line quick reference guide is designed to help you master Linux quicker.

TOP 10 COMMANDS

These may not be the most common commands used by everyone but they will certainly feature frequently for many users of Linux and the command line.

cd

The 'cd' command is one of the commands you will use the most at the command line in Linux. It allows you to change your working directory. You use it to move around within the hierarchy of your file system. You can also use chdir.

ls

The 'ls' command shows you the files in your current directory. Used with certain options, it lets you see file sizes, when files where created and file permissions. For example, `ls ~` shows you the files that are in your home directory.

cp

The 'cp' command is used to make copies of files and directories. For example, `cp file sub` makes an exact copy of the file whose name you entered and names the copy sub but the first file will still exist with its original name.

pwd

The 'pwd' command prints the full pathname of the current working directory (pwd stands for "print working directory"). Note that the GNOME terminal also displays this information in the title bar of its window.

clear

The 'clear' command clears your screen if this is possible. It looks in the environment for the terminal type and then in the terminfo database to figure out how to clear the screen. This is equivalent to typing Control-L when using the bash shell.

mv

The 'mv' command moves a file to a different location or renames a file. For example `mv file sub` renames the original file to sub. `mv sub ~/Desktop` moves the file 'sub' to your desktop directory but does not rename it. You must specify a new filename to rename a file.

chown

The 'chown' command changes the user and/or group ownership of each given file. If only an owner (a user name or numeric user ID) is given, that user is made the owner of each given file, and the files' group is not changed.

chmod

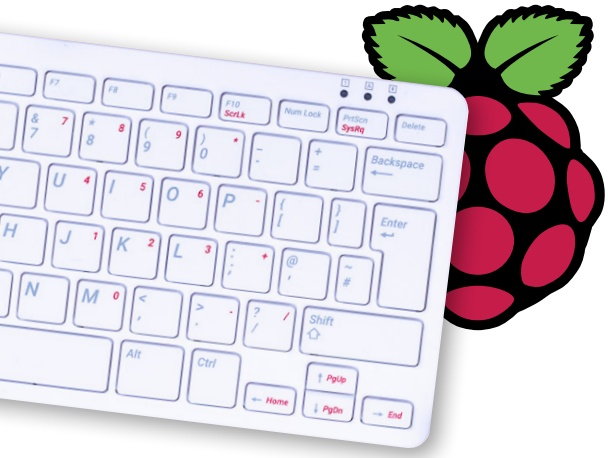
The 'chmod' command changes the permissions on the files listed. Permissions are based on a fairly simple model. You can set permissions for user, group and world and you can set whether each can read, write and or execute the file.

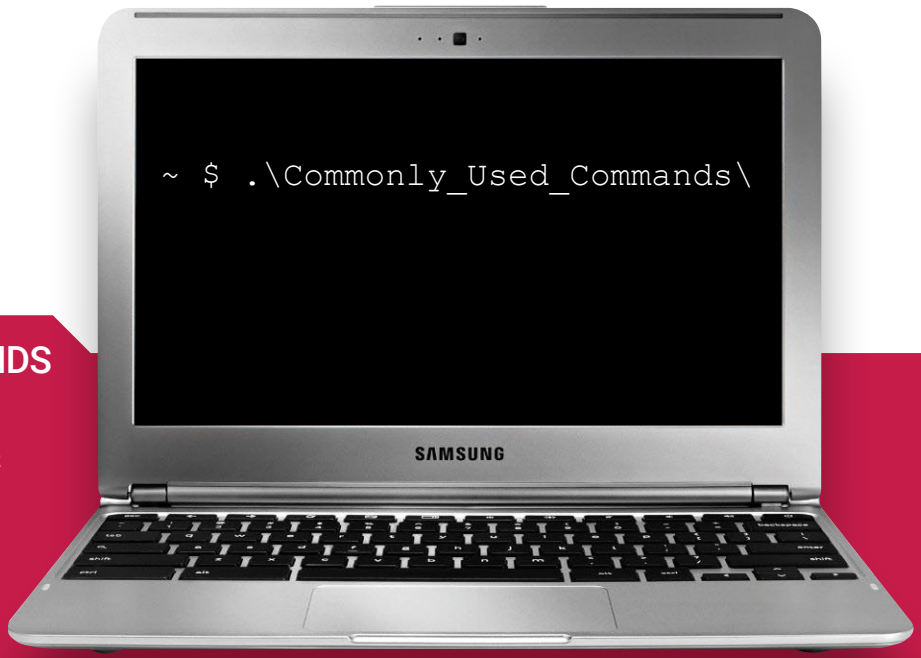
rm

The 'rm' command removes (deletes) files or directories. The removal process unlinks a filename in a filesystem from data on the storage device and marks that space as usable by future writes. In other words, removing files increases the amount of available space on your disk.

mkdir

Short for "make directory", 'mkdir' is used to create directories on a file system, if the specified directory does not already exist. For example, `mkdir work` creates a work directory. More than one directory may be specified when calling mkdir.





USEFUL HELP/INFO COMMANDS

The following commands are useful for when you are trying to learn more about the system or program you are working with in Linux. You might not need them every day, but when you do, they will be invaluable.

free — The 'free' command displays the total amount of free and used physical and swap memory in the system. For example, `free -m` gives the information using megabytes.

df — The 'df' command displays filesystem disk space usage for all partitions. The command `df-h` is probably the most useful (the -h means human-readable).

top — The 'top' program provides a dynamic real-time view of a running system. It can display system summary information, as well as a list of processes.

uname — The 'uname' command with the `-a` option prints all system information, including machine name, kernel name, version and a few other details.

ps — The 'ps' command allows you to view all the processes running on the machine. Every operating system's version of ps is slightly different but all do the same thing.

grep — The 'grep' command allows you to search inside a number of files for a particular search pattern and then print matching lines. An example would be: `grep blah file`.

sed — The 'sed' command opens a stream editor. A stream editor is used to perform text transformations on an input stream: a file or input from a pipeline.

adduser — The 'adduser 'command adds a new user to the system. Similarly, the `addgroup` command adds a new group to the system.

deluser — The 'deluser' command removes a user from the system. To remove the user's files and home directory, you need to add the `-remove-home` option.

delgroup — The 'delgroup' command removes a group from the system. You cannot remove a group that is the primary group of any users.

man man — The 'man man' command brings up the manual entry for the man command, which is a great place to start when using it.

man intro — The 'man intro' command is especially useful. It displays the Introduction to User Commands, which is a well written, fairly brief introduction to the Linux command line.

A-Z of Linux Commands

There are literally thousands of Linux commands, so while this is not a complete A-Z, it does contain many of the commands you will most likely need. You will probably find that you end up using a smaller set of commands over and over again but having an overall knowledge is still very useful.

A

- `adduser` Add a new user
- `arch` Print machine architecture
- `awk` Find and replace text within file(s)

B

- `bc` An arbitrary precision calculator language

C

- `cat` Concatenate files and print on the standard output
- `chdir` Change working directory
- `chgrp` Change the group ownership of files
- `chroot` Change root directory
- `cksum` Print CRC checksum and byte counts
- `cmp` Compare two files
- `comm` Compare two sorted files line by line
- `cp` Copy one or more files to another location
- `crontab` Schedule a command to run at a later time
- `csplit` Split a file into context-determined pieces
- `cut` Divide a file into several parts

D

- `date` Display or change the date & time
- `dc` Desk calculator
- `dd` Data Dump, convert and copy a file
- `diff` Display the differences between two files
- `dirname` Convert a full path name to just a path
- `du` Estimate file space usage

E

- `echo` Display message on screen
- `ed` A line oriented text editor (edlin)
- `egrep` Search file(s) for lines that match an extended expression
- `env` Display, set or remove environment variables
- `expand` Convert tabs to spaces
- `expr` Evaluate expressions

F

- `factor` Print prime factors
- `fdisk` Partition table manipulator for Linux
- `fgrep` Search file(s) for lines that match a fixed string
- `find` Search for files that meet a desired criteria
- `fmt` Reformat paragraph text
- `fold` Wrap text to fit a specified width
- `format` Format disks or tapes
- `fsck` Filesystem consistency check and repair

G

- `gawk` Find and Replace text within file(s)
- `grep` Search file(s) for lines that match a given pattern
- `groups` Print group names a user is in
- `gzip` Compress or decompress named file(s)

H

- `head` Output the first part of file(s)
- `hostname` Print or set system name

I

- `id` Print user and group ids
- `info` Help info
- `install` Copy files and set attributes

J

- `join` Join lines on a common field

K

- `kill` Stop a process from running

L

- `less` Display output one screen at a time
- `ln` Make links between files
- `locate` Find files
- `logname` Print current login name
- `lpc` Line printer control program
- `lpr` Off line print
- `lprm` Remove jobs from the print queue

M

man	See Help manual
mkdir	Create new folder(s)
mkfifo	Make FIFOs (named pipes)
mknod	Make block or character special files
more	Display output one screen at a time
mount	Mount a file system

N

nice	Set the priority of a command or job
nl	Number lines and write files
nohup	Run a command immune to hangups

P

passwd	Modify a user password
paste	Merge lines of files
pathchk	Check file name portability
pr	Convert text files for printing
printcap	Printer capability database
printenv	Print environment variables
printf	Format and print data

Q

quota	Display disk usage and limits
quotacheck	Scan a file system for disk usage
quotactl	Set disk quotas

R

ram	Ram disk device
rcp	Copy files between two machines
rm	Remove files
rmdir	Remove folder(s)
rpm	Remote Package Manager
rsync	Remote file copy (synchronise file trees)

S

screen	Terminal window manager
sdiff	Merge two files interactively
select	Accept keyboard input
seq	Print numeric sequences
shutdown	Shutdown or restart Linux
sleep	Delay for a specified time
sort	Sort text files
split	Split a file into fixed-size pieces
su	Substitute user identity
sum	Print a checksum for a file
symlink	Make a new name for a file
sync	Synchronise data on disk with memory

T

tac	Concatenate and write files in reverse
tail	Output the last part of files
tar	Tape Archiver
tee	Redirect output to multiple files
test	Evaluate a conditional expression
time	Measure Program Resource Use
touch	Change file timestamps
top	List processes running on the system
traceroute	Trace Route to Host
tr	Translate, squeeze and or delete characters
tsort	Topological sort

U

umount	Unmount a device
unexpand	Convert spaces to tabs
uniq	Uniquify files
units	Convert units from one scale to another
unshar	Unpack shell archive scripts
useradd	Create new user account
usermod	Modify user account
users	List users currently logged in

V

vdir	Verbosely list directory contents (`ls -l -b')

W

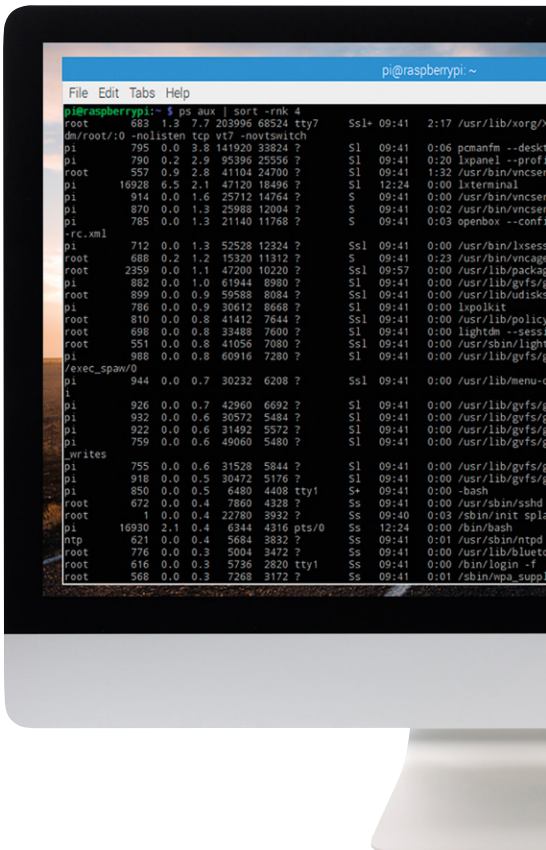
watch	Execute or display a program periodically
wc	Print byte, word, and line counts
whereis	Report all known instances of a command
which	Locate a program file in the user's path
who	Print all usernames currently logged in
whoami	Print the current user id and name

X

xargs	Execute utility, passing constructed argument list(s)

Y

yes	Print a string until interrupted



15 Essential Commands

In addition to the commands we've previously covered, here are fifteen extra, essential commands that you'll find useful. As your Pi and Linux skills grow, these will also become more frequently used; covering networking, administration and more.

1

`lsblk` You will often find the need to list the available storage devices of your Linux system. The lsblk is one of the most used Linux commands for this purpose. This handy terminal command will present you with a tree structure of your storage devices.

2

`Service` Service is the de-facto command to invoke system-wide services from the terminal. A powerful Linux terminal command for manipulating the system; you can leverage this command for running any script directly from the terminal window.

3

`In` The ln command is a clever Linux instruction for creating symbolic links to a specific file. You can use this flexible command to produce multiple instances of a symbolic link to a file, or directory, on your disk space.

4

`Alias` This is one of the most used Linux commands by system admins, as it allows them to replace a word by another string directly from the terminal. Among other functions, this is one of the best terminal commands through which you can customize the shell and manipulate environment variables.

5

`Cal` The Cal command is one of those fun Linux commands, which shows you the current date as a calendar in an ASCII text format. Type this command, with parameters like month and year, to get specified information right into the terminal.

6

`History` Want to check the history of your terminal command sessions? The History command lets you do precisely this. When typed without any parameters, this will print out the bash history of your terminal session.

7

`Wget` This is one of the best Linux commands for downloading files from the web directly from the terminal. It's also one of those handy little terminal commands that can be used in scripts, and provides users with the ability to use the HTTP, HTTPS, and FTP Internet protocols.

8

`Iptables` The Iptables command is a terminal utility that lets system admins control the incoming and outgoing Internet traffic on any machine. It is among the most used Linux commands to define authentic traffics and for blacklisting suspicious or untrusted network requests.

9

`Curl` Curl, or rather cURL, is a very powerful network tool for transferring files over a network. This is one of those Linux commands designed to work without user interaction and is often employed in network related shell scripts.

10

`Sudo` Probably the most used Linux command. You've already used it throughout this book, and it's one that you'll keep coming back to as you progress with Linux. Basically, it lets non-privileged users gain access to, and modify, files that only high-level users can access.

11

`Exit` This can be used to exit from the Terminal session, or from the current user group to the last group. It's also used when you need to exit from a remote connection inside the Terminal.

12

`Head` This handy command will display the first ten lines of a file; extremely useful for when connecting to a remote system and trawling through multiple files.

13

`Type` The Type command will display information regarding any of the current commands installed in the system. For example, it will display the location of any of the Terminal commands, so you know where to look for any lost packages.

14

`Vi` Opens the Vi text editor in the Terminal. Vi is a traditional command, which first appeared in 1976 – over 43 years ago, and is still one of the top, go-to text editors for seasoned system admins and users alike.

15

`W` A simple enough command, although quite powerful. The W command will display exactly who is logged in to the system and what they're doing. Used by system admins to see what's going on in their systems, it's something you'll probably come to use as your Linux skills improve.

Python on the Pi

Being able to code is part of making not just the Raspberry Pi, but all the connected devices that can access the Internet, do what you want them to. The Raspberry Pi is a perfect coding base that comes pre-installed with the latest version of one of the world's most popular programming languages: Python.

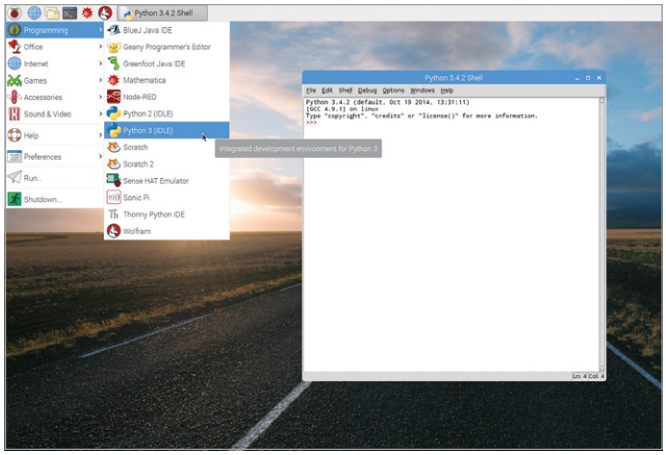
Starting Python for the First Time

If you're using the new Raspberry Pi, together with its latest release of Raspbian, then you will need to manually install the Python IDLE. This is due to the Pi team removing the core Python IDLE in favour of replacing it with their own coding text editor.

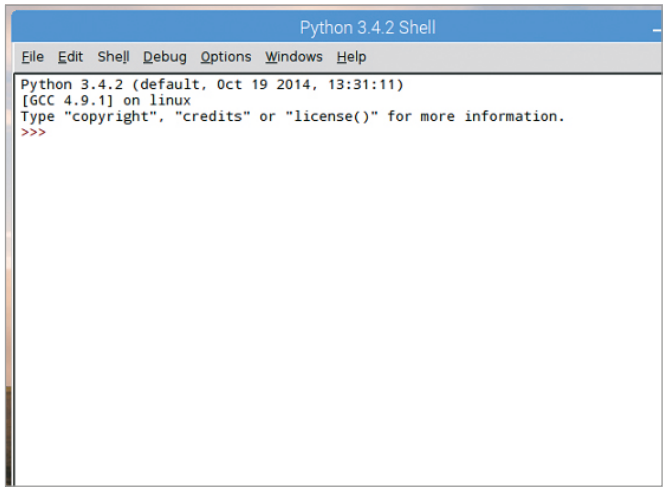
STARTING PYTHON

For those using the Pi 4 and new Raspbian, drop into a Terminal and enter: `sudo apt-get install idle3`. Older versions of Raspbian already have the official Python IDLE pre-installed.

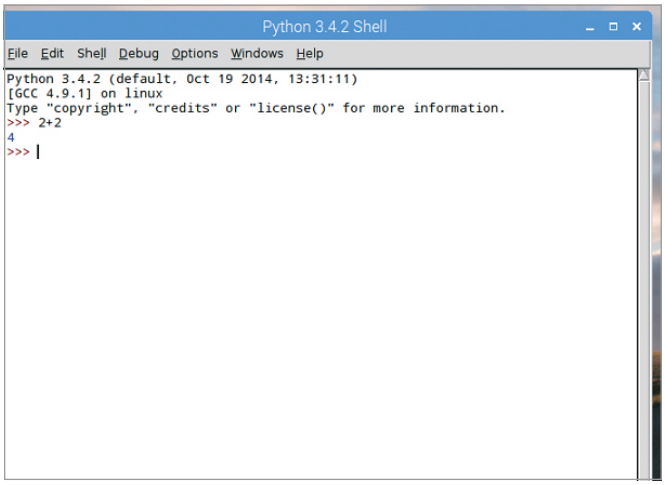
STEP 1 With the Raspbian desktop loaded, click on the Menu button followed by Programming > Python 3 (IDLE). This will open the Python 3 Shell. Windows and Mac users can find the Python 3 IDLE Shell from within the Windows Start button menu and via Finder.



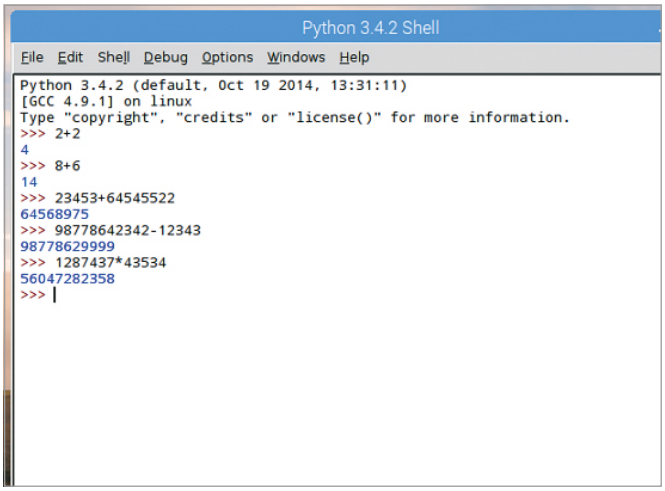
STEP 2 The Shell is where you can enter code and see the responses and output of code you've programmed into Python. This is a kind of sandbox, where you're able to try out some simple code and processes.



STEP 3 For example, in the Shell enter: `2+2` After pressing Enter, the next line will display the answer: 4. Basically, Python has taken the 'code' and produced the relevant output.



STEP 4 The Python Shell acts very much like a calculator, since code is basically a series of mathematical interactions with the system. Integers, which are the infinite sequence of whole numbers can easily be added, subtracted, multiplied and so on.

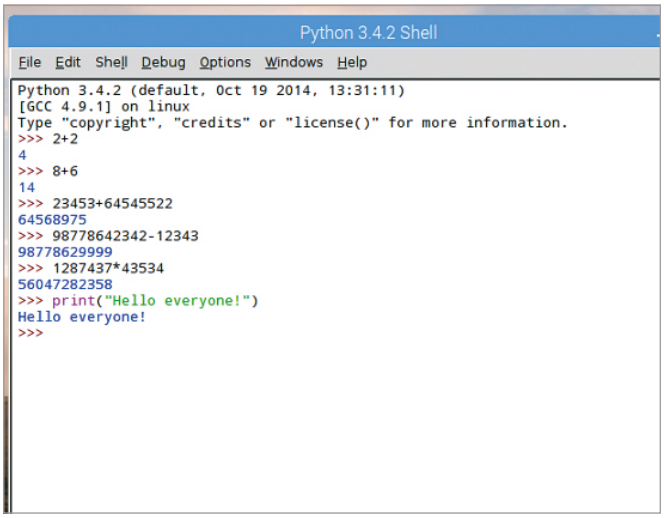


STEP 5

While that's very interesting, it's not particularly exciting. Instead, try this:

```
print("Hello everyone!")
```

Just enter it into the IDLE as you've done in the previous steps.



STEP 6

This is a little more like it, since you've just produced your first bit of code. The Print command is fairly self-explanatory, it prints things. Python 3 requires the brackets as well as quote marks in order to output content to the screen, in this case the Hello everyone! bit.

STEP 7

You may have noticed the colour coding within the Python IDLE. The colours represent different elements of Python code. They are:

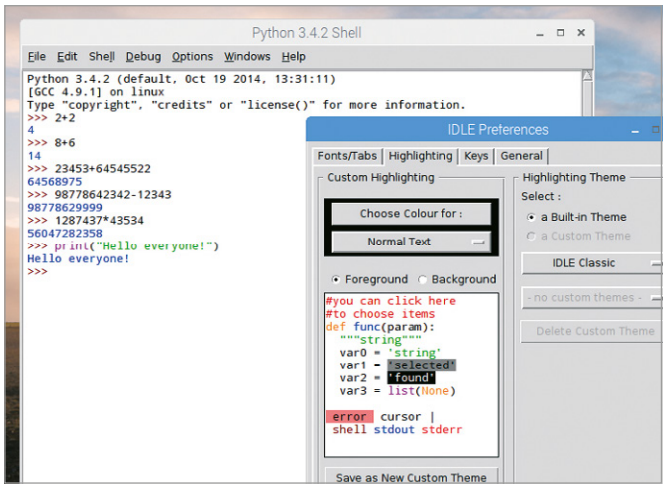
Black – Data and Variables
Green – Strings
Purple – Functions
Orange – Commands
Blue – User Functions
Dark Red – Comments
Light Red – Error Messages

IDLE Colour Coding

Colour	Use for	Examples
Black	Data & variables	`23.6 area`
Green	Strings	`"Hello World"`
Purple	Functions	`len() print()`
Orange	Commands	`if for else`
Blue	User functions	`get_area()`
Dark red	Comments	`#Remember VAT`
Light red	Error messages	`SyntaxError:`

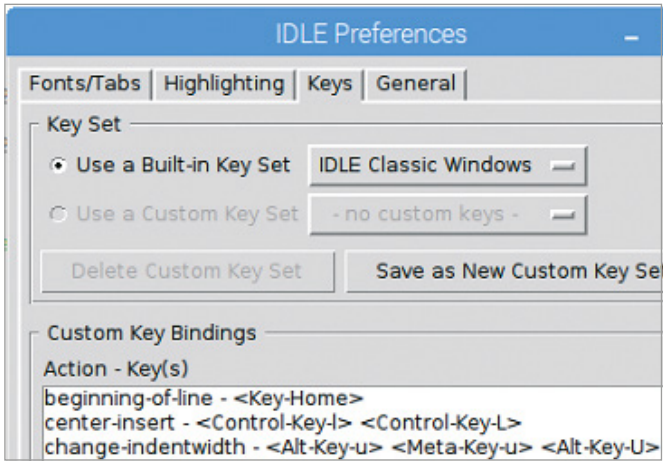
STEP 8

The Python IDLE is a configurable environment. If you don't like the way the colours are represented, then you can always change them via Options > Configure IDLE and clicking on the Highlighting tab. However, we don't recommend that as you won't be seeing the same as our screenshots.



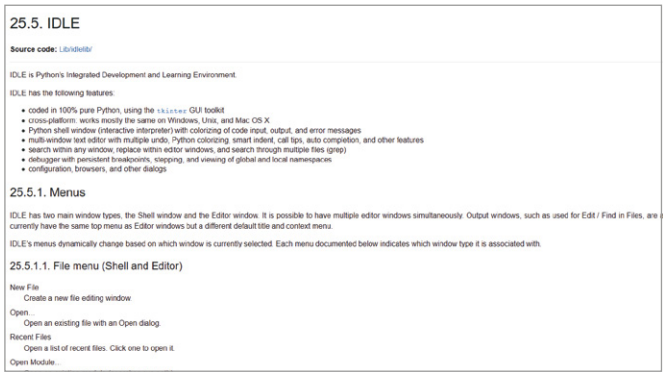
STEP 9

Just like most programs available, regardless of the operating system, there are numerous shortcut keys available. We don't have room for them all here but within the Options > Configure IDLE and under the Keys tab, you can see a list of the current bindings.



STEP 10

The Python IDLE is a powerful interface, and one that's actually been written in Python using one of the available GUI toolkits. If you want to know the many ins and outs for the Shell, we recommend you take a few moments to view **www.docs.python.org/3/library/idle.html**, which details many of the IDLE's features.



Your First Code

Essentially, you've already written your first piece of code with the 'print("Hello everyone!")' function from the previous tutorial. However, let's expand that and look at entering your code and playing around with some other Python examples.

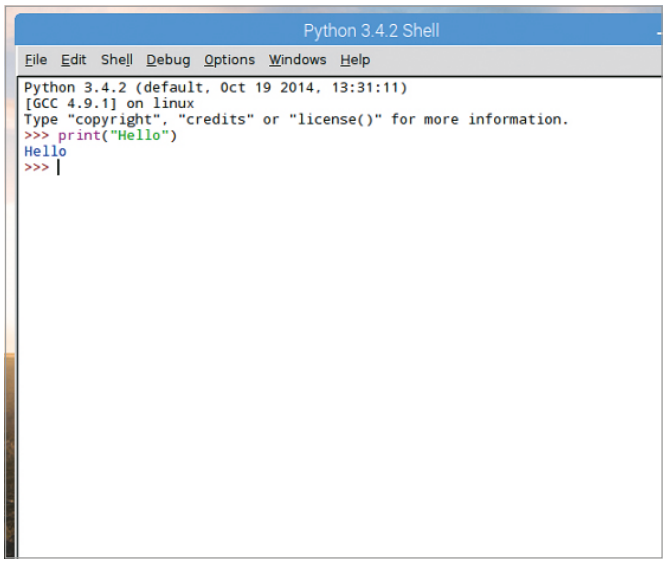
PLAYING WITH PYTHON

With most languages, computer or human, it's all about remembering and applying the right words to the right situation. You're not born knowing these words, so you need to learn them.

STEP 1

If you've closed Python 3 IDLE, reopen it in whichever operating system version you prefer. In the Shell, enter the familiar following:

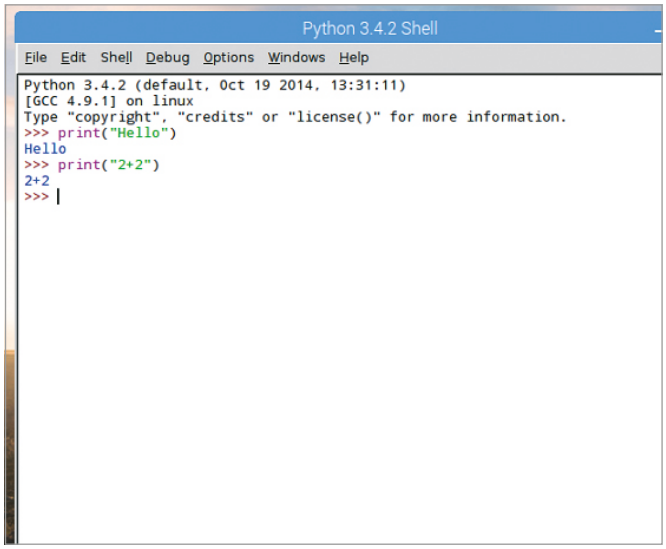
```
print("Hello")
```



STEP 2

Just as predicted, the word Hello appears in the Shell as blue text, indicating output from a string. It's fairly straightforward and doesn't require too much explanation. Now try:

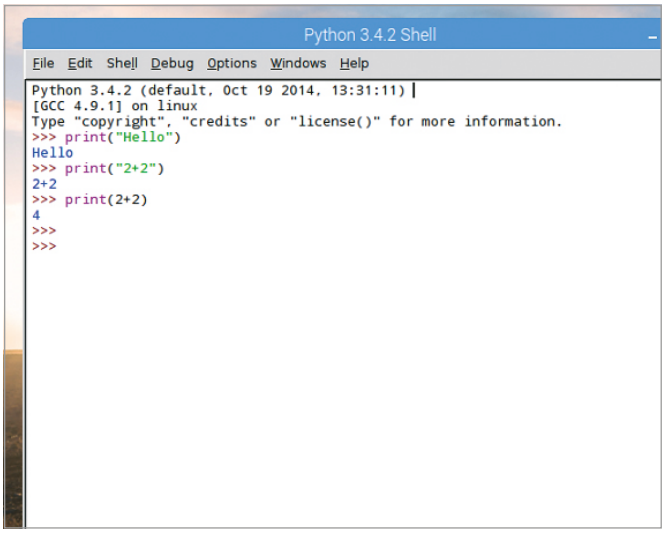
```
print("2+2")
```



STEP 3

You can see that instead of the number 4, the output is the 2+2 you asked to be printed to the screen. The quotation marks are defining what's being outputted to the IDLE Shell; to print the total of 2+2 you need to remove the quotes:

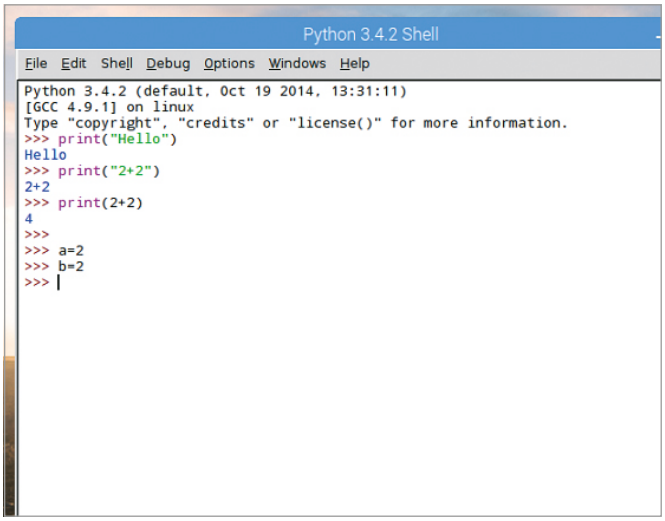
```
print(2+2)
```



STEP 4

You can continue as such, printing 2+2, 464+2343 and so on to the Shell. An easier way is to use a variable, which is something we will cover in more depth later. For now, enter:

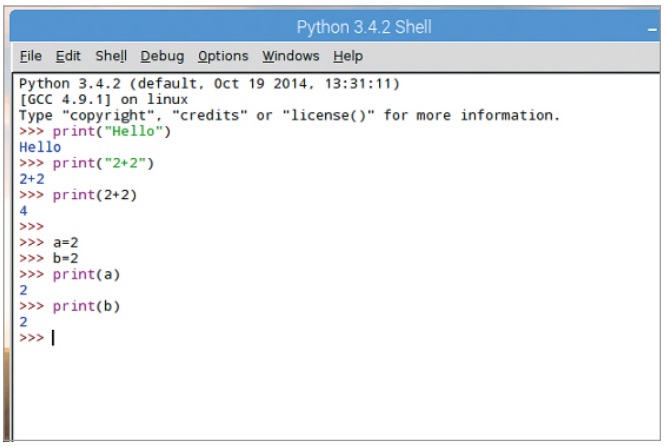
```
a=2
b=2
```



STEP 5

What you have done here is assign the letters a and b two values: 2 and 2. These are now variables, which can be called upon by Python to output, add, subtract, divide and so on for as long as their numbers stay the same. Try this:

```
print(a)
print(b)
```

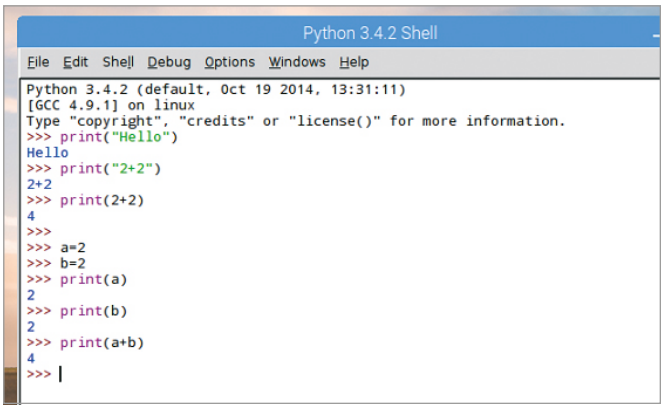


STEP 6

The output of the last step displays the current values of both a and b individually, as you've asked them to be printed separately. If you want to add them up, you can use the following:

```
print(a+b)
```

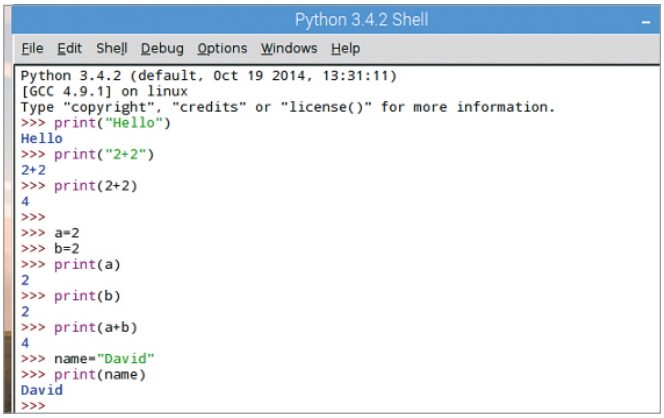
This code simply takes the values of a and b, adds them together and outputs the result.



STEP 7

You can play around with different kinds of variables and the Print function. For example, you could assign variables for someone's name:

```
name="David"
print(name)
```

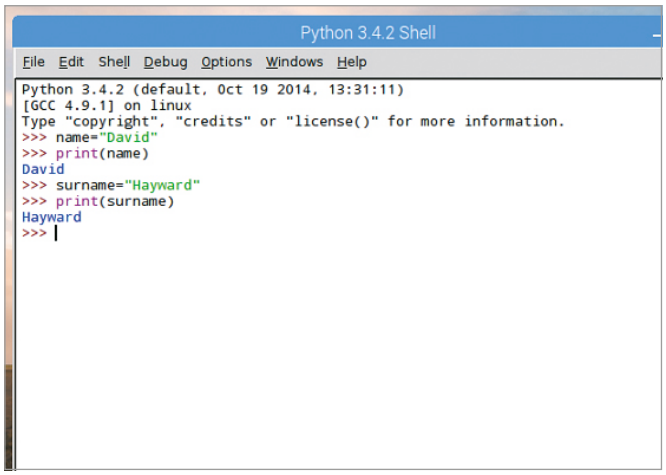


STEP 8

Now let's add a surname:

```
surname="Hayward"
print(surname)
```

You now have two variables containing a first name and a surname and you can print them independently.

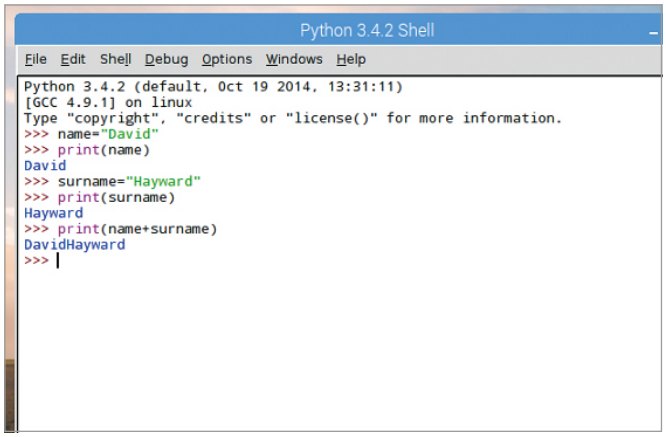


STEP 9

If we were to apply the same routine as before, using the + symbol, the name wouldn't appear correctly in the output in the Shell. Try it:

```
print(name+surname)
```

You need a space between the two, defining them as two separate values and not something you mathematically play around with.



STEP 10

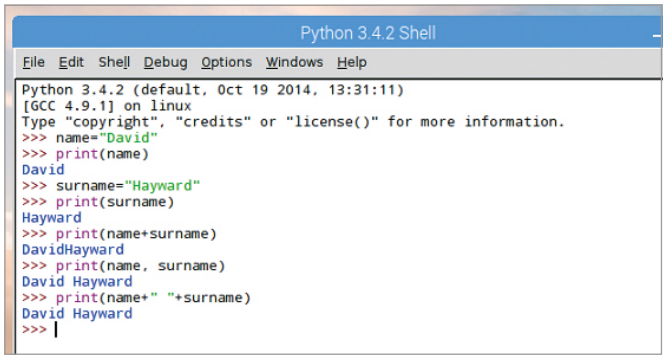
In Python 3 you can separate the two variables with a space using a comma:

```
print(name, surname)
```

Alternatively, you can add the space ourselves:

```
print(name+" "+surname)
```

The use of the comma is much neater, as you can see. Congratulations, you've just taken your first steps into the wide world of Python.



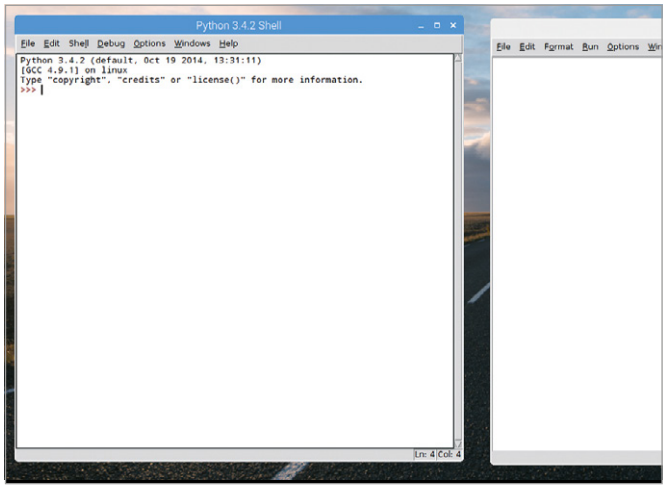
Saving and Executing Your Code

While working in the IDLE Shell is perfectly fine for small code snippets, it's not designed for entering longer program listings. In this section you're going to be introduced to the IDLE Editor, where you will be working from now on.

EDITING CODE

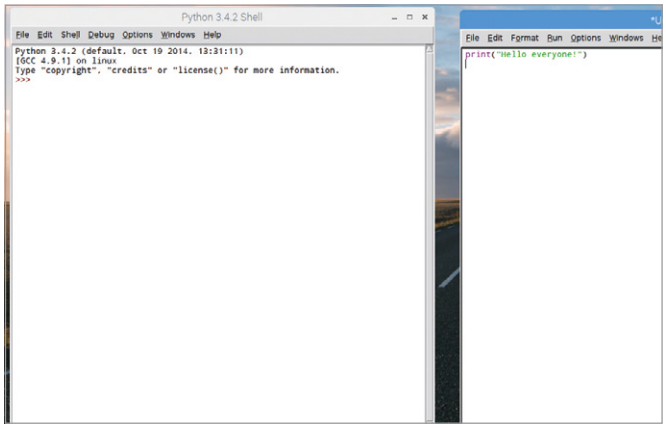
You will eventually reach a point where you have to move on from inputting single lines of code into the Shell. Instead, the IDLE Editor will allow you to save and execute your Python code.

STEP 1 First, open the Python IDLE Shell and when it's up, click on File > New File. This will open a new window with Untitled as its name. This is the Python IDLE Editor and within it you can enter the code needed to create your future programs.

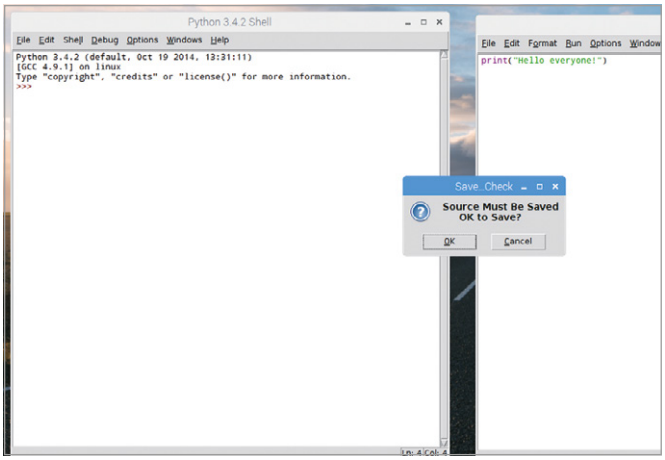


STEP 2 The IDLE Editor is, for all intents and purposes, a simple text editor with Python features, colour coding and so on; much in the same vein as Sublime. You enter code as you would within the Shell, so taking an example from the previous tutorial, enter:

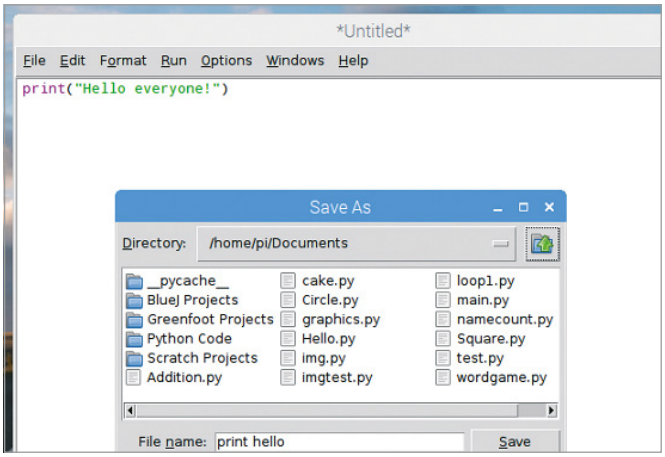
```
print("Hello everyone!")
```



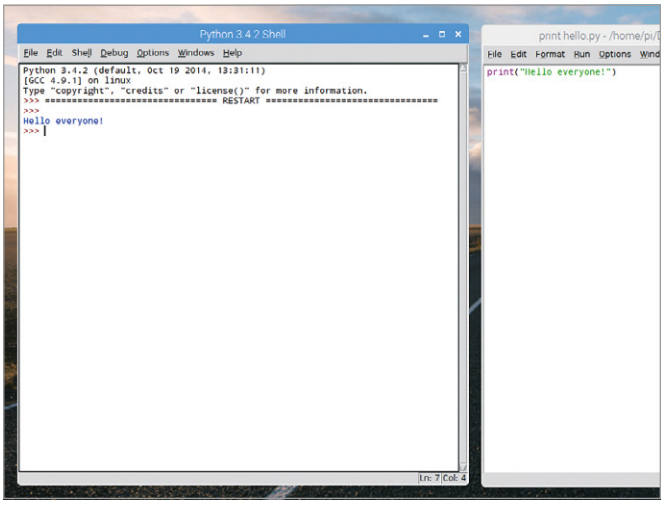
STEP 3 You can see that the same colour coding is in place in the IDLE Editor as it is in the Shell, enabling you to better understand what's going on with your code. However, to execute the code you need to first save it. Press F5 and you get a Save…Check box open.



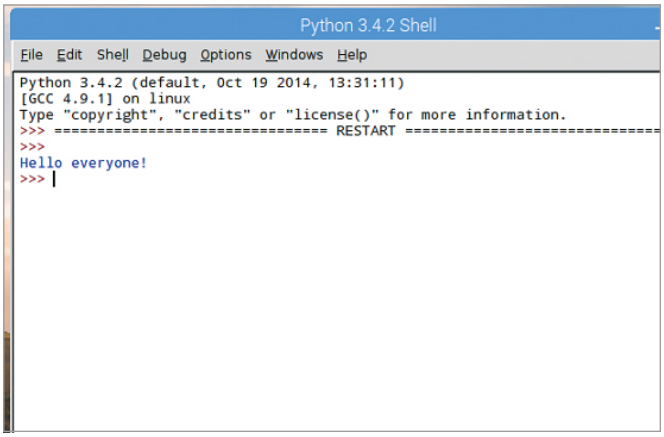
STEP 4 Click on the OK button in the Save box and select a destination where you'll save all your Python code. The destination can be a dedicated folder called Python or you can just dump it wherever you like. Remember to keep a tidy drive though, to help you out in the future.



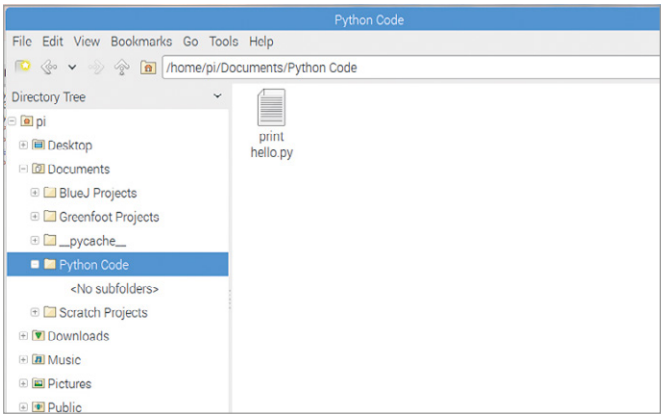
STEP 5 Enter a name for your code, 'print hello' for example, and click on the Save button. Once the Python code is saved it's executed and the output will be detailed in the IDLE Shell. In this case, the words 'Hello everyone!'.



STEP 6 This is how the vast majority of your Python code will be conducted. Enter it into the Editor, hit F5, save the code and look at the output in the Shell. Sometimes things will differ, depending on whether you've requested a separate window, but essentially that's the process. It's the process we will use throughout this book, unless otherwise stated.



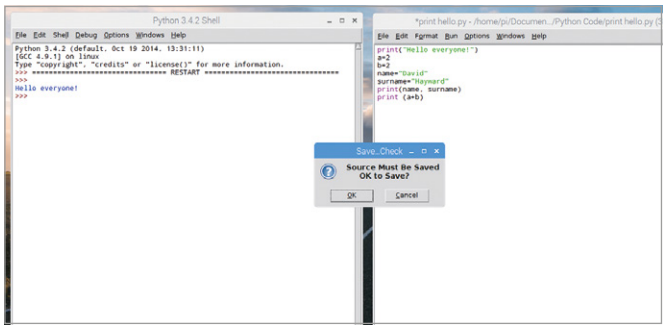
STEP 7 If you open the file location of the saved Python code, you can see that it ends in a .py extension. This is the default Python file name. Any code you create will be whatever.py and any code downloaded from the many internet Python resource sites will be .py. Just ensure that the code is written for Python 3.



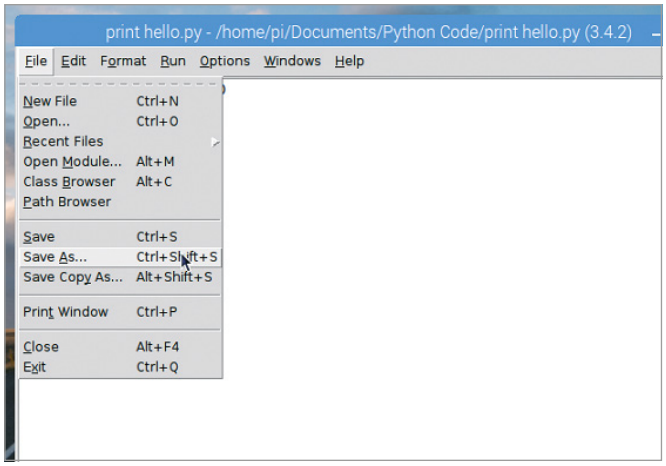
STEP 8 Let's extend the code and enter a few examples from the previous tutorial:

```
a=2
b=2
name="David"
surname="Hayward"
print(name, surname)
print (a+b)
```

If you press F5 now you'll be asked to save the file, again, as it's been modified from before.



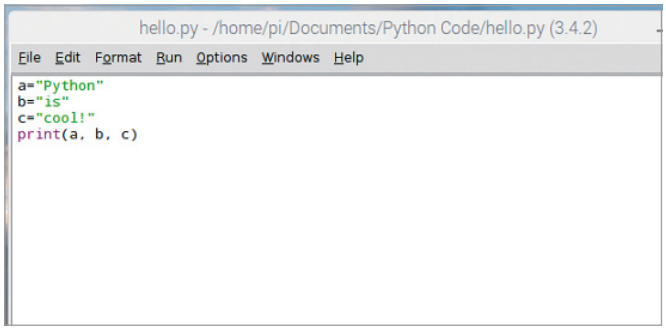
STEP 9 If you click the OK button, the file will be overwritten with the new code entries, and executed, with the output in the Shell. It's not a problem with just these few lines but if you were to edit a larger file, overwriting can become an issue. Instead, use File > Save As from within the Editor to create a backup.



STEP 10 Now create a new file. Close the Editor, and open a new instance (File > New File from the Shell). Enter the following and save it as hello.py:

```
a="Python"
b="is"
c="cool!"
print(a, b, c)
```

You will use this code in the next tutorial.



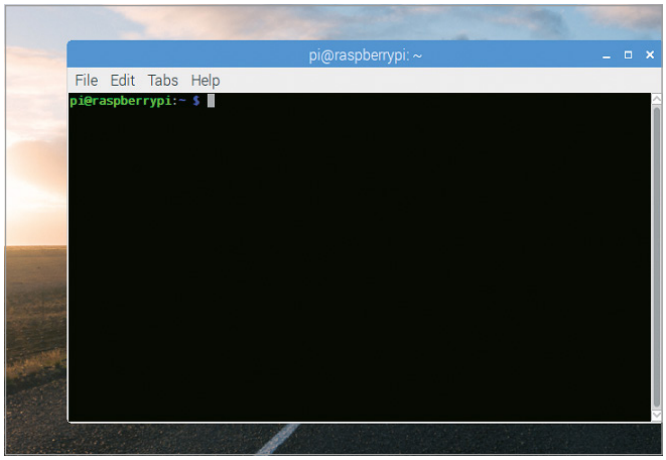
Executing Code from the Command Line

Although we're working from the GUI IDLE throughout this book, it's worth taking a look at Python's command line handling. We already know there's a command line version of Python but it's also used to execute code.

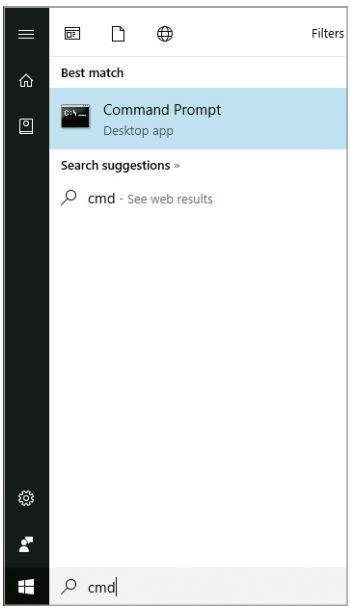
COMMAND THE CODE

Using the code we created in the previous tutorial, the one we named hello.py, let's see how you can run code that was made in the GUI at the command line level.

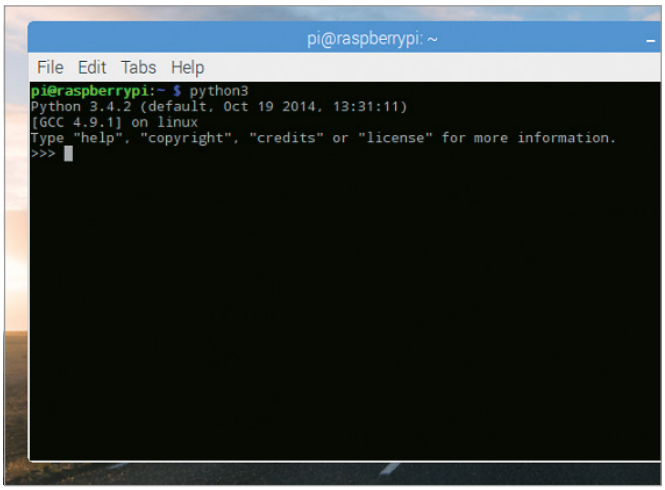
STEP 1 Python, in Linux, comes with two possible ways of executing code via the command line. One of the ways is with Python 2, whilst the other uses the Python 3 libraries and so on. First though, drop into the command line or Terminal on your operating system.



STEP 2 Just as before, we're using a Raspberry Pi: Windows users will need to click the Start button and search for CMD, then click the Command Line returned search; and macOS users can get access to their command line by clicking Go > Utilities > Terminal.



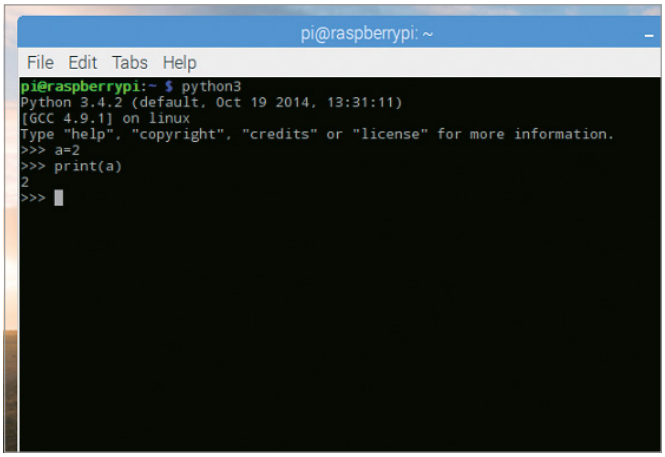
STEP 3 Now you're at the command line we can start Python. For Python 3 you need to enter the command python3 and press Enter. This will put you into the command line version of the Shell, with the familiar three right-facing arrows as the cursor (>>>).



STEP 4 From here you're able to enter the code you've looked at previously, such as:

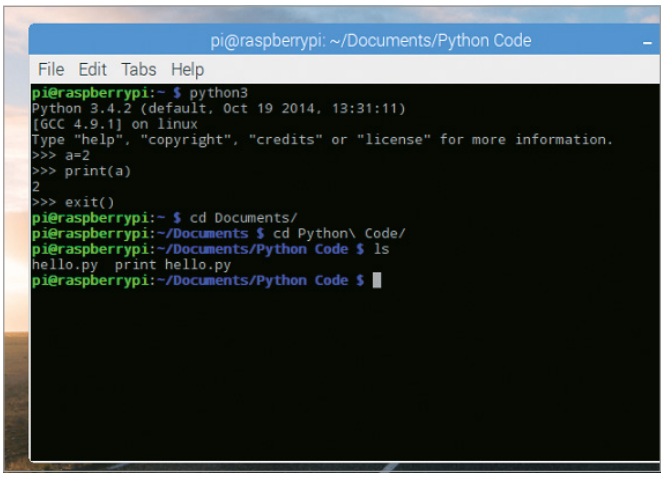
```
a=2
print(a)
```

You can see that it works exactly the same.



STEP 5

Now enter: `exit()` to leave the command line Python session and return you back to the command prompt. Enter the folder where you saved the code from the previous tutorial and list the available files within; hopefully you should see the hello.py file.



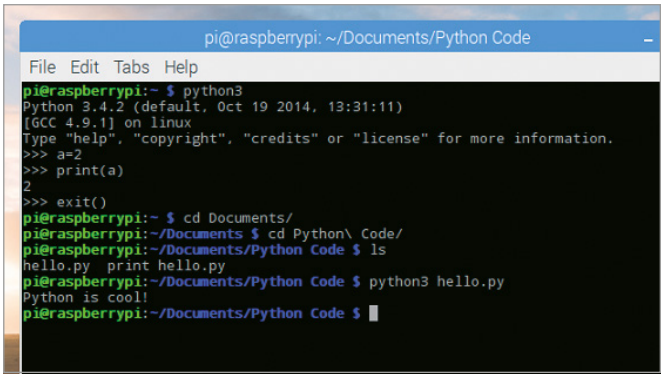
STEP 6

From within the same folder as the code you're going to run, enter the following into the command line:

```
python3 hello.py
```

This will execute the code we created, which to remind you is:

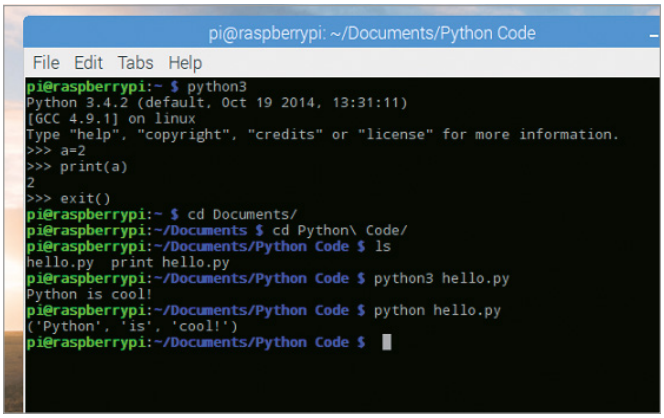
```
a="Python"
b="is"
c="cool!"
print(a, b, c)
```



STEP 7

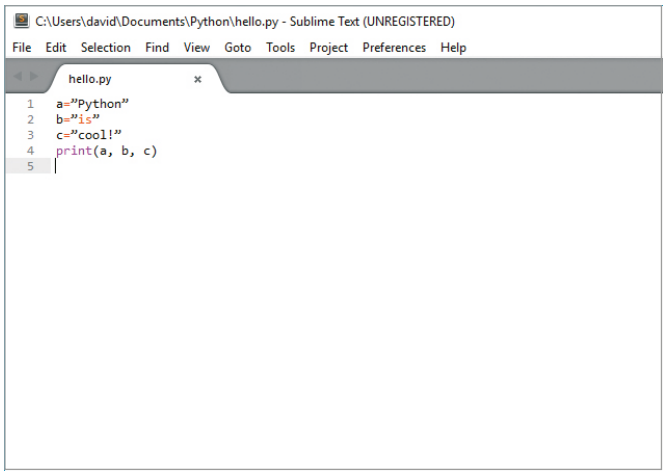
Naturally, since this is Python 3 code, using the syntax and layout that's unique to Python 3, it only works when you use the python3 command. If you like, try the same with Python 2 by entering:

```
python hello.py
```



STEP 8

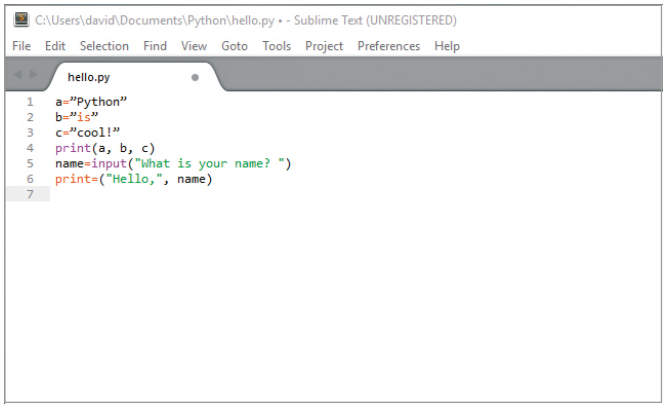
The result of running Python 3 code from the Python 2 command line is quite obvious. Whilst it doesn't error out in any way, due to the differences between the way Python 3 handles the Print command over Python 2, the result isn't as we expected. Using Sublime for the moment, open the hello.py file.



STEP 9

Since Sublime Text isn't available for the Raspberry Pi, you're going to temporarily leave the Pi for the moment and use Sublime as an example that you don't necessarily need to use the Python IDLE. With the hello.py file open, alter it to include the following:

```
name=input("What is your name? ")
print("Hello,", name)
```

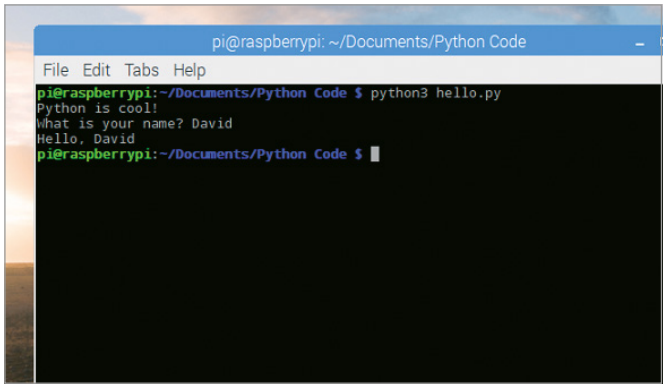


STEP 10

Save the hello.py file and drop back to the command line. Now execute the newly saved code with:

```
python3 hello.py
```

The result will be the original Python is cool! statement, together with the added input command asking you for your name, and displaying it in the command window.



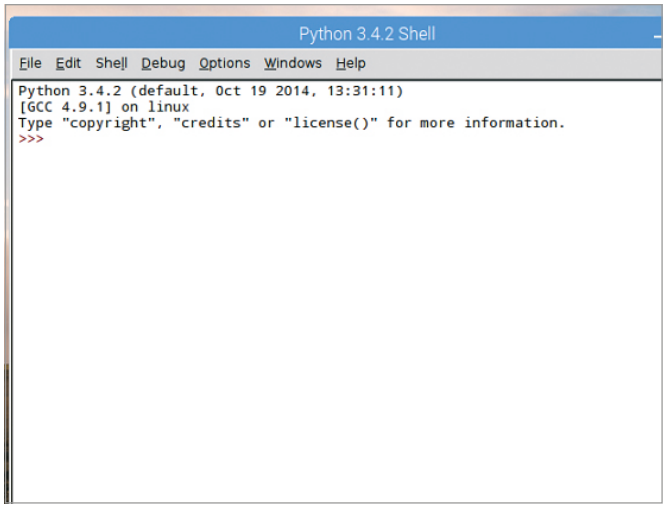
Numbers and Expressions

We've seen some basic mathematical expressions with Python, simple addition and the like. Let's expand on that now and see just how powerful Python is as a calculator. You can work within the IDLE Shell or in the Editor, whichever you like.

IT'S ALL MATHS, MAN

You can get some really impressive results with the mathematical powers of Python; as with most, if not all, programming languages, maths is the driving force behind the code.

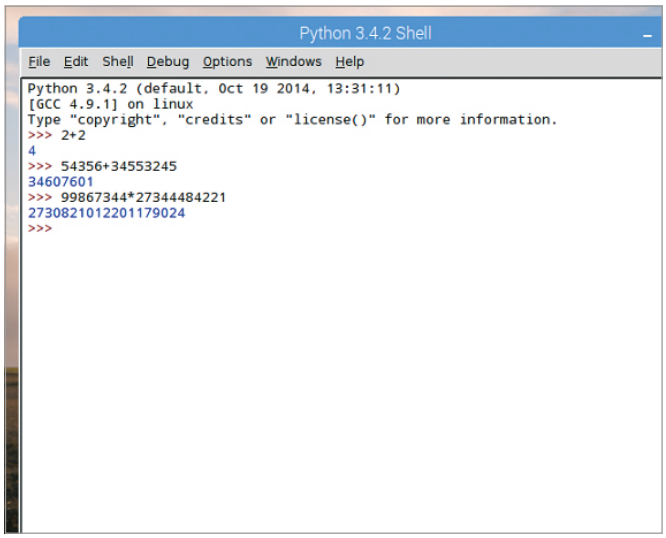
STEP 1 Open up the GUI version of Python 3, as mentioned you can use either the Shell or the Editor. For the time being, you're going to use the Shell just to warm our maths muscle, which we believe is a small gland located at the back of the brain (or not).



STEP 2 In the Shell enter the following:

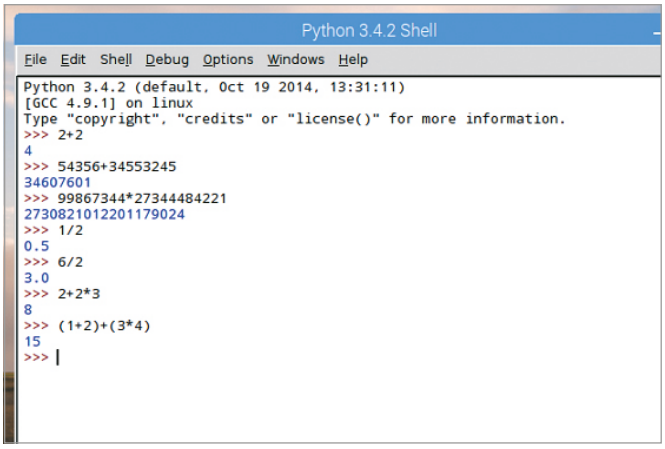
```
2+2
54356+34553245
99867344*27344484221
```

You can see that Python can handle some quite large numbers.



STEP 3 You can use all the usual mathematical operations: divide, multiply, brackets and so on. Practise with a few, for example:

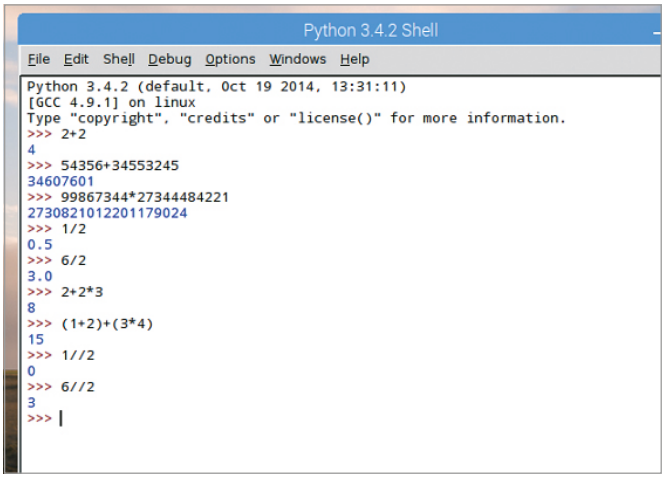
```
1/2
6/2
2+2*3
(1+2)+(3*4)
```



STEP 4 You've no doubt noticed, division produces a decimal number. In Python these are called floats, or floating point arithmetic. However, if you need an integer as opposed to a decimal answer, then you can use a double slash:

```
1//2
6//2
```

And so on.



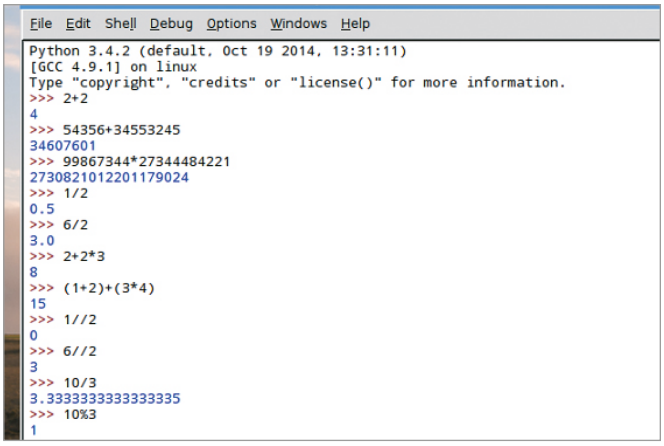
STEP 5

You can also use an operation to see the remainder left over from division. For example: `10/3`

Will display 3.333333333, which is of course 3.3-recurring.

If you now enter: `10%3`

This will display 1, which is the remainder left over from dividing 10 by 3.



STEP 6

Next up we have the power operator, or exponentiation if you want to be technical. To work out the power of something you can use a double multiplication symbol or double-star on the keyboard:

```
2**3
10**10
```

Essentially, it's 2x2x2 but we're sure you already know the basics behind maths operators. This is how you would work it out in Python.

STEP 7

Numbers and expressions don't stop there. Python has numerous built-in functions to work out sets of numbers, absolute values, complex numbers and a host of mathematical expressions and Pythagorean tongue-twisters. For example, to convert a number to binary, use:

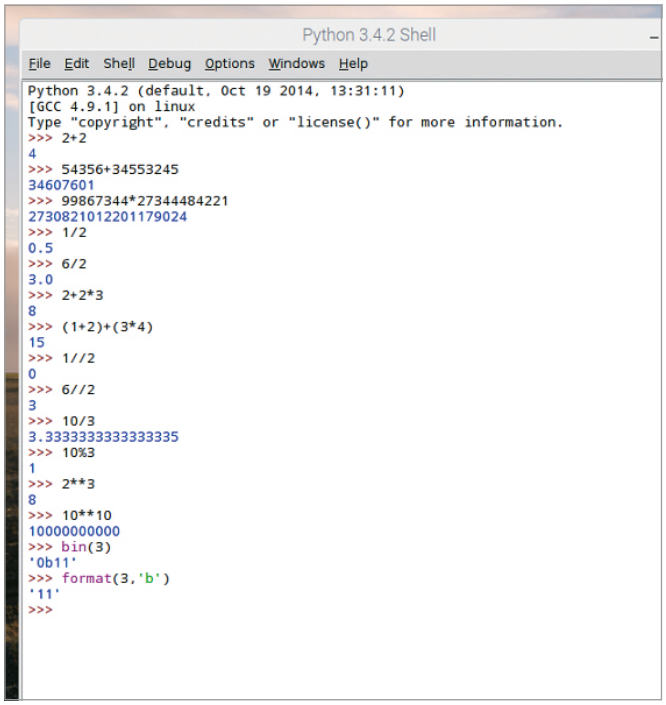
```
bin(3)
```

STEP 8

This will be displayed as '0b11', converting the integer into binary and adding the prefix 0b to the front. If you want to remove the 0b prefix, then you can use:

```
format(3, 'b')
```

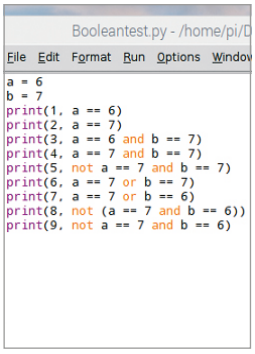
The Format command converts a value, the number 3, to a formatted representation as controlled by the format specification, the 'b' part.



STEP 9

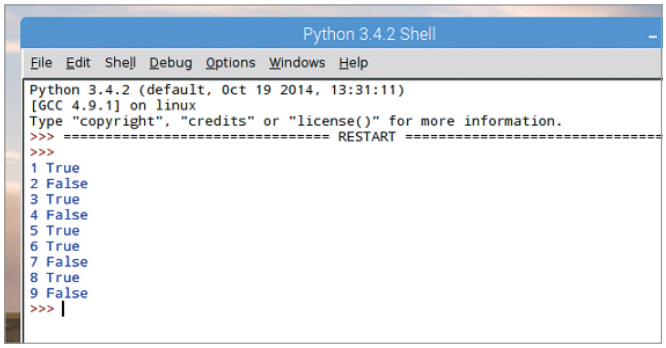
A Boolean Expression is a logical statement that will either be true or false. We can use these to compare data and test to see if it's equal to, less than or greater than. Try this in a New File:

```
a = 6
b = 7
print(1, a == 6)
print(2, a == 7)
print(3, a == 6 and b == 7)
print(4, a == 7 and b == 7)
print(5, not a == 7 and b == 7)
print(6, a == 7 or b == 7)
print(7, a == 7 or b == 6)
print(8, not (a == 7 and b == 6))
print(9, not a == 7 and b == 6)
```



STEP 10

Execute the code from Step 9, and you can see a series of True or False statements, depending on the result of the two defining values: 6 and 7. It's an extension of what you've looked at, and an important part of programming.



Using Comments

When writing your code, the flow, what each variable does, how the overall program will operate and so on is all inside your head. Another programmer could follow the code line by line but over time, it can become difficult to read.

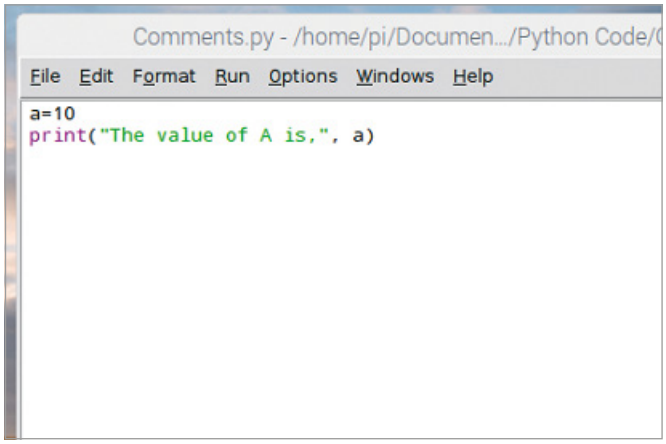
#COMMENTS!

Programmers use a method of keeping their code readable by commenting on certain sections. If a variable is used, the programmer comments on what it's supposed to do, for example. It's just good practise.

STEP 1 Start by creating a new instance of the IDLE Editor (File > New File) and create a simple variable and print command:

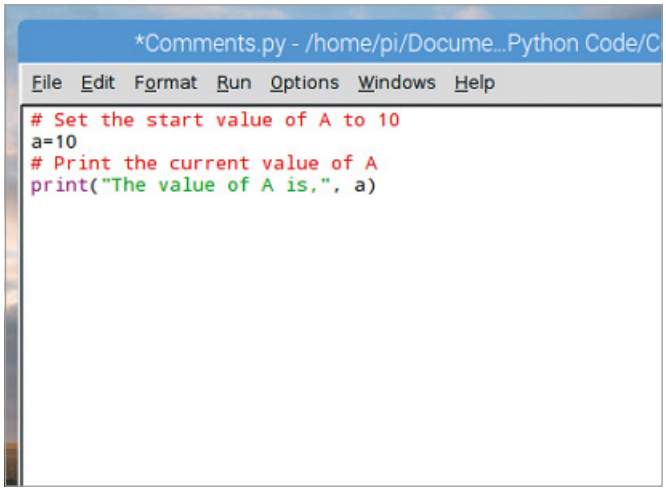
```
a=10
print("The value of A is,", a)
```

Save the file and execute the code.

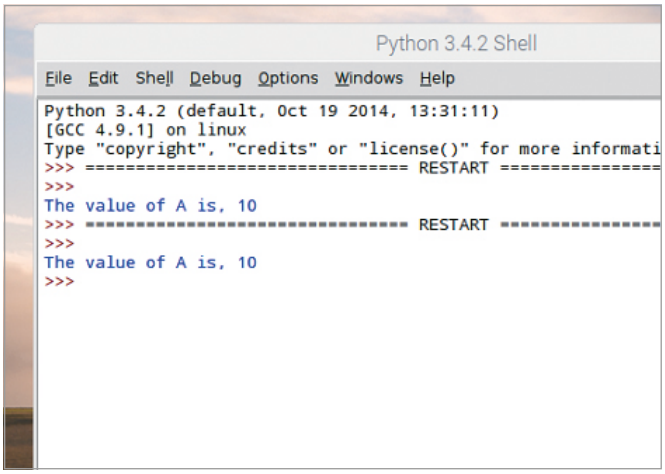


STEP 2 Running the code will return the line: The value of A is, 10 into the IDLE Shell window, which is what we expected. Now, add some of the types of comments you'd normally see within code:

```
# Set the start value of A to 10
a=10
# Print the current value of A
print("The value of A is,", a)
```

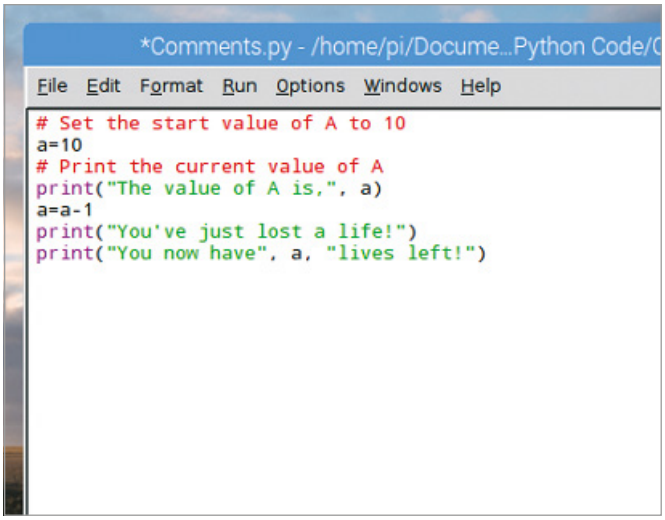


STEP 3 Resave the code and execute it. You can see that the output in the IDLE Shell is still the same as before, despite the extra lines being added. Simply put, the hash symbol (#) denotes a line of text the programmer can insert to inform them, and others, of what's going on without the user being aware.



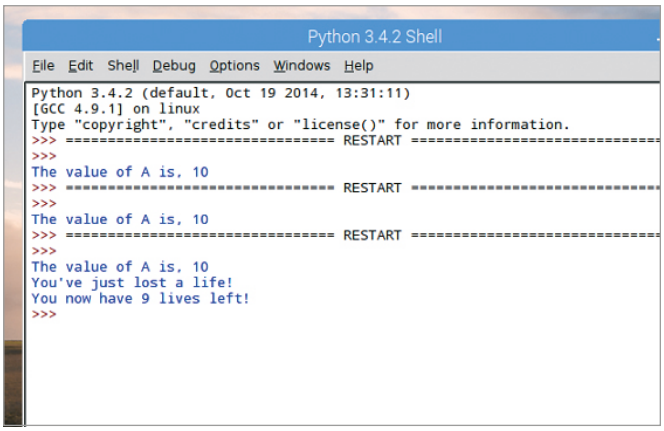
STEP 4 Let's assume that the variable A that we've created is the number of lives in a game. Every time the player dies, the value is decreased by 1. The programmer could insert a routine along the lines of:

```
a=a-1
print("You've just lost a life!")
print("You now have", a, "lives left!")
```



STEP 5

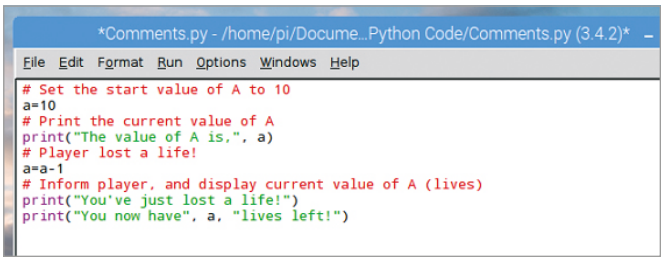
While we know that the variable A is lives, and that the player has just lost one, a casual viewer or someone checking the code may not know. Imagine for a moment that the code is twenty thousand lines long, instead of just our seven. You can see how handy comments are.



STEP 6

Essentially, the new code together with comments could look like:

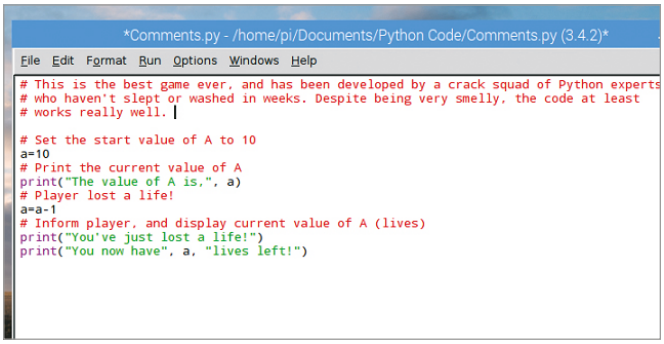
```
# Set the start value of A to 10
a=10
# Print the current value of A
print("The value of A is,", a)
# Player lost a life!
a=a-1
# Inform player, and display current value of A
(lives)
print("You've just lost a life!")
print("You now have", a, "lives left!")
```



STEP 7

You can use comments in different ways. For example, Block Comments are a large section of text that details what's going on in the code, such as telling the code reader what variables you're planning on using:

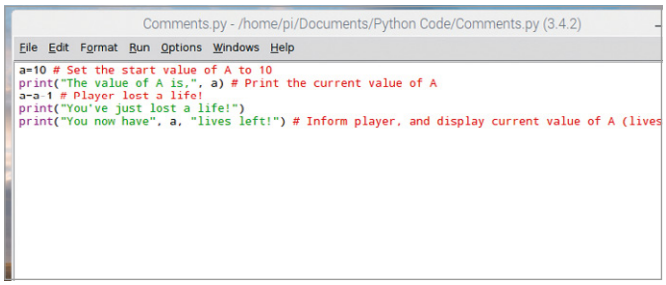
```
# This is the best game ever, and has been developed
by a crack squad of Python experts
# who haven't slept or washed in weeks. Despite
being very smelly, the code at least
# works really well.
```



STEP 8

Inline comments are comments that follow a section of code. Take our examples from above, instead of inserting the code on a separate line, we could use:

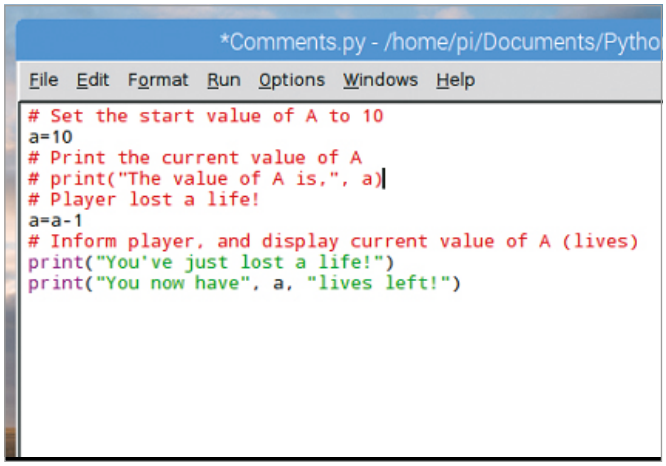
```
a=10 # Set the start value of A to 10
print("The value of A is,", a) # Print the current
value of A
a=a-1 # Player lost a life!
print("You've just lost a life!")
print("You now have", a, "lives left!") # Inform
player, and display current value of A (lives)
```



STEP 9

The comment, the hash symbol, can also be used to comment out sections of code you don't want to be executed in your program. For instance, if you wanted to remove the first print statement, you would use:

```
# print("The value of A is,", a)
```



STEP 10

You also use three single quotes to comment out a Block Comment or multi-line section of comments. Place them before and after the areas you want to comment for them to work:

```
'''
This is the best game ever, and has been developed
by a crack squad of Python experts who haven't slept
or washed in weeks. Despite being very smelly, the
code at least works really well.
'''
```

Working with Variables

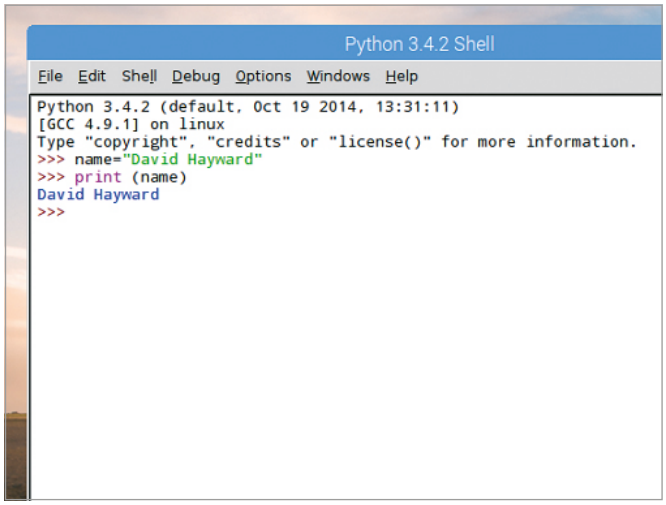
We've seen some examples of variables in our Python code already but it's always worth going through the way they operate and how Python creates and assigns certain values to a variable.

VARIOUS VARIABLES

You'll be working with the Python 3 IDLE Shell in this tutorial. If you haven't already, open Python 3 or close down the previous IDLE Shell to clear up any old code.

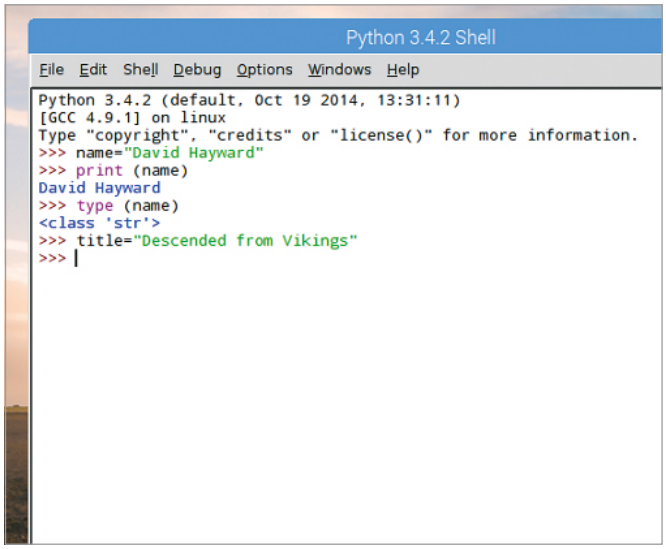
STEP 1

In some programming languages you're required to use a dollar sign to denote a string, which is a variable made up of multiple characters, such as a name of a person. In Python this isn't necessary. For example, in the Shell enter: name="David Hayward" (or use your own name, unless you're also called David Hayward).



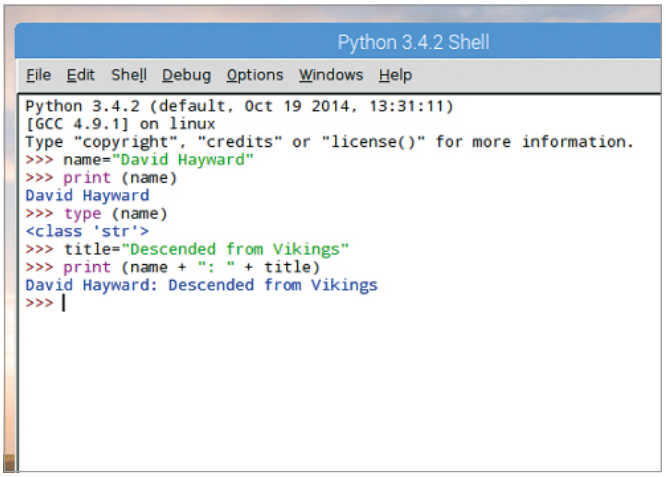
STEP 2

You can check the type of variable in use by issuing the **type ()** command, placing the name of the variable inside the brackets. In our example, this would be: `type (name)`. Add a new string variable: `title="Descended from Vikings"`.



STEP 3

You've seen previously that variables can be concatenated using the plus symbol between the variable names. In our example we can use: `print (name + ": " + title)`. The middle part between the quotations allows us to add a colon and a space, as variables are connected without spaces, so we need to add them manually.



STEP 4

You can also combine variables within another variable. For example, to combine both name and title variables into a new variable we use:

```
character=name + ": " + title
```

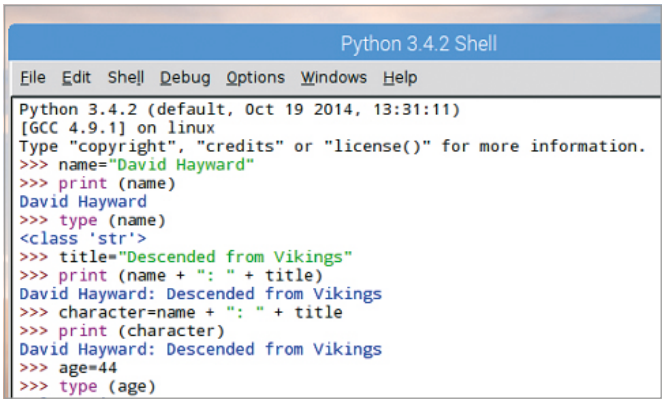
Then output the content of the new variable as:

```
print (character)
```

Numbers are stored as different variables:

```
age=44
Type (age)
```

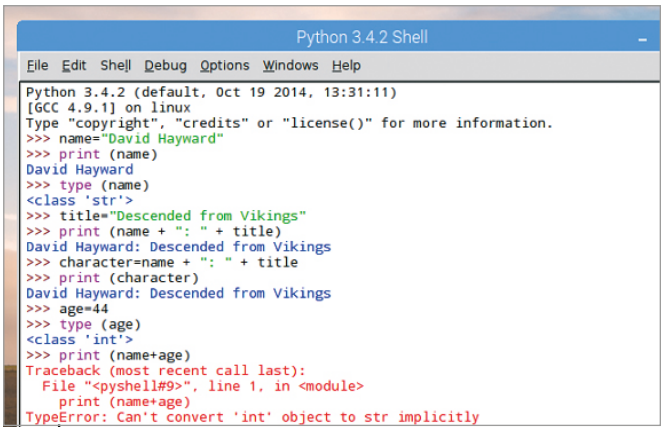
Which are integers, as we know.



STEP 5

However, you can't combine both strings and integer type variables in the same command, as you would a set of similar variables. You need to either turn one into the other or vice versa. When you do try to combine both, you get an error message:

```
print (name + age)
```



STEP 6

This is a process known as TypeCasting. The Python code is:

```
print (character + " is " + str(age) + " years old.")
```

or you can use:

```
print (character, "is", age, "years old.")
```

Notice again that in the last example, you don't need the spaces between the words in quotes as the commas treat each argument to print separately.

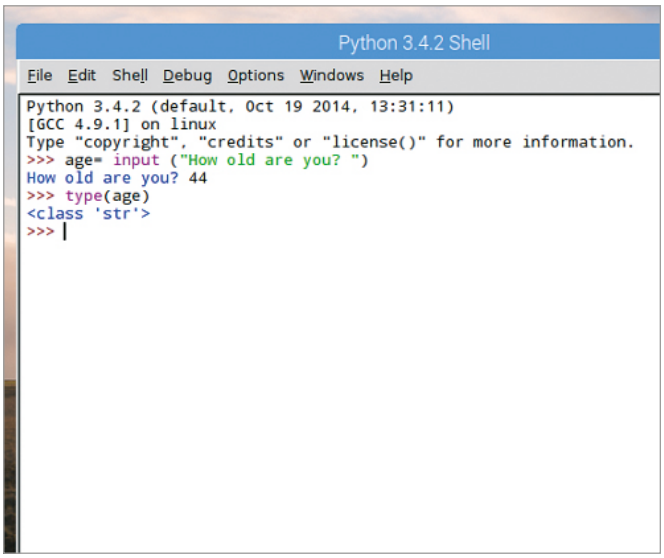
```
>>> print (name + age)
Traceback (most recent call last):
  File "<pyshell#18>", line 1, in <module>
    print (name + age)
TypeError: Can't convert 'int' object to str implicitly
>>> print (character + " is " + str(age) + " years old.")
David Hayward: Descended from Vikings is 44 years old.
>>> print (character, "is", age, "years old.")
David Hayward: Descended from Vikings is 44 years old.
>>>
>>>
```

STEP 7

Another example of TypeCasting is when you ask for input from the user, such as a name. for example, enter:

```
age= input ("How old are you? ")
```

All data stored from the Input command is stored as a string variable.

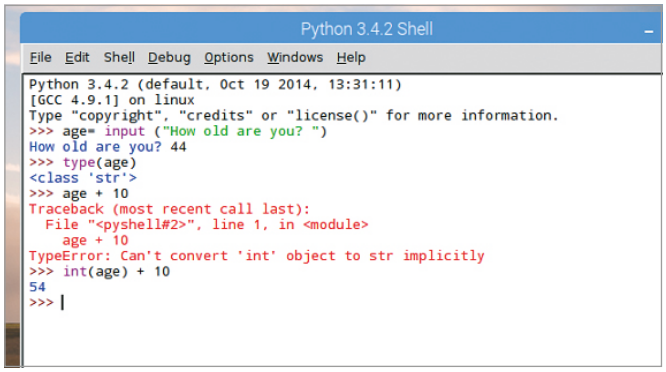


STEP 8

This presents a bit of a problem when you want to work with a number that's been inputted by the user, as age + 10 won't work due to being a string variable and an integer. Instead, you need to enter:

```
int(age) + 10
```

This will TypeCast the age string into an integer that can be worked with.

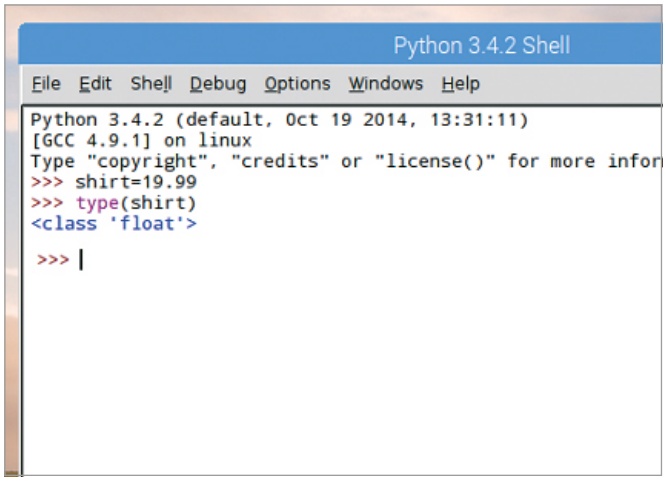


STEP 9

The use of TypeCasting is also important when dealing with floating point arithmetic; remember: numbers that have a decimal point in them. For example, enter:

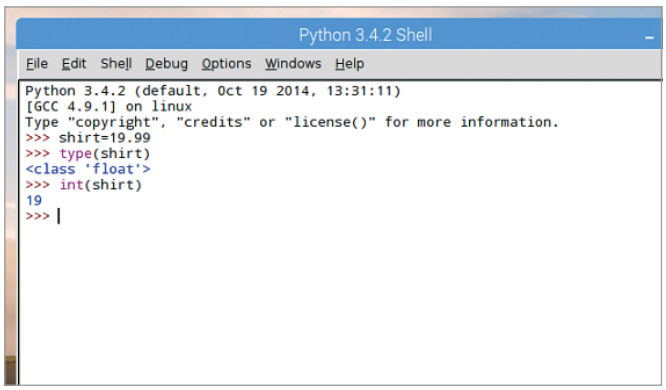
```
shirt=19.99
```

Now enter: `type(shirt)` and you'll see that Python has allocated the number as a 'float', because the value contains a decimal point.



STEP 10

When combining integers and floats Python usually converts the integer to a float, but should the reverse ever be applied it's worth remembering that Python doesn't return the exact value. When converting a float to an integer, Python will always round down to the nearest integer, called truncating; in our case instead of 19.99 it becomes 19.



User Input

We've seen some basic user interaction with the code from a few of the examples earlier, so now would be a good time to focus solely on how you would get information from the user then store and present it.

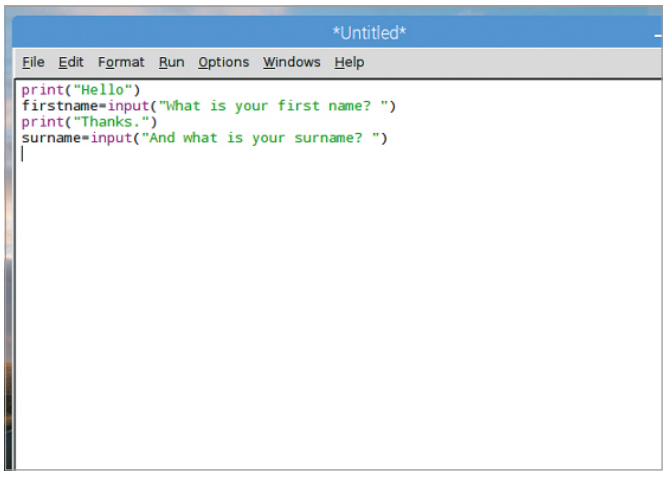
USER FRIENDLY

The type of input you want from the user will depend greatly on the type of program you're coding. For example, a game may ask for a character's name, whereas a database can ask for personal details.

STEP 1

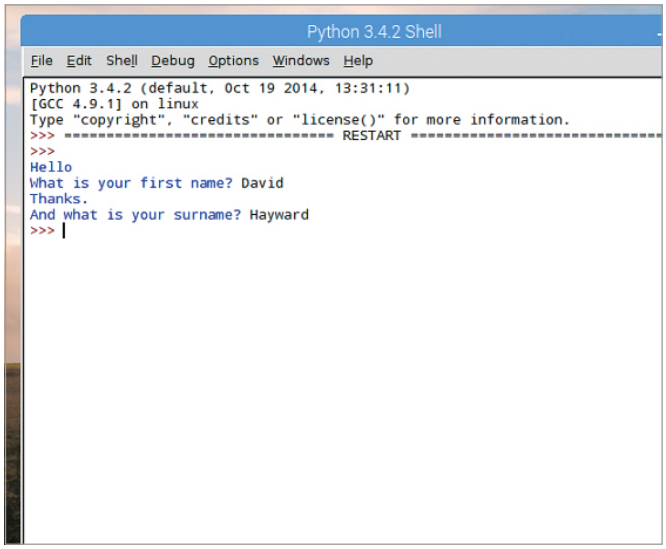
If it's not already, open the Python 3 IDLE Shell, and start a New File in the Editor. Let's begin with something really simple, enter:

```
print("Hello")
firstname=input("What is your first name? ")
print("Thanks.")
surname=input("And what is your surname? ")
```



STEP 2

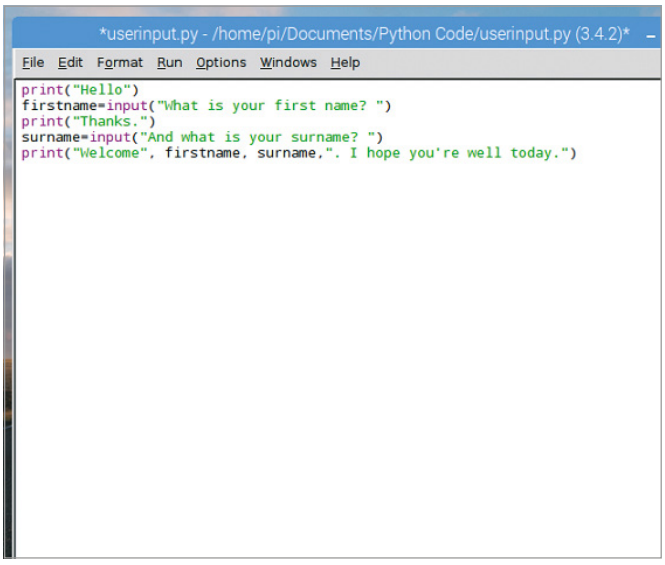
Save and execute the code, and as you already no doubt suspected, in the IDLE Shell the program will ask for your first name, storing it as the variable firstname, followed by your surname; also stored in its own variable (surname).



STEP 3

Now that we have the user's name stored in a couple of variables we can call them up whenever we want:

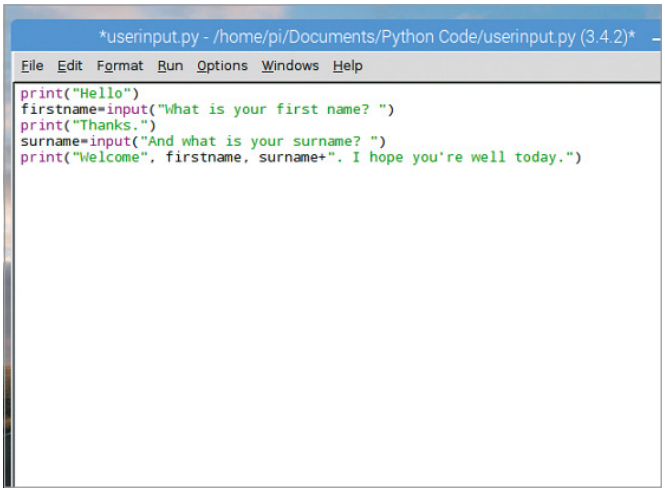
```
print("Welcome", firstname, surname, ". I hope you're well today.")
```



STEP 4

Run the code and you can see a slight issue, the full stop after the surname follows a blank space. To eliminate that we can add a plus sign instead of the comma in the code:

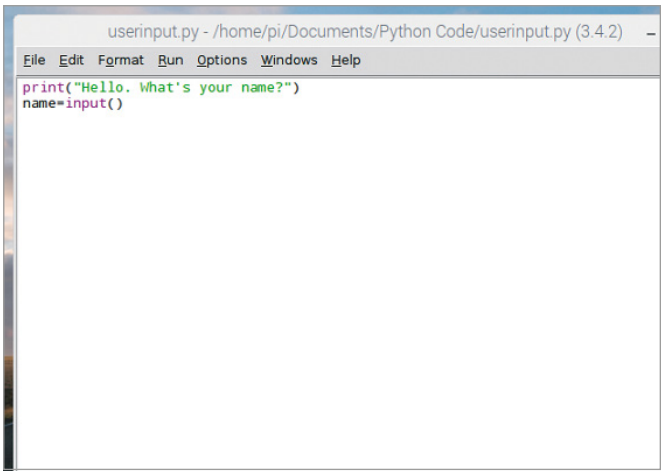
```
print("Welcome", firstname, surname+". I hope you're well today.")
```



STEP 5

You don't always have to include quoted text within the input command. For example, you can ask the user their name, and have the input in the line below:

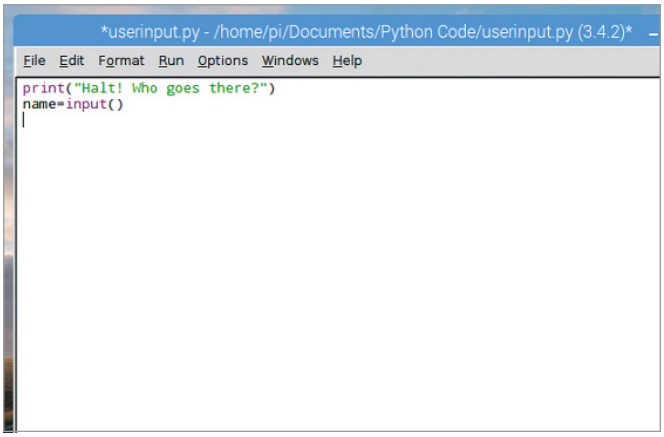
```
print("Hello. What's your name?")
name=input()
```



STEP 6

The code from the previous step is often regarded as being a little neater than having a lengthy amount of text in the input command, but it's not a rule that's set in stone, so do as you like in these situations. Expanding on the code, try this:

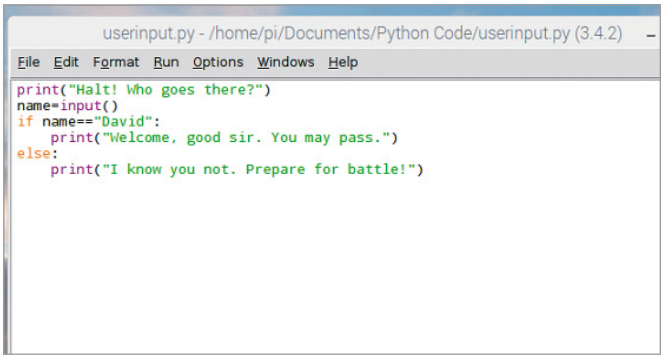
```
print("Halt! Who goes there?")
name=input()
```



STEP 7

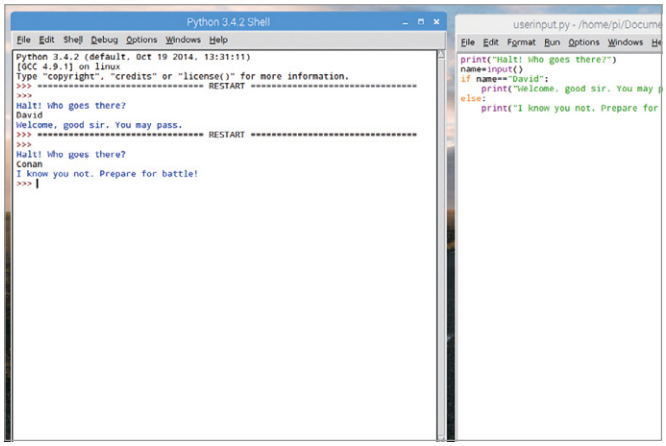
It's a good start to a text adventure game, perhaps? Now you can expand on it and use the raw input from the user to flesh out the game a little:

```
if name=="David":
  print("Welcome, good sir. You may pass.")
else:
  print("I know you not. Prepare for battle!")
```



STEP 8

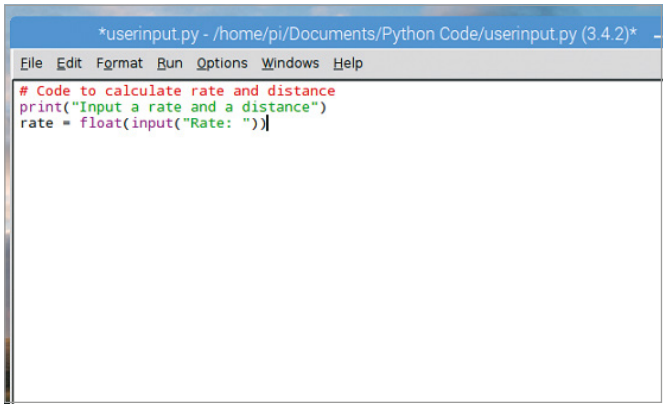
What you've created here is a condition, which we will cover soon. In short, we're using the input from the user and measuring it against a condition. So, if the user enters David as their name, the guard will allow them to pass unhindered. Else, if they enter a name other than David, the guard challenges them to a fight.



STEP 9

Just as you learned previously, any input from a user is automatically a string, so you need to apply a TypeCast in order to turn it into something else. This creates some interesting additions to the input command. For example:

```
# Code to calculate rate and distance
print("Input a rate and a distance")
rate = float(input("Rate: "))
```

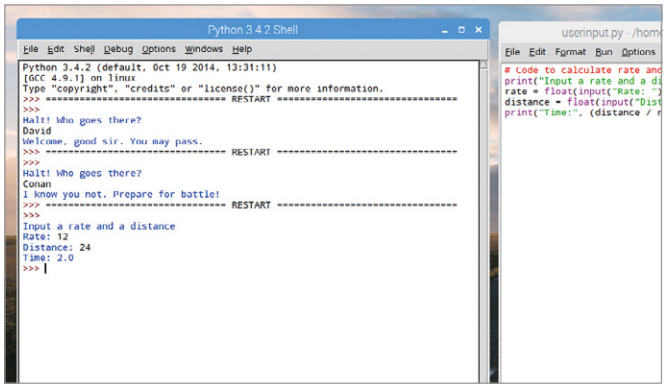


STEP 10

To finalise the rate and distance code, we can add:

```
distance = float(input("Distance: "))
print("Time:", (distance / rate))
```

Save and execute the code and enter some numbers. Using the float(input element, we've told Python that anything entered is a floating point number rather than a string.



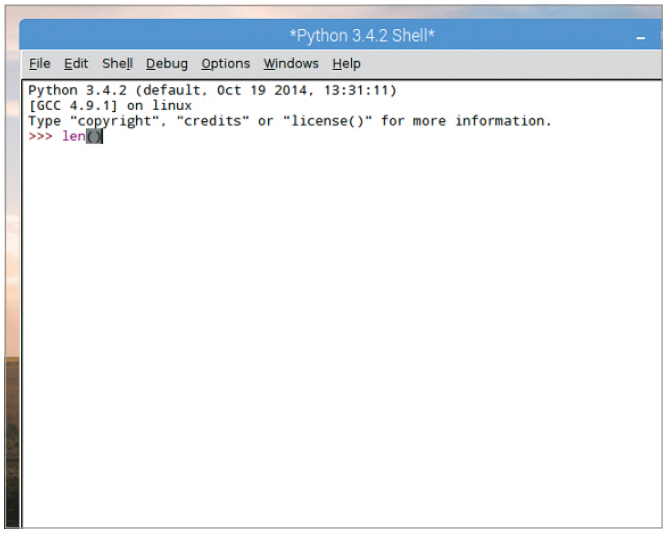
Creating Functions

Now that you've mastered the use of variables and user input, the next step is to tackle functions. You've already used a few functions, such as the print command but Python enables you to define your own functions.

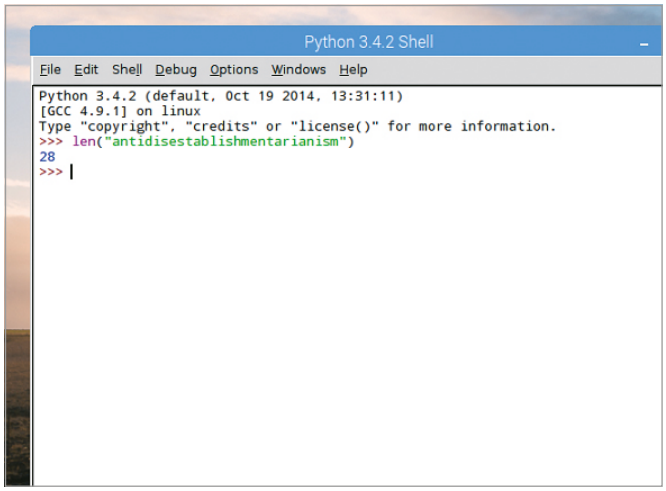
FUNKY FUNCTIONS

A function is a command that you enter into Python to do something. It's a little piece of self-contained code that takes data, works on it and then returns the result.

STEP 1 It's not just data that a function works on. They can do all manner of useful things in Python, such as sort data, change items from one format to another and check the length or type of items. Basically, a function is a short word that's followed by brackets. For example, **len()**, **list()** or **type()**.



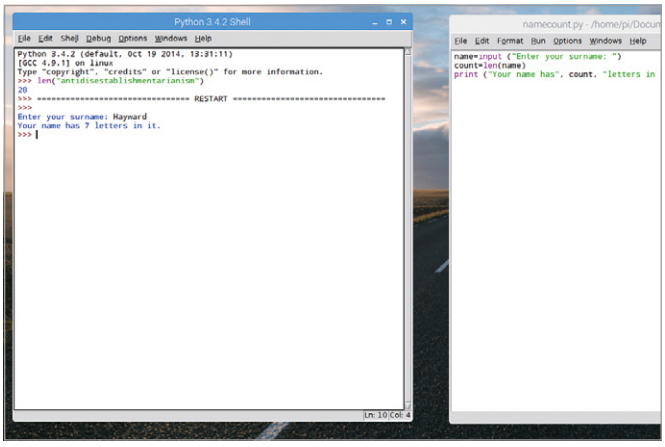
STEP 2 A function takes data, usually a variable, works on it depending on what the function is programmed to do and returns the end value. The data being worked on goes inside the brackets, so if you wanted to know how many letters are in the word antidisestablishmentarianism, then you'd enter: `len("antidisestablishmentarianism")` and the number 28 would return.



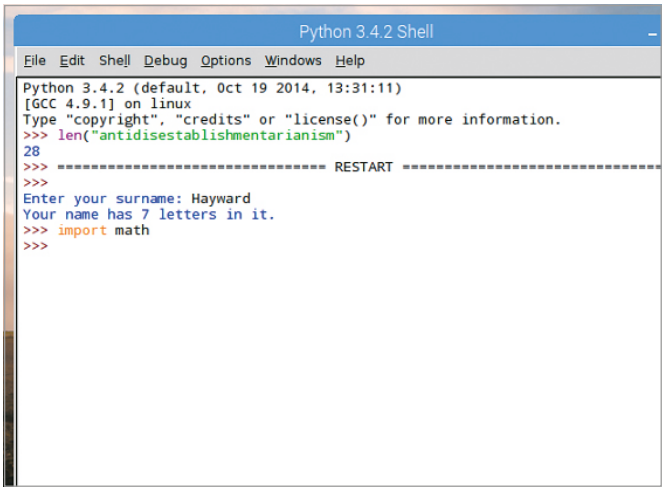
STEP 3 You can pass variables through functions in much the same manner. Let's assume you want the number of letters in a person's surname, you could use the following code (enter the text editor for this example):

```
name=input ("Enter your surname: ")
count=len(name)
print ("Your surname has", count, "letters in it.")
```

Press F5 and save the code to execute it.



STEP 4 Python has tens of functions built into it, far too many to get into in the limited space available here. However, to view the list of built-in functions available to Python 3, navigate to **www.docs.python.org/3/library/functions.html**. These are the predefined functions, but since users have created many more, they're not the only ones available.

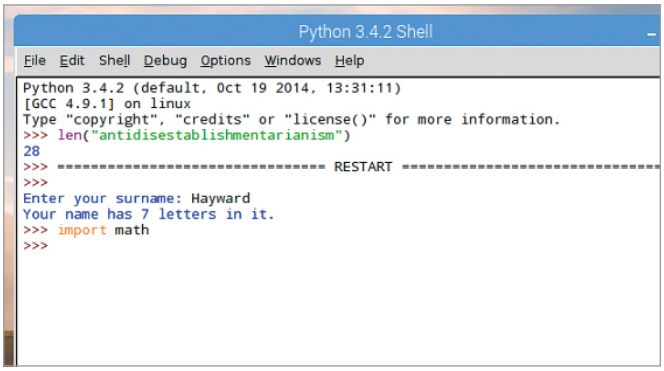


STEP 5

Additional functions can be added to Python through modules. Python has a vast range of modules available that can cover numerous programming duties. They add functions and can be imported as and when required. For example, to use advanced mathematics functions enter:

```
import math
```

Once entered, you have access to all the Math module functions.

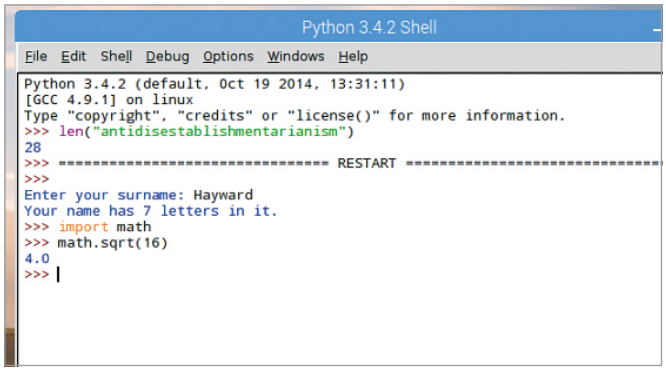


STEP 6

To use a function from a module enter the name of the module followed by a full stop, then the name of the function. For instance, using the Math module, since you've just imported it into Python, you can utilise the square root function. To do so, enter:

```
math.sqrt(16)
```

You can see that the code is presented as module.function(data).



FORGING FUNCTIONS

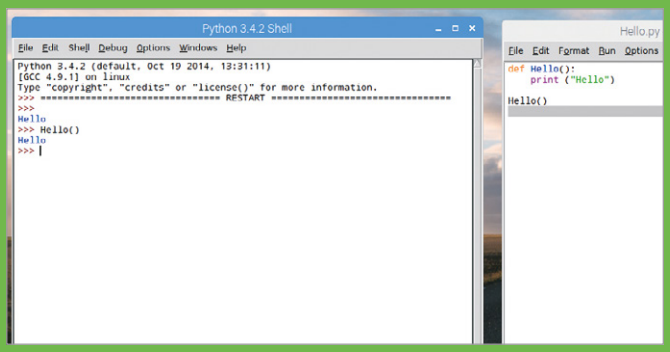
There are many different functions you can import, created by other Python programmers, and you will undoubtedly come across some excellent examples in the future; you can also create your own with the def command.

STEP 1

Choose File > New File to enter the editor, let's create a function called Hello, that greets a user. Enter:

```
def Hello():
   print ("Hello")
Hello()
```

Press F5 to save and run the script. You can see Hello in the Shell, type in Hello() and it returns the new function.

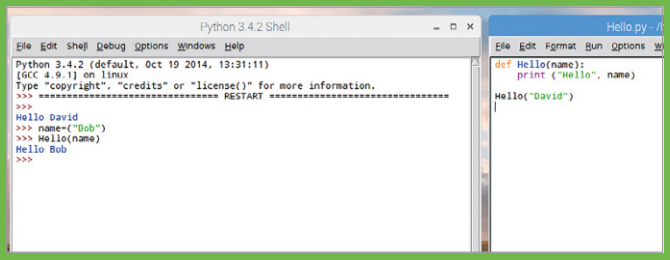


STEP 2

Let's now expand the function to accept a variable, the user's name for example. Edit your script to read:

```
def Hello(name):
   print ("Hello", name)
Hello("David")
```

This will now accept the variable name, otherwise it prints Hello David. In the Shell, enter: `name=("Bob"), then: Hello(name)`. Your function can now pass variables through it.

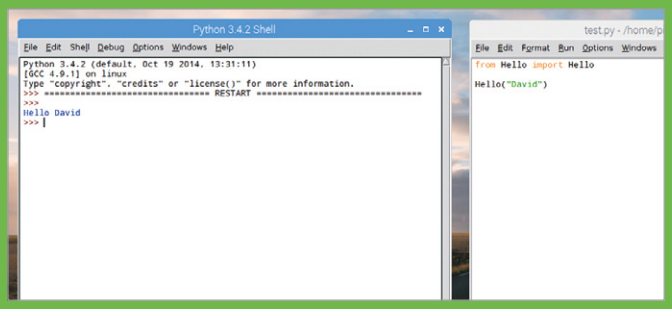


STEP 3

To modify it further, delete the Hello("David") line, the last line in the script and press Ctrl+S to save the new script. Close the Editor and create a new file (File > New File). Enter the following:

```
from Hello import Hello
Hello("David")
```

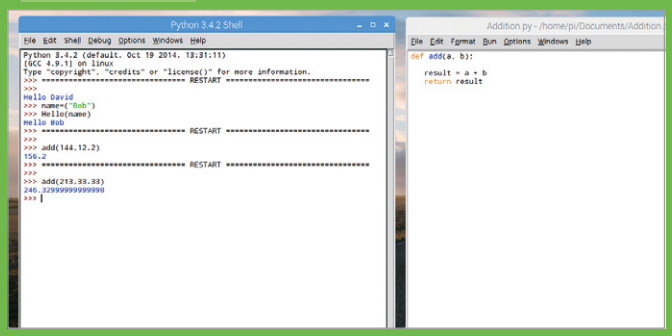
Press F5 to save and execute the code.



STEP 4

What you've just done is import the Hello function from the saved Hello.py program and then used it to say hello to David. This is how modules and functions work: you import the module then use the function. Try this one, and modify it for extra credit:

```
def add(a, b):
  result = a + b
  return result
```



Conditions and Loops

Conditions and loops are what makes a program interesting; they can be simple or rather complex. How you use them depends greatly on what the program is trying to achieve; they could be the number of lives left in a game or just displaying a countdown.

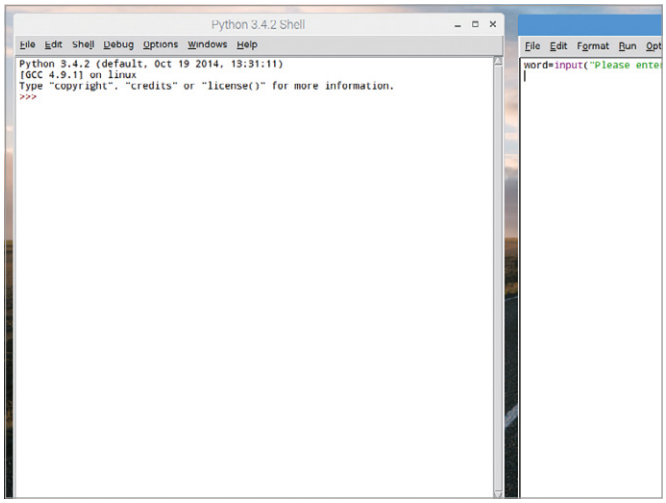
TRUE CONDITIONS

Keeping conditions simple to begin with makes learning to program a more enjoyable experience. Let's start then by checking if something is TRUE, then doing something else if it isn't.

STEP 1

Let's create a new Python program that will ask the user to input a word, then check it to see if it's a four-letter word or not. Start with File > New File, and begin with the input variable:

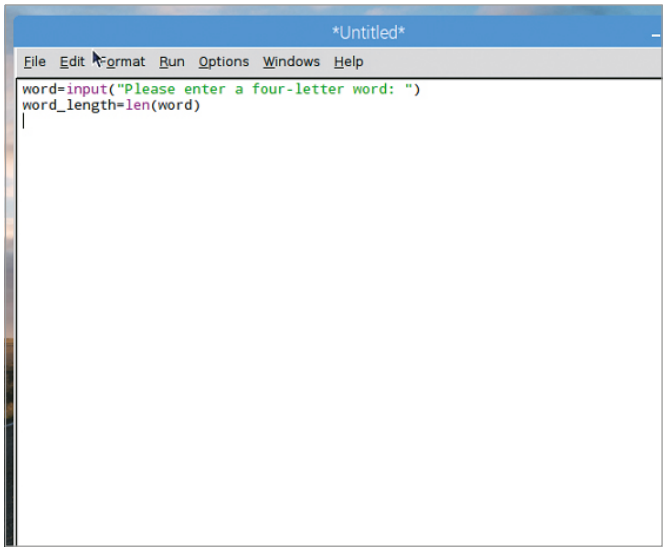
```
word=input("Please enter a four-letter word: ")
```



STEP 2

Now we can create a new variable, then use the len function and pass the word variable through it to get the total number of letters the user has just entered:

```
word=input("Please enter a four-letter word: ")
word_length=len(word)
```

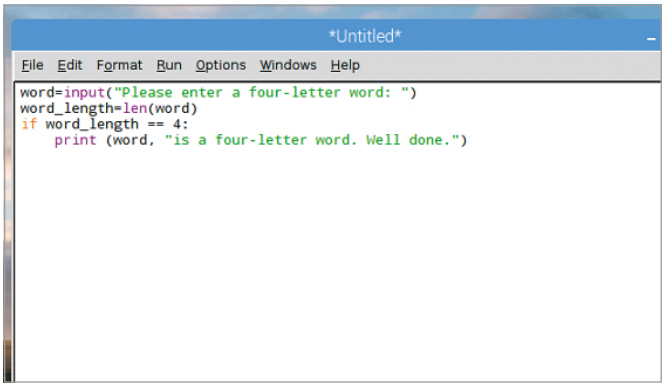


STEP 3

Now you can use an if statement to check if the word_length variable is equal to four and print a friendly conformation if it applies to the rule:

```
word=input("Please enter a four-letter word: ")
word_length=len(word)
if word_length == 4:
   print (word, "is a four-letter word. Well done.")
```

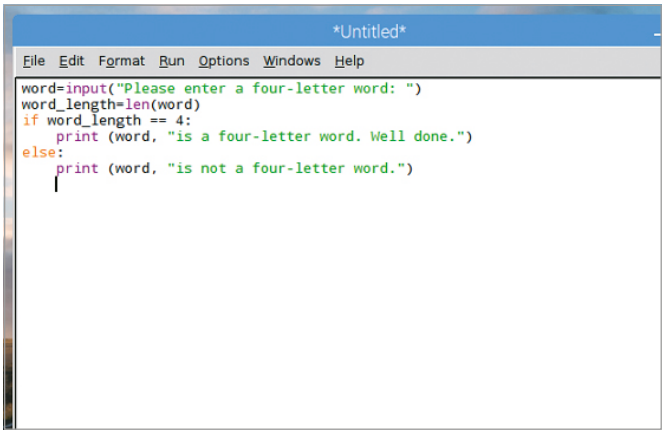
The double equal sign (==) means check if something is equal to something else.



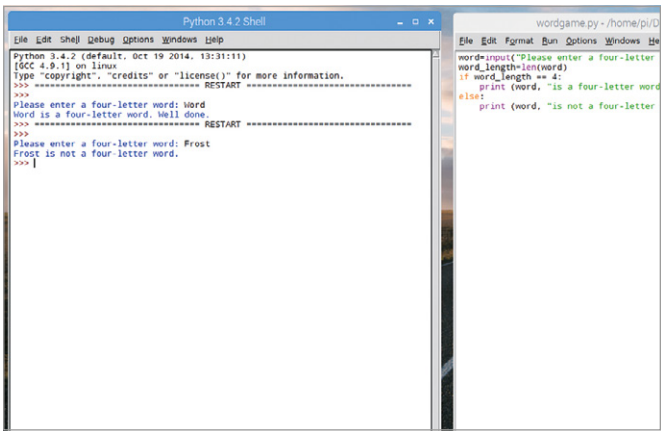
STEP 4

The colon at the end of IF tells Python that if this statement is true do everything after the colon that's indented. Next, move the cursor back to the beginning of the Editor:

```
word=input("Please enter a four-letter word: ")
word_length=len(word)
if word_length == 4:
  print (word, "is a four-letter word. Well done.")
else:
   print (word, "is not a four-letter word.")
```

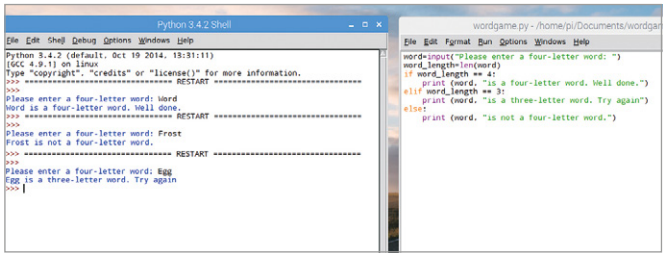


STEP 5 Press F5 and save the code to execute it. Enter a four-letter word in the Shell to begin with, you should have the returned message that it's the word is four letters. Now press F5 again and rerun the program but this time enter a five-letter word. The Shell will display that it's not a four-letter word.



STEP 6 Now expand the code to include another conditions. Eventually, it could become quite complex. We've added a condition for three-letter words:

```
word=input("Please enter a four-letter word: ")
word_length=len(word)
if word_length == 4:
  print (word, "is a four-letter word. Well done.")
elif word_length == 3:
  print (word, "is a three-letter word. Try again.")
else:
  print (word, "is not a four-letter word.")
```

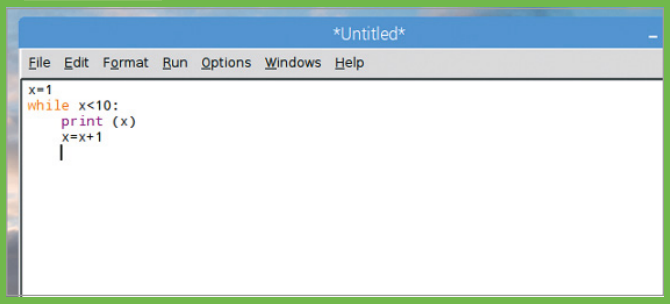


LOOPS

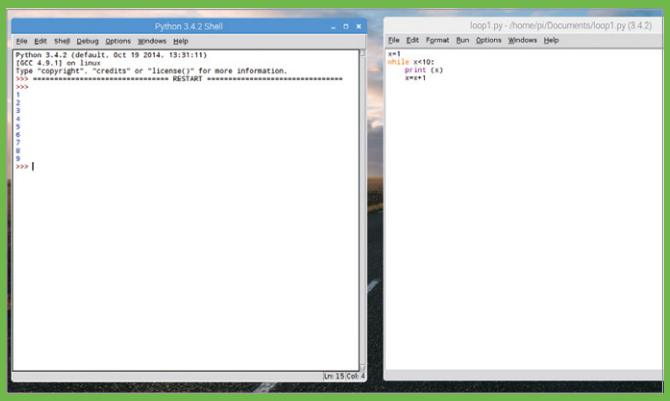
A loop looks quite similar to a condition but they are somewhat different in their operation. A loop will run through the same block of code a number of times, usually with the support of a condition.

STEP 1 Let's start with a simple While statement. Like IF, this will check to see if something is TRUE, then run the indented code:

```
x = 1
while x < 10:
  print (x)
  x = x + 1
```

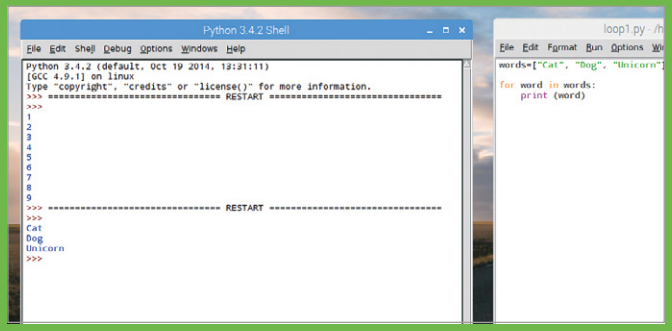


STEP 2 The difference between if and while is when while gets to the end of the indented code, it goes back and checks the statement is still true. In our example x is less than 10. With each loop it prints the current value of x, then adds one to that value. When x does eventually equal 10 it stops.



STEP 3 The For loop is another example. For is used to loop over a range of data, usually a list stored as variables inside square brackets. For example:

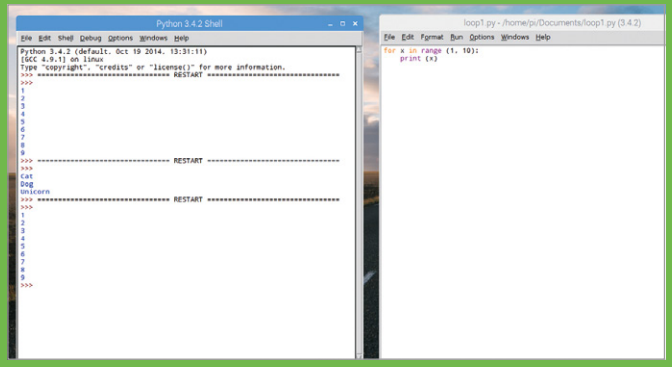
```
words=["Cat", "Dog", "Unicorn"]
for word in words:
  print (word)
```



STEP 4 The For loop can also be used in the countdown example by using the range function:

```
for x in range (1, 10):
  print (x)
```

The x=x+1 part isn't needed here because the range function creates a list between the first and last numbers used.



Python Modules

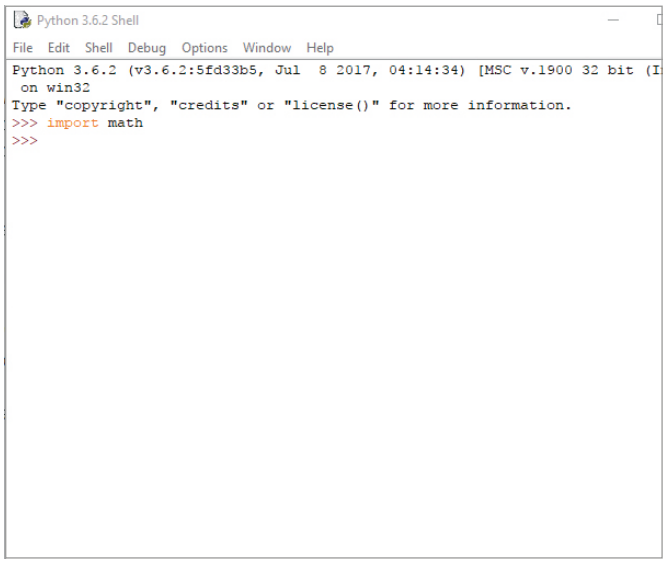
We've mentioned modules previously, (the Math module) but as modules are such a large part of getting the most from Python, it's worth dedicating a little more time to them. In this instance we're using the Windows version of Python 3.

MASTERING MODULES

Think of modules as an extension that's imported into your Python code to enhance and extend its capabilities. There are countless modules available and as we've seen, you can even make your own.

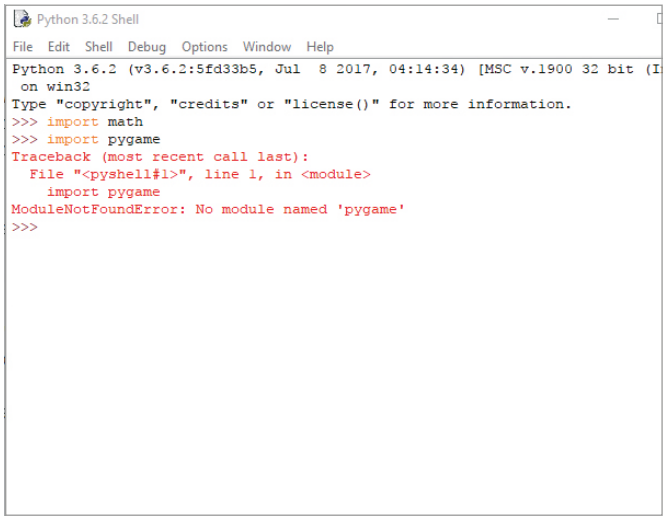
STEP 1

Although good, the built-in functions within Python are limited. The use of modules, however, allows us to make more sophisticated programs. As you are aware, modules are Python scripts that are imported, such as import math.



STEP 2

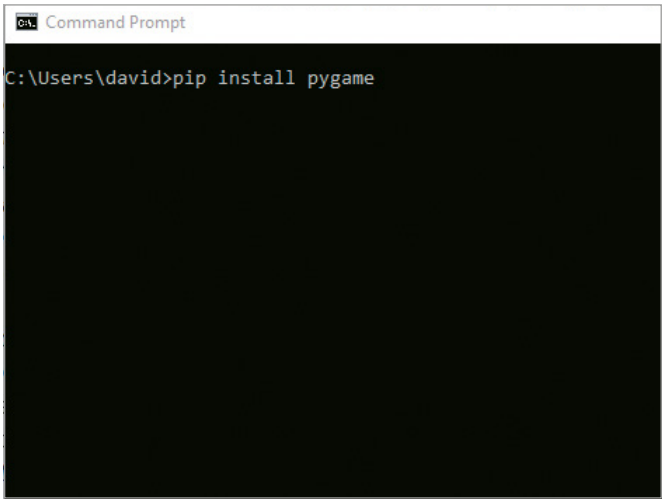
Some modules, especially on the Raspberry Pi, are included by default, the Math module being a prime example. Sadly, other modules aren't always available. A good example on non-Pi platforms is the Pygame module, which contains many functions to help create games. Try: **import pygame**.



STEP 3

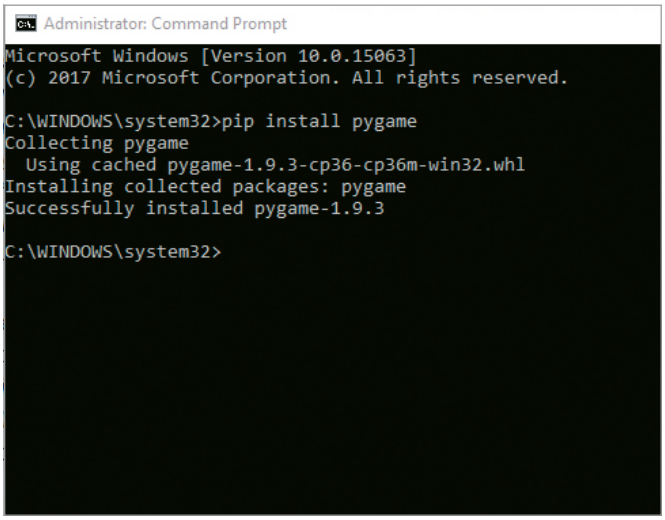
The result is an error in the IDLE Shell, as the Pygame module isn't recognised or installed in Python. To install a module we can use PIP (Pip Installs Packages). Close down the IDLE Shell and drop into a command prompt or Terminal session. At an elevated admin command prompt, enter:

```
pip install pygame
```



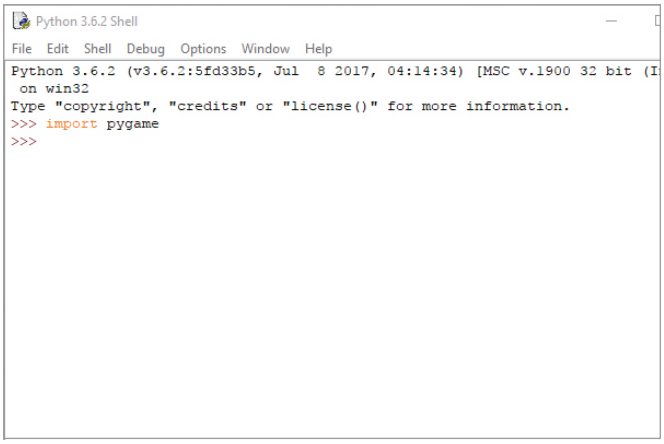
STEP 4

The PIP installation requires an elevated status due it installing components at different locations. Windows users can search for CMD via the Start button and right-click the result then click Run as Administrator. Linux and Mac users can use the Sudo command, with sudo pip install package.



STEP 5

Close the command prompt or Terminal and relaunch the IDLE Shell. When you now enter: `import pygame`, the module will be imported into the code without any problems. You'll find that most code downloaded or copied from the internet will contain a module, mainstream of unique, these are usually the source of errors in execution due to them being missing.



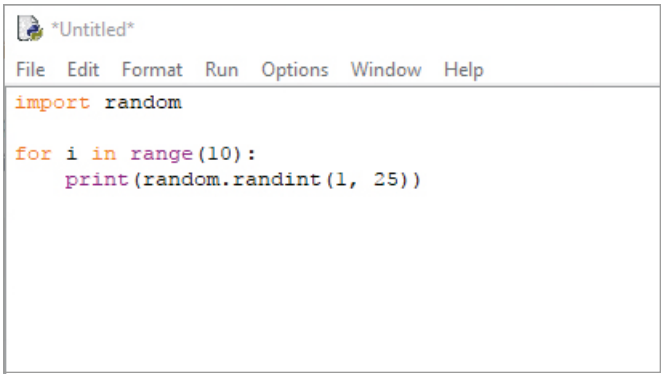
STEP 6

The modules contain the extra code needed to achieve a certain result within your own code, as we've previously experimented with. For example:

```
import random
```

Brings in the code from the random number generator module. You can then use this module to create something like:

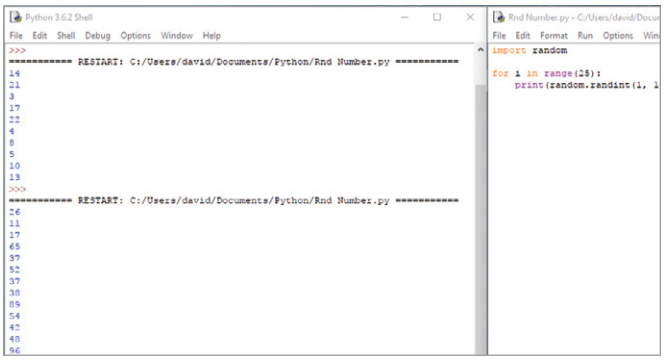
```
for i in range(10):
  print(random.randint(1, 25))
```



STEP 7

This code, when saved and executed, will display ten random numbers from 1 to 25. You can play around with the code to display more or less, and from a great or lesser range. For example:

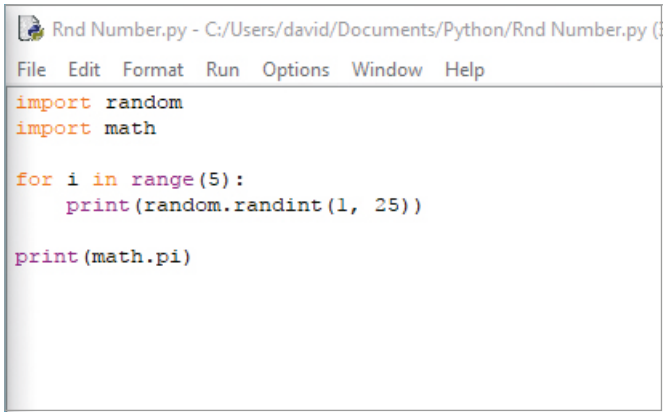
```
import random
for i in range(25):
  print(random.randint(1, 100))
```



STEP 8

Multiple modules can be imported within your code. To extend our example, use:

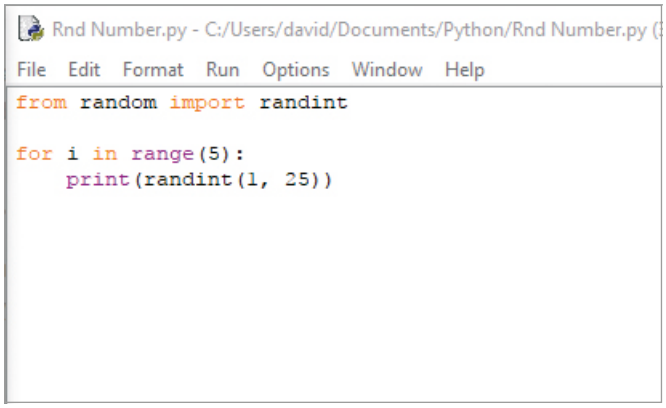
```
import random
import math
for I in range(5):
  print(random.randint(1, 25))
print(math.pi)
```



STEP 9

The result is a string of random numbers followed by the value of Pi as pulled from the Math module using the print(math.pi) function. You can also pull in certain functions from a module by using the from and import commands, such as:

```
from random import randint
for i in range(5):
  print(randint(1, 25))
```

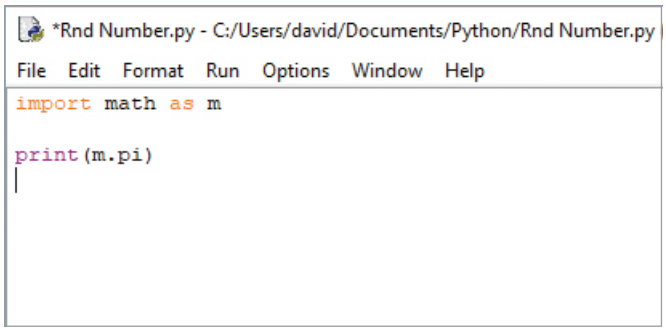


STEP 10

This helps create a more streamlined approach to programming. You can also use import module*, which will import everything defined within the named module. However, it's often regarded as a waste of resources but it works nonetheless. Finally, modules can be imported as aliases:

```
import math as m
print(m.pi)
```

Of course, adding comments helps to tell others what's going on.



Python Errors

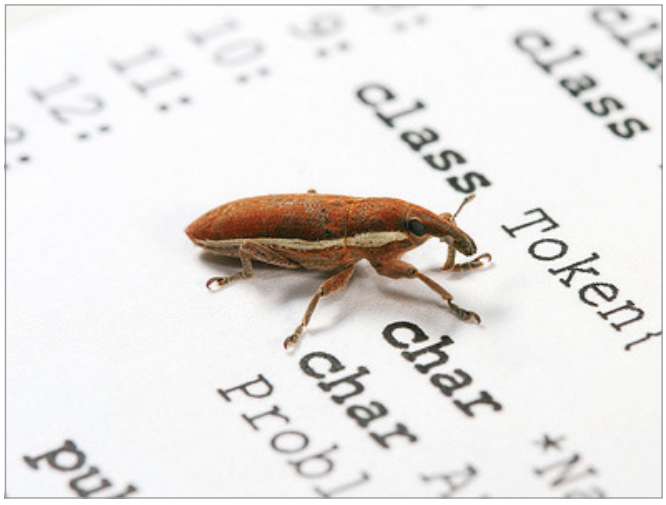
It goes without saying that you'll eventually come across an error in your code, where Python declares it's not able to continue due to something being missed out, wrong or simply unknown. Being able to identify these errors makes for a good programmer.

DEBUGGING

Errors in code are called bugs and are perfectly normal. They can often be easily rectified with a little patience. The important thing is to keep looking, experimenting and testing. Eventually your code will be bug free.

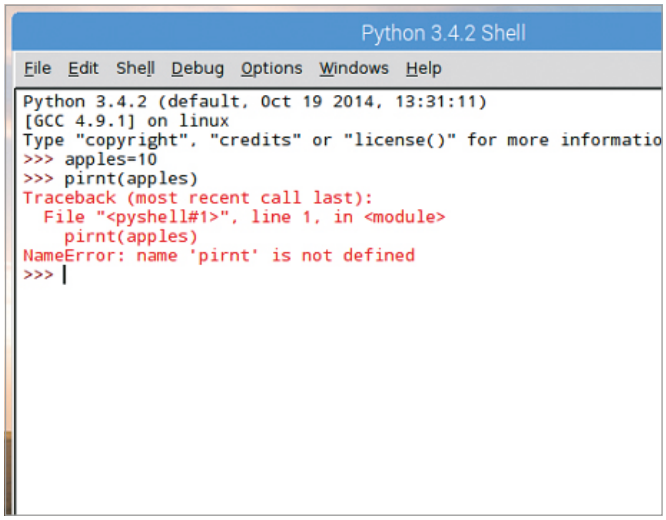
STEP 1

Code isn't as fluid as the written word, no matter how good the programming language is. Python is certainly easier than most languages but even it is prone to some annoying bugs. The most common are typos by the user and whilst easy to find in simple dozen-line code, imagine having to debug multi-thousand line code.



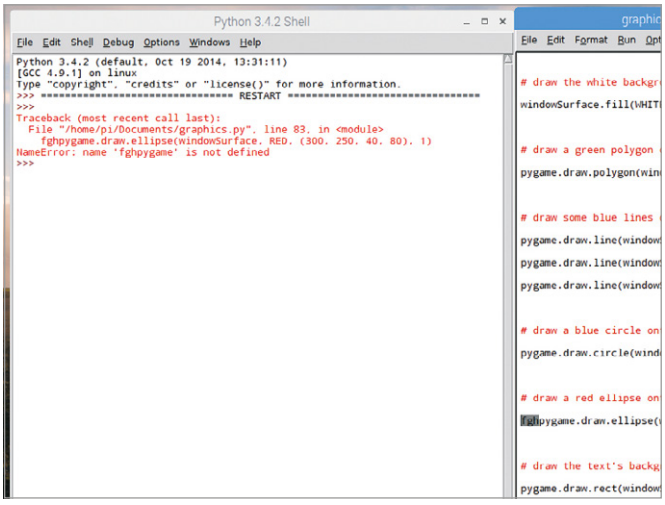
STEP 2

The most common of errors is the typo, as we've mentioned. The typos are often at the command level: mistyping the print command for example. However, they also occur when you have numerous variables, all of which have lengthy names. The best advice is to simply go through the code and check your spelling.



STEP 3

Thankfully Python is helpful when it comes to displaying error messages. When you receive an error, in red text from the IDLE Shell, it will define the error itself along with the line number where the error has occurred. Whilst in the IDLE Editor this is a little daunting for lots of code; text editors help by including line numbering.



STEP 4

Syntax errors are probably the second most common errors you'll come across as a programmer. Even if the spelling is correct, the actual command itself is wrong. In Python 3 this often occurs when Python 2 syntaxes are applied. The most annoying of these is the print function. In Python 3 we use **print("words")**, whereas Python2 uses **print "words"**.

STEP 5

Pesky brackets are also a nuisance in programming errors, especially when you have something like:

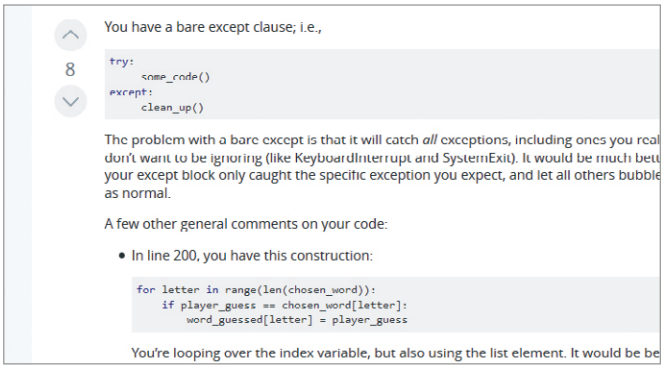
```
print(balanced_check(input()))
```

Remember that for every '(' there must be an equal number of ')'.

```
import sys

def balanced_check(data):
    stack = []
    characters = list(data)

    for character in characters:
        reference = {
            '(': ')',
            '{': '}',
            '[': ']'
        }
        if character in reference.keys():
            stack.append(character)

        elif character in reference.values() and len(stack) > 0:
            char = stack.pop()
            if reference.get(char) != character:
                return "NO"
        else:
            return "NO"

    if len(stack) == 0:
        return "YES"
```

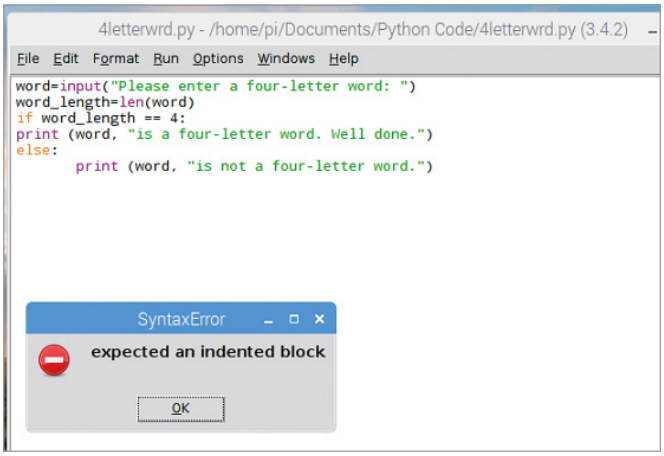
STEP 6

There are thousands of online Python resources, code snippets and lengthy discussions across forums on how best to achieve something. Whilst 99 per cent of it is good code, don't always be lured into copying and pasting random code into your editor. More often than not, it won't work and the worst part is that you haven't learnt anything.



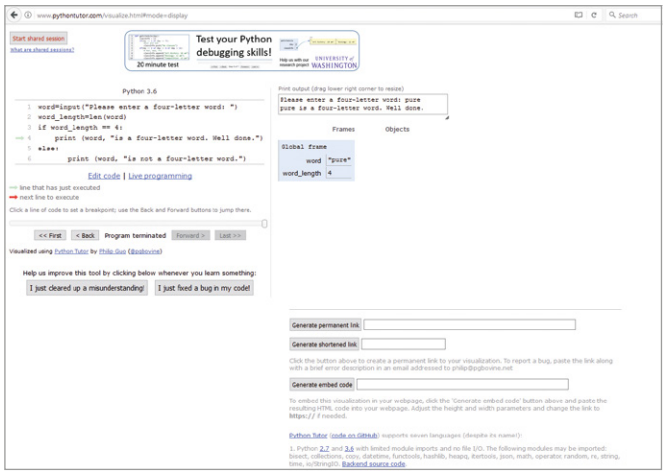
STEP 7

Indents are a nasty part of Python programming that a lot of beginners fall foul of. Recall the If loop from the Conditions and Loops section, where the colon means everything indented following the statement is to be executed as long as it's true? Missing the indent, or having too much of indent, will come back with an error.



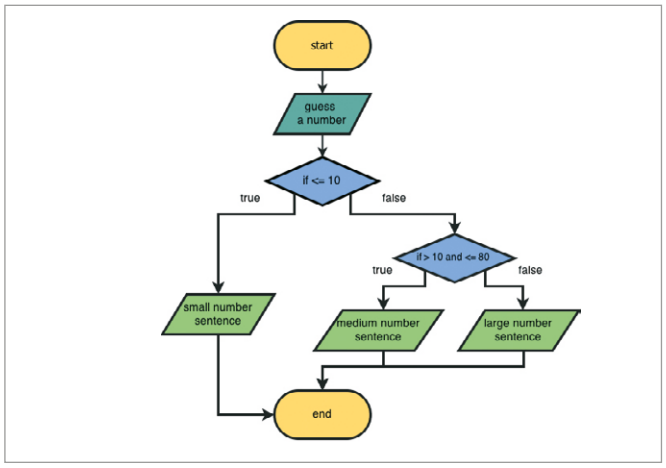
STEP 8

An excellent way to check your code step-by-step is to use Python Tutor's Visualise web page, found at **www.pythontutor.com/visualize.html#mode=edit**. Simply paste your code into the editor and click the Visualise Execution button to run the code line-by-line. This helps to clear bugs and any misunderstandings.



STEP 9

Planning makes for good code. Whilst a little old school, it's a good habit to plan what your code will do before sitting down to type it out. List the variables that will be used and the modules too; then write out a script for any user interaction or outputs.



STEP 10

Purely out of interest, the word debugging in computing terms comes from Admiral Grace Hopper, who back in the '40s was working on a monolithic Harvard Mark II electromechanical computer. According to legend Hopper found a moth stuck in a relay, thus stopping the system from working. Removal of the moth was hence called debugging.

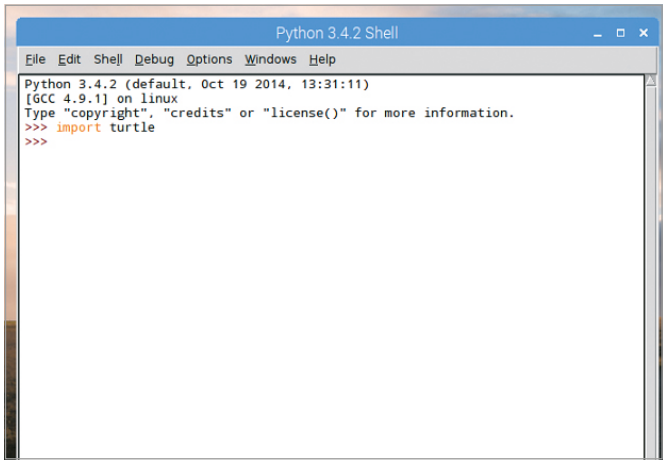
Python Graphics

While dealing with text on the screen, either as a game or in a program, is great, there will come a time when a bit of graphical representation wouldn't go amiss. Python 3 has numerous ways in which to include graphics and they're surprisingly powerful too.

GOING GRAPHICAL

You can draw simple graphics, lines, squares and so on, or you can use one of the many Python modules available, to bring out some spectacular effects.

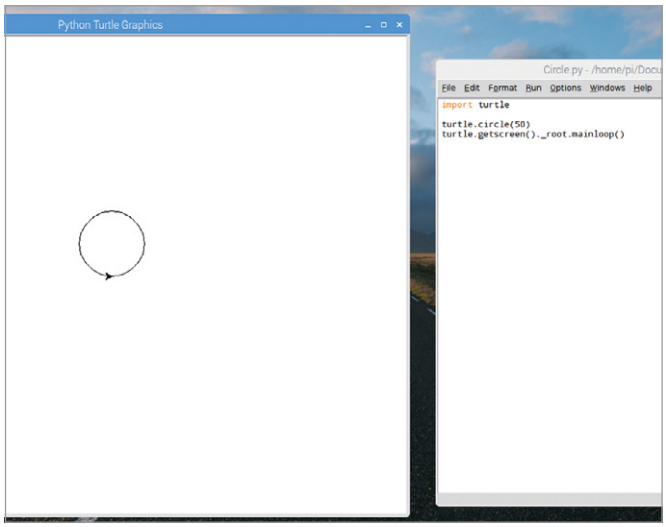
STEP 1 One of the best graphical modules to begin learning Python graphics is Turtle. The Turtle module is, as the name suggests, based on the turtle robots used in many schools, that can be programmed to draw something on a large piece of paper on the floor. The Turtle module can be imported with: `import turtle`.



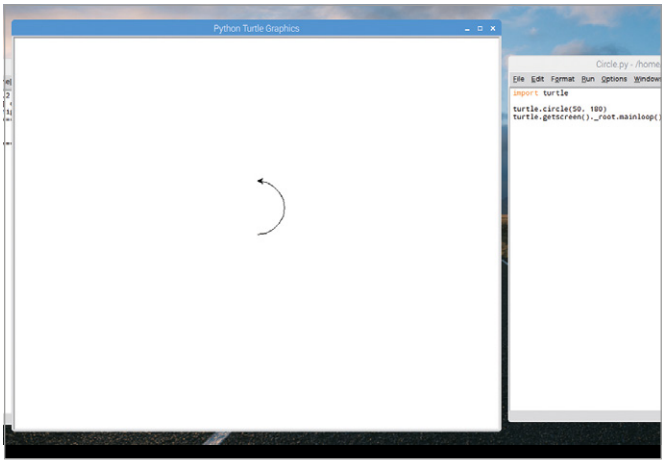
STEP 2 Let's begin by drawing a simple circle. Start a New File, then enter the following code:

```
import turtle
turtle.circle(50)
turtle.getscreen()._root.mainloop()
```

As usual press F5 to save the code and execute it. A new window will now open up and the 'Turtle' will draw a circle.



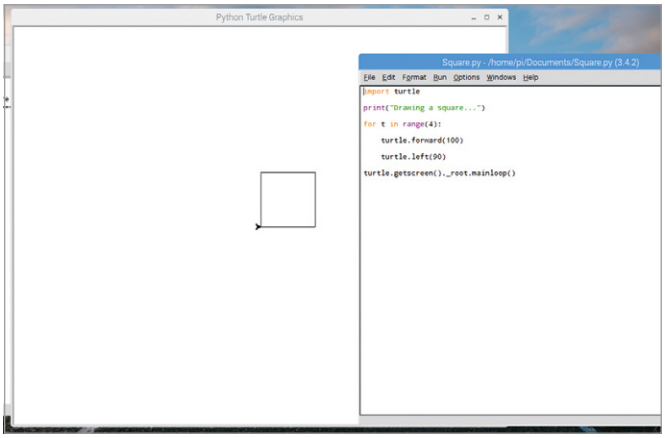
STEP 3 The command turtle.circle(50) is what draws the circle on the screen, with 50 being the size. You can play around with the sizes if you like, going up to 100, 150 and beyond; you can draw an arc by entering: `turtle.circle(50, 180)`, where the size is 50, but you're telling Python to only draw 180° of the circle.



STEP 4 The last part of the circle code tells Python to keep the window where the drawing is taking place to remain open, so the user can click to close it. Now, let's make a square:

```
import turtle
print("Drawing a square...")
for t in range(4):
    turtle.forward(100)
    turtle.left(90)
turtle.getscreen()._root.mainloop()
```

You can see that we've inserted a loop to draw the sides of the square.



STEP 5

You can add a new line to the square code to add some colour:

```
turtle.color("Red")
```

Then you can even change the character to an actual turtle by entering:

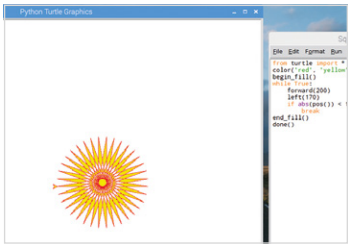
```
turtle.shape("turtle")
```

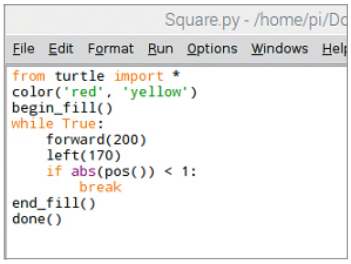
You can also use the command `turtle.begin_fill()`, and `turtle.end_fill()` to fill in the square with the chosen colours; red outline, yellow fill in this case.

STEP 6

You can see that the Turtle module can draw out some pretty good shapes and become a little more complex as you begin to master the way it works. Enter this example:

```
from turtle import *
color('red', 'yellow')
begin_fill()
while True:
  forward(200)
  left(170)
  if abs(pos()) < 1:
    break
end_fill()
done()
```

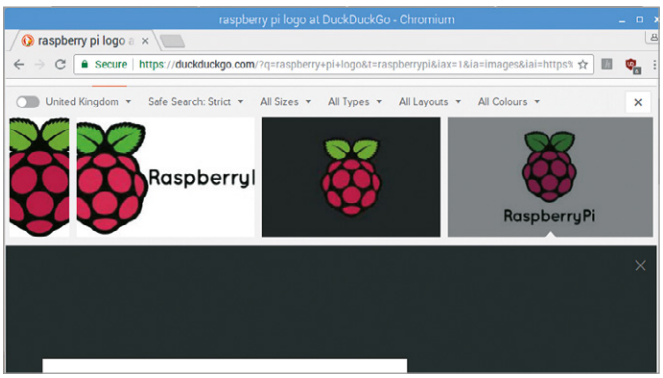




It's a different method, but very effective.

STEP 7

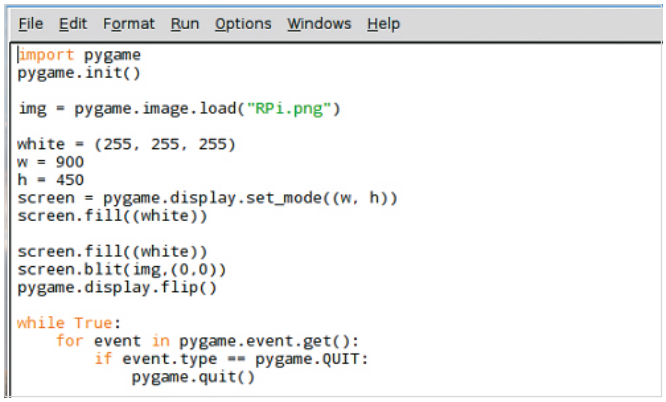
Another way in which you can display graphics is by using the Pygame module. There are numerous ways in which pygame can help you output graphics to the screen but for now let's look at displaying a predefined image. Start by opening a browser and finding an image, then save it to the folder where you save your Python code.



STEP 8

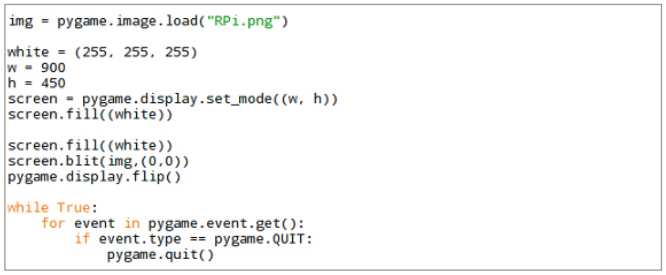
Now let's get the code by importing the Pygame module:

```
import pygame
pygame.init()
img = pygame.image.load("RPi.png")
white = (255, 255, 255)
w = 900
h = 450
screen = pygame.display.
set_mode((w, h))
screen.fill((white))
screen.fill((white))
screen.blit(img,(0,0))
pygame.display.flip()
while True:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      pygame.quit()
```



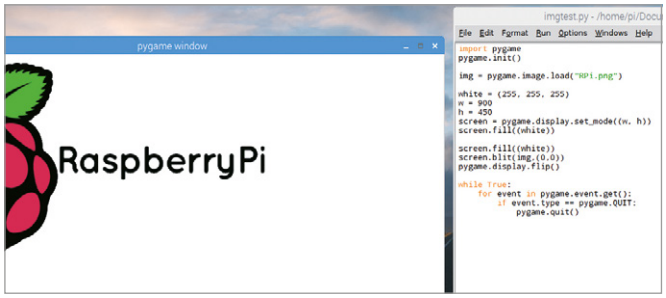
STEP 9

In the previous step you imported pygame, initiated the pygame engine and asked it to import our saved Raspberry Pi logo image, saved as RPi.png. Next you defined the background colour of the window to display the image and the window size as per the actual image dimensions. Finally you have a loop to close the window.



STEP 10

Press F5 to save and execute the code and your image will be displayed in a new window. Have a play around with the colours, sizes and so on and take time to look up the many functions within the Pygame module too.



Glossary of Terms

Just like most technology, Python contains many confusing words and acronyms. Here then, for your own sanity, is a handy glossary to help you keep on top of what's being said when the conversation turns to Python programming.

Argument
The detailed extra information used by Python to perform more detailed commands. Can also be used in the command prompt to specify a certain runtime event.

Block
Used to describe a section or sections of code that are grouped together.

Break
A command that can be used to exit a for or while loop. For example, if a key is pressed to quit the program, Break will exit the loop.

Class
A class provides a means of bundling data and functionality together. They are used to encapsulate variables and functions into a single entity.

Comments
A comment is a section of real world wording inserted by the programmer to help document what's going on in the code. They can be single line or multi-line and are defined by a # or "'.

Debian
A Linux-based distro or distribution that forms the Debian Project. This environment offers the user a friendly and stable GUI to interact with along with Terminal commands and other forms of system level administration.

Def
Used to define a function or method in Python.

Dictionaries
A dictionary in Python is a data structure that consists of key and value pairs.

Distro
Also Distribution, an operating system that uses the Linux Kernel as its core but offers something different in its presentation to the end user.

Editor
An individual program, or a part of the graphical version of Python, that enables the user to enter code ready for execution.

Exceptions
Used as a means of breaking from the normal flow of a code block in order to handle any potential errors or exceptional conditions within the program.

Expression
Essentially, Python code that produces a value of something.

Float
An immutable floating point number used in Python.

Function
Used in Python to define a sequence of statements that can be called or referenced at any time by the programmer.

GitHub
A web-based version control and collaboration portal designed for software developers to better manage source code.

Global Variable
A variable that is useable anywhere in the program.

Graphics
The use of visual interaction with a program, game or operating system. Designed to make it easier for the user to manage the program in question.

GUI
Graphical User Interface. The interface which most modern operating systems use to enable the user to interact with the core programming of the system. A friendly, easy to use graphical desktop environment.

High-Level Language
A programming language that's designed to be easy for people to read.

IDLE
Stands for Integrated Development Environment or Integrated Development and Learning Environment.

Immutable
Something that cannot be changed after it is created.

Import
Used in Python to include modules together with all the accompanying code, functions and variables they contain.

Indentation
Python uses indentation to delimit blocks of code. The indents are four spaces apart, and are often created automatically after a colon is used in the code.

Integer
A number data type that must be a whole number and not a decimal.

Interactive Shell
The Python Shell, which is displayed whenever you launch the graphical version of Python.

Kernel
The core of an operating system, which handles data processing, memory allocation, input and output, and processes information between the hardware and programs.

Linux
An open source operating system that's modelled on UNIX. Developed in 1991 by Finnish student Linus Torvalds.

Lists
A Python data type that contains collections of values, which can be of any type and can readily be modified.

Local Variable
A variable that's defined inside a function and is only useable inside that function.

Loop
A piece of code that repeats itself until a certain condition is met. Loops can encase the entire code or just sections of it.

Module
A Python file that contains various functions that can be used within another program to further extend the effectiveness of the code.

Operating System
Also OS. The program that's loaded into the computer after the initial boot sequence has completed. The OS manages all the other programs, graphical user interface (GUI), input and output and physical hardware interactions with the user.

Output
Data that is sent from the program to a screen, printer or other external peripheral.

PIP
Pip Installs Packages. A package management system used to install and manage modules and other software written in Python.

Print
A function used to display the output of something to the screen.

Prompt
The element of Python, or the Command Line, where the user enters their commands. In Python it's represented as >>> in the interactive shell.

Pygame
A Python module that's designed for writing games. It includes graphics and sound libraries and was first developed in October 2000.

Python
An awesome programming language that's easy to learn and use, whilst still being powerful enough to enjoy.

Random
A Python module that implements a pseudo-random character generator using the Mersenne Twister PRNG.

Range
A function that used to return a list of integers, defined by the arguments passed through it.

Root
The bottom level user account used by the system itself. Root is the overall system administrator and can go anywhere, and do anything, on the system.

Sets
Sets are a collection of unordered but unique data types.

Strings
Strings can store characters that can be modified. The contents of a string are alphanumerical and can be enclosed by either single or double quote marks.

Terminal
Also Console or Shell. The command line interface to the operating system, namely Linux, but also available in macOS. From there you can execute code and navigate the filesystem.

Tkinter
A Python module designed to interact with the graphical environment, specifically the tk-GUI (Tool Kit Graphical User Interface).

Try
A try block allows exceptions to be raised, so any errors can be caught and handled according to the programmer's instructions.

Tuples
An immutable Python data type that contains an ordered set of either letters or numbers.

UNIX
A multitasking, multiuser operating system designed in the '70s at the Bell Labs Research Centre. Written in C and assembly language

Variables
A data item that has been assigned a storage location in the computer's memory.

X
Also X11 or X-windows. The graphical desktop used in Linux-based systems, combining visual enhancements and tools to manage the core operating system.

Zen of Python
When you enter: `import this` into the IDLE, the Zen of Python is displayed.

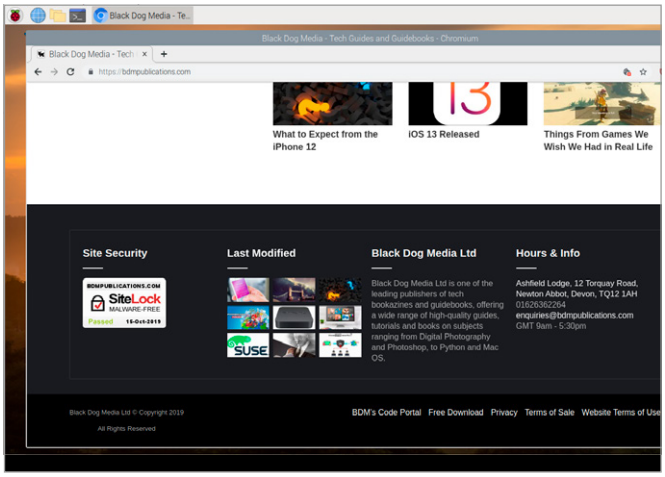
BDM's Code Repository

What better way to learn Python than from playing around with actual code? Back in the 80s, we typed in lines of BASIC or Assembler code from the monthly magazines of the time. It was laborious, but we hacked the code and made it our own; making us all-round better coders.

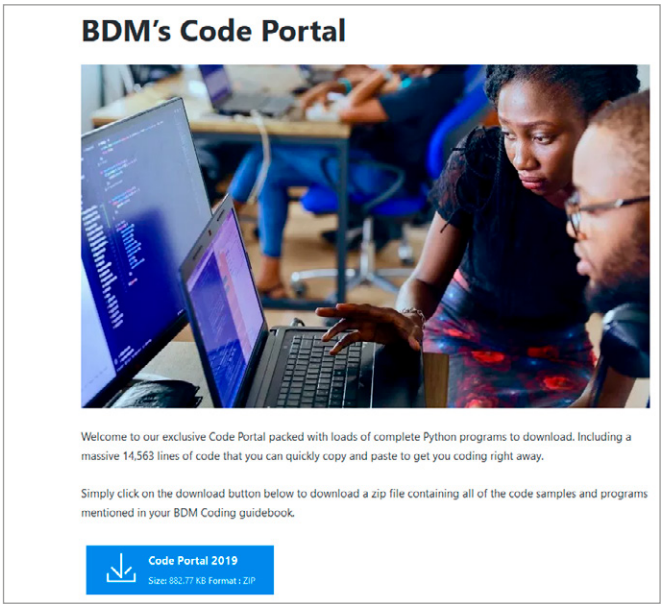
GET THE CODE

We've got a huge repository of code that's free for you to download, use, insert into your own Python routines and make even better. Here's how to get hold of it.

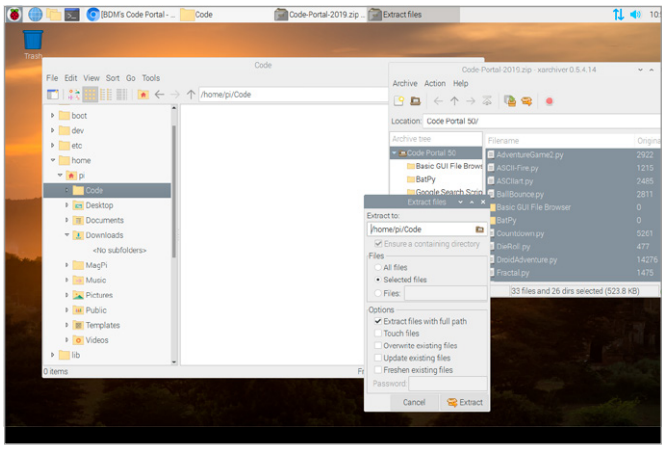
STEP 1 Begin by powering up your Pi and opening a web browser. Navigate to the BDM Site at **https://bdmpublications.com/**, and then scroll to the bottom of the main page and click on the BDM's Code Portal link.



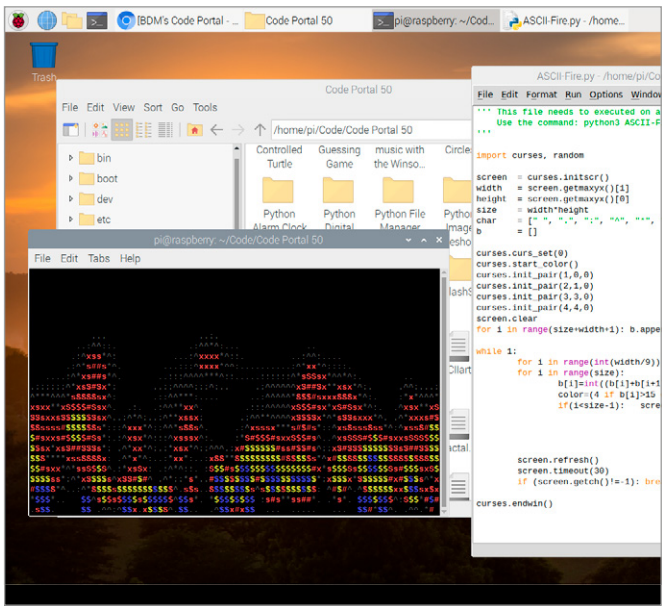
STEP 2 You will need to register a user account with BDM first, before being able to download the code. Simply click on the Join Us link, or if you've already registered, then click on the Login link. Once logged in, you can download the code by clicking on the blue Code Portal button at the bottom of the page.



STEP 3 Within the file structure of your Raspberry Pi, create a new folder called Code; it's in here that you can create your own code, as well as unpack the Code Portal contents. Open the downloaded, compressed Code Portal, select all the files within the archive, and extract them into the newly created Code folder.

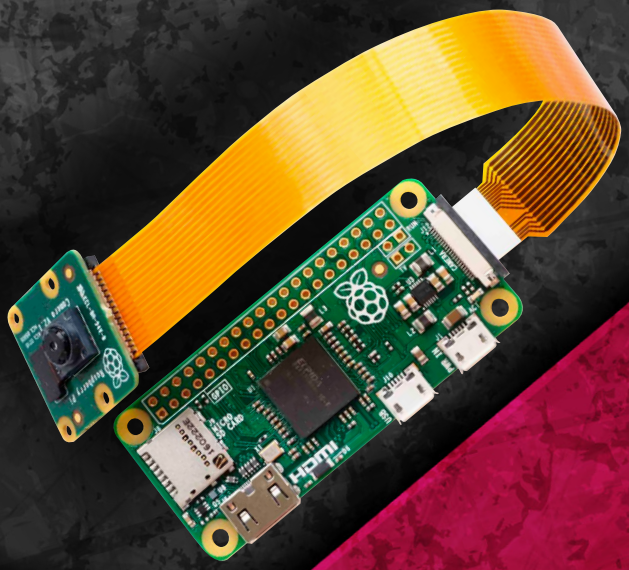


STEP 4 You are now able to load up any of the code within your favourite IDLE, which means you're free to use the code in your scripts and programs. Remember, if you've got some great code you want to share, then get in contact and tell us all about it.



Pi Projects: Ideas and Code

The Raspberry Pi is all about projects. For instance we've seen the Pi attached to a satellite, taking high-resolution images of the Earth while in orbit. Scientists have also used a Pi to monitor habitats in some of the most extreme places on the planet and enthusiasts the world over have used a Raspberry Pi as the driving force of their unique projects.



Creating a Loading Screen

If you're looking to add a little something extra to your Python code, then consider including a loading screen. The loading screen is a short introduction, or piece of art, that appears before the main part of your code.

LOAD""

Back in the 80s, in the 8-bit home computing era, loading screens were often used to display the cover of a game as it loaded from tape. The image would load itself in, usually one-line-at-a-time, then proceed to colour itself in while the loading raster bars danced around in the borders of the screen.

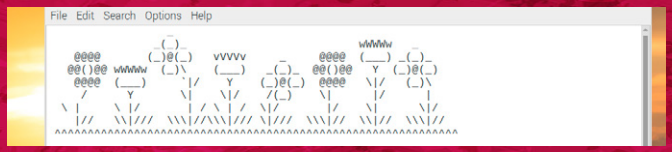
Loading screens were a part of the package and often the buy-in for the whole game as an experience. Some loading screens featured animations, or a countdown for time remaining as the game loads, while others even went so far as to include some kind of playable game. The point being: a loading screen is not just an artistic part of computing history, but an introduction to the program that's about to be run.

While these days loading screens may no longer be with us, in terms of modern gaming we can still include them in our own Python content. Either just for fun, or to add a little retro-themed spice to the mix.

SCREEN$

Creating a loading screen in Python is remarkably easy. You have several options to hand: you can create a tkinter window and display an image, followed by a brief pause, before starting your main code, or you could opt for a console-based ASCII art version that loads one-line-at-a-time. Let's look at the latter and see how it works.

First you'll need some ASCII art, you can look up plenty of examples online, or use an image to ASCII Art converter to transform any images you have to ASCII. When you have your ASCII art, drop it into a newly created folder inside a normal text file.

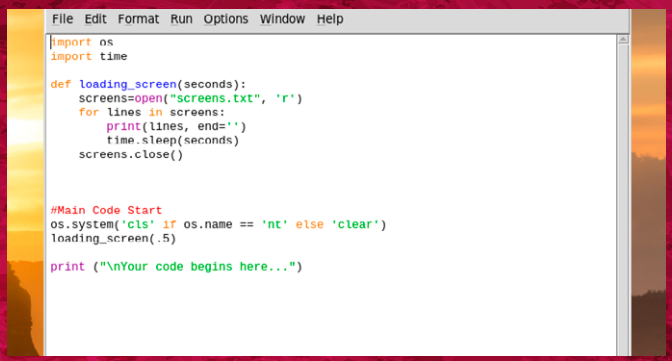


Save the file, call it screens.txt for now.

THE CODE

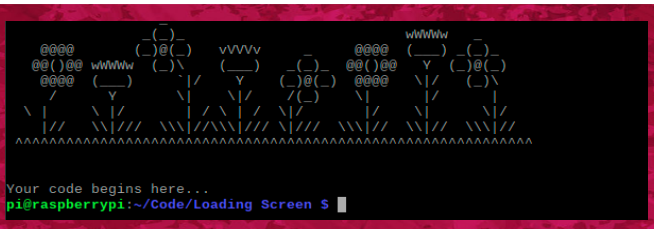
Launch Python and enter the following code to a New File:

```
import os
import time

def loading_screen(seconds):
    screens=open("screens.txt", 'r')
    for lines in screens:
        print(lines, end='')
        time.sleep(seconds)
    screens.close()

#Main Code Start
os.system('cls' if os.name == 'nt' else 'clear')
loading_screen(.5)

print ("\nYour code begins here...")
```



The code is quite simple: import the OS and Time modules and then create a Python function called loading_screen with a (seconds) option. Within the function, open the text file with the ASCII art as read-only and create a `For` loop that'll read the text file one-line-at-a-time. Next, print the lines – incidentally, the `lines, end=''` element will strip the newline from the text document, without it you'll end up with a double-line spaced display. Include the timing in seconds and close the text file buffer.

The final part of the code, `#Main Code Start`, is where you'll clear the screen (CLS for Windows, Clear for Linux (Raspberry Pi) and macOS) and call the function, together with the output number of seconds to wait for each line to be written to the screen – in this case, .5 of a second.



Save the code as screens.py, drop into a Command Prompt or Terminal and execute it. The screen will clear and your ASCII art inside the text file will load in line-by-line, creating a loading screen effect.

LOADING...

Another favourite introduction screen is that of a simple loading animation, where the word loading is displayed, followed by some characters, and a percentage of the program loaded. While it may not take long for your Python code to load, the effect can be interesting.

Create a New File in Python and enter the following code:

```
import os
import time

def loading_bar(seconds):
  for loading in range(0,seconds+1):
    percent = (loading * 100) // seconds
    print("\n")
    print("Loading...")
    print("<" + ("-" * loading) + (" " *
(seconds-loading)) + "> " + str(percent) + "%")
    print("\n")
    time.sleep(1)
    os.system('cls' if os.name == 'nt' else
'clear')

#Main Code Start
loading_bar(10)

print ("\nYour code begins here...")
```

The code works in much the same way as the previous, except, instead of reading from a text file, it's running through a For loop that prints Loading... followed by an animation of sorts, along with a percentage counter; clearing the screen every second and displaying the new results. It's simple, yes, but quite effective in its design.

COMBINING THE TWO

How about combining the two elements we've looked at? Let's begin with a Loading... progress bar, followed by the loading screen. After that, you can include your own code and continue your program. Here's the code:

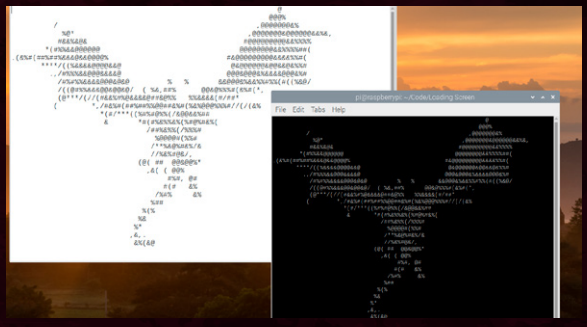
```
import os
import time

def loading_bar(seconds):
  for loading in range(0,seconds+1):
    percent = (loading * 100) // seconds
    print("\n")
    print("Loading...")
    print("<" + ("-" * loading) + (" " *
(seconds-loading)) + "> " + str(percent) + "%")
    print("\n")
    time.sleep(1)
    os.system('cls' if os.name == 'nt' else
'clear')

def loading_screen(seconds):
    screens=open("screens.txt", 'r')
    for lines in screens:
        print(lines, end='')
        time.sleep(seconds)
    screens.close()

#Main Code Start
loading_bar(10)
os.system('cls' if os.name == 'nt' else 'clear')
loading_screen(.5)

print ("\nYour code begins here...")
```

You can, of course, pull those functions up wherever and whenever you want in your code and they'll display, as they should, at the beginning. Remember to have the ASCII text file in the same folder as the Python code, or, at the `screens=open("screens.txt", 'r')` part of the code, point to where the text file is located.

ADVENTURE TIME

A good example of using loading screen, ASCII art text images is when coding a text adventure. Once you've established your story, created the characters, events and so on, you could easily incorporate some excellently designed ASCII art to your game.

Imagine coming across a dragon, in game, and displaying its representation as ASCII. You can then load up the image lines, one-by-one, and continue with the rest of the adventure code. It's certainly worth having a play around with and it'll definitely add a little something else extra.



Tracking the ISS with Python

Of the many, amazing human achievements over the past decade or so, the International Space Station tops the bill. This incredible collaboration between nations sees vital experiments carried out in space, as well as observations of our own planet.

TO BOLDLY GO...

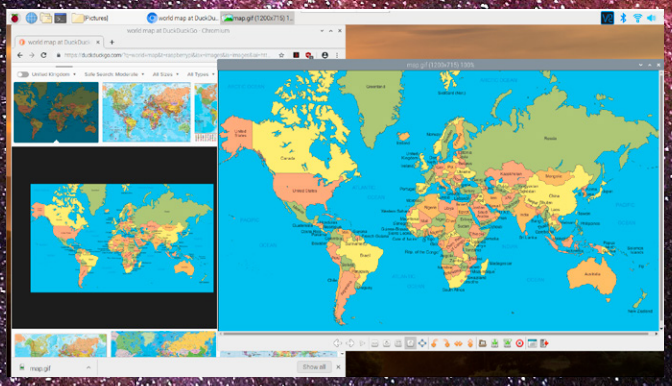
Indeed, the ISS is something most of us consider as a worthy example of what can happen when we work together. NASA, among other agencies, uses a wealth of Python code onboard the ISS to help automate routines, alongside its function as an in-between link to translate code from one language to another, and then into a human-readable format. If you're considering a career in space, then learning Python is a must-have skill.

While we're not able to fill you in on all the details regarding the code the ISS utilises, we can look at a fun Python script that will track the ISS, display the number of astronauts on board, and update the station's current latitude and longitude every five seconds, while also displaying your current latitude and longitude.

Displaying all that information in a single screen can become a little cluttered, so in this instance we're going to look at spreading all those details over three screens: a text document window -displaying the astronauts and your current latitude and longitude, a command line (or Terminal window) - displaying the continually updating latitude and longitude of the ISS as it orbits Earth, and a final, large window displaying a map of the world, together with an icon representing the ISS, that's updated as it or[illegible]nterested? Read

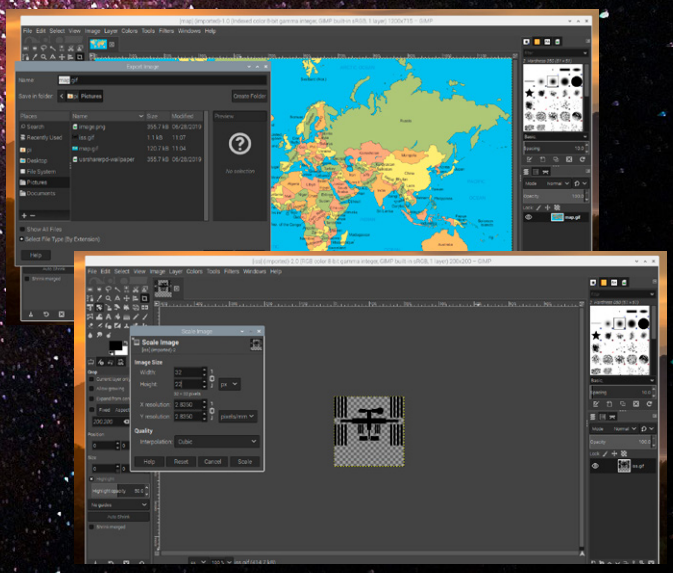
THE GRAPHICS

Firstly, we need to get hold of a map of the world and an image of the ISS, to use as an icon that will be updated according to the position of the space station as it travels over the surface. A quick Google of World Map will help you out here. Look for one that's reasonably large, the one we used for this example was 1280 x 700, and one that has the names of the countries – if you're using this with young people, to help with putting shapes of countries to names.



Next, look for an ISS icon. As this is going to be a graphical representation of the location of the ISS, we need the image to be reasonably small so it doesn't drown out the locations on the map, but also prominent enough to see when the map is loaded. We opted for an image that's 32 x 22 pixels in size. Don't worry too much if you're not able to find one that small, you can always resize it in an image-editing app such as GIMP.

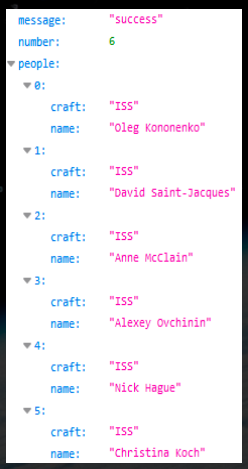
As we're going to be using Turtle, a component of tkinter, the downloaded images will need to be converted to GIF, since this is the default and recommended image format. You can easily look up a converter online, but using GIMP, which is cross-platform and therefore works on both the Raspberry Pi and a Windows PC, will suffice. Simply load the image up in your image editor app and choose Save As, call them map and iss respectively, and click GIF as the image format. Remember to also resize the ISS image before saving it as a GIF.



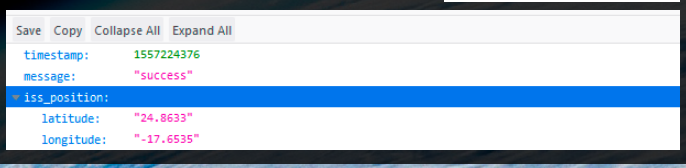
THE CODE

The code we're using here utilises an open source API (Application Programming Interface) to retrieve real-time data online regarding the status of the ISS. An API enables applications to communicate with one another by providing the raw data that a programmer can pull out and interact with in their own code. In this case, the API in question is a web-based collection of raw data that's stored in a JSON (JavaScript Object Notation) format – an accessible and easy to use data-interchange interface.

In order for Python to interact with the JSON provided data, it needs to use the urllib.request and json modules. We're also going to be using a new module called geocoder, which will pull up a users' current latitude and longitude based on their IP address. The two JSON APIs can be located at: **http://api.open-notify.org/astros.json**, and, **http://api.open-notify.org/iss-now.json**. One of which contains the data regarding the astronauts onboard the ISS, and the other contains the data regarding the current location of the ISS in longitude and latitude.



```
message:    "success"
number:     6
people:
  0:
    craft:  "ISS"
    name:   "Oleg Kononenko"
  1:
    craft:  "ISS"
    name:   "David Saint-Jacques"
  2:
    craft:  "ISS"
    name:   "Anne McClain"
  3:
    craft:  "ISS"
    name:   "Alexey Ovchinin"
  4:
    craft:  "ISS"
    name:   "Nick Hague"
  5:
    craft:  "ISS"
    name:   "Christina Koch"
```



```
Save  Copy  Collapse All  Expand All
timestamp:    1557224376
message:      "success"
iss_position:
  latitude:   "24.8633"
  longitude:  "-17.6535"
```

Let's begin by breaking the code into bite-sized chunks:

```
import json, turtle, urllib.request, time, webbrowser
import geocoder # need to pip install geocoder for your lat/long to work.
```

First, we need to import the modules used in the code: json, turtle, urlib.request, time, and webbrowser. The json and urlib.request modules deal with the data being pulled from the APIs, turtle will display the graphics, and time you already know. The Webbrowser module will open text files in the default browser or default text reader application. The geocoder module is a new element and you will need to install it with: `pip install geocoder` (or try `pip3 install geocoder`, if the pip install command still won't load the module). Geocoder can retrieve a users' location based on their IP address, as each ISP will have a geo-specific IP.

```
#Retrieve the names of all the astronauts currently on board the ISS, and own lat/long - write to a file and display
url = "http://api.open-notify.org/astros.json"
response = urllib.request.urlopen(url)
result = json.loads(response.read())
a=open("iss.txt","w")
a.write("There are currently " + str(result["number"]) + " astronauts on the ISS:\n\n")

people = result["people"]

for p in people:
  a.write(p["name"] + " - on board" + "\n")

g=geocoder.ip('me') # need to pip install geocoder, and import as in the headers above
a.write("\nYour current Lat/Long is: " + str(g.latlng)) # prints your current lat/long in the text file.
a.close()
webbrowser.open("iss.txt")
```

This section will use the json and urllib.request modules to pull the data from the API that contains the names of the astronauts onboard the ISS. It then creates a new text file called iss.txt, where it'll write 'There are currently X astronauts on the ISS…' and list them by name. The second part of this section will then use the geocoder module to retrieve your current latitude and longitudes, based on your IP address, and also write that information to the end of the text file that contains the names of the astronauts. The last line, webbrowser.open("iss.txt") will use the webbroser module to open and display the newly written text file in either the system's default text file reading app or the default web browser, either will work just fine.

```
#Setup world map in Turtle
screen = turtle.Screen()
screen.setup(1280, 720)
screen.setworldcoordinates(-180, -90, 180, 90)
#Load the world map image
screen.bgpic("map.gif")

screen.register_shape("iss.gif")
iss = turtle.Turtle()
iss.shape("iss.gif")
iss.setheading(45)
iss.penup()
```

This section of the code sets up the graphical window containing the world map and the ISS icon. To begin, we set up the turtle screen, using the resolution of the world map image we downloaded at the start of this tutorial (1280 x 720). The screen.setworldcoordinates syntax will mark the boundaries of the screen, creating the x and y coordinates of the four corners of the canvas, so that the ISS icon can freely travel across the map of the world and wrap itself back to the opposite side when it reaches an edge. The ISS icon is set as the turtle pen shape, giving it an angle of 45 degrees when moving.

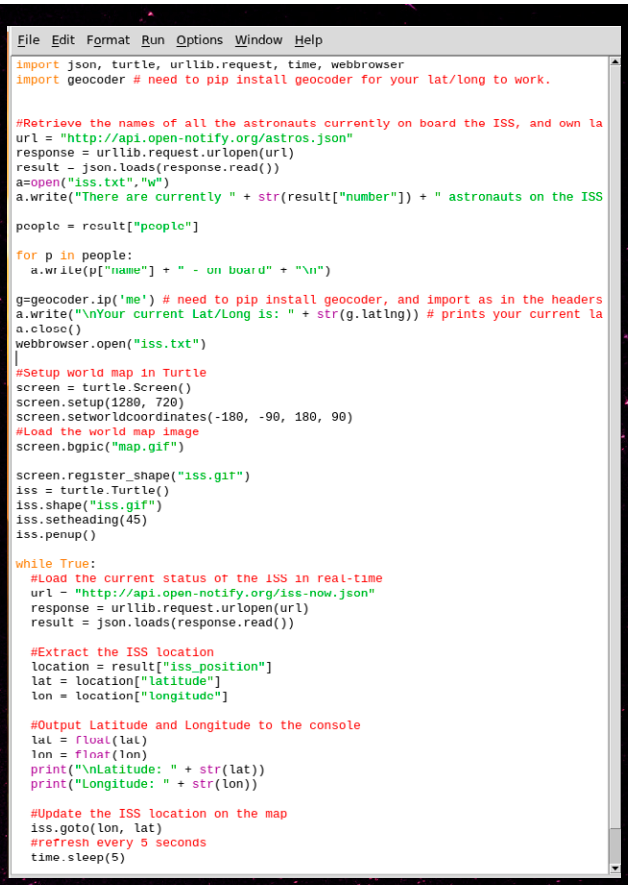
```
while True:
  #Load the current status of the ISS in real-time
  url = "http://api.open-notify.org/iss-now.json"
  response = urllib.request.urlopen(url)
  result = json.loads(response.read())

  #Extract the ISS location
  location = result["iss_position"]
  lat = location["latitude"]
  lon = location["longitude"]

  #Output Latitude and Longitude to the console
  lat = float(lat)
  lon = float(lon)
  print("\nLatitude: " + str(lat))
  print("Longitude: " + str(lon))

  #Update the ISS location on the map
  iss.goto(lon, lat)
  #refresh every 5 seconds
  time.sleep(5)
```

Now for the final part of the code: Here we collect the location data from the ISS status API, pulling out the latitude and longitude elements of the JSON file. The code then prints the latitude and longitude data to the console/Terminal, transferring the data from a float to a string in order to print it correctly. The last section will use the latitude and longitude as variables lat and lon, to update the ISS icon on the map every five seconds.

Here's the code in its entirety:

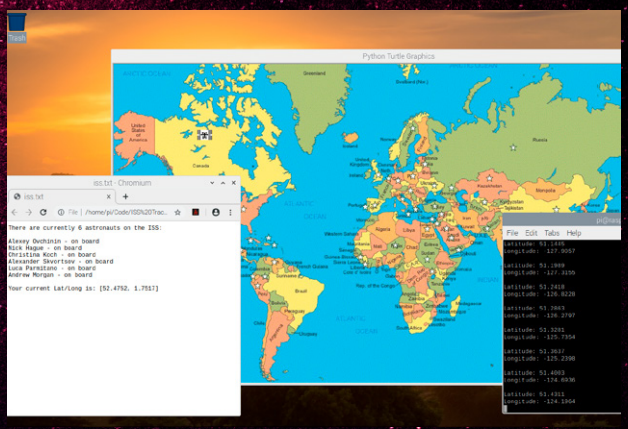
```
import json, turtle, urllib.request, time, webbrowser
import geocoder # need to pip install geocoder for
your lat/long to work.

#Retrieve the names of all the astronauts currently on
board the ISS, and own lat/long - write to a file and
display
url = "http://api.open-notify.org/astros.json"
response = urllib.request.urlopen(url)
result = json.loads(response.read())
a=open("iss.txt","w")
a.write("There are currently " + str(result["number"])
+ " astronauts on the ISS:\n\n")

people = result["people"]

for p in people:
  a.write(p["name"] + " - on board" + "\n")

g=geocoder.ip('me') # need to pip install geocoder,
and import as in the headers above
a.write("\nYour current Lat/Long is: " + str(g.
latlng)) # prints your current lat/long in the text
file.
a.close()
webbrowser.open("iss.txt")

#Setup world map in Turtle
screen = turtle.Screen()
screen.setup(1280, 720)
screen.setworldcoordinates(-180, -90, 180, 90)
#Load the world map image
screen.bgpic("map.gif")

screen.register_shape("iss.gif")
iss = turtle.Turtle()
iss.shape("iss.gif")
iss.setheading(45)
iss.penup()

while True:
  #Load the current status of the ISS in real-time
  url = "http://api.open-notify.org/iss-now.json"
  response = urllib.request.urlopen(url)
  result = json.loads(response.read())

  #Extract the ISS location
  location = result["iss_position"]
  lat = location["latitude"]
  lon = location["longitude"]

  #Output Latitude and Longitude to the console
  lat = float(lat)
  lon = float(lon)
  print("\nLatitude: " + str(lat))
  print("Longitude: " + str(lon))

  #Update the ISS location on the map
  iss.goto(lon, lat)
  #refresh every 5 seconds
  time.sleep(5)
```

Create a new folder in your system, called ISSTrack (for example), and place the two graphics as well as the Python code itself.



RUNNING THE CODE

The code is best executed from the command line, or Terminal. Clear your desktop, enter your command line, and navigate to where you've saved the code plus the two graphics. Launch the code with either: `python3 ISSTrack.py`, or, `python ISSTrack.py` (depending on whether you're using a Raspberry Pi/Linux, or a Windows PC, and what you've called the Python code).



The code will launch two extra windows together with the command line window you already have open. One will be the text file, containing the named astronauts, along with your current latitude and longitude and the other will be the world map with the ISS icon located wherever the ISS is currently orbiting. The command line window will start scrolling through the changing latitude and longitude of the ISS.

Text Animations

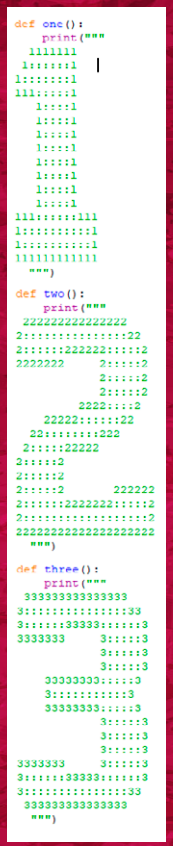
There's a remarkable amount you can do with some simple text and a little Python know-how. Combining what you've already learned, we can create some interesting animation effects from the command line.

THE FINAL COUNTDOWN

Let's begin with some example code that will display a large countdown from ten, then clear the screen and display a message. The code itself is quite simple, but lengthy. You will need to start by importing the OS and Time modules, then start creating functions that display the numbers (see image below) and so on to 10. It'll take some time, but it's worth it in the end. Of course, you can always take a different approach and design the numbers yourself.

The next step of the process is to initialise the code settings and start the countdown:

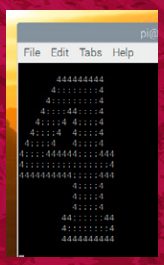
```
#Initialise settings
start = 10
message = ">       BLAST OFF!!          <"

#Start the countdown
for counter in range(start, 0, -1):
    if counter == 10:
        ten()
    elif counter == 9:
        nine()
    elif counter == 8:
        eight()
    elif counter == 7:
        seven()
    elif counter == 6:
        six()
    elif counter == 5:
        five()
    elif counter == 4:
        four()
    elif counter == 3:
        three()
    elif counter == 2:
        two()
    elif counter == 1:
        one()
    time.sleep(1)
    os.system('cls' if os.name ==
'nt' else 'clear')
```



And finally, we can add a display for the message:

```
#Display the message
print("v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v")
print("<                              >")
print(message)
print("<                              >")
print("v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v")
print("\n\n\n")
```

The code in its entirety can be viewed from within our Code Repository:



https://bdmpublications.com/code-portal, where you're free to copy it to your own Python IDLE and use it as you see fit. The end effect is quite good and it'll be worth adding to your own games, or presentations, in Python.

To extend the code, or make it easier to use, you can always create the number functions in their own Python code, call it Count.py for example, then import Count at the beginning of a new Python file called Countdown.py, along with the OS and Time modules:

```
import os
import time
import count
```

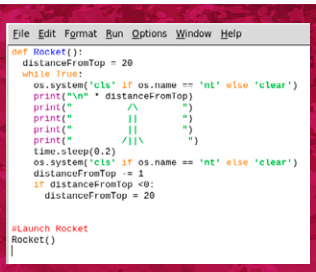
From there, you will need to specify the imported code in the Countdown section:

```
#Start the countdown
for counter in range(start, 0, -1):
    if counter == 10:
        count.ten()
    elif counter == 9:
        count.nine()
    elif counter == 8:
        count.eight()
    elif counter == 7:
        count.seven()
    elif counter == 6:
        count.six()
    elif counter == 5:
        count.five()
    elif counter == 4:
        count.four()
    elif counter == 3:
        count.three()
    elif counter == 2:
        count.two()
    elif counter == 1:
        count.one()
```

This will pull the functions from the imported Count.py and print them to the screen.

ROCKET LAUNCH

Building on the previous countdown example, we can create an animated rocket that'll launch after the Blast Off!! message has been printed. The code would look something like:



```
def Rocket():
  distanceFromTop = 20
  while True:
    os.system('cls' if os.name == 'nt' else 'clear')
    print("\n" * distanceFromTop)
    print("          /\          ")
    print("          ||          ")
    print("          ||          ")
    print("         /||\         ")
    time.sleep(0.2)
    os.system('cls' if os.name == 'nt' else 'clear')
    distanceFromTop -= 1
    if distanceFromTop <0:
      distanceFromTop = 20

#Launch Rocket
Rocket()
```

Here, we've created a new function called Rocket, which produces the effect of an ASCII-like rocket taking off and scrolling upwards; using the distanceFromTop variable.

To use this, add it to the end of the previous countdown code and, at the end of the Blast Off!! message, add the following lines:

```
print("\n\n\n\n")
input("Press Enter to launch rocket...")
```

This will allow your message to be displayed and then, when the user has hit the Enter button, the rocket will launch.

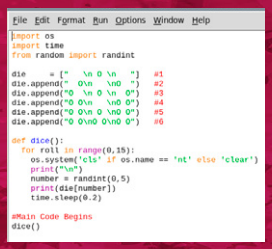
Again, the code in its entirety can be found in the Code Repository at: **https://bdmpublications.com/ code-portal**.

ROLLING DIE

Aside from the rocket animation, together with its countdown, another fun bit of text-based animation is that of a rolling dice.

A rolling dice can be a great animation to include in an adventure game, where the player rolls to see what their score is compared to that of an enemy. The highest roller wins the round and the losers' health drops as a result. It's an age-old combat sequence, used mainly in the Dungeon and Dragons board games and Fighting Fantasy novels, but it works well.

The code you'll need to animate a dice roll is:



```
import os
import time
from random import randint

die      = ["   \n O \n   "]   #1
die.append("  O\n   \nO  ")   #2
die.append("O  \n O \n  O")   #3
die.append("O O\n   \nO O")   #4
die.append("O O\n O \nO O")   #5
die.append("O O\nO O\nO O")   #6

def dice():
  for roll in range(0,15):
    os.system('cls' if os.name == 'nt' else 'clear')
    print("\n")
    number = randint(0,5)
    print(die[number])
    time.sleep(0.2)

#Main Code Begins
dice()
```

You may need to tweak the O entries, to line up the dots on the virtual dice. Once it's done, though, you'll be able to add this function to your adventure game code and call it up whenever your character, or the situation, requires some element of luck, combat, or chance roll of the dice.

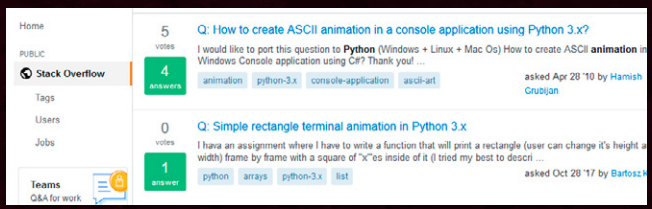
DISCOVER ANIMATIONS

The great thing about Python code is that it's so accessible. These few examples will help you add some fun, or something different, to your programs, but they're just the tip of the proverbial iceberg. If there's something you want to include in your code, and you're at a sticking point, then consider heading over to Stack Overflow and search for Python 3 content.

Stack Overflow is a great online help and resource portal, where you can ask questions and experts will try and help you. It doesn't always work out, but most of the time, you'll find what you're looking for within this great resource.

Also, if you're after a simple animation then take to Google and spend some time searching for it. While you may not find exactly what it is you're after, you're bound to come across something very like the desired effect; all you need to do is modify it slightly to accomplish your own goals.

You'll find that many professionals will often take to the Internet to find content they're after. True, they're able to code it themselves, but even experts get stuck sometimes, so don't worry about hunting code snippets down; you're in good company.



Retro Coding

There's a school of thought, that to master the foundations of good coding skills you need to have some experience of how code was written in the past. In the past is a bit of a loose term, but mostly, it means coding from the 80s.

```
5  HOME
10  PRINT "---UNIT CONVERTER---"
20  INPUT "[F]AH-CEL OR [C]EL-FA
    H: ";C$
30  INPUT "ENTER UNIT: ";UN
40 X = (UNIT - 32) * 5 / 9
50 Y = (UNIT * 9 / 5) + 32
60  IF C$ = "F" THEN  PRINT "CEL
    SIUS: ";X
70  IF C$ = "C" THEN  PRINT "FAH
```

THE GOLDEN ERA OF CODE

While it may seem a little counterproductive to learn how to code in a language that's virtually obsolete, there are some surprising benefits to getting your hands dirty with a bit of retro coding. Firstly, learning old code will help you build the structure of code as, regardless of whether it is a language that was developed yesterday, or forty years ago, code still demands strict discipline to work correctly. Secondly, everyday coding elements, such as loops, sub routines and so on, are a great visual aid to learn in older code, especially BASIC. Lastly, it's simply good fun.

```
1050 REM FOR I=DLSTART TO DLEND
1060 REM PRINT I,PEEK(I)
1070 REM NEXT I
1080 REM
1090 POKE 512,0
1100 POKE 513,6
1110 REM
1120 FOR I=1536 TO 1550
1130 READ A
```

GOING BASIC

The easiest retro language to play around with is, without doubt, BASIC. Developed back in the mid-sixties, BASIC (Beginner's All-Purpose Symbolic Instruction Code) is a high-level programming language whose design was geared toward ease of use. In a time when computers were beginning to become more accessible, designers John Kemeny and Thomas Kurtz needed a language that students could get to grips with, quickly and easily. Think of BASIC as a distant relation to Python.

THE BEEB

The problem with BASIC is that there were so many different versions available, across multiple 8-bit platforms, with each having its own unique elements on top of the core BASIC functions. The BASIC that was packaged with the Commodore 64 was different to that on the ZX Spectrum, or the Atari home computers, due to the differing hardware of each system. However, it's widely recognised that one of the 'best', and possibly most utilised, form of BASIC from the 80s was that of BBC BASIC.

BBC BASIC was used on the Acorn BBC Micro range of computers, utilising the MOS 6502-based processor technologies. It was one of the quickest examples of BASIC and, thanks to an inline assembler, it was also capable of allowing the developers of the time to write code for different processor types, such as the Zilog Z80 – a CPU present in the ZX Spectrum, as well as many arcade machines.

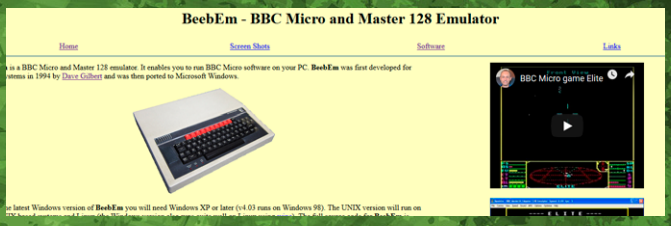
The BBC Micro was designed and built by Acorn Computers – a company that is historically responsible for the creation of the ARM CPU - the processor that's used in virtually every Android phone and tablet, smart TV, set top box and so on, as well as the Raspberry Pi – so essentially, the BBC Micro is the grandfather of the Raspberry Pi.

The BBC Micro was born in a time when the UK government was looking for a countrywide computer platform to be used throughout education. Different companies bid, but it was the BBC's Computer Literacy Project (the BBC Micro) that was chosen, due to its ruggedness, upgradability, and potential for education. As a result, the BBC Micro, or the Beeb as it's affectionately known, became the dominant educational computer throughout the 80s.

BEEBEM

Naturally, you could scour eBay and look for a working BBC Micro to play around on, and it'll be a lot of fun. However, for the sake of just getting hands-on with some retro code, we'll use one of the best BBC Micro emulators available: BeebEm.

BeebEm was originally developed for UNIX in 1994 by Dave Gilbert and later ported to Windows. It is now developed by Mike Wyatt and Jon Welch, who maintain the Mac port of the emulator, and is therefore available for Windows 10, Linux and macOS, as well as other platforms.



If you're using Windows 10, simply navigate to **http://www.mkw.me.uk/beebem/index.html**, and download the BeebEM414.exe that's displayed in the main screen.

Once downloaded, launch the executable and follow the on-screen instructions to install it. MacOS users can get everything they need from: **http://www.g7jjf.com/**. However, Raspberry Pi and Linux users will have to do a little nifty keyboard work before they can enjoy the benefits of the Beeb on their screens. Here's how to get it working under Linux:

First, drop to a Terminal and enter:

```
sudo apt-get update && upgrade
```

```
wget http://beebem-unix.bbcmicro.com/download/beebem-0.0.13.tar.gz
```

Then, extract the compressed files with:

`tar zxvf beebem-0.0.13.tar.gz`, then enter the newly created Beebem folder with: `cd beebem-0.0.13/.`

Now enter the following lines, hitting Enter and answering Y to accept any changes:

```
sudo apt-get install libgtk2.0-dev libsdl1.2-dev
```

```
./configure
```

```
make
```

```
sudo make install
```

This may take some time, but when it's all done, simply enter: **beebem**, to start the BBC Micro emulator.

BBC BASIC

Once installed and powered up, BeebEm will display the default BBC system start-up, along with a couple of beeps. Those of you old enough to have been in a UK school in the 80s will certainly recall this setup.

In BASIC, we use line numbers to determine which lines of code run in sequence. For example, to print something to screen we'd enter:

```
10 print "hello"
```

```
BBC Computer 32K

Acorn DFS

BASIC

>
```

Once you've typed the above in, press Enter and then type:

```
run
```

```
BBC Computer 32K

Acorn DFS

BASIC

>10 PRINT "HELLO"
>RUN
HELLO
>_
```

We can of course expand the code to include variables, multi-line print statements, and so on:

```
1 CLS
10 Input "Hello, what's your name? " n$
20 print
30 print "Hi, " n$ " I hope you're well today."
```

Type RUN to execute the code, you can also type LIST to view the code you've entered.

```
HI, DAVID I HOPE YOU'RE WELL TODAY.
>LIST
   1 CLS
  10 INPUT "HELLO, WHAT'S YOUR NAME? "
N$
  20 PRINT
  30 PRINT "HI, " N$ " I HOPE YOU'RE WE
LL TODAY."
>
```

As you can see, variables are handled in much the same way as Python, a print statement on its own displays a blank line, and CLS clears the screen; although the Pi uses Clear instead of CLS. We're also able to do some maths work, and play around with variables too:

```
1 CLS
10 input "how old are you? " a
20 print
30 if a > 40 print "You're over 40 years old."
40 if a < 40 print "You're under 40 years old."
50 print
```

```
YOU'RE OVER 40 YEARS OLD.

>LIST
   1 CLS
  10 INPUT "HOW OLD ARE YOU? " A
  20 PRINT
  30 IF A>40 PRINT "YOU'RE OVER 40 YEAR
S OLD."
  40 IF A<40 PRINT "YOU'RE UNDER 40 YEA
RS OLD."
  50 PRINT
>
```

BBC BASIC also has some interesting built-in features, such as the value of PI:

```
1 REM Area of a circle
2 CLS
20 Input "Enter the radius: " r
30 let area = PI*r*r
40 print "The area of your circle is: "; area
50 print ''
```

```
>LIST
   1 REM AREA OF A CIRCLE
   2 CLS
  20 INPUT "ENTER THE RADIUS: " R
  30 LET AREA = PI*R*R
  40 PRINT "THE AREA OF YOUR CIRCLE IS:
 ";AREA
  50 PRINT '''
>_
```

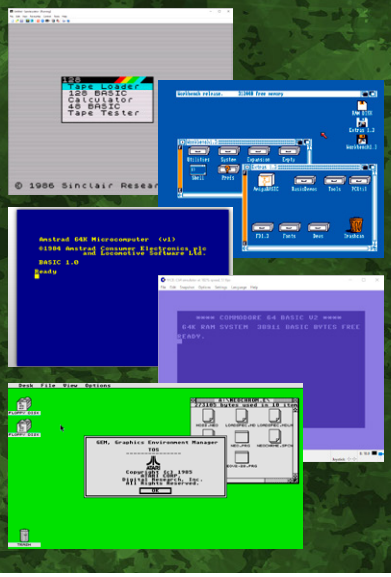
As you'll notice, variables with a dollar ($) represent strings, nothing after the variable, or a hash (#) represent floating point decimals; a whole integer has a % character, and a byte has an ampersand (&). The single quotes after the Print on line 50 indicate a blank line, one for each tick, while REM on line 1 is a comment, and thus ignored by the BASIC compiler.

Needless to say, there's a lot you can learn, as well as having fun, with BBC BASIC. It's a rainy day project and something that's interesting to show the kids – this is how we rolled back in the 80s, kids!

There are a number of sites you can visit to learn BBC BASIC, such as **http://archive.retro-kit.co.uk/bbc.nvg.org/docs.php3.html**. See what you can come up with using BBC BASIC, or other BASIC types for different systems, and let us know what you've created.

OTHER SYSTEMS

Naturally, you don't have to look to the BBC Micro to play around with some retro code. If you grew up with a Commodore 64, then you can always try VICE, the C64 emulator. Likewise, the ZX Spectrum has a slew of great emulators available for every modern system to play around on. In fact, you can probably find an emulator for virtually every 8-bit or 16-bit machine that was produced over the years. Each has their own unique perspective and coding nuances, so find a few and see what you can create.



Using Text Files for Animation

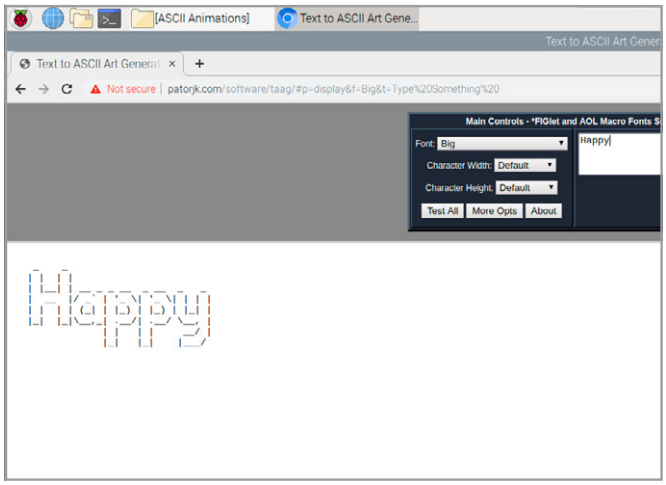
Animation in Python can be handled with the likes of the Tkinter and Pygame modules, however, there's more than one way to achieve a decent end result. Using some clever, text file reading code, we can create command-line animations.

ASCII ANIMATION

Let's assume you wanted to create an animated ASCII Happy Birthday Python script, with the words Happy and Birthday alternating in appearance. Here's how it's done.

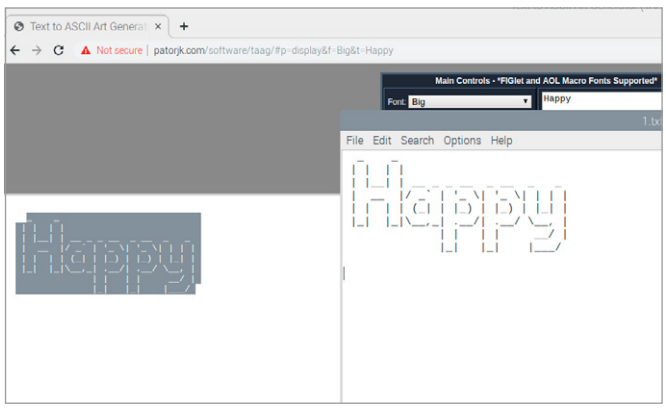
STEP 1

First we need to create some ASCII-like text, head over to **http://patorjk.com/software/taag**. This is an online Text to ASCII generator, created by Patrick Gillespie. Start by entering **Happy** into the text box, the result will be displayed in the main window. You can change the font with the drop-down menu to the side of the text box; we've opted for **Big**.



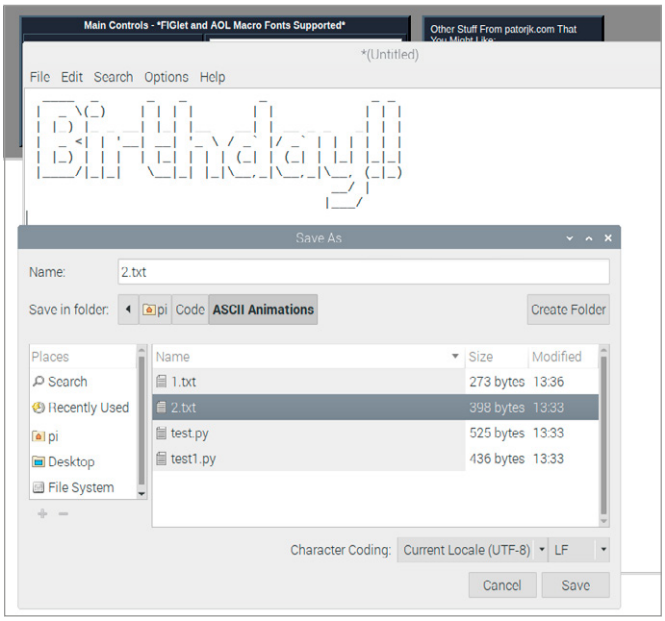
STEP 2

Now, on your computer create a folder in your Python code directory (call it **Test**, for now) and open either Leafpad for the Raspberry Pi or, if you're using Windows 10, then Notepad. Click on the **Select & Copy** button, at the bottom of the ASCII Generator webpage, and paste the contents into the text editor.



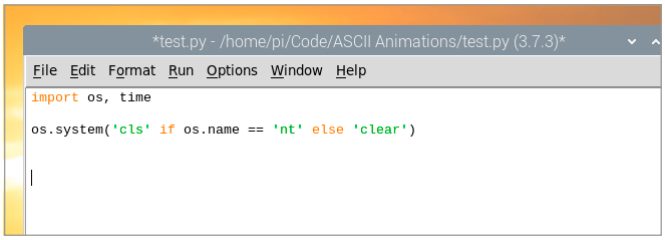
STEP 3

Save the text file as **1.txt** (you can call it anything, but now for ease of use 1.txt will suffice) and save the file in the newly created Test folder. When it's saved, do exactly the same for the word Birthday. You can select a new font from the ASCII Generator, or add extra characters, and when you're ready save the file as **2.txt**.



STEP 4

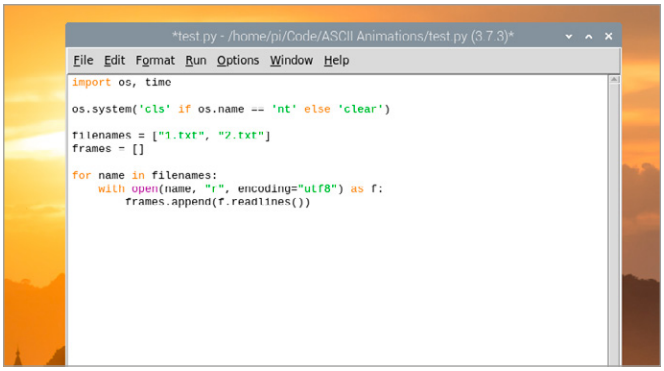
Open up Python and create a **New File**. We're going to need to import the OS and Time modules for this example, followed by a line to clear the screen of any content. If you're using Windows, then you'll use the CLS command, whereas it's Clear for the Raspberry Pi's Linux OS. We can create a simple if/else statement to handle the command.



STEP 5

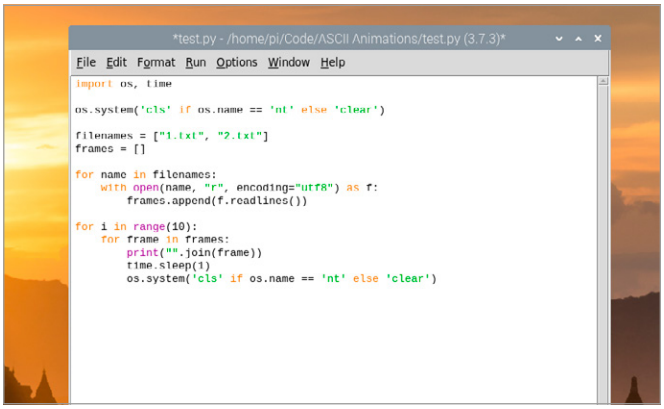
Next we need to create a list with the names of the text files we want to open, and then we need to open them for display in the Terminal.

```
filenames = [“1.txt”, “2.txt”]
frames = []

for name in filenames:
    with open(name, “r”, encoding=”utf8”) as f:
        frames.append(f.readlines())
```



STEP 6

We've used the UTF8 standard when opening the text files, as ASCII art as text, within a text file, often requires you to save the file as UTF compliant – due to the characters used. Now we can add a loop to display the files as 1.txt, then 2.txt, creating the illusion of animation while clearing the screen after each file is displayed.

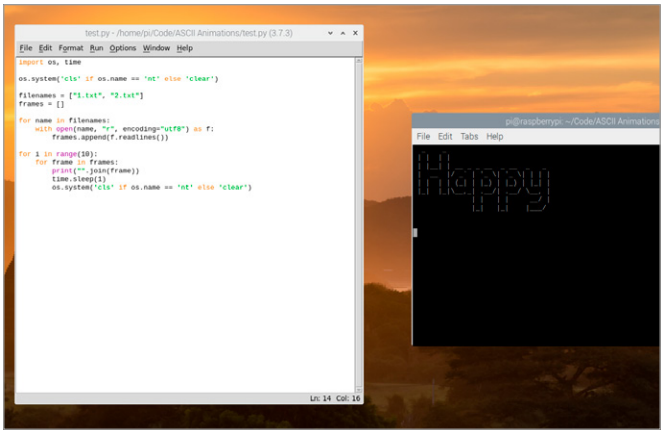


STEP 7

Save the Python code in the same folder as the text files and drop into a Terminal or Command Prompt. Navigate to the folder in question, and enter the command:

```
python3 NAME.py
```

Where NAME is whatever you called your saved Python code.



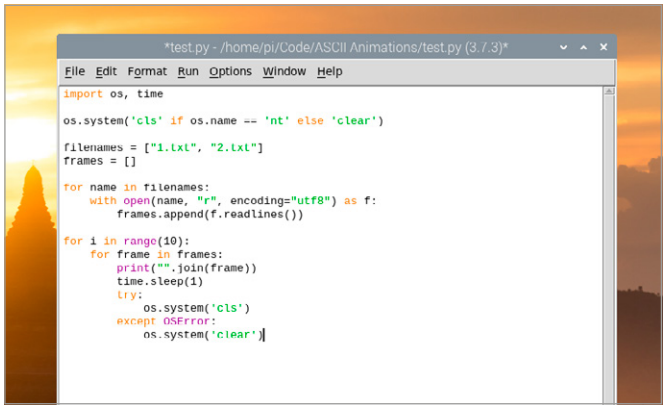
STEP 8

Here's the code in full:

```
import os, time

os.system('cls' if os.name == 'nt' else 'clear')


filenames = [“1.txt”, “2.txt”]
frames = []

for name in filenames:
    with open(name, “r”, encoding=”utf8”) as f:
        frames.append(f.readlines())

for i in range(10):
    for frame in frames:
        print(“”.join(frame))
        time.sleep(1)
        os.system('cls' if os.name == 'nt' else
'clear')
```

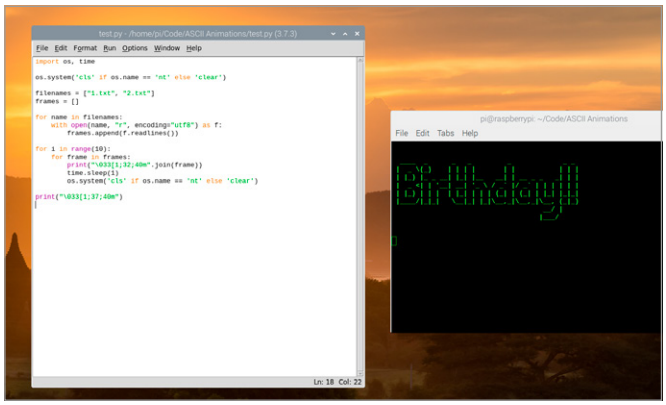
STEP 9

Note, from the loop, within the code, we've used the same CLS and Clear if/else statement as before. Again, if you're running on Windows then the OS module will use the CLS command, 'ELSE' if you're using Linux or a Mac, the Clear command will work correctly. If you want, you could use a Try/Except statement instead.



STEP 10

You can spice things up a little by adding system/Terminal colours. You'll need to Google the system codes for the colours you want. The code in our example turns the Raspberry Pi Terminal to green text on a black background, then changes it back to white on black at the end of the code. Either way, it's a fun addition to your Python code.



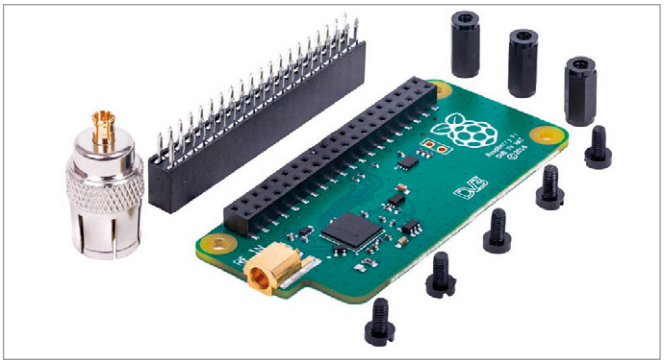
Stream Digital TV with a HAT - Part 1

A HAT, in case you're wondering, is a Hardware Attached on Top add-on board that connects to a Raspberry Pi's 40-pin GPIO. They can extend the capabilities of a Pi by adding motors, LCDs, sensors, controllers and more. In addition, they can be programmed via the Pi and Python.

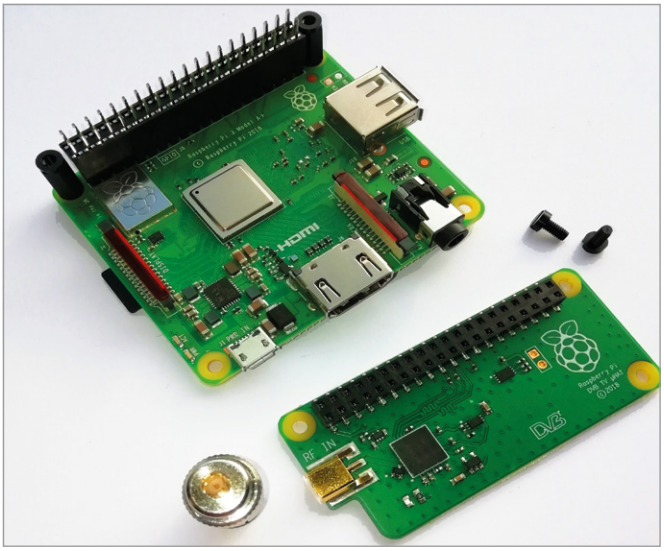
TV HAT

In this tutorial, we're using the Raspberry Pi TV HAT as sold by The Pi Hut (https://thepihut.com/products/raspberry-pi-tv-hat?variant=13539182673982). It's a Sony CXD2880 TV Tuner supporting DVB-T and 2nd-gen DVB-T2 standards.

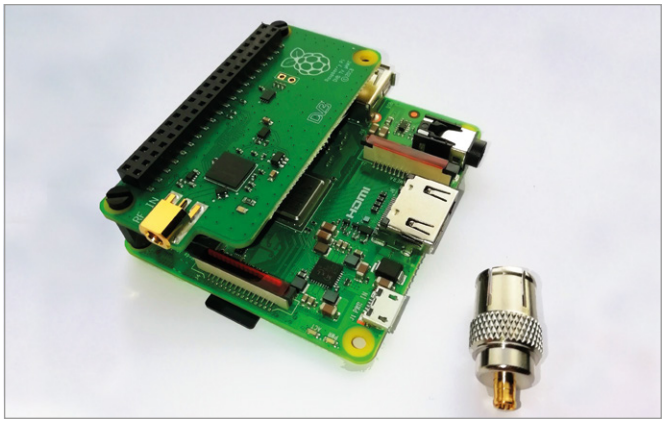
STEP 1 The Raspberry Pi TV HAT is compatible with the Pi Zero, Pi 3 A+ and B+ models. The kit comes with everything you need to connect the HAT to your RPi, including spacers, screws and the 40-pin header. Begin by opening the box and spreading the contents out on a clear area.



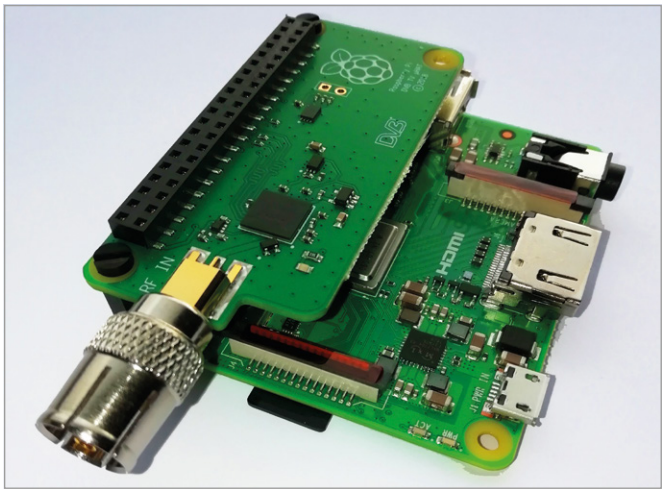
STEP 2 Start by connecting the 40-pin header to the 40-pin GPIO on top of your Raspberry Pi. Now take the spacers and screws and fit them to the corresponding holes in the corners of the main Raspberry Pi and the TV HAT; use three spacers and screws for the Pi Zero and two for the Pi 3 Models A+/B+.



STEP 3 With the spacers attached to the Raspberry Pi, line up the TV HAT with the 40-pin header, ensuring that the side of the HAT with the Pi logo is facing up and the HAT's gold coloured coaxial attachment is on the side of the SD card. When in place, screw the HAT to its spacers.

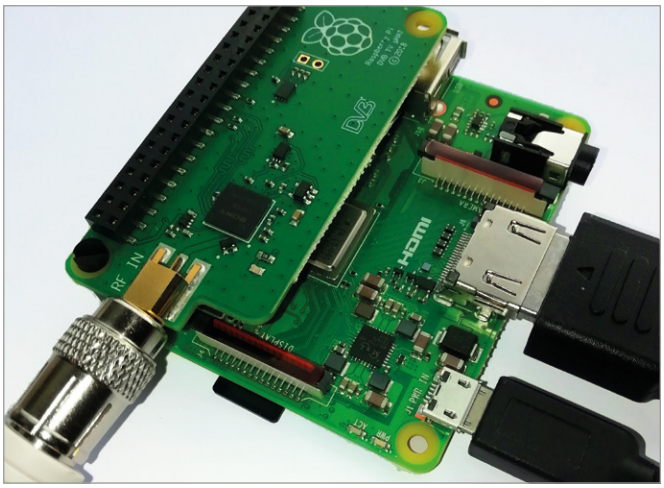


STEP 4 Now attach the coaxial adapter to the gold-coloured pin at the side of the TV HAT. It may need a firm push to lock into place, it will 'click' when fully inserted and in position. Now you will need to connect the TV HAT to your TV aerial and provide power to the Raspberry Pi.



STEP 5

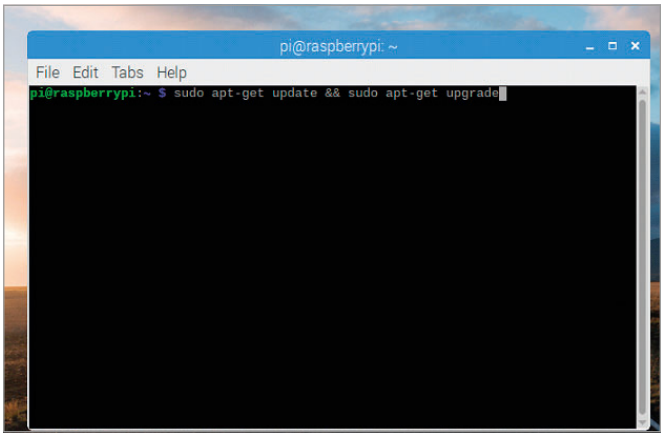
You can use the HDMI port to connect your Pi and its TV HAT to the TV/monitor, or you can just power the Pi (the TV HAT gets its power from the Pi) and connect remotely. The most important element is to ensure that the TV HAT is connected to an aerial that you know can receive a TV signal.



STEP 6

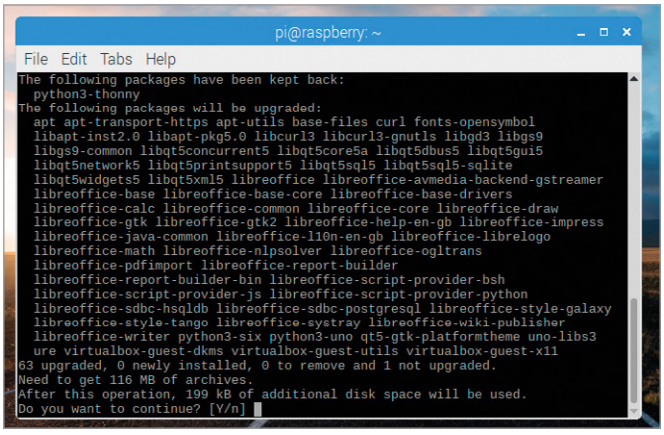
Power up your Raspberry Pi and when you get to the Raspbian desktop, open up the Terminal app. When at the Terminal, enter the following to ensure the Pi is up to date and running the latest versions of its software and system:

```
sudo apt-get update && sudo apt-get upgrade
```



STEP 7

Press Enter and allow the Pi to run through the update process. If you have any significant updates, you may need to answer 'Y' to any questions the Pi asks regarding these. Answering yes will replace the older software with the newer versions and is necessary for a smoothly running Pi.

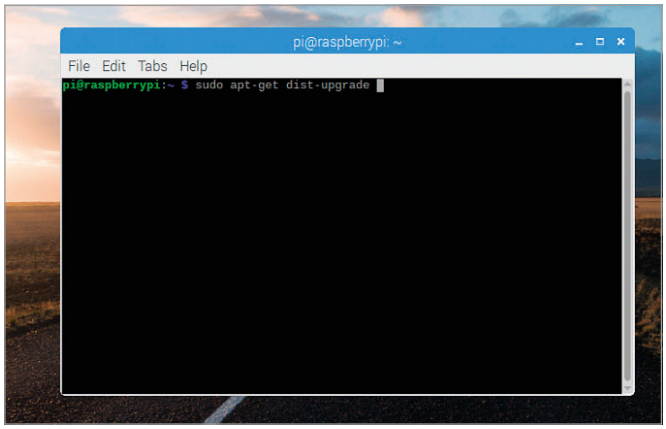


STEP 8

If you haven't used your Raspberry Pi since, at least, November 2018, then you may need to upgrade the core OS and synchronise the version of Raspbian with what's currently available from the Pi Foundation's downloads page. It's not totally necessary, but if you choose to, enter:

```
sudo apt-get dist-upgrade
```

Press Enter and follow the on-screen instructions.



STEP 9

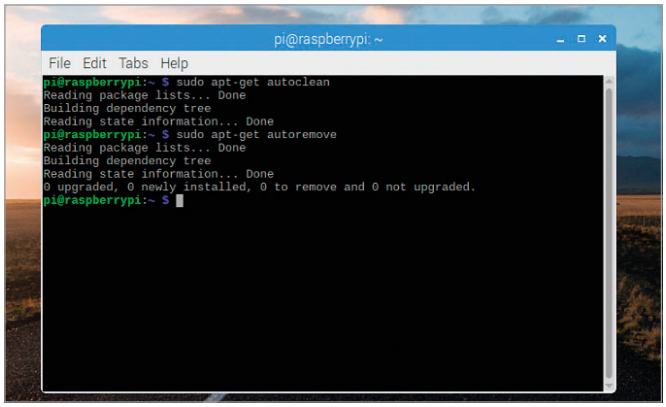
When the updates are complete, there's a good chance you've got some leftover, older setup files and packages in the system. To save space, use the following commands:

```
sudo apt-get autoclean
```

and:

```
sudo apt-get autoremove
```

Answer 'Y' to clear the system of the unnecessary packages and files.



STEP 10

At this point, it's worth noting that in the UK it's necessary to have a valid TV License in order to watch or record programmes as they are being shown on TV or live via an online service. It is an offence under Section 363 of the Communications Act 2003 to use a TV receiving device without a valid TV License.

Stream Digital TV with a HAT - Part 2

In part 1 of this tutorial, we set up the Raspberry Pi and the TV HAT. The core Pi software and system should now be up to date and the TV Hat connected to an aerial. You should also ensure you have a valid UK TV Licence.

TURN ON, TUNE IN

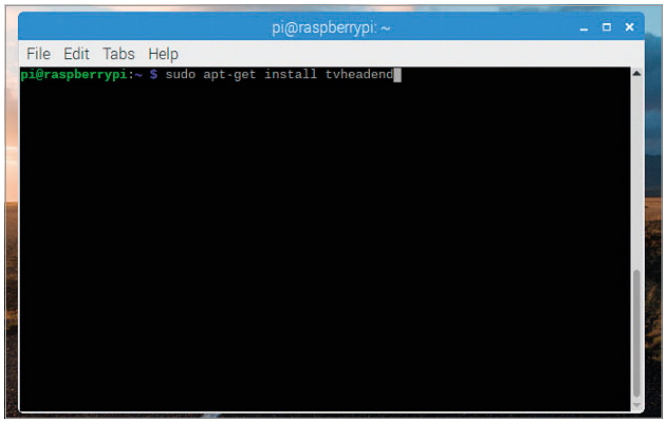
With the hardware ready, it's now time to begin the TV tuning software installation and setup. If you haven't already, reboot the Raspberry Pi and enter the Terminal for this next part.

STEP 1

To use the TV tuner, we need to install the TVheadend software. Open a Terminal session and enter the following:

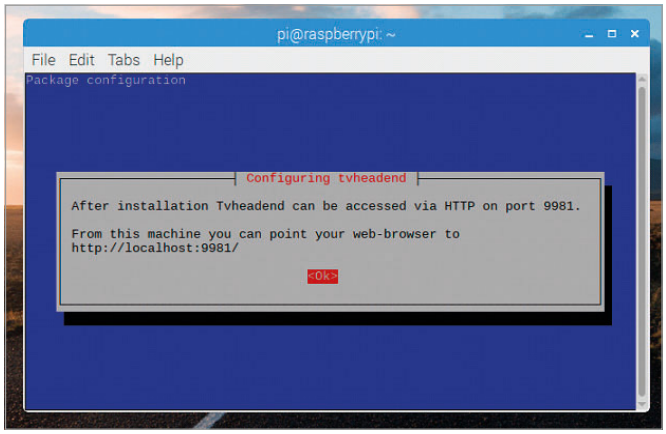
```
sudo apt-get install tvheadend
```

Press Enter and 'Y' if necessary to confirm the installation.



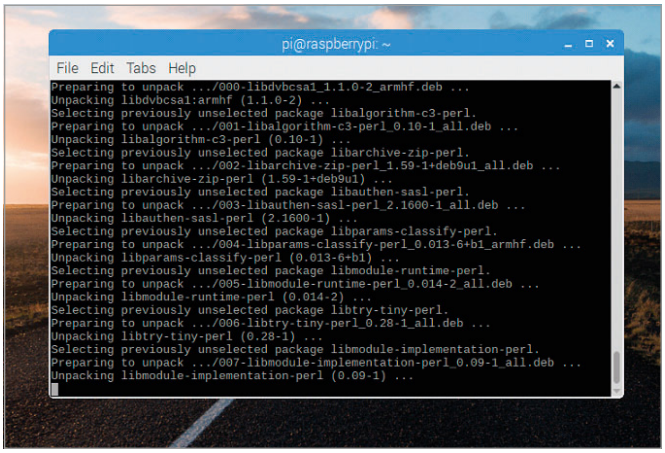
STEP 2

As the installation continues, you will be presented with a configuration screen. Enter a username and password to enable access to the TVheadend server. Once entered, make a note of the web address access for the server. When accessing on the TV Pi, it'll be **http://localhost:9981/**. If accessing from another computer on the network, use the Pi's IP address. For example: **http://192.168.1.223:9981/**.



STEP 3

The remainder of the setup will now continue. It'll take around three minutes to complete, depending on which model Raspberry Pi you're using. When the setup has finished, you can exit the Terminal session.

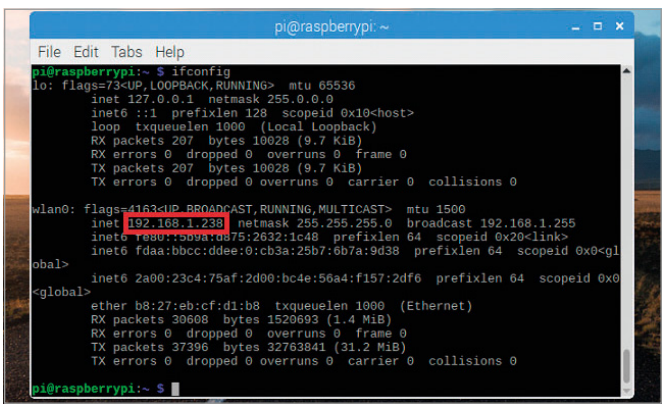


STEP 4

You can now either open the web browser on the Pi, if you're connected to the TV through the Pi, or, if you're connecting remotely, open any web browser on your computer with the address from Step 2. If you don't know the Pi's IP address enter:

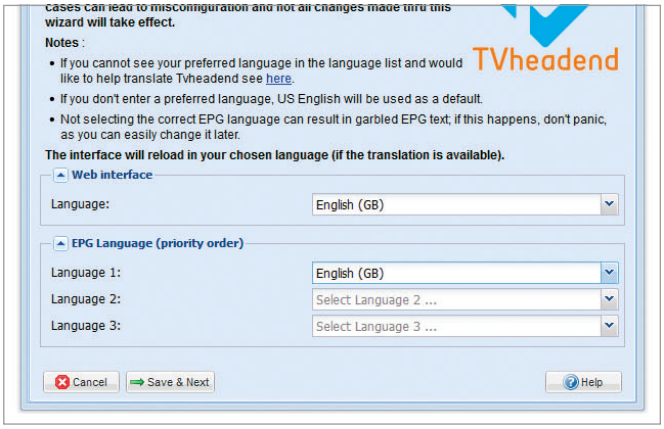
```
ifconfig
```

in the Terminal on the Pi. The **inet** entry is the Pi's IP address. Here, the example is: **192.168.1.238**.



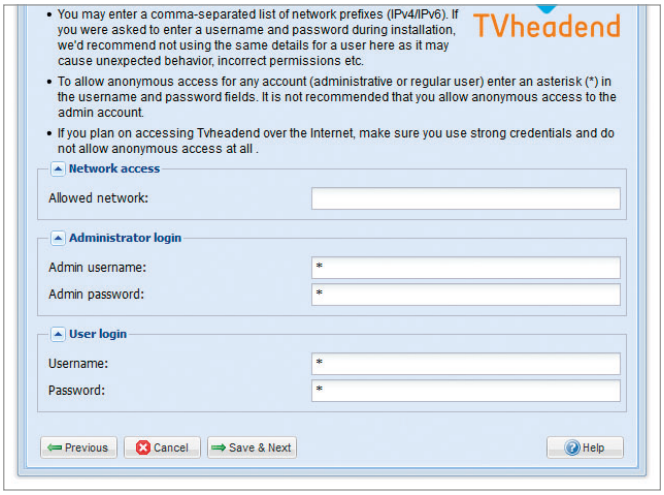
STEP 5

For this example, we will assume you're connecting remotely (from another computer on the home network). Enter the Pi's IP address with the port 9981, e.g.: **http://192.168.1.238:9981/**. To start the Configuration Wizard: enter the username and password you set up from Step 2 and log in to the TVheadend Server, then set the Language and Language 1 options to your preference.



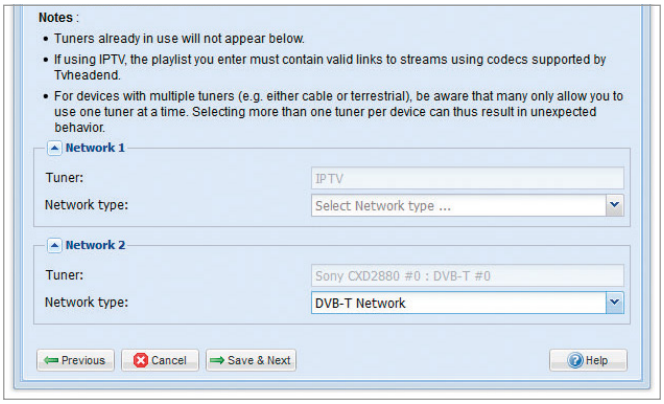
STEP 6

Click the **Save & Next** button to continue. You will need to enable network access to the server. Leave the Allowed Network field blank, but add an asterisk (*) in each of the other fields. Click **Save & Next** for the next step in the setup process.



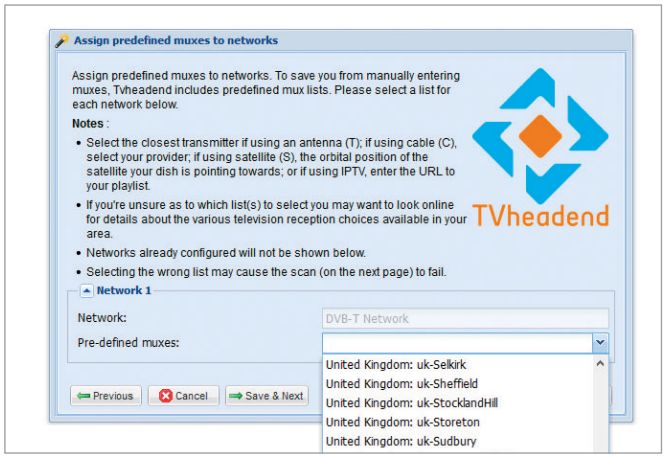
STEP 7

For the Network Settings page, leave the first three fields blank, but use the pull-down menu to select DVB-T Network. The TVheadend server will already have pre-selected the Sony CXD2880 tuner (the TV HAT) from its available choices. Click the **Save & Next** button when ready.



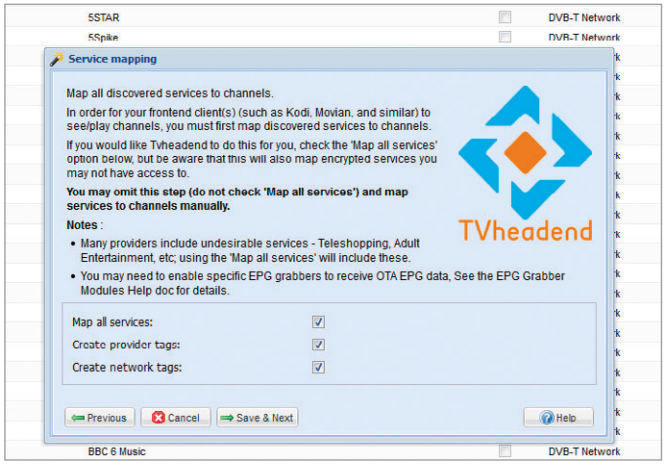
STEP 8

For this next step you need to select the transmitter closest to your location. You can find the closest transmitter by entering your details at: **http://www.digitaluk.co.uk/coveragechecker/**. Simply choose the transmitter from the pull-down list. When ready, click on the **Save & Next** button. The TV HAT will now scan for all available TV signals from the chosen transmitter.



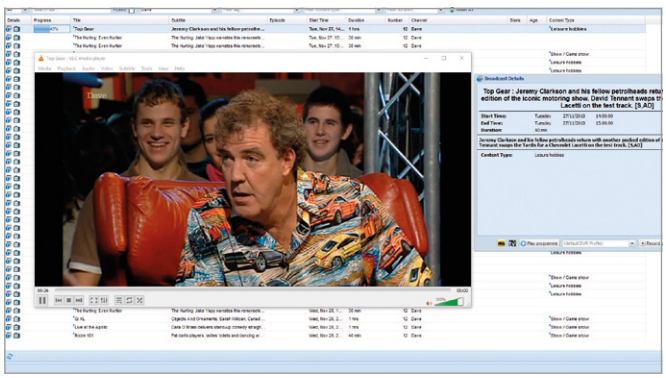
STEP 9

On the next screen, tick all three of the available boxes. Click the **Save & Next** box followed by the **Finish** button on the next screen; It's recommended that you now reboot the Pi and when it's fully rebooted, navigate back to the TVheadend server webpage.



STEP 10

The TVheadend server webpage will now display the list of available channels. After picking one, either click the title of the programme showing to expand the details, then click the **Play Programme** button to view its content, or click the small TV icon in the details column of the channel. You can now watch live TV across your home network.



Pi Projects: Desktop Pi

As you are now no doubt aware, the Raspberry Pi is certainly a versatile little piece of technology. Let's look at some of the popular projects you can apply to your Pi, starting with one that's not only the easiest, but also one of the best.

COMPUTER PI

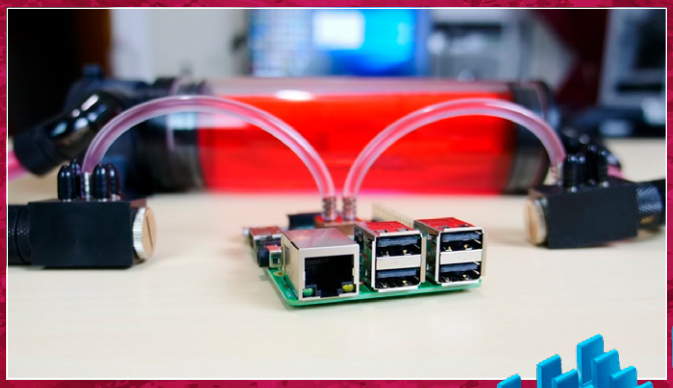
First and foremost, the Raspberry Pi is a computer. It has an operating system, built-in productivity apps, the ability to connect to both a home network and the Internet (either wirelessly or through Ethernet), and you can connect a mouse, keyboard, and monitor. It stands to reason, then, that it makes for a remarkably cheap desktop computer.

The obvious advantage of using the Raspberry Pi as a desktop computer though, is the fact that it's so cheap. For around £100 you could easily purchase the Pi, monitor, keyboard and mouse, providing you with a fully-functional computer capable of doing most, if not all, of the things you would do on a computer that costs ten times that amount. However, there are ways in which you can further enhance the desktop Pi project.

CASES

To begin with, you will need the Raspbian Stretch with desktop and recommended software OS version from the Raspberry Pi Foundation page, **www.raspberrypi.org/downloads/raspbian**. Once you've installed that to your Pi, booted it, and logged in with a personal user account, you can then start looking at one of the many colourful, and functional, cases available for the Pi.

The Pi Hut offers a superb selection of cases for the various versions of the Raspberry Pi, for this project, we'd recommend the Raspberry Pi 3 Model B+, as it's the fastest and most capable of the current models. For just £6, you can have a colourful – white and red – official Raspberry Pi case, with pre-formed holes that line up perfectly with the ports on the Pi. For around ten pound more, there's also the FLIRC case, an impressive design that's finished in brushed aluminium. On the other hand, if you want something a little different from the norm, then how about a case that's designed to look like a Nintendo Entertainment System, a SNES, or even a SEGA Megadrive?



COOLING

Another thing to consider, if you're going to be using your Pi as a desktop computer, is cooling. While the Pi's low use of resources means that its internals don't get too hot, prolonged and intense use may see your Pi getting a little warm, especially if it's inside a case and the ambient temperature of the environment is also warm. There are options available to help cool your Pi down, chiefly the Raspberry Pi Heatsink. This is a high-quality aluminium heat dispersing element that fits on top of the Pi's processor, leeching heat away from the vital circuit board and out into the surrounding air. However, bigger and more effective heatsinks are available, as well as fans, and even a water-cooling kit. These are a little extreme though, and only necessary if your Pi is doing some serious number crunching for an extended period of time.

STORAGE

While the Raspberry Pi desktop project is a great idea, the sad fact of the matter is that the Pi doesn't exactly offer the user a wealth of storage; compared to a Windows or Mac desktop, that is. Naturally, you can use a larger capacity SD card on which to install Raspbian, and utilise the remaining space as your Home folder, but again you're limited. Thankfully, it's not too much expense to include an external hard drive. A portable, USB connected, slimline 2TB hard drive can cost in the region of £60 and give you as much storage capacity as one of the more traditional desktop computers. One thing worth noting with regards to using an external hard drive, make sure you always have a backup of what's on there.

There are many options available to you if you decide to go down the Raspberry Pi desktop computer project route; there's even a Pi laptop kit available called Pi-Top. Needless to say, with this amount of choice you're able to customise your desktop and its setup in a more personal way than on a Windows PC, or a Mac.

PERIPHERALS

To cut down on the amount of cable clutter on your desktop, you could consider opting for a Bluetooth keyboard and mouse. The Pi 3 Model B+ offers Bluetooth connectivity alongside its Wi-Fi capabilities, so effectively you're able to use any recently purchased Bluetooth keyboard and mouse kit available. The major advantage here is that the Pi can be mounted to the rear of your monitor and, together with connecting to your home network via Wi-Fi, the only cables involved would be the power and the HDMI cable to the monitor, both of which could be neatly tucked away.

DON'T THROW AWAY YOUR PC YET

While the Raspberry Pi is a great little desktop computer, it does have its limitations. If you're accustomed to using a reasonably powerful Windows PC, or Mac, then you may find that, while being efficient, cheap and offering more space on your desk, the Pi does tend to struggle from time to time.

This is purely down to the fact that the Raspberry Pi isn't the most powerful computing device available today. It won't be able to handle the latest, triple-A games, intense graphics, Virtual Reality, or even some of the higher definition media content. You may find it stuttering when trying to playback fast moving 1080HD scenes, and true 4K playback isn't an option.

However, if you're looking to substitute your day-to-day workstation with something cheaper and smaller, and you're not going to push the Pi's processor too much, then you'll find the Pi to be a wonderful desktop computer. If you want more, though, hang on to that more powerful PC for the moment.

Pi Projects: Retro Gaming

Those of you old enough to recall the golden era of the home computer, the 80s, will have fond memories of playing on a Commodore 64, ZX Spectrum, Amiga, Atari ST, and the countless wonderful arcade machines that consumed more than their fair share of our pocket money. Hold on, because you're in luck.

GOING RETRO

The Raspberry Pi makes for an amazing retro gaming computer. With it you're able to emulate and play the classic home computers, consoles and arcade machines that brought us so much joy back in the day.

Thankfully, the processing power of most of the systems of the past is well within the capabilities of the Raspberry Pi's processor. There are a few examples that don't run too well, such as a PS2, or those systems that utilised a specialised 3D component, but on the whole, there's likely to be a fully working emulator, available for the Pi, that covers the home computer, console, or arcade that you remember playing.

RetroPie

RetroPie is the foremost retro gaming project available for the Raspberry Pi. It's a set of modules that are built upon Raspbian, an older project called EmulationStation, and Linus distribution called RetroArch. It contains built-in emulation that covers dozens of systems, from an Amiga through to an Atari 2600, Amstrad CPC to SEGA Dreamcast, ZX Spectrum to an Apple II; and the list is continually growing thanks to the contributions from the community.

RetroPie can work as an installation on top of an existing operating system, such as Raspbian, or you can install RetroPie to an SD card and boot the Pi directly into it, choosing to add further software later if you want. Once installed, you're able to connect USB controllers, or even a PS4 controller via Bluetooth, and if you're feeling up to it, there's also support for original controllers when connected to the Raspberry Pi's GPIO pins.

If you don't want to go down the RetroPie route and instead you only want one or two systems emulated, then Raspbian has a wealth of individual emulators available for you to install. You can search for the system via Google, or if you already know the name of it, simply install it to Raspbian through the Terminal. It's worth mentioning that there will often be multiple emulators designed for the same system. While one may work perfectly with the vast majority of games, it may struggle with some of the better games available for that particular system. On the other

hand, another emulator may play all the games for a system almost perfectly, with perhaps a loss of sound in some parts, or a slow down in others. It's therefore up to you what you want. You can have multiple emulators installed for a single system and use them depending on which game you want to play, based on how well the emulator supports them, or you can find the single emulator that works reasonably well with everything. It's trial and error, finding the perfect setup.

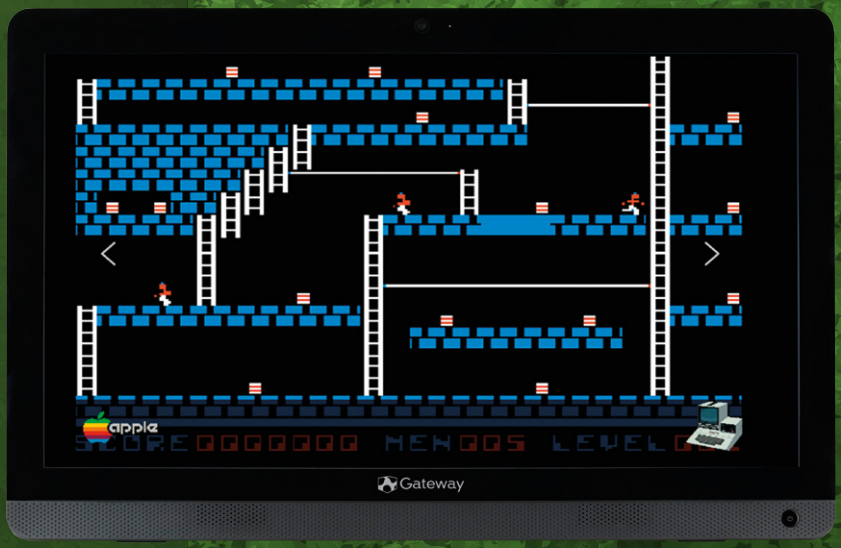
There are further options you can pursue when building a Raspberry Pi retro gaming system. You can encase your Pi inside one of the many retro-themed cases, such as the Mega-Pi SEGA Megadrive case, or you can build your Raspberry Pi into the Picade desktop arcade machine.

Picade is a great project, featuring authentic sticks and buttons, and an external speaker. There's artwork available for the arcade cabinet setup, along with full instructions on how to set up the Pi and connect everything to the GPIO pins. The 8-inch display is perfect for old-school gaming, and you can improve on the visuals by adding extra stickers, a different acrylic marquee, and posters. This, combined with a RetroPie installation, is an excellent project idea that'll keep you entertained for hours.

ABOUT ROMs

A ROM is the actual contents of a game, or the BIOS of an old system, be that a home computer or a console. These ROMs are often ripped from the original tape, cartridge, or chip, and are available to download from the Internet.

However, the use of ROMs is a continual legal headache. Most ROMs are illegal, meaning that they are available to download without the permission of the developer who created the game, the publishing house that released the game, and the company that owns the rights to the system on which the game is intended to be played.



There's a school of thought that if you own the original tape, cartridge and so on, of the game then you're legally allowed to obtain a ROM of the same game and play it on an emulator, but that's not always correct. In the example of music, it is technically illegal to copy music from a CD you own to playback on a media device; and in some ways the same applies to a ROM.

There are however, some examples of a situation when a title, or a system, has fallen out of copyright, or has been abandoned. In these instances, it is perfectly legal for someone to generate a ROM from the game and distribute it on the Internet. Sometimes, another developer improves the original game, adding extra levels, effects, and so on. Some original developers of older games have allowed the use of their game to be distributed as a ROM, and therefore it is legal for you to download and play it.

In short, if you download a ROM from the Internet, and it doesn't specify that it's abandonware or allowed by the developer, publishing house, or company, then you're doing so illegally. It's highly unlikely that the police will drop in through your bedroom window and arrest you on the spot – unless you actively host illegal content on a website you own – so it's purely down to your own conscious.

Remember, even if the game is over thirty years old, somewhere out there is a developer who spent the time creating it, so by playing a pirated version of it - an illegal ROM, you're taking something away from the individual or team that programmed the game in the first place.

LEGAL EMULATION

Interestingly, while it's illegal to download most ROMs, it is not illegal to install and use an emulator. In fact there are emulators available for the PS4 that allow you play old SEGA Megadrive games, so while Sony, or whoever released the emulator for the PS4, have paid for the rights to use the ROMs in their emulator, it isn't illegal for them to develop and use the SEGA Megadrive emulator.

In the same respect, it's not illegal for you to download an emulator for any of the systems, you will just need to find legal ROMs to play on it.

Pi Projects: Media Centre

Of all the Raspberry Pi projects out there, turning the tiny computer into a powerful media centre is probably top of the list. And thanks to the Pi's diminutive dimensions, and reasonable processing performance, it is wholly achievable.

FILM NIGHT

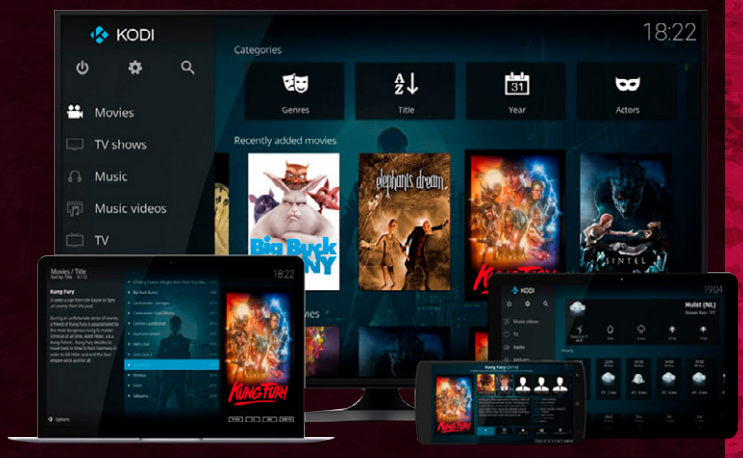
First off, what is a media centre? Basically, a media centre, with regards to what we're talking about here, is a computer that's capable of playing, possibly recording, and even sorting a media collection made up of videos, music and images. It's a home cinema system, connected to a TV that will stream media from online, or from a network attached storage device, or storage connected directly to the computer itself.

A media centre is often capable of sorting your collection of videos, music, and photos into a logical order, either by album, genre, year, alphabetically, or some custom setup. It can connect to the Internet and display information regarding the media, such as album cover, movie poster, location an image was taken, or even connect to an online database such as IMDB and provide further information. In short, it's a one-stop location for all your content.

KODI

Since the Pi's first appearance, there has been some form of media centre software available for it. Back then, when it launched in 2012, the major player was XBMC, Xbox Media Centre, and was considered one of the best cross-platform media centre applications ever developed. By 2014, though, the team behind XBMC decided to change its name to Kodi.

Kodi itself is an entertainment hub that collates and organises all your digital media into a single, user-friendly, and beautifully designed user interface. Thanks to its design, Kodi is capable of playing all types of music media and video media, as well as streaming TV shows (via online, or through a compatible TV tuner) and photos, plus you can record live TV through a PVR. It's extraordinarily customisable, allowing you to install further add-ons that can connect to other online services like SoundCloud, YouTube and so on.



AUTO-START KODI

If you want to start Kodi as soon as your Raspberry Pi powers up, then you will need to alter one of the configuration files. While in the Terminal, after installing Kodi, enter:

```
sudo nano /etc/default/kodi
```

Look for the entry ENABLED, and change it to 1:

```
ENABLED=1
```

Press Ctrl+Z to save and exit Nano, then reboot your Pi with:

```
sudo reboot
```

Your Pi will now reboot and Kodi will start up once the main boot sequence has completed.

LIBREELEC VS. OSMC VS. RASPBIAN

Which OS to install? We could argue all day over the benefits, advantages, and disadvantages of each available operating system and you will probably still be as undecided as when we began.

In truth, there is no perfect operating system, as each offers its own unique way of doing things. In the end, it simply comes down to which one do you prefer. Raspbian has everything, yet will be slower, LibreELEC and OSMC are faster, but aren't as fully featured as Raspbian. What you need to do is experiment with all three, have a look around to see what else is available, and then decide on which works best for you and your particular goals with the Raspberry Pi.

OTHER OPTIONS

Of course, you don't necessarily need to go down the full media centre road to utilise the Pi as an excellent media playback device. As you have already noticed, the Raspberry Pi comes pre-installed with its own specialised version of VLC, a very capable media player. VLC on the Pi has been designed to make use of the Pi's processing power, and has some hardware acceleration benefits coded directly into it, making it one of the best media players available.

All you need, therefore, is to have a software updated Pi with VLC, and access to wherever you've stored your media collection. While it may not be as graphically impressive as Kodi, you can still playback most, if not all, forms of digital media.

INSTALLING KODI – OPTION 1

Should you want to install Kodi on your Raspberry Pi and use it as an all-powerful media centre, there are two options available to you. The first is to simply boot up Raspbian, drop into a Terminal session, perform an update and upgrade, then enter:

```
sudo apt-get install kodi
```

Once the setup is complete, you can start Kodi by entering: kodi, into the Terminal.

INSTALLING KODI – OPTION 2

The second option for installing Kodi on the Pi is to use one of the Kodi-optimised operating systems suitable for the Raspberry Pi. If you recall, from when you first installed Raspbian via NOOBS, there are several OS options available to you of which two are LibreELEC and OSMC. Both of these operating systems are especially designed to work better with Kodi than the fully installed version of Raspbian, due to being a more lightweight OS and less system resource heavy.

This option is often regarded as the preferred method of turning the Pi into a media centre, as both LibreELEC and OSMC are much faster operating systems, giving more of the Pi's resources over to Kodi and allowing it to playback content without causing too many problems.

Pi Projects: BBS Client

In a digital world, before the Internet was a common household name, there existed a connected community of surfers. These individuals didn't surf the WWW, instead they dialled up Bulletin Board Systems (BBS) and opened a whole new world of content.

WARGAMES

If you're old enough to recall, or have since watched, the excellent movie Wargames, then you'll be roughly familiar with the way in which a Bulletin Board System works; and if you haven't watched Wargames, then we recommend you get hold of it.

In the movie, the young protagonist spends his days at the keyboard of his early 80s computer, using his modem to dial into remote systems. Once inside these remote systems, he then goes about traversing the remote host's file system looking for anything interesting.

The movie plot aside, this, essentially, is how a BBS works. It's a remote computer that runs specialist BBS server software with a mix of content either pre-installed, or added by the system admin (sysadmin or sysop). A user of the BBS can then dial, in the old modem sense, the BBS server's phone number and gain access to the system with a valid username and password; or if they're new, they have the option to create a new user.

These days, of course, the dial-up aspect has pretty much gone the way of the Dodo (although there are still some retro stalwarts who relish in the chronic noise of a dial-up connection), however, we can still enjoy the retro feeling of a traditional BBS using the legacy protocol, Telnet.

WHY?

In a world of Internet snooping, a BBS is probably one of the last bastions of digital privacy; to some degree. A private BBS is somewhere you can connect with likeminded individuals, to chat, swap code, reminisce, play a text-based adventure, or simply just hang out. True, you can get hold of copyrighted or explicit content, but that's only if you connect to those BBSes that serve such content, just as with the Internet.

Most BBSes follow a theme, whether that's old DOS-based adventures, ZX Spectrum fans, Commodore 64 gamers, or even something non-techie related, such as a Ford Cortina owners club; no doubt swapping owner manuals, old photos of the Cortina E and such.

In truth, a modern BBS is a bit of fun. Connecting to a system someone has installed and built, set around a particular theme, and designed with fantastic looking ANSI graphics, is a great pastime. It's a form of respect, in some ways, to acknowledge the work that's gone into creating the BBS by connecting to it and you also get to learn a little more about how protocols work, and how everything is connected.

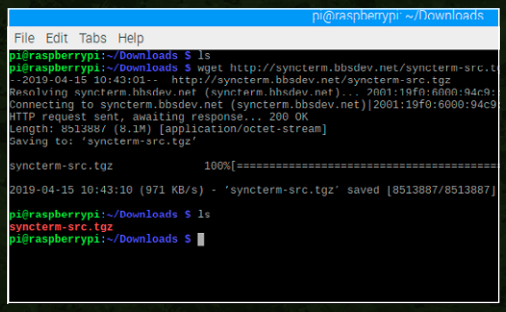
HOW?

Connecting to a BBS via the Raspberry Pi is quite easy, but to get the most from it you will need to get your hands dirty in the Terminal.

As mentioned, we're going to be using a form of the protocol Telnet, in this instance specifically the program SyncTERM, to emulate the old-style Terminals that support ANSI art and IBM fonts, while connecting to the remote BBS with the Telnet protocol. You can simply use telnet under the Terminal (once you've installed it), but you'll miss some of the glorious artwork displayed within the majority of the BBSes.

To begin with, drop into a Terminal session on your Pi. When the Terminal is fired up, enter sudo apt-get update && sudo apt-get upgrade to ensure your system is up to date. If everything is okay, enter: sudo apt-get install telnet. While this stage isn't strictly necessary, it's always a good idea to have the base protocol client installed.

When telnet is installed, you can then start the procedure of installing SyncTERM. In order to get SyncTERM working, you'll need to build it from source. By now, you should be a dab hand at this, but here's the process in case you've forgotten (along with some added elements to help everything go to plan).



Begin by changing directory to the Downloads folder and downloading the source code:

```
cd Downloads\
wget http://syncterm.bbsdev.net/syncterm-src.tgz
ls
```

With ls entered, you should see the newly downloaded tgz file. To unpack the downloaded file, enter:

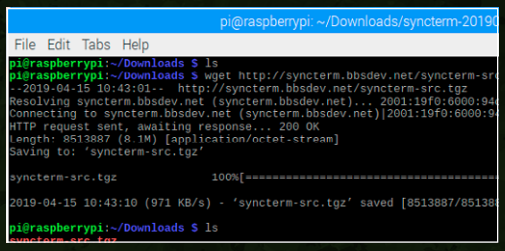
```
tar -xf syncterm-src.tgz
```

This will create a new syncterm-(DATE) folder, where DATE is the current date when you've unpacked the contents of the tgz file.

You will now need to change directories to:

```
cd syncterm-(DATE)
cd src
cd syncterm
```

You can mesh these directories together, but, for the sake of keeping things simple, we'll stick to one folder at a time. Also, remember you can hit the Tab key to auto-complete a directory name.



Once in the syncterm directory, you can begin to build from source. However, before you do that, it's worth installing a couple of extras to ensure the BBS session works to perfection. Start by installing the following:

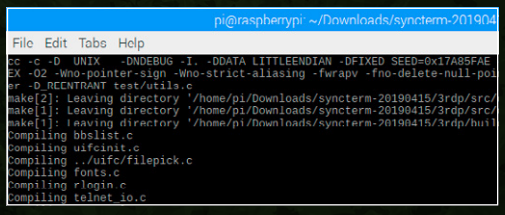
```
sudo apt-get install libncurses5-dev
sudo apt-get install libsdl1.2-dev
```

Once these two are installed, start the build process by entering:

```
sudo make
```

Followed by:

```
sudo make install
```



The process may take a few minutes, so be patient. When everything is installed, you can enter the command: syncterm, to start the program and change the screen settings, if you wish. However, to get straight into connecting to a BBS, try one of these commands:

```
syncterm dura-bbs.net: 6359
syncterm bbs.kernelerror.com: 10023
syncterm particlebbs.dyndns.org: 6400
syncterm heatwave.ddns.net: 9640
syncterm sysgod.org:23000
```

Naturally, some, or even all, of these BBSes may be offline when you come to test them; they are, after all, being operated by individuals like you and I. If they are offline, you can always get hold of a comprehensive list of active servers by visiting **https://www.telnetbbsguide.com/bbs/list/brief/**.

GET CONNECTED

It's worth spending some time finding the sort of BBS that suits your tastes. As you'll see by visiting the aforementioned website, there's over 500 BBSes currently listed, so somewhere in there a BBS could be your new online haunt.

Pi Projects: Weather Station

The Raspberry Pi has offered its userbase a superb platform to science. There are Raspberry Pi projects that involve the International Space Station, alongside NASA and ESA-led applications. However, it's an Earth-based project that's taken the community by storm.

IT'S RAINING PI, HALLELUJAH!

Weather stations have been available to the project-minded public since long before the Raspberry Pi entered the scene, but thanks to the Pi's unique specifications, a new generation of weather station has emerged; and they're really quite remarkable.

As with a lot of Raspberry Pi projects, the market has become flooded with various hardware, kits, and HATs that will enable you to set up a simple, or complex, weather station. The more experienced weather station and Raspberry Pi user will probably lean toward buying in the components individually, as opposed to an entire kit, but for the rest of us the kit-form versions are an excellent start to a project that can quickly grow over time.

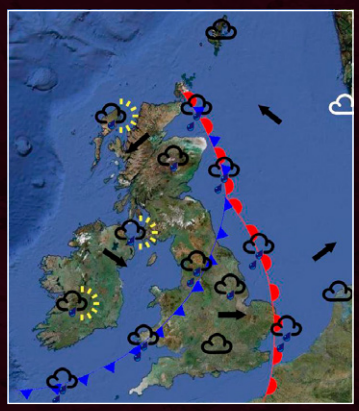
WHY?

With our smart devices connected to everything around us all the time, having a multi-limbed weather station, perched precariously on the end of the garden shed, measuring the amount of rainfall and in what direction the wind is blowing, may seem a little unnecessary; since your phone could tell you all that already with just a swipe or two. However, that discounts the pleasure of creating your own project and measuring the results as they change from one day to the next.

Any project, regardless of whether it's on a Raspberry Pi or not, is something from which to learn. In the case of a Raspberry Pi driven weather station, the user can learn about all manner of electronics, connectivity, some simple engineering, and coding, even before we get onto the varying aspects of meteorology.

The weather station project doesn't necessarily need to be an all-powerful outdoor contraption either. Basically, there are two routes you can opt to take: the outdoor model, measuring wind speed/direction, precipitation and so on, and an indoor model that measures ambient temperature, pressure and humidity.

Why indoors? There are a number of reasons. Perhaps you're in charge of a collection of servers, and while the company you work for may not be able to stretch to a fully sealed environment machine room, you could fill the gap with a great collection of sensors attached to a Raspberry Pi. You may have some from of allergy, where precise control of your indoor environment is important. Maybe you look after some exotic animals, in a room a where air pressure, humidity and such are vital to the creature's health. Needless to say, there are plenty more examples.



In short, setting up a weather station is simply a fun project. Setting aside the things you'll learn from getting one up and running, you'll eventually have a physical piece of hardware that can, with moderate accuracy, measure your environment and display that data in a number of graphs and charts. It's quite cool, when you stop to think about it.

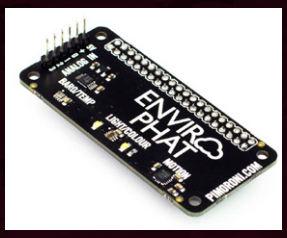
SOME OPTIONS?

To go into all the currently available Raspberry Pi weather station options will take more space than we're currently able to offer here, but let's have a quick look at some of the best selling kits and components.

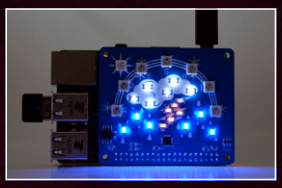
Weather Meter Kit – For outdoor weather monitoring, we have the Weather Meter Kit as sold by CPC for around £77. The kit comprises of three main component sensors that measure wind speed, wind direction, and rainfall. The rain gauge is a self-emptying bucket that's capable of activating for each 0.011-inch of rainfall collected, while the anemometer activates a closure switch every 1.492mph once per second. The wind direction module can display up to sixteen different positions, and the kit comes complete with full assembly instructions.

Velleman WS3080 Weather Station – The Velleman WS3080 Weather Station is one of the higher-end products that can interact with the Raspberry Pi. This outdoor mounted station is capable of monitoring rainfall, wind speed and wind direction, air pressure, wind chill temperatures, air temperature, and UV light levels. It features alarms for temperature, humidity, wind chill, dew point, wind speed, air pressure and storms, and comes with a separate LCD transmitter.

Pimoroni Enviro Phat – The Enviro pHAT is an add-on sensor for the Raspberry Pi that includes four different sensors to monitor temperature, pressure, light levels, and motion. It's ideal for indoor environment monitoring and, thanks to some nifty Python libraries and code, you're able to display the data via a web page and view the variables remotely.



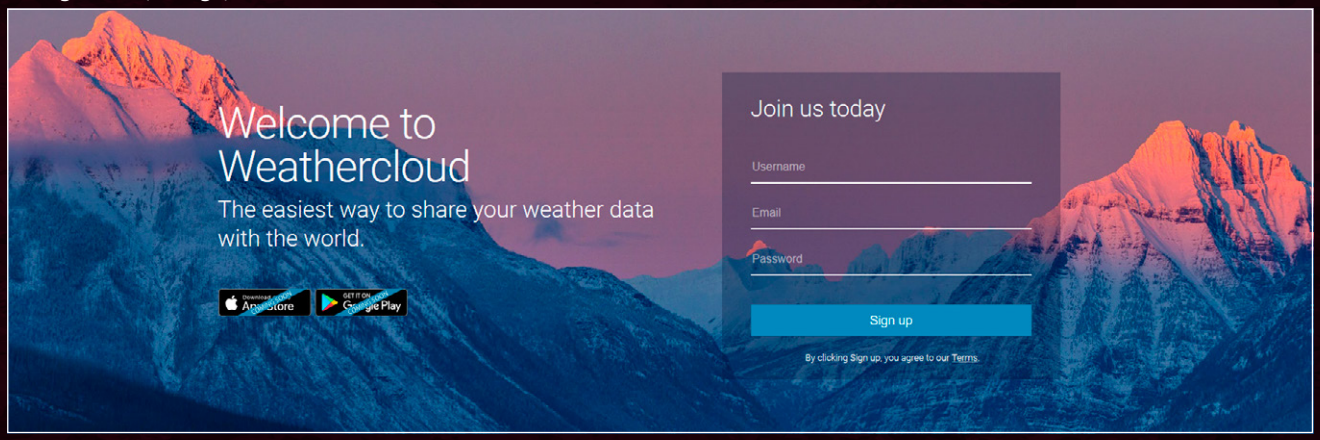
Cyntech WeatherHAT – The WeatherHAT from Cyntech can turn the Raspberry Pi into a weather display station. While not obtaining data directly from an external source (instead using one of the many localised, weather monitoring stations), it does offer seven RGB LEDs to highlight the current and upcoming weather. Controlled via Python, the HAT can help provide a good coding base for future weather station projects.



WeeWX – When you collect the necessary hardware together (the various sensors you'll need and so on), you'll need a good piece of software to help collate all that data and display it in a readable fashion; this is where WeeWX comes in. WeeWX is a Python-based program that can interact with many different weather stations to produce graphs and reports, and even publish the data to HTML on a web site.

WHERE NEXT?

As you can imagine, it'll take some time to get everything together, program the components, and collate the data you want to view. Once all that's gathered, though, what next?



There are some websites available that display localised weather from independent sources such as a back garden weather station, so when you've got everything up and running, try looking out for such projects and start collaborating. Also, consider creating a Twitter account for the Raspberry Pi weather station and Tweeting the local weather.

Common Raspberry Pi Problems

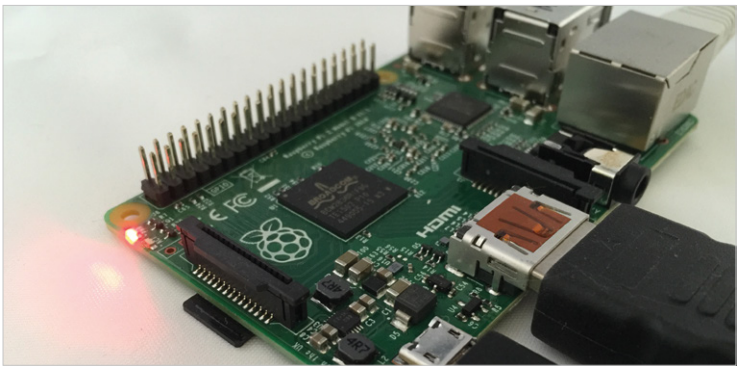
The Raspberry Pi hardware and software is pretty reliable and problems are more often due to set up errors rather than anything to do with the hardware. However, there are times when hardware can seem to be at fault, so here are a few of the more common issues you might encounter when using your Raspberry Pi.

TROUBLESHOOTING YOUR RPI

The Raspberry Pi is a surprisingly stable bit of kit, but there is always the chance of encountering problems. If you're really stuck there are apps available which can help diagnose problems on the RPi.

RED POWER LED IS BLINKING

A blinking red power LED indicates problems with the power supply. On model A and B, it is hard-wired to the 3.3V power supply rail. If it is blinking, it means the 5V power supply is dropping out. Use a different power supply. On the model B+ and also the A+, the circuit has been improved to give a much more reliable warning of poor power quality. The red power LED is wired to an APX803 supervisor which kicks in when the 5V power supply drops below 4.63V. If it does, the LED will blink. Check your connections, cable and power supply.



COLOURED SPLASH SCREEN

With the current firmware, a coloured splash screen is displayed after GPU firmware (start.elf) is loaded. This should be replaced by Linux console a second later. However if the coloured screen remains, it suggests the kernel.img file is failing to boot. Try replacing it with a known good one.

Immediately after displaying the splash screen, the Pi starts consuming a little more current. If the Pi resets at that moment, it is an indication that the power supply isn't able to deliver the full current that your Pi requires but dips its output voltage below a minimum when loaded with the full current the Pi needs.

GREEN LED BLINKS IN A SPECIFIC PATTERN

1 flash: Possibly you have a RPi from Micron. Take a good look at the processor if it says M with an orbit round it. Using the latest software will solve your problem; also make sure you have a 4Gb SD card, as a 2Gb doesn't work.

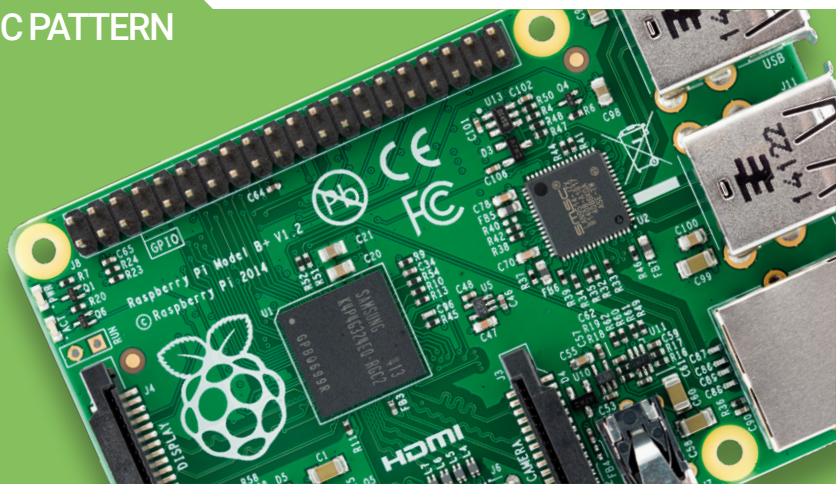
2 flashes: The SD Card cannot be read. A solution could be to forma the card and flash Raspbian with Pi Installer from Terminal.

3 flashes: Start.elf not found.

4 flashes: Start.elf not launched.

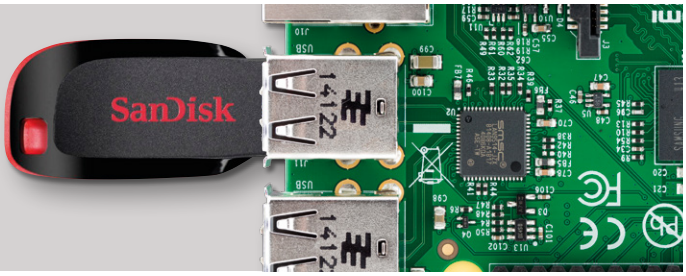
7 flashes: Kernel.img not found.

8 flashes: SDRAM not recognised. You need newer bootcode.bin/start.elf firmware.



NO USB DEVICE WORKS

The most common cause of USB devices working is low power supply voltage from a bad PSU, cable or USB hub; but it could also be that no clock signal is present. Return the board for a replacement if you think this is the case but before coming to this conclusion, confirm known good peripherals. A significant number of USB keyboards are not compatible with Raspberry so make sure you are using one made for Pi.



RASPBERRY PI NOT RESPONDING TO KEY PRESSES

This is most often caused by inadequate power. Use a good power supply and a good power cable. Some cheap cables that work with a mobile phone, cannot fully power the Pi. Some USB devices require a lot of power; most will have a label showing the voltage and mA requirements. Each one should be 5v 100mA max. Any more than this and they must be used with a powered USB hub. Try unplugging every USB device except the keyboard. You should also note that some keyboards have built in hubs and can try to draw 150mA; Pi can only handle 100mA per USB slot without a hub. Use the latest software too.

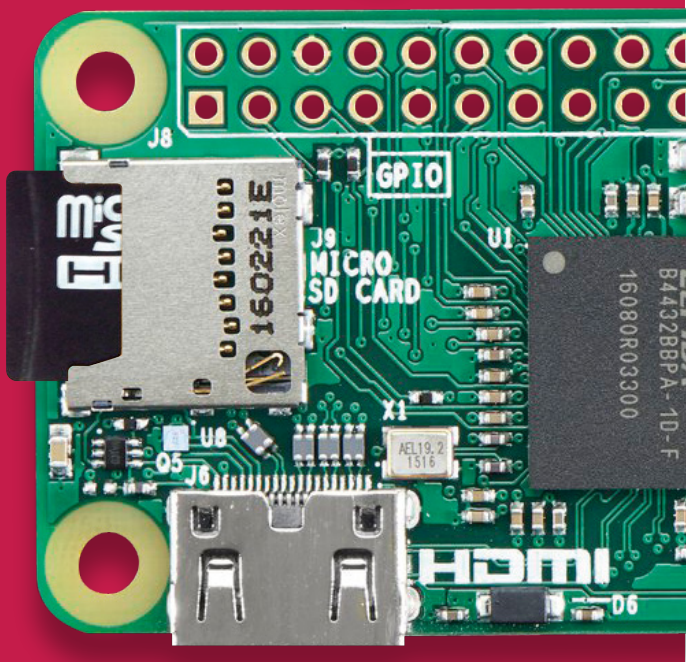
KEYBOARD OR MOUSE INTERFERES WITH A USB WI-FI DEVICE

Connecting a keyboard and or mouse while a USB Wi-Fi device is connected, may cause one or both devices to malfunction. Tests point to interferences in the 2.4 GHz frequency band in which both Wi-Fi sticks, as well as USB keyboards transmit data. Changing the channel on the wireless access point should fix the problem completely.

SD CARD PROBLEMS

If you have problems, check you have the latest firmware version first. If that is not the problem, try the following.

- Some SD cards do not work on the Pi, so check the list of known SD cards on the official Pi website.
- If you are having problems setting up your SD card you might want to start by erasing it completely, especially if it has been used elsewhere and still contains data or partitions.
- Windows and Mac users can download a formatting tool from the SD Association: **https://www.sdcard.org/downloads/formatter_3/**
- Reformatting cards is also easy to do in a digital camera.
- After writing the image to the SD card, verify that you can see the boot partition when you insert the SD card into your computer. The partition should contain a number of files, including start.elf and kernel.img. If you do not see these files on the SD card, you have made an error writing the image file.
- If you are manually preparing your SD card on Linux or macOS using the dd command, this operation will completely erase any existing data and partitions. Make sure you write to the whole card (e.g. /dev/sdd) and not to an existing partition (e.g. /dev/sdd1).
- If you put the SD card into your PC in an attempt to write the Pi operating system onto it and the PC tells you the card is write-protected, even with the write-protect tab in the correct forward position you may have a faulty SD-card rewriter.



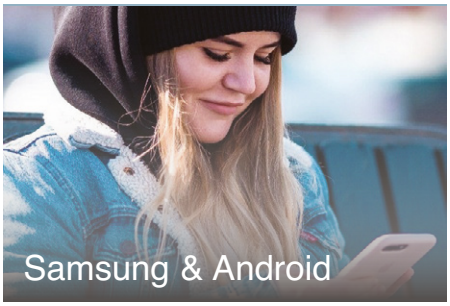
Black Dog Media

Master Your Tech — From Beginner to Expert

To continue learning more about your tech visit us at:

www.bdmpublications.com

FREE Tech Guides



EXCLUSIVE Offers on our Tech Guidebooks

- Print & digital editions
- Featuring the very latest updates
- Step-by-step tutorials and guides
- Created by BDM experts

Check out our latest titles today!

bdmpublications.com/ultimate-photoshop

Buy our Photoshop guides and download tutorial images for free!

Simply sign-up and get creative.

PLUS

Special Deals and Bonus Content

Sign up to our monthly newsletter and get the latest updates, offers and news from BDM. We are here to help you Master Your Tech!

The Ultimate Jargon Buster

Avoid tech confusion, either when reading this book or when talking to friends, with this glossary of technology terms and phrases. We have looked across the tech boarders to bring to you the definitive jargon buster, but it should help you to understand the common terms people use when talking about their devices and their software.

0-9

3G
The third generation of mobile data networking used by both the iPhone and iPad. This connection is slower than Wi-Fi, but is more readily available and is used to transfer data from your device when you are on the go. It uses the mobile phone network.

4G
The fourth generation of mobile data networking.

5G
The fifth generation of mobile data networking offers increased speed when transferring data on the go but it is still in its early stages of adoption by mobile phone networks.

A...

Accessibility
A series of tools and features designed to make an Apple device such as the Mac and mobile devices easier to use by those with disabilities such as vision or hearing impairments. You can find the Mac's Accessibility features and customise them in System Preferences.

ADB
Android Debug Bridge. Part of the Android Software Development Kit, used to send commands from a computer to an attached phone.

Adobe Bridge
Bridge is a browser application produced by Adobe Systems as part of the Creative Suite and is usually installed alongside Photoshop. Its main function is as the file management hub of the Creative Suite. It can be used to open, manage, rate and rename files as well as edit their metadata.

Adobe RGB
A device independent colour space developed by Adobe. It provides a relatively large range of colours, i.e. grey-balanced and perceptually uniform. It is widely used for image editing.

ADSL
Asymmetric Digital Subscriber Line. It's a means of connecting to the Internet through your telephone line. Sometimes just called 'DSL'

Airplane Mode
All airlines warn you to turn off mobile electronic devices when on board an aircraft, so this iPad setting turns off all incoming and outgoing signals to your device, including data, Bluetooth and Wi-Fi.

AirPlay
A protocol for streaming sounds and video from an Apple device to a set of compatible speakers or a device such as an Apple TV. It's wireless and easy to use.

AMOLED
Active Matrix Organic Light Emitting Diode. A bright and colourful display technology popular on smartphones (although it has now been superseded by Super AMOLED and qHD.)

Android
The name of the operating on your smartphone (we are assuming you own an Android phone if you are reading this magazine). There have so far been eleven versions/updates released.

Android Market
The previous name for the Google Play Store. The place to go to find apps, books and movies to install on your phone.

Anti-Aliasing Filter
This is an optical filter, also known as low-pass filter, which is placed on the camera sensor to create a slight blur or softening that helps counteract aliasing or Moiré interference.

Apk (.apk)
The file extension of Android applications.

Apps (Applications)
The programs, such as Angry Birds, Facebook or Soundhound, that you install and run on you Android phone.

App Store
The App Store is where you can download free and paid programs to your device using your Apple ID. You can access it through the application found on your home screen.

App Inventor
A web-based system that lets anyone develop apps for Android. Originally created and run by Google, but now run as an open-source project.

Apple ID
This is the email address and password that you have registered with Apple. It will be required to access most online applications on your iPad, including iTunes, App Store and Books.

Apple Menu
The menu that's opened by clicking on the Apple icon in the left of the menu bar, when using a Mac or MacBook computer. It gives access to system functions such as Preferences, App Store, Force Quit and more.

Archos
A manufacturer of Android tablets.

ASUS
A well-known manufacturer of Android smartphones and tablets.

AT&T
One of the "Big Four" of American carriers

B...

Bit
A contraction of binary digit, the smallest unit of information storage or digital information that can take on one of two values, 0 and 1.

Bit Depth
Defines how many bits of colour data are used to describe each pixel or channel. For example, 2 bits per pixel only allows for black or white. 8 bits provides 256 colours. When referring to an 8-bit colour image, 256 is multiplied by the three primary channels (red, green and blue) to create what is commonly called 24-bit colour, with a possible 16,777,266 colours.

Black Point
In image editing, the black point is a tonal adjustment that sets the point at which the deepest shadow detail in the histogram is clipped to black.

Bloatware
The name given to unwanted applications preloaded onto your phone. Bloatware cannot usually be removed by the end user unless they decide to root their handset.

BlueTooth
Short range file data system built into almost every Android smartphone ever made. Can be used to send files and connect speakers or headphones wirelessly to your phone.

Books
Apple's eBook reader, available from the App Store. It handles the standard electronic publishing formats protected by FairPlay DRM and PDF. It was introduced in 2010 along with the iPad.

Bootloader
A normally hidden mode in Android that helps with flashing ROMs when rooting an Android phone.

Broadband
Wide bandwidth data transmission, that is, fast Internet as opposed to the older, dial-up services.

Browser
An app used to access websites found on the worldwide web. The iPad and iPhone come with Apple's Safari browser preinstalled but others are available in the App Store. Android devices use the Chrome browser

BSI
Backside Illumination. Sometimes used to improve smartphone camera performance.

C...

Calendar
This is one of several preloaded apps found on most devices. Use it to keep track of events, invitations and reminders on your phone and tablet.

Camera Raw
Proprietary raw file formats designed to hold image data and metadata generated by digital cameras. These formats are non-standard and undocumented, although they are usually based on the TIFF/EP file format standard.

Carrier
Another name for a mobile network provider (Vodafone, AT&A, Sprint, etc.)

Casting
The process of converting one data-type into another. For example, sometimes a number may stored as text but need to be converted in to an integer.

CCD (Charged Coupled Device)
A type of image sensor found in digital cameras and scanners. It is a light-sensitive chip that converts light into an electrical charge that is then processed by an analogue to digital converter. CCD differs from the other common sensor type (CMOS) in the way that it processes the electrical charges captured by sensor elements.

CDMA
One of the two main cell phone communication standards. Not often used in phones outside of the U.S.

Chromatic Aberration
Known also as colour fringing, chromatic aberration is caused when a camera lens does not focus the different wavelengths of light onto the exact same focal plane. The effect is visible as a thin coloured halo around objects in the scene, often the border between dark and light objects.

Class
A class provides a means of bundling data and functionality together. They are used to encapsulate variables and functions into a single entity.

Clipping
The loss or either highlight or shadow details when tone information is forced to pure white or black. For example, over-exposure can produce clipping by forcing highlights that should contain detail to register as pure white. Clipping can also be caused either intentionally as a creative effect or unintentionally because of excessive corrections. Saturation clipping can occur when colours are pushed beyond the range of a colour space.

Comments
A comment is a section of real world wording inserted by the programmer to help document what's going on in the code. They can be single line or multi-line and are defined by a # or '".

Constant
A number that does not change. It is good practice to name constants in capitals e.g. SPEED_OF_LIGHT

CMOS (Complementary Metal Oxide Semiconductor)
A type of image sensor found in digital cameras and scanners. It is a light-sensitive chip that converts light into an electrical charge, which is then processed by an analogue to digital converter. CMOS differs from the other common sensor type (CCD) in the way that it processes the electrical charges captured by sensor elements.

CMYK
Also commonly referred to as process colour, CMYK is a subtractive colour model using cyan, magenta, yellow and black inks in colour printing.

Colour Profile
Also known as an ICC profile, the Colour Profile defines the information required to by a colour management system (CMS), to make the colour transformations between colour spaces. They can be device specific such as monitors, scanners or printers or abstract editing spaces.

Compression
The process of re-encoding digital information using fewer bits than the original file or source. This reduces transmission time and storage requirements. There are a number of different algorithms that provide either "lossy" or lossless compression. JEPG is a common file format that employs lossy compression to achieve smaller file sizes at the expense of image quality.

Cupcake
The nickname for Android version 1.5.

CyanogenMod
One of the best known and most often used series of custom ROMs.

D...

DECT
Digital Enhanced Cordless Telecommunications. It's a wireless standard used mostly for cable-free telephone systems.

DLNA
Dynamic Living Network Alliance. A technology found on some high-end Android phones that lets users stream photos and videos from their phone to a compatible TV.

DNG (Digital Negative)
An open standard file format developed by Adobe Systems that provides an alternative to proprietary camera raw files. The DNG specification incorporates rich metadata along with embedded previews, camera profiles and editable notes. DNG uses lossless compression that can result in a significant file size reduction over the original proprietary raw.

Download
The term used when taking a file from the Internet or from a connected device such as a computer, to your phone or tablet.

Dock
The opaque strip at the bottom of the home screen. Apps in the dock remain in a special row of icons (or Folders post iOS 4) along the bottom of iPhone, iPod touch and iPad screens and do not change when you swipe between home screens.

DPI (Dots Per Inch)
The measurement of print resolution expressed in how many dots of ink are laid down either horizontally or vertically per inch. A higher number indicates a greater amount of output resolution. Not to be confused with pixel per inch (PPI). There is not necessarily a direct correlation between DPI and PPI.

Dream (HTC Dream or G1)
The very first phone to use the Android operating system.

Dynamic Range
In the context of photography, dynamic range describes the difference between the brightest and darkest light intensities of a scene. From capture to output, there can be a large difference in the size of the dynamic range that each device is capable capturing or reproducing. Dynamic range is commonly expressed in the number of f-stops that can be captured or the contrast ratio of the scene or device.

E...

Eclair
The nickname for Android version 2.0/2.1.

Emoticon
A small drawing used to augment a message or text. Typically these are yellow faces showing a variety of expressions.

Escape Sequence
When characters that have certain meanings in the Python coding language are required in strings they have to be "escaped" so that the computer knows they do not have their usual meaning. This is done by putting a slash in front of them e.g. \"

Ethernet
The format used for local cabled networks (LAN). Your router comes supplied with Ethernet cables and has ports for plugging them in.

Exposure
The total amount of light that strikes the sensor or film during an image capture. An optimal exposure takes full advantage of the dynamic range of the sensor without under-exposing the shadows or over-exposing the highlights.

Extender
A device that extends the range of a wireless network by creating a second entry point, which may, or may not, merge with the main one.

F...

Facebook
Currently the most popular social networking site on the Internet; there are currently over 835 million registered users.

FaceTime
Apple's video calling service. Requires a Wi-Fi connection and is currently only supported via a phone number on iPhone and Apple ID email address on iPod touch 4 and Mac.

Factory Reset
An option on your Android phone that allows you to return it to the state it was when it left the factory.

File Format
The structure of how information is encoded in a computer file. File formats are designed to store specific types of information, such as JPEG and TIFF for image or raster data, AI for vector data or PDF for document exchange.

Folder
An icon representing a container for a group of apps, files or icons.

Force Quit
In the Fast App Switcher, tapping and holding an app will put it in 'jiggly mode' and tapping the x badge will force it to quit. Built-in apps like Mail and Messages will automatically restart while third-party apps will restart the next time you launch them.

Froyo
The nickname given to Android version 2.2.

G...

G1
The very first phone to run the Android operating system. Also known and the HTC Dream.

Game Center
Apple's gaming service, where you can discover new games and share your game experiences with friends from around the world.

Gamut
The range of colours and tonal values that can be produced by a capture or output device or represented by a colour space.

Galaxy
A range of hugely popular handsets from Samsung, the biggest smartphone manufacturer in the world.

Geotagging
The act of digitally attaching your location to photos taken on your phone.

Gingerbread
The nickname given to Android version 2.3.

Gmail
Google's web-based email software. Comes pre-installed on every Android smartphone.

Google
Owner (although not the original creator) of Android. Also own a fairly well known search engine...

Google Now
An enhanced Google search app which bases the information displayed on current location. Currently only found in Jelly Bean.

Google Play
Previously known as Android Market, this is where you go to download Android compatible apps, books, music and movies.

Gorilla Glass
Increasingly popular scratch-resistant glass used for smartphone displays.

GPS
Global Positioning System. A system that uses satellites to pinpoint your current location.

Grayscale
A monochromatic digital image file with pixel values that use shades of grey to represent tonal information. The term is often used to describe digital black and white photographs.

GSM
One of the two main cell phone communication standards. Used in most countries outside of the U.S.

H...

Hacking
Most often means rooting when talking about Android.

Hard Reset
Also called Factory Reset. Returns the phone to its post-factory state.

HDR (High Dynamic Range)
A process that combines multiple exposure variations of an image to achieve a dynamic range exceeding that of a single exposure. Algorithms are used to blend the exposures into a high-bit file format that can then be converted to either 8 or 16 bit for printing or web presentation.

Histogram
A graphical representation of the tone and colour distribution in a digital image. This is typically based on a particular colour or working space by plotting the number of pixels for each tone or colour value. It can be used to interpret photographic exposure and reveal shadow or highlight clipping.

Home Button
The physical hardware button on the front of early models of the iPhone, iPod touch, iPad and many Android devices, located just below the screen. It's used to wake the device, return to the Home Screen and several other functions.

Home Screen
The front end of your smartphone or tablet. The screen you see, containing app icons, widgets, etc., when you first unlock the device.

Honeycomb
The nickname given to Android version 3.0. The only version designed specifically for tablets, but now superseded by ICS.

HTC
A large Taiwanese smartphone manufacturer.

HTTP
Hypertext Transfer Protocol, the protocol used by the World Wide Web (Internet) that defines how messages are sent, received and read by browsers and other connected software layers.

HTTPS
Hypertext Transfer Protocol Secure, an encrypted and far more secure version of HTTP.

I...

Ice Cream Sandwich
The nickname given to Android version 4.0/4.1. The majority of new Android tablets now use this.

IMEI
International Mobile Equipment Identity. This is a unique identification number assigned to every phone.

Intel
Well known PC processor manufacturer. Has now started producing smartphone processors.

Internet
A global system of interconnected computers and networks which use the Internet Protocol Suite (TCP/IP) to link online devices.

Indentation
The coding language Python uses indentation to delimit blocks of code. The indents are four spaces apart, and are often created automatically after a colon is used in the code

iOS
Apple mobile operating system and the software that powers the iPhone, iPod touch, iPad and Apple TV.

IPS
In Plane Switching is a type of display used on some phones that increases the viewing angle of the screen.

I-Tunes
Mac and Windows music playing software, also used to activate and sync iPhone, iPod touch and iPad. It is also used to purchase and manage music, movies, TV shows, apps, books and other media.

ISO (International Organisation for Standardisation)
In photography, ISO refers to the standard for measurement of the sensitivity of film or digital sensors to light.

J...

Jelly Bean
The nickname given to Android 4.2, the latest version of the operating system.

JIT
The Just In Time compiler was introduced in Android 2.2. It helps to speed up apps on Android.

JPEG, JPG (Joint Photographic Experts Group)
A standard created by the Joint Photographic Experts Group for the compression of photographic images and the accompanying file format. It employs lossy compression that can reduce file size but at the expense of image quality and detail.

K...

Kernel
The basic Linux building block of Android.

Keyboard
Tablets and smartphones can feature either a physical or software keyboard.

Keyword
An element of metadata that is used to make a file more easily discoverable to searches. Keywords can be individual words or short phrases and can have a hierarchical structure.

L...

Landscape Mode
This describes a phone or tablet when you hold it horizontally; this is when it's wider than it is tall and the Home button is on the right or left of the screen.

Launcher
This is the part of the Android user interface on home screens that lets you launch apps and make phone calls.

LAN
Local Area Network. Devices that are connected to your router using Ethernet cables, are part of the LAN (see also WLAN).

LG
A large Korean electronics and smartphone manufacturer.

Linux
An open-source operating system that is used as the basis of Android.

Live Wallpapers
Animated wallpapers introduced in Android 2.1.

Loop
A piece of code that repeats itself until a certain condition is met. Loops can encase the entire code or just sections of it.

LTE
Long-Term Evolution. A name sometimes given to 4G data networks.

Luminance
The intensity of light as emitted or reflected by an object or surface. This is usually expressed in candelas per square meter (cd/m2). It is a measurement of the brightness of an object or light source.

M...

Magic
HTC phone also known as the MyTouch 3G. The first phone to use an Android operating system.

Mail
Built-in Apple app for handling POP3, IMAP, MobileMe and Exchange/ActiveSync email accounts.

Messages
One of Apple's built-in iPhone apps that handles SMS text messages and MMS multimedia messages. SMS messages are also more generally called "messages" on most devices.

MMS: (MultimediaMmessages)
MMS supports images, videos, sound, contact cards and location data. Sent and received via the Messages app.

Megapixel
A term used to describe digital camera resolution, 1 megapixel equals one million pixels or sensor elements. To calculate the megapixel value for a camera, multiply the horizontal by the vertical pixel counts of the recorded image.

Mesh
A means of combining two wireless access points into one, so they use the same settings and appear as a single network to devices that join it.

Metadata
Embedded or associated information describing a file's contents, used in digital photography to hold exposure information, GPS location data, copyright information and more. There are several metadata formats such as EXIF, IIM, IPTC Core, Dublin Core, DICOM and XMP.

Modem
Short for modulate-demodulate, a modem converts data into a signal that can be transferred over a phone line, and does so in reverse for incoming data.

Motorola
A large manufacturer or electronics and smartphones.

N...

News
Is an app in iOS that collected together magazine and newspaper apps and allowed the automatic downloading of new stories.

Nexus
A range of smartphones and tablets developed by Google. The Nexus range runs a pure version of Android.

NFC
Near Field Communication. A technology which allows data to be between phones or between your phone and another device.

Noise
The unwanted colour or luminance variations of pixels that degrade the overall quality of an image. Noise can result from several different sources including a low signal to noise ratio, the use of high ISO settings, long exposures, stuck sensor pixels and compression artefacts. It can appear as random colour speckles, a grain-like effect or banding.

Notification Centre:
A pull-down list of recent notifications, accessible from any iOS Home Screen or from within any iOS app. Similar to the Notification Panel found on Android.

O...

OEM
Original Equipment Manufacturer. A company which manufacturers devices for another brand (e.g. ASUS is the OEM of Google's Nexus 7.)

One Series
A range of smartphones from HTC. Includes the One X, One V and the One S.

Open GL
An open source graphics library, used on some smartphones.

Open Source
Software which is available to be studied, used and adapted by anyone. Android is open source software.

Operating System
Also OS. The program that's loaded into the computer after the initial boot sequence has completed. The OS manages all the other programs, graphical user interface (GUI), input and output and physical hardware interactions with the user.

Optimus
A range of smartphones from LG

OTA
Over The Air. A method which upgrades are wirelessly sent to smartphones.

Output
Data that is sent from the program to a screen, printer or other external peripheral.

P...

Pantech
A South Korean smartphone manufacturer.

PDF (Portable Document Format)
Developed by Adobe Systems, PDF is an open standard file format for cross-platform document exchange. PDF is highly extensible, preserves the integrity of the original document, is searchable and provides document security.

Photos
Built-in Apple app that handles your photo albums on your iPhone and iPod touch 4, and synced photos and videos for iPhone and all generations of iPad and iPod touch.

Photo Stream
Part of iCloud, Photo Stream stores your last thirty days or 1000 photos online and on your iOS devices, and all your photos on your Mac.

Pixel
Derived from the term picture element, this is the smallest unit of information in a digital image. It is also commonly used to describe the individual elements on a capture device such as a camera sensor.

PIN
Stands for Personal Identification Number. Used to lock smartphones and SIM cards.

Plug-In
A software application or module that provides extended and specific functionality from within a larger host application.

Portrait Mode
This describes a smartphone or tablet when you hold it vertically; this is when it's taller than it is wide and the Home button is at the top or bottom of the screen.

PSD
The .psd (Photoshop Document) format is a popular proprietary file format from Adobe Systems, Inc. It has support for most of the imaging options available in Photoshop, such as layer masks, transparency, text and alpha channels. In addition, spot colours, clipping paths and even duotone settings can be saved if you are preparing images for printing.

Project Butter
Software enhancements introduced in Android 4.1. Designed to smooth out frame rates and animations.

Q...

QR Code
A type of barcode which can be scanned by smartphones to reveal information such as text and website URL's.

QuickTime
Apple's 2D video and graphics player, used to play movies and other video on iOS.

R...

Raw Files
A Raw file is the unprocessed data captured by a digital camera sensor. In most cases, cameras write Raw files using a proprietary file format. Raw files give the photographer the advantage of managing image processing during post-production rather than letting the camera make the processing decisions, as happens when shooting in JPEG format. See also: DNG.

Recovery Mode
A separate operating mode of Android. Mainly used for device administration and repair.

Retina Display
Super sharp display available on Mac computers and iOS devices.

Resolution

A measurement of the ability of an optical, capture, or output system to record and reproduce detail. It can be defined in several different metrics such as Line Pairs, PPI, DPI, SPI and LPI. Also see DPI and PPI.

RGB
A colour model that uses the three primary additive colours (red, green, blue) that can be mixed in different ratios to make all other colours.

ROM
Read Only Memory. In Android a ROM is used to load software updates. Custom ROMs are software updates developed by third parties.

Root
In Android, to Root means to unlock the device to allow more access to the core software (or root).

Router
A device that manages and organises your home network devices, whether they connect to the router using a cable (LAN), or wirelessly (WLAN).

S...

Safari
Apple's web browser, both for Mac OS X and iOS (sometimes called Mobile Safari). Based on KHTML/ WebKit renderer and the Nitro JavaScript engine.

Samsung
A huge Korean smartphone and electronics manufacturer.

SD Card
A small memory card which can often be inserted into smartphones to increase storage capacity.

Sense
The user interface designed by and used on HTC phones.

Sharpening
The process of increasing or emphasising contrast around the edges of details in an image, to give the impression that the image is sharper than it really is.

Sideload
The process of installing an app onto your phone outside of the Google Play store.

SIM Card
The small plastic chip required in all GSM phones to connect to the mobile network.

Siri
Apple's intelligent virtual assistant, that replaces Voice Control on the iPhone.

Sleep/Wake Button
Physical hardware button. Used to power on, wake from sleep, put to sleep and power down most smartphones and tablets.

Sony Ericsson
The company formed by Sony and Ericsson to manufacture and distribute mobile devices.

Sprint
One of the large US mobile carriers.

SSID
Service Set ID. In a nutshell, this is the 'name' of your wireless network, and can be changed using your router.

Super AMOLED
An improvement of AMOLED displays, providing brighter, less power hungry and less reflective screens.

T...

T-Mobile
Large US mobile carrier and manufacturer of smartphones.

Tegra 2
NVIDIA's dual-core mobile processor.

Tegra 3
NVIDIA's newer, quad-core, mobile processor.

Tethering
Using your smartphones data connection to provide internet access for another device (laptops, etc.)

Text Field
Any area where you can add text. For example, the search field is where you type something you're looking for. Tap on a text field to bring up the virtual keyboard.

Thumbnail Image
A small, low-resolution image preview used on the web to link to a high-resolution version of the file. Thumbnails can also be embedded in file formats such as TIFF and PSD.

TIFF or TIF (Tagged Image File Format)
An open standard file format specifically designed for images. TIFF can incorporate several types of compression, including LZW, JPEG and ZIP. The format is suitable for the storage of high quality archive images. The DNG format is based on the main TIFF standard.

TouchWiz
Samsung's custom user interface.

Twitter
One of the most popular social networks built around a follower and following system rather than friends.

U...

UPnP
Universal Plug and Play. A protocol used by digital media players for enjoying video, music, and pictures over your home network.

URL
Uniform Resource Locator. This is a web address, used to access a web page on the Internet, and usually starts 'www' and ends in '.com', or some other top-level domain.

USB
Universal Serial Bus. The connection method now used by most smartphones to connect to a computer or power source (MicroUSB).

V...

Vanilla
Sometimes used to describe Android without any custom user interface applied.

VDSL
Very High Speed Digital Subscriber Line. It's another protocol for getting on the Internet using your phone line, and is sometimes shortened to DSL.

Verizon
One of the four large US mobile carriers.

Virtual Environment
A cooperatively isolated runtime environment that allows Python users and applications to install and upgrade Python distribution packages without interfering with the behaviour of other Python applications running on the same system.

Virtual Machine
A computer defined entirely in software. Can be used to test/run/create code that won't affect the host system.

VPN
(Virtual Private Network):
This provides secure access over the Internet to private networks, such as the network at your company or school.

W...

While Loop
A coding loop that repeats code while a comparative statement returns the value True.

White Balance (WB)
In digital photography, white balance establishes the colour balance of the image in relationship to colour temperature of the lighting conditions. Most digital cameras have several built-in white balance presets (tungsten, daylight, cloudy, fluorescent, etc.) along with an auto setting and the ability to set a custom WB.

Widgets
The name given to the home-screen gadgets which allow you to see app updates, news, etc.

Wi-Fi
A group of backwards-compatible radio technologies used to connect peripherals to a network wirelessly.

WLAN
Wireless Local Area Network. Your network of wireless devices, as opposed to devices connected with a cable (see LAN).

World Phone
A device which works on both CDMA and GSM networks outside of its home country.

WPS
Wi-Fi Protected Setup, an easier way of connecting wireless devices to your router.

XYZ...

Xperia
A range of smartphones developed by Sony Ericsson. Includes the Xperia T and the Xperia Play, the PlayStation smartphone.

YouTube
Google-owned, web-based video streaming service. A YouTube app is usually pre-installed on Android devices.