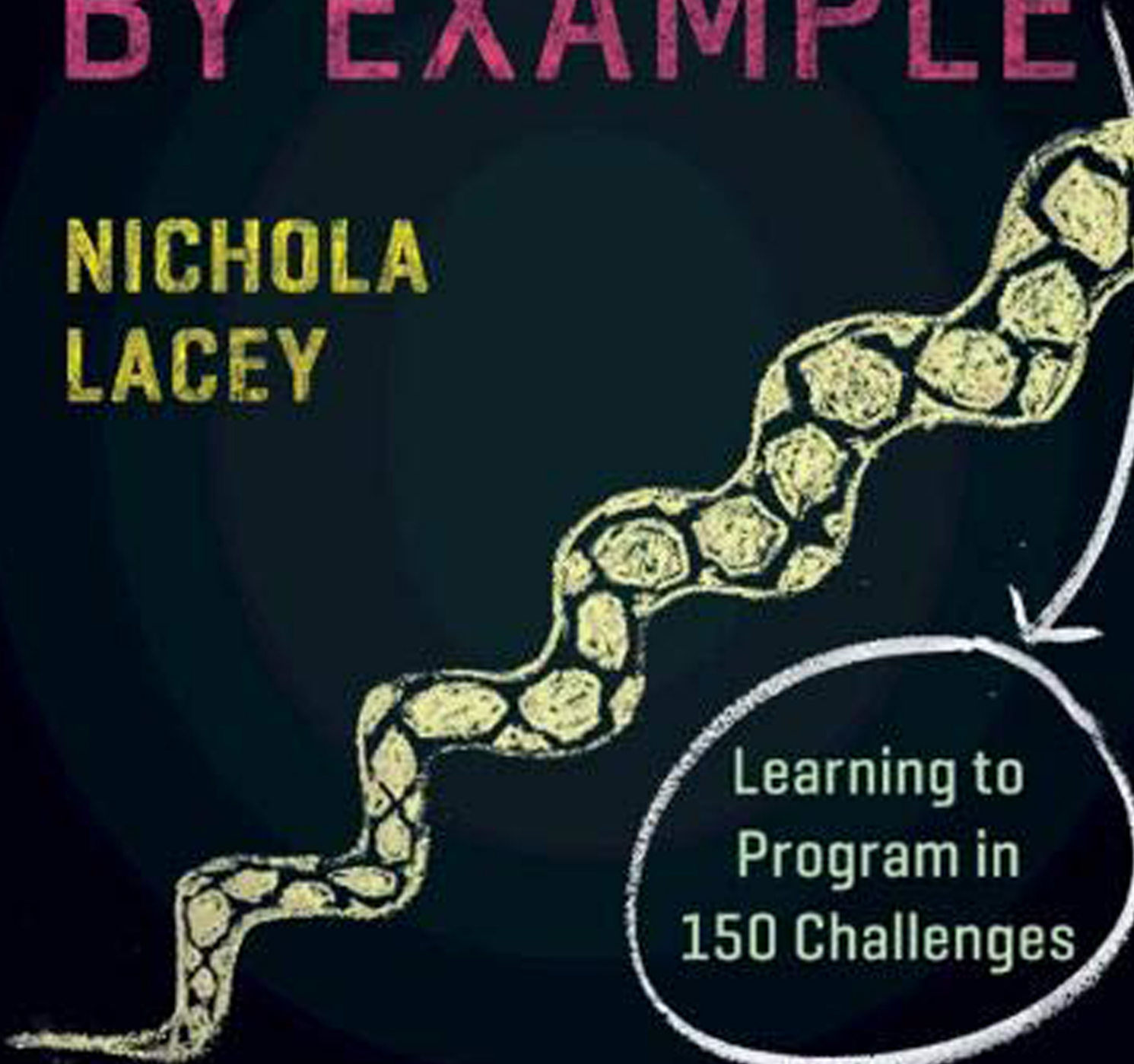


PYTHON BY EXAMPLE

NICHOLA
LACEY

A yellow and black patterned snake is drawn in a stylized, hand-drawn manner. It starts at the bottom left and moves towards the top right. A white arrow originates from the snake's head and points down to a white circle. Inside the circle, the text "Learning to Program in 150 Challenges" is written in white.

Learning to
Program in
150 Challenges

Python by Example

Learning to Program in 150 Challenges

Python is today's fastest growing programming language. This engaging and refreshingly different guide breaks down the skills into clear step-by-step chunks and explains the theory using brief easy-to-understand language. Rather than bamboozling readers with pages of mind-numbing technical jargon, this book includes 150 practical challenges, putting the power in the reader's hands. Through creating programs to solve these challenges the reader will quickly progress from mastering the basics to confidently using subroutines, a graphical user interface, and linking to external text, csv and SQL files. This book is perfect for anyone who wants to learn how to program with Python. In particular, students starting out in computer science and teachers who want to improve their confidence in Python will find here a set of ready-made challenges for classroom use.

NICHOLA LACEY is Director of Nichola Wilkin Ltd. She is a trusted source for teaching resources, having sold thousands of resources to schools around the world. As one of the most popular authors on TES, Nichola enjoys an extremely high review rating with hundreds of thousands of downloads. She was a programmer before moving into corporate training and then retraining as a teacher, and she gained a unique skill set of programming and practical classroom experience after being promoted to head of computer science in a private boys' school.

PYTHON BY EXAMPLE

Learning to Program in 150 Challenges

NICHOLA LACEY

Nichola Wilkin Ltd



CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre, New Delhi – 110025, India

79 Anson Road, #06-04/06, Singapore 079906

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning, and research at the highest international levels of excellence.

www.cambridge.org

Information on this title: www.cambridge.org/9781108716833

DOI: 10.1017/9781108591942

© Nichola Lacey 2019

This publication is in copyright. Subject to statutory exception and to the provisions of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published 2019

Printed in Singapore by Markono Print Media Pte Ltd

A catalogue record for this publication is available from the British Library.

ISBN 978-1-108-71683-3 Paperback

Cambridge University Press has no responsibility for the persistence or accuracy of URLs for external or third-party internet websites referred to in this publication and does not guarantee that any content on such websites is, or will remain, accurate or appropriate.

Contents

<i>Image Credits</i>	vi
<i>Introduction</i>	1
<i>Downloading Python</i>	4
<i>Some Tips</i>	6

Part I: Learning Python

Challenges 1 - 11: The Basics	11
Challenges 12 - 19: If Statements	17
Challenges 20 - 26: Strings	24
Challenges 27 - 34: Maths	31
Challenges 35 - 44: For Loop	35
Challenges 45 - 51: While Loop	40
Challenges 52 - 59: Random	45
Challenges 60 - 68: Turtle Graphics	51
Challenges 69 - 79: Tuples, Lists and Dictionaries	58
Challenges 80 - 87: More String Manipulation	67
Challenges 88 - 95: Numeric Arrays	72
Challenges 96 - 103: 2D Lists and Dictionaries	79
Challenges 105 - 110: Reading and Writing to a Text File	86
Challenges 111 - 117: Reading and Writing to a .csv File	91
Challenges 118 - 123: Subprograms	99
Challenges 124 - 132: Tkinter GUI	110
Challenges 133 - 138: More Tkinter	124
Challenges 139 - 145: SQLite	134

Part II: Chunky Challenges

Challenge 146: Shift Code	150
Challenge 147: Mastermind	153
Challenge 148: Passwords	156
Challenge 149: Times Table (GUI)	161
Challenge 150: Art Gallery	164

<i>What Next?</i>	169
<i>Glossary</i>	170
<i>Index</i>	182



Image Credits

Animal Drawings:

Pages 1, 9, 11, 31, 35, 37, 41, 45, 92, 125, 127, 138, 150, 165: HelenField/Shutterstock.com

Pages 16, 20, 94 and 141(bottom): Victoria Novak/Shutterstock.com

Pages 27 and 157: Dimonika/Shutterstock.com

Pages 46, 58, 73, 93, 95, 103, 128, 135, 147, 169: mart/Shutterstock.com

Pages 59, 68, 151, 154: lynea/Shutterstock.com

Page 80: MoreVector/Shutterstock.com

All other animal drawings: Olga_Angelloz/Shutterstock.com

Other decorative icons:

MG Drachal/Shutterstock.com

Mila Petkova/Shutterstock.com

Nikolaeva/Shutterstock.com

Tiwat K/Shutterstock.com

Introduction



If you have ever picked up a programming manual and felt your forehead go clammy and your eyes cross as you attempt to make sense of the long-winded explanations, this is the guide for you.

I have been in your position, attempting to learn how to program and having to rely on the traditional style of guides. I know from painful experience how quickly I glaze over and my brain solidifies; after only a few pages the tedium leaves me blindly reading words without any real notion of what they mean any more. Inevitably I give up and the whole process makes me feel like a limp failure, gasping for breath after I surface from drowning in technical jargon.

I hated having to read through pointless drivel and then be presented with a short program telling me exactly what to type in and then spend the next 20 pages reading about what I have just done and the 101 ways I could run it. I hated having no control over trying things out for myself and I hated the way these guides would only contain one or two challenges at the end of a chapter of theory.



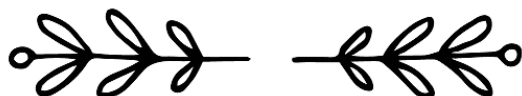
I knew there had to be a better way, and thankfully there is. I wrote it and you are presently reading it, so aren't you lucky? This guide is refreshingly different and helps you learn how to program with Python by using practical examples rather than self-important explanations.



Many programmers learn through experimentation, looking at others' code and working out what method is best for a given situation. This book is a hands-on approach to learning programming. After minimal reading you are set a number of challenges to create the programs. You can explore and experiment with the programming language and look at the example solutions to learn how to think like a programmer. There are no chapters entitled "the architecture of a computer", "the theory of programming" or any other gobbledy-gook other authors like to waste time with. I don't want to baffle you with theory or blind you with overbearing explanations that suck out your enthusiasm for learning to program.

Hopefully, you want to get stuck into creating programs, solving problems and enjoying the sense of accomplishment that you get as you proudly look over your lines of code, knowing that you created something that works. That is great, your eagerness is to be applauded and I salute those who are reading this while already sitting at their computers, fingers poised and ready to get going. If that is the case, that you already have Python open on your screen and are itching to get going, then away you go and I'll see you in the first chapter called "The Basics" on page 11.

For everyone who is still with us and is feeling a little more timid, there are just a few more things to tell you about before you take the plunge.



How to Use This Book

This book builds from very simple programs to more complex ones. If you are new to programming or new to Python, start with "The Basics" and work through the chapters in order.

If you are familiar with Python programming and feel confident with the basics, the theory and logic surrounding programming, then you can just dip in and out of the book to get help on the specifics you need.



The book is split into two sections:

Part I

In Part I, each chapter takes you through some basic programming rules and challenges for you to complete and includes:

- a **simple explanation** giving you pointers, which is useful if you are new to programming in Python;
- **examples of code** with a short explanation, which you can use as a basis to solve the challenges;
- a **list of challenges** for you to work through that get harder as you move through them. Each challenge should only take between a couple of minutes and 20 minutes to solve; however, some of the more complex challenges near the end of Part I will take longer as you build up the techniques you will be using. Don't panic if you take longer than this, as long as you solve the problems without *too* much copying from the suggested solution, you are doing fine;
- code containing a **possible solution** for each challenge; there is often more than one answer available, but we include just a single program as a possible solution that you can refer to if you get stuck on a particular aspect of the code.

Part II

In Part II, you are given some larger challenges which utilize the programming skills you learnt in Part I and allow you to consolidate and reinforce the techniques you have been practising. In this section, you are not given the help and example code that is given in Part I and it will take longer to solve each challenge. After each challenge, you are given one possible answer that you may find useful if you are stuck. However, you may have found another solution that works just as well.



Who Is This Book For?

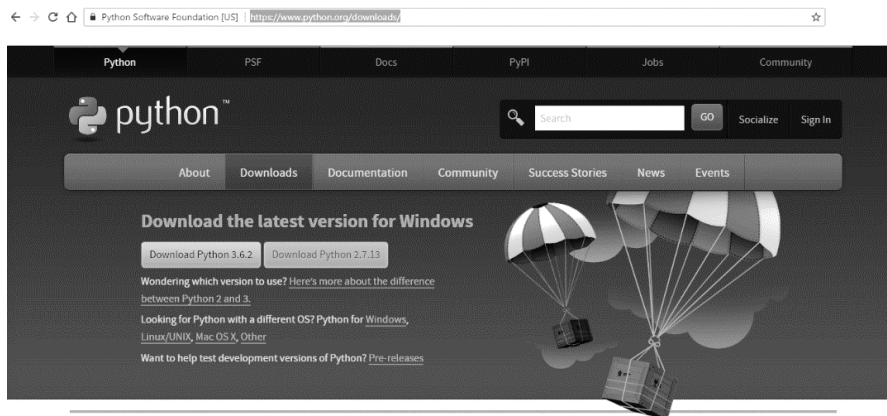
This book is suitable for anyone who wants to learn how to program with Python. It is an essential tool for teachers and students in Key Stage 3 or those studying computer science who need help and ready-made examples to practise programming techniques and build confidence. It can also be used to help with a computer science programming project resource bank, to help pupils needing additional support or just a quick reminder of the syntax when creating programs.



Downloading Python

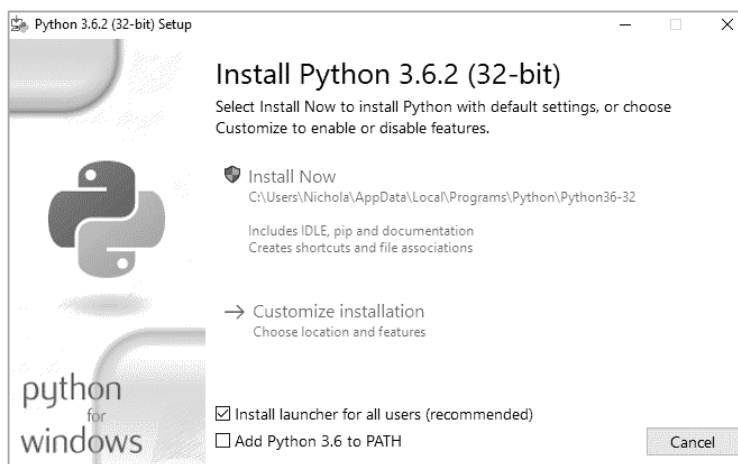
You can download Python for free from the official Python website:

www.python.org/downloads/



Click on the latest version (in the example above, click on the **Download Python 3.6.2** button) to start the installation.

The program will download an executable (.exe) file. When you run this program, you will see an install window like the one shown below.

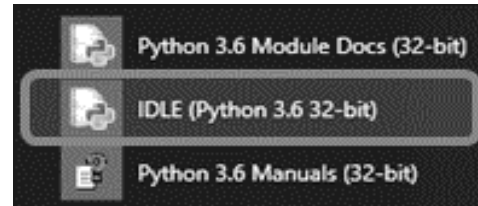
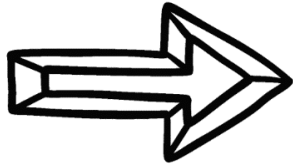


Click the **Install Now** option and the program will start installing Python onto your system.



Running Python

To start Python on a Windows system, click on the **Windows** icon or **Start** menu and select the **IDLE (Python version number)** option as highlighted below.



Some Tips

File Location

On a Windows system, the Python folder is usually found in the C:\ drive and will be named Python36 (or similar) and the files will automatically be saved in the same location, unless you save them specifically in another location.



Using Comments

Comments are a very useful tool for programmers. They serve two purposes:

- adding an explanation of how the program works;
- stopping parts of the program from working temporarily so you can run and test other parts of the program.



The first, and original, purpose of explaining how a program works is so other programmers can make sense of your programs in case your code needs to be altered and updated in the future and to remind you about why you wrote particular lines of code.

```
print("This is a simple program")
print() #Outputs a blank line to help with layout
name = input("Please input your name: ")#Asks for an input
print("Hello", name) #Joins "Hello" and their name together
```

In this example, comments have been added at the end of the last three lines. They are shown in red and start with the # symbol.

In reality, you would not add comments on lines which contain obvious code as it would clutter the screen; you would only add comments where necessary.

As Python knows to ignore anything after a # symbol, programmers soon started to use # at the start of lines of their code to block out sections they do not want to run so they can focus on and test others.

```
#print("This is a simple program")
print()
name = input("Please input your name: ")
print("Hello", name)
```

In this example, the # has been added to the first line of the program to temporarily stop it from running. To bring it back into the running order simply delete the # and the code will be reactivated.



In this guide, we have not included any comments to the programs so you have to read the code to make sense of it. That way you will really learn how to code! If you are creating programs as part of your coursework you should add comments to explain your programming to the examiner.



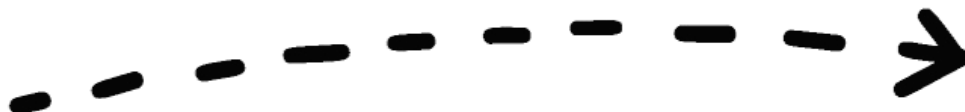
Formatting Python

In most versions of Python IDLE it is possible to quickly add comments and indent code using the menus. This way, if you need to block out entire areas using a comment you simply highlight the lines and then select the **Format** menu and select **Comment Out Region**. Similarly, if you need to indent a region (we will look at the reason for indenting code later) then you can also easily do this with the menu.



File	Edit	Format	Run	Options	Windows	Help
		Indent Region			Ctrl+]	
		Dedent Region			Ctrl+[
		Comment Out Region			Alt+3	
		Uncomment Region			Alt+4	

Okay, that is all the "housekeeping" out of the way. No more procrastinating; take a deep breath and away we go...



Part 1

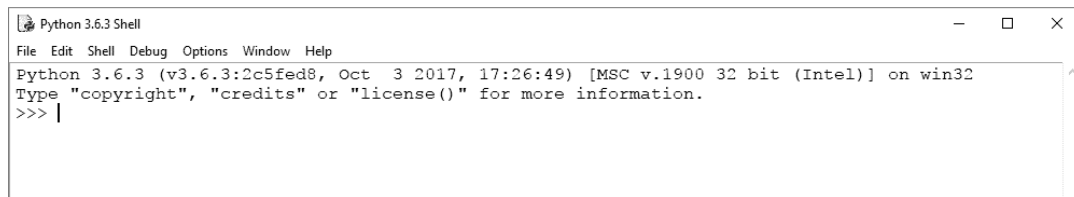
Learning Python



The Basics

Explanation

This is the **shell** window and is the first screen you see when you launch Python.



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
```

It is possible to write Python code straight into the shell, but as soon as you hit [Return] at the end of a line, it will run that line of code. This may be suitable for using Python as a quick calculator; for instance, you can type in **3*5** at the prompt and Python will show the answer **15** on the next line; however, this style of inputting is not useful for more complex programs.

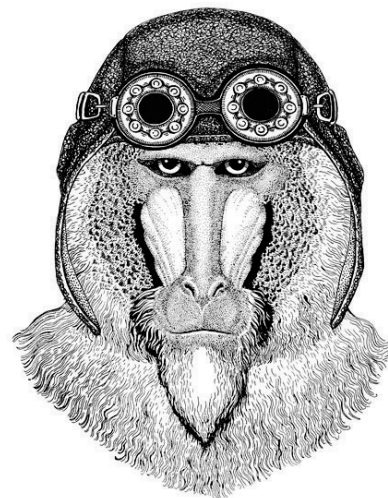


It is much better to start a new window, create all the code in the new window, save your code and run it.

To create a new window in which to write your code, select **File** and **New**.

Once you enter your code in this new window you can save it and run it all in one go. This will then run the code in the shell window.

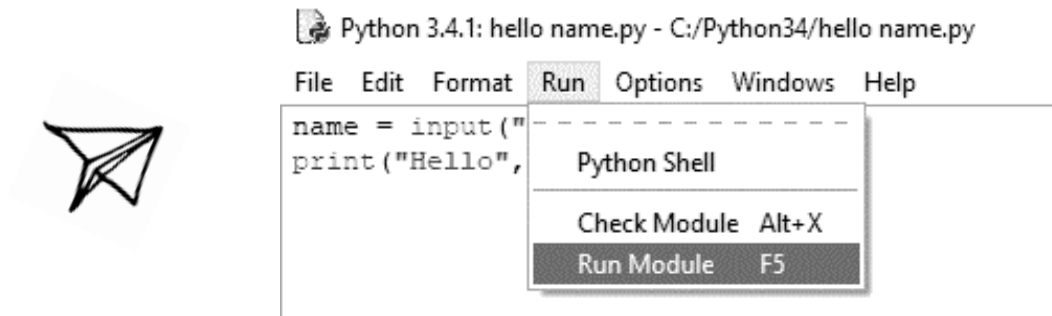
Alternatively, Python programs can be written using any text editor and must be saved with the file name extension **.py** in order to work. These programs can then be run from the command prompt by typing in the full directory root and file name.



Running Your Program

Every time you run the code your program will need to be saved afresh in case there have been any changes to it.

In this version of Python, you can run the program by selecting the **Run** menu and selecting **Run Module**. Alternatively, you can press the **[F5]** key. If this is the first time the program is saved, Python will prompt you to name and save the file before it will allow the program to run.



Important Things to Note When Writing Your Programs

Python is case sensitive so it is important that you use the correct case, otherwise your code will not work.



Text values need to appear in speech marks (") but numbers do not.

When naming **variables** (i.e. values that you want to store data in) you cannot use any dedicated words such as print, input, etc. otherwise your code will not work.

When saving your files **do not save them with any dedicated words** that Python already uses, such as print, input, etc. If you do this it will not run and you will need to rename the file before it works.

To edit a program you have saved and closed, right-click on the file and select **Edit with IDLE**. If you just double-click on the file it will only try to run it and you will not be able to edit it.

Example Code

```
num1 = 93
```

Set the value of a **variable**, if there is not a variable already created, it will create one. A variable is a container for a value (in this case the variable will be called "num1" and store the value 93). The value stored in the variable can change while the program is running. The variable can be called whatever you want (except Python dedicated words such as print, save, etc.) and it must start with a letter rather than a number or symbol and have no spaces.



1+2=?

```
answer = num1 + num2
```

Adds together num1 and num2 and stores the answer in a variable called answer.

```
answer = num1 - num2
```

Subtracts num2 from num1 and stores the answer in a variable called answer.

```
answer = num1 * num2
```

Multiplies num1 by num2 and stores the answer in a variable called answer.

```
answer = num1 / num2
```

Divides num1 by num2 and stores the answer in a variable called answer.

```
answer = num1 // num2
```

A whole number division (i.e. $9//4 = 2$) and stores the answer in a variable called answer.

```
print ("This is a message")
```

Displays the message in the brackets. As the value we want displayed is a text value it has the speech marks, which will not be displayed in the output. If you wanted to display a numerical value or the contents of a variable, the speech marks are not needed.

```
print ("First line\nSecond line")
```

"\n" is used as a line break.

```
print ("The answer is", answer)
```

Displays the text "The answer is" and the value of the variable answer.

```
textValue = input("Enter a text value: ")
```

Displays the question "Enter a text value: " and stores the value the user enters in a variable called textValue. The space after the colon allows a space to be added before the user enters their answer, otherwise they appear squashed unattractively together.

```
numValue = int(input("Enter a number: "))
```

Displays the question "Enter a number: " and stores the value as an integer (a whole number) in a variable called numValue. Integers can be used in calculations but variables stored as text cannot.



Challenges

001

Ask for the user's first name and display the output message **Hello [First Name]**.

002

Ask for the user's first name and then ask for their surname and display the output message **Hello [First Name] [Surname]**.



003

Write code that will display the joke "What do you call a bear with no teeth?" and on the next line display the answer "A gummy bear!" Try to create it using only one line of code.

004

Ask the user to enter two numbers. Add them together and display the answer as **The total is [answer]**.

005

Ask the user to enter three numbers. Add together the first two numbers and then multiply this total by the third. Display the answer as **The answer is [answer]**.

006

Ask how many slices of pizza the user started with and ask how many slices they have eaten. Work out how many slices they have left and display the answer in a user-friendly format.

007

Ask the user for their name and their age. Add 1 to their age and display the output **[Name] next birthday you will be [new age]**.

008

Ask for the total price of the bill, then ask how many diners there are. Divide the total bill by the number of diners and show how much each person must pay.

009

Write a program that will ask for a number of days and then will show how many hours, minutes and seconds are in that number of days.

Keep going, you are doing well.



010

There are 2,204 pounds in a kilogram. Ask the user to enter a weight in kilograms and convert it to pounds.

011

Task the user to enter a number over 100 and then enter a number under 10 and tell them how many times the smaller number goes into the larger number in a user-friendly format.

Answers

001

```
firstname = input("Please enter your first name: ")
print ("Hello",firstname)
```

002

```
firstname = input("Please enter your first name: ")
surname = input("Please enter your surname: ")
print ("Hello",firstname, surname)
```

003

```
print("What do you call a bear with no teeth?\nA gummy bear!")
```

004

```
num1 = int(input("Please enter your first number: "))
num2 = int(input("Please enter your second number: "))
answer = num1 + num2
print("The answer is", answer)
```

005

```
num1 = int(input("Please enter your first number: "))
num2 = int(input("Please enter your second number: "))
num3 = int(input("Please enter your third number: "))
answer = (num1 + num2) * num3
print("The answer is", answer)
```

006

```
startNum = int(input("Enter the number of slices of pizza you started with: "))
endNum = int(input("How many slices have you eaten? "))
slicesLeft = startNum - endNum
print("You have", slicesLeft, "slices remaining")
```

007

```
name = input("What is your name? ")
age = int(input("How old are you? "))
newAge = age + 1
print(name, "next birthday you will be", newAge)
```

008

```
bill = int(input("What is the total cost of the bill? "))
people = int(input("How many people are there? "))
each = bill/people
print("Each person should pay £", each)
```

009

```
days = int(input("Enter the number of days: "))
hours = days*24
minutes = hours*60
seconds = minutes*60
print("In", days, "days there are...")
print(hours, "hours")
print(minutes, "minutes")
print(seconds, "seconds")
```

010

```
kilo = int(input("Enter the number of kilos: "))
pound = kilo * 2.204
print("That is", pound, "pounds")
```

011

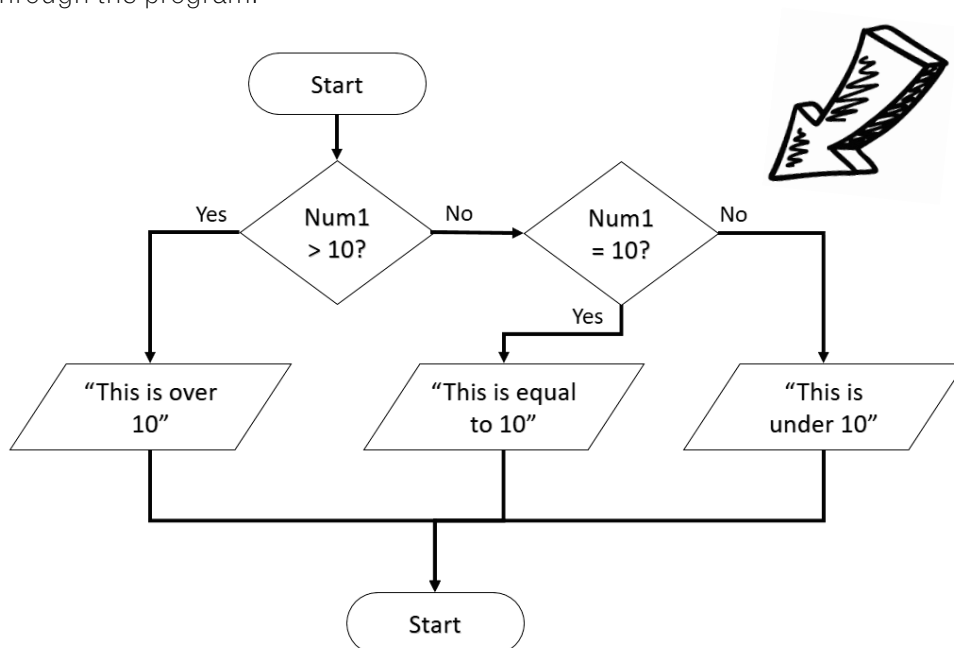
```
larger = int(input("Enter a number over 100: "))
smaller = int(input("Enter a number under 10: "))
answer = larger//smaller
print(smaller, "goes into", larger, answer, "times")
```



If Statements

Explanation

If statements allow your program to make a decision and change the route that is taken through the program.



Below is how the if statement for this flow chart would look in Python.



```
if num1 > 10:
    print("This is over 10")
elif num1 == 10:
    print("This is equal to 10")
else:
    print("This is under 10")
```



Indenting Lines of Code

Indenting is very important in Python as it shows the lines that are dependent on others, as shown in the example on the previous page. In order to indent text you can use your **[Tab]** key or you can press your **[space key]** five times. The **[backspace]** key will remove indents.

The first line of the if statement tests a condition, and if that condition is met (i.e. the first condition is true) then the lines of code directly below it are run. If it is not met (i.e. the first condition is false) it will test the second condition, if there is one, and so on. Below are examples of the different comparison and logical operators you can use in the condition line of your if statement.

Comparison Operators

Operator	Description
<code>==</code>	Equal to
<code>!=</code>	Not equal to
<code>></code>	Greater than
<code><</code>	Less than
<code>>=</code>	Greater than or equal to
<code><=</code>	Less than or equal to

Logical Operators

Operator	Description
<code>and</code>	Both conditions must be met
<code>or</code>	Either condition must be met



Example Code

Please note: In the examples shown, num is a variable entered by the user that has been stored as an integer.



```
if num > 10:
    print("This is over 10")
else:
    print("This is not over 10")
```

If num1 is over 10, it will display the message "This is over 10", otherwise it will display the message "This is under 10".

```
if num > 10:
    print("This is over 10")
elif num == 10:
    print("This is equal to 10")
else:
    print("This is under 10")
```

If num1 is over 10, it will display the message "This is over 10", otherwise it will check the next condition. If num1 is equal to 10, it will display the message "This is equal to 10". Otherwise, if neither of the first two conditions have been met, it will display the message "This is under 10".



```
if num >= 10:
    if num <= 20:
        print("This is between 10 and 20")
    else:
        print("This is over 20")
else:
    print("This is under 10")
```

If num1 is 10 or more then it will test another if statement to see if num1 is less than or equal to 20. If it is, it will display the message "This is between 10 and 20". If num1 is not less than or equal to 20 then it will display the message "This is over 20". If num1 is not over 10, it will display the message "This is under 10".



```
text = str.lower(text)
```

Changes the text to lower case. As Python is case sensitive, this changes the data input by the user into lower case so it is easier to check.

```
num = int(input("Enter a number between 10 and 20: "))
if num >= 10 and num <= 20:
    print("Thank you")
else:
    print("Out of range")
```

This uses **and** to test multiple conditions in the if statement. Both the conditions must be met to produce the output "Thank you".

```
num = int(input("Enter an EVEN number between 1 and 5: "))
if num == 2 or num == 4:
    print("Thank you")
else:
    print("Incorrect")
```

This uses **or** to test the conditions in the if statement. Just one condition must be met to display the output "Thank you".



Challenges

012

Ask for two numbers. If the first one is larger than the second, display the second number first and then the first number, otherwise show the first number first and then the second.

013

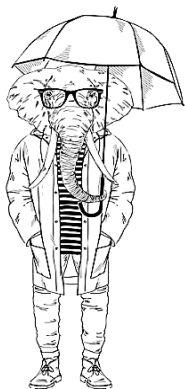
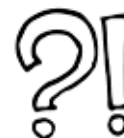
Ask the user to enter a number that is under 20. If they enter a number that is 20 or more, display the message "Too high", otherwise display "Thank you".

014

Ask the user to enter a number between 10 and 20 (inclusive). If they enter a number within this range, display the message "Thank you", otherwise display the message "Incorrect answer".

015

Ask the user to enter their favourite colour. If they enter "red", "RED" or "Red" display the message "I like red too", otherwise display the message "I don't like [colour], I prefer red".

**016**

Ask the user if it is raining and convert their answer to lower case so it doesn't matter what case they type it in. If they answer "yes", ask if it is windy. If they answer "yes" to this second question, display the answer "It is too windy for an umbrella", otherwise display the message "Take an umbrella". If they did not answer yes to the first question, display the answer "Enjoy your day".

017

Ask the user's age. If they are 18 or over, display the message "You can vote", if they are aged 17, display the message "You can learn to drive", if they are 16, display the message "You can buy a lottery ticket", if they are under 16, display the message "You can go Trick-or-Treating".

018

Ask the user to enter a number. If it is under 10, display the message "Too low", if their number is between 10 and 20, display "Correct", otherwise display "Too high".

019

Ask the user to enter 1, 2 or 3. If they enter a 1, display the message "Thank you", if they enter a 2, display "Well done", if they enter a 3, display "Correct". If they enter anything else, display "Error message".

Answers

012

```
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
if num1 > num2:
    print(num2, num1)
else:
    print(num1, num2)
```

013

```
num = int(input("Enter a value less than 20: "))
if num >= 20:
    print("Too high")
else:
    print("Thank you")
```

014

```
num = int(input("Enter a value between 10 and 20: "))
if num >= 10 and num <= 20:
    print("Thank you")
else:
    print("Incorrect answer")
```

015

```
colour = input("Type in your favourite colour: ")
if colour == "red" or colour == "RED" or colour == "Red":
    print("I like red too.")
else:
    print("I don't like that colour, I prefer red")
```

016

```
raining = input("Is it raining? ")
raining = str.lower(raining)
if raining == "yes":
    windy = input("Is it windy? ")
    windy = str.lower(windy)
    if windy == "yes":
        print("It is too windy for an umberella")
    else:
        print("Take an umberella")
else:
    print("Enjoy your day")
```

017

```
age = int(input("What is your age? "))
if age >= 18:
    print ("You can vote")
elif age == 17:
    print ("You can learn to drive")
elif age == 16:
    print ("You can buy a lottery ticket")
else:
    print ("You can go Trick-or-Treating")
```

018

```
num = int(input("Enter a number: "))
if num <10:
    print("Too low")
elif num >=10 and num <=20:
    print("Correct")
else:
    print("Too high")
```

019

```
num = input("Enter 1, 2 or 3: ")
if num == "1":
    print("Thank you")
elif num == "2":
    print("Well done")
elif num == "3":
    print("Correct")
else:
    print("Error message")
```

Strings

Explanation

String is the technical name for text. To define a block of code as a string, you need to include it in either double quotes (") or single quotes ('). It doesn't matter which you use so long as you are consistent.

There are some characters you need to be particularly careful with when inputting them into strings. These include:

" '\

That is because these symbols have special meanings in Python and it can get confusing if you use them in a string.

If you want to use one of these symbols you need to precede it with a backslash symbol and then Python will know to ignore the symbol and will treat it as normal text that is to be displayed.

Symbol	How to type this into a Python string
"	\"
'	\'
\	\\



Strings and Numbers as Variables

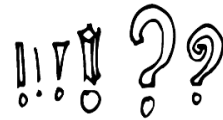
If you define a variable as a string, even if it only contains numbers, you cannot later use that string as part of a calculation. If you want to use a variable that has been defined as a string in a calculation, you will have to convert the string to a number before it can be used.

```
num = input("Enter a number: ")
total = num + 10
print(total)
```

In this example, the author has asked for a number, but has not defined it as a numeric value and when the program is run they will get the following error:

```
Enter a number: 45
Traceback (most recent call last):
  File "C:/Python34/CHALLENGES/String/example.py", line 2, in <module>
    total = num + 10
TypeError: Can't convert 'int' object to str implicitly
>>>
```

Although this error message looks scary, it is simply saying that the line **total = num + 10** isn't working as the variable num is defined as a string.



This problem can be solved in one of two ways. You can either define it as a number when the variable is being originally created, using this line:

```
num = int(input("Enter a number: "))
```

or you can convert it to a number after it has been created using this line:

```
num = int(num)
```

The same can happen with strings.

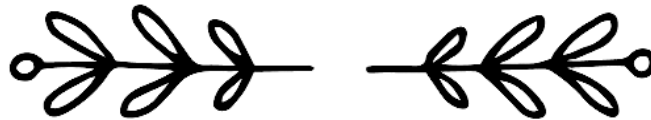
```
name = input("Enter a name: ")
num = int(input("Enter a number: "))
ID = name + num
print(ID)
```

In this program, the user is asked to enter their name and a number. They want it joined together and with strings the addition symbol is used as **concatenation**. When this code is run you will get a similar error message to before:

```
Enter a name: Bob
Enter a number: 23
Traceback (most recent call last):
  File "C:/Python34/CHALLENGES/String/example.py", line 3, in <module>
    ID = name + num
TypeError: Can't convert 'int' object to str implicitly
>>>
```

To get around this, either don't define the variable as a number in the first place or convert it to a string afterwards using the line:

```
num = str(num)
```



Multiple-Line Strings

If you want to input a string across multiple lines you can either use the line break (`\n`) or you can enclose the entire thing in triple quotes. This will preserve the formatting of the text.

```
address="""123 Long Lane
Oldtown
AB1 23CD"""
print(address)
```



Example Code

Please note: In the following examples, the terms word, phrase, name, firstname and surname are all variable names.

len (word)

Finds the length of the variable called word.

word.upper ()

Changes the string into upper case.

print (word.capitalize ())

Displays the variable so only the first word has a capital letter at the beginning and everything else is in lower case.

word.lower ()

Changes the string into lower case.

name = firstname+surname

Joins the first name and surname together without a space between them, known as concatenation

word.title ()

Changes a phrase so that every word has a capital letter at the beginning with the rest of the letters in the word in lower case (i.e.Title Case).

```
text = " This is some text. "
```

```
print (text.strip (" "))
```

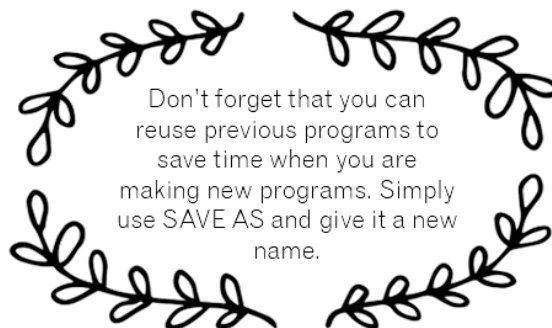
Removes extra characters (in this case spaces) from the start and end of a string.

```
print ("Hello world"[7:10])
```

Each letter is assigned an index number to identify its position in the phrase, including the space. Python starts counting from 0, not 1.

0	1	2	3	4	5	6	7	8	9	10
H	e	l	l	o		w	o	r	l	d

Therefore, in this example, it would display the value of positions 7, 8 and 9, which is "orl".



Challenges

020

Ask the user to enter their first name and then display the length of their name.



021

Ask the user to enter their first name and then ask them to enter their surname. Join them together with a space between and display the name and the length of whole name.

022

Ask the user to enter their first name and surname in lower case. Change the case to title case and join them together. Display the finished result.

023

Ask the user to type in the first line of a nursery rhyme and display the length of the string. Ask for a starting number and an ending number and then display just that section of the text (remember Python starts counting from 0 and not 1).



024

Ask the user to type in any word and display it in upper case.

025

Ask the user to enter their first name. If the length of their first name is under five characters, ask them to enter their surname and join them together (without a space) and display the name in upper case. If the length of the first name is five or more characters, display their first name in lower case.

Don't forget, you can always look back, remind yourself of some of the earlier skills you have learnt. You have learnt a great deal so far.

026

Pig Latin takes the first consonant of a word, moves it to the end of the word and adds on an "ay". If a word begins with a vowel you just add "way" to the end. For example, pig becomes igpay, banana becomes anabay, and aadvark becomes aadvarkway. Create a program that will ask the user to enter a word and change it into Pig Latin. Make sure the new word is displayed in lower case.



Answers

020

```
name = input("Enter your first name: ")
length = len(name)
print(length)
```

021

```
firstname = input("Enter your first name: ")
surname = input("Enter your surname: ")
name = firstname + " " + surname
length = len(name)
print(name)
print(length)
```

022

```
firstname = input("Enter your first name in lowercase: ")
surname = input("Enter your surname in lowercase: ")
firstname = firstname.title()
surname = surname.title()
name = firstname + " " + surname
print(name)
```

023

```
phrase = input("Enter the first line of a nursery rhyme: ")
length = len(phrase)
print("This has", length, "letters in it")
start = int(input("Enter a starting number: "))
end = int(input("Enter an end number: "))
part = (phrase[start:end])
print(part)
```

024

```
word = input("Enter a word: ")
word = word.upper()
print(word)
```

025

```
name = input("Enter your first name: ")
if len(name) < 5:
    surname = input("Enter your surname: ")
    name = name + surname
    print(name.upper())
else:
    print(name.lower())
```

026

```
word = input("Please enter a word: ")
first = word[0]
length = len(word)
rest = word[1:length]
if first != "a" and first != "e" and first != "i" and first != "o" and first != "u":
    newword = rest + first + "ay"
else:
    newword = word + "way"
print(newword.lower())
```

Maths

Explanation

Python can perform several mathematical functions, but these are only available when the data is treated as either an **integer** (a whole number) or a **floating-point** (a number with a decimal place). If data is stored as a string, even if it only contains numeric characters, Python is unable to perform calculations with it (see page 24 for a fuller explanation).



Example Code

Please note: In order to use some of the mathematical functions (`math.sqrt(num)` and `math.pi`) you will need to import the maths library at the start of your program. You do this by typing `import math` as the first line of your program.

```
print(round(num,2))
```

Displays a number rounded to two decimal places.

```
**
```

To the power of (e.g. 10^2 is 10^{**2}).

```
math.sqrt(num)
```

The square root of a number, but you must have the line `import math` at the top of your program for this to work.

```
num=float(input("Enter number: "))
```

Allows numbers with a decimal point dividing the integer and fraction part.

```
math.pi
```

Gives you pi (π) to 15 decimal places, but you must have the line `import math` at the top of your program for this to work.

```
x // y
```

Whole number division (e.g. $15//2$ gives the answer 7).

```
x % y
```

Finds the remainder (e.g. $15\%2$ gives the answer 1).



Challenges

027

Ask the user to enter a number with lots of decimal places. Multiply this number by two and display the answer.

028

Update program 027 so that it will display the answer to two decimal places.

029

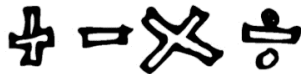
Ask the user to enter an integer that is over 500. Work out the square root of that number and display it to two decimal places.

030

Display pi (π) to five decimal places.

031

Ask the user to enter the radius of a circle (measurement from the centre point to the edge). Work out the area of the circle ($\pi \times \text{radius}^2$).



032

Ask for the radius and the depth of a cylinder and work out the total volume (circle area \times depth) rounded to three decimal places.

033

Ask the user to enter two numbers. Use whole number division to divide the first number by the second and also work out the remainder and display the answer in a user-friendly way (e.g. if they enter 7 and 2 display "7 divided by 2 is 3 with 1 remaining").



034

Display the following message:

- 1) Square
- 2) Triangle

Enter a number:

If the user enters 1, then it should ask them for the length of one of its sides and display the area. If they select 2, it should ask for the base and height of the triangle and display the area. If they type in anything else, it should give them a suitable error message.

You are starting to think like a programmer.



Answers

027

```
num = float(input("Enter a number with lots of decimal places: "))
print(num*2)
```

028

```
num = float(input("Enter a number with lots of decimal places: "))
answer = num*2
print(answer)
print(round(answer, 2))
```

029

```
import math
num = int(input("Enter a number over 500: "))
answer = math.sqrt(num)
print(round(answer, 2))
```

030

```
import math
print(round(math.pi, 5))
```

031

```
import math
radius = int(input("Enter the radius of the circle: "))
area = math.pi*(radius**2)
print(area)
```

032

```
import math
radius = int(input("Enter the radius of the circle: "))
depth = int(input("Enter depth: "))
area = math.pi*(radius**2)
volume = area*depth
print(round(volume, 3))
```

033

```
num1=int(input("Enter a number: "))
num2=int(input("Enter another number: "))
ans1 = num1//num2
ans2 = num1%num2
print(num1, "divided by", num2, "is", ans1, "with", ans2, "remaining.")
```

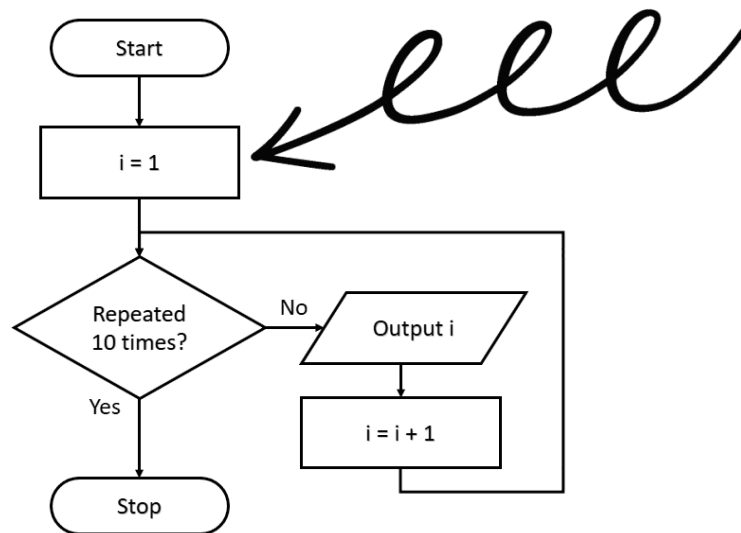
034

```
print("1) Square")
print("2) Triangle")
print()
menuselection = int(input("Enter a number: "))
if menuselection == 1:
    side = int(input("Enter the length of one side: "))
    area = side*side
    print("The area of your chosen shape is", area)
elif menuselection == 2:
    base = int(input("Enter the length of the base: "))
    height = int(input("Enter the height of the triangle: "))
    area = (base*height)/2
    print("The area of your chosen shape is", area)
else:
    print("Incorrect option selected")
```

For Loop

Explanation

A **for loop** allows Python to keep repeating code a set number of times. It is sometimes known as a **counting loop** because you know the number of times the loop will run before it starts.



In this case, it starts at 1 and will keep repeating the loop (displaying i) until it reaches 10 and then stops. This is how this loop would look in Python

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

In this example, the outputs would be 1, 2, 3, 4, 5, 6, 7, 8 and 9.

When it gets to 10 the loop would stop so 10 would not be shown in the output.

Remember to indent the lines of code within the for loop.



Example Code

The range function is often used in for loops and lists the starting number, the ending number and can also include the steps (e.g. counting in 1s, 5s or any other value you wish).

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

This loop uses a variable called "i" to keep track of the number of times the loop has been repeated. It will start i at 1 (as that is the starting value in the range function) and repeat the loop, adding 1 to i each time and displaying the value of i until it reaches 10 (as dictated by the range function), where it will stop. Therefore, it will not repeat the loop a tenth time and will only have the following output:

1, 2, 3, 4, 5, 6, 7, 8, 9



```
for i in range(1,10,2):
    print(i)
```

This range function includes a third value which shows how much is added to i in each loop (in this case 2). The output will therefore be: **1, 3, 5, 7, 9**

```
for i in range(10,1,-3):
    print(i)
```

This range will subtract 3 from i each time. The output will be: **10, 7, 4**



Using loops is a powerful programming tool that you will use a lot in the more challenging programs.

```
for i in word:
    print(i)
```

This would display each character in a string called "word" as a separate output (i.e. on a separate line).



Challenges

035

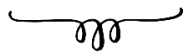
Ask the user to enter their name and then display their name three times.

038

Change program 037 to also ask for a number. Display their name (one letter at a time on each line) and repeat this for the number of times they entered.

041

Ask the user to enter their name and a number. If the number is less than 10, then display their name that number of times; otherwise display the message "Too high" three times.



036

Alter program 035 so that it will ask the user to enter their name and a number and then display their name that number of times.

037

Ask the user to enter their name and display each letter in their name on a separate line.

039

Ask the user to enter a number between 1 and 12 and then display the times table for that number.



040

Ask for a number below 50 and then count down from 50 to that number, making sure you show the number they entered in the output.

042

Set a variable called total to 0. Ask the user to enter five numbers and after each input ask them if they want that number included. If they do, then add the number to the total. If they do not want it included, don't add it to the total. After they have entered all five numbers, display the total.

043

Ask which direction the user wants to count (up or down). If they select up, then ask them for the top number and then count from 1 to that number. If they select down, ask them to enter a number below 20 and then count down from 20 to that number. If they entered something other than up or down, display the message "I don't understand".

044

Ask how many people the user wants to invite to a party. If they enter a number below 10, ask for the names and after each name display "[name] has been invited". If they enter a number which is 10 or higher, display the message "Too many people".

Answers

035

```
name = input("Type in your name: ")
for i in range (0,3):
    print(name)
```

036

```
name = input("Type in your name: ")
number = int(input("Enter a number: "))
for i in range (0,number):
    print(name)
```

037

```
name = input("Enter your name: ")
for i in name:
    print(i)
```

038

```
num = int(input("Enter a number: "))
name = input("Enter your name: ")
for x in range(0,num):
    for i in name:
        print(i)
```

039

```
num = int(input("Enter a number between 1 and 12: "))
for i in range(1, 13):
    answer = i * num
    print (i, "x", num, "=", answer)
```

040

```
num = int(input("Enter a number below 50: "))
for i in range(50,num-1, -1):
    print(i)
```

041

```
name = input("Enter your name: ")
num = int(input("Enter a number: "))
if num < 10:
    for i in range(0, num):
        print(name)
else:
    for i in range(0, 3):
        print("Too high")
```

042

```
total = 0
for i in range(0, 5):
    num = int(input("Enter a number: "))
    ans = input("Do you want this number included? (y/n) ")
    if ans == "y":
        total = total + num
print(total)
```

043

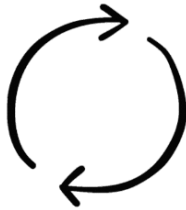
```
direction = input("Do you want to count up or down? (u/d) ")
if direction == "u":
    num = int(input("What is the top number? "))
    for i in range(1, num+1):
        print(i)
elif direction == "d":
    num = int(input("Enter a number below 20: "))
    for i in range(20, num-1, -1):
        print(i)
else:
    print("I don't understand")
```

044

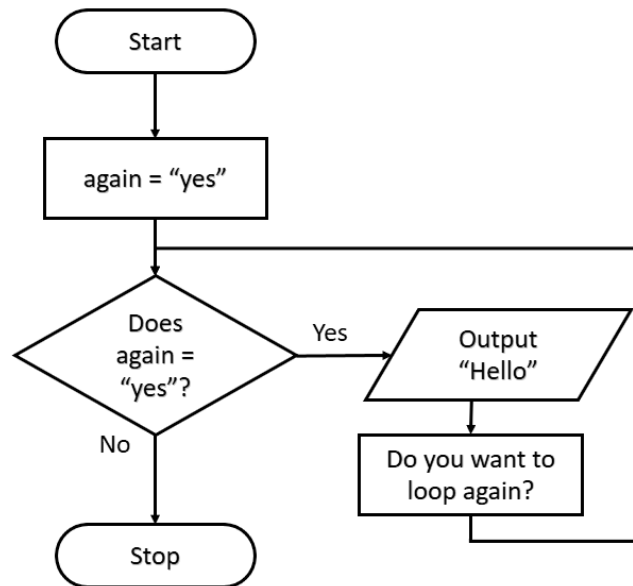
```
num = int(input("How many friends do you want to invite to the party? "))
if num < 10:
    for i in range(0, num):
        name = input("Enter a name: ")
        print(name, "has been invited")
else:
    print("Too many people")
```

While Loop

Explanation



A **while loop** allows code to be repeated an unknown number of times as long as a condition is being met. This may be 100 times, just the once or even never. In a while loop the condition is checked before the code is run, which means it could skip the loop altogether if the condition is not being met to start with. It is important, therefore, to make sure the correct conditions are in place to run the loop before it starts.



In Python the example for the flow chart above would look as follows:

```
again = "yes"
while again == "yes":
    print ("Hello")
    again=input("Do you want to loop again? ")
```

It will keep repeating this code until the user enters anything other than "yes".

Example Code



```
total = 0
while total < 100:
    num = int(input("Enter a number: "))
    total = total + num
print("The total is", total)
```

The above program will create a variable called total and store the value as 0. It will ask the user to enter a number and will add it to the total. It will keep repeating this as long as the total is still below 100. When the total equals 100 or more, it will stop running the loop and display the total.



Comparison Operators

Operator	Description
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Logical Operators

Operator	Description
and	Both conditions must be met
or	Either condition must be met

Remember: text values must appear in speech marks and numeric values do not.



Challenges

045

Set the total to 0 to start with. While the total is 50 or less, ask the user to input a number. Add that number to the total and print the message "The total is... [total]". Stop the loop when the total is *over 50*.

047

Ask the user to enter a number and then enter another number. Add these two numbers together and then ask if they want to add another number. If they enter "y", ask them to enter another number and keep adding numbers until they do not answer "y". Once the loop has stopped, display the total.

049

Create a variable called `compnum` and set the value to 50. Ask the user to enter a number. While their guess is not the same as the `compnum` value, tell them if their guess is too low or too high and ask them to have another guess. If they enter the same value as `compnum`, display the message "Well done, you took [count] attempts".



048

Ask for the name of somebody the user wants to invite to a party. After this, display the message "[name] has now been invited" and add 1 to the count. Then ask if they want to invite somebody else. Keep repeating this until they no longer want to invite anyone else to the party and then display how many people they have coming to the party.

050

Ask the user to enter a number between 10 and 20. If they enter a value under 10, display the message "Too low" and ask them to try again. If they enter a value above 20, display the message "Too high" and ask them to try again. Keep repeating this until they enter a value that is between 10 and 20 and then display the message "Thank you".



046

Ask the user to enter a number. Keep asking until they enter a value *over 5* and then display the message "The last number you entered was a [number]" and stop the program.

051

Using the song "10 green bottles", display the lines "There are [num] green bottles hanging on the wall, [num] green bottles hanging on the wall, and if 1 green bottle should accidentally fall". Then ask the question "how many green bottles will be hanging on the wall?" If the user answers correctly, display the message "There will be [num] green bottles hanging on the wall". If they answer incorrectly, display the message "No, try again" until they get it right. When the number of green bottles gets down to 0, display the message "There are no more green bottles hanging on the wall".

Answers

045

```
total = 0
while total <= 50:
    num = int(input("Enter a number: "))
    total = total + num
    print("The total is...",total)
```

046

```
num = 0
while num <= 5:
    num = int(input("Enter a number: "))
print("The last number you entered was a", num)
```

047

```
num1 = int(input("Enter a number: "))
total = num1
again = "y"
while again == "y":
    num2 = int(input("Enter another number: "))
    total = total + num2
    again = input("Do you want to add another number? (y/n) ")
print("The total is ",total)
```

048

```
again = "y"
count = 0
while again == "y":
    name = input("Enter a name of somebody you want to invite to your party: ")
    print(name, "has now been invited")
    count = count + 1
    again = input("Do you want to invite somebody else? (y/n) ")
print("You have", count, "people coming to your party")
```

049

```
compnum = 50
guess = int(input("Can you guess the number I am thinking of? "))
count = 1
while guess != compnum:
    if guess < compnum:
        print("Too low")
    else:
        print("Too high")
    count = count+1
    guess = int(input("Have another guess: "))
print("Well done, you took", count, "attempts")
```

050

```
num = int(input("Enter a number between 10 and 20: "))
while num <10 or num >20:
    if num <10:
        print("Too low")
    else:
        print("Too high")
    num = int(input("Try again: "))
print("Thank you")
```

051

```
num = 10
while num >0:
    print("There are ", num, "green bottles hanging on the wall.")
    print( num, "green bottles hanging on the wall.")
    print("And if 1 green bottle should accidentally fall,")
    num = num - 1
    answer = int(input("How many green bottles will be hanging on the wall? "))
    if answer == num:
        print("There will be", num, "green bottles hanging on the wall.")
    else:
        while answer!=num:
            answer = int(input("No, try again: "))
print("There are no more green bottles hanging on the wall.")
```

Random

Explanation

Python can generate **random** values. In reality, the values are not completely random as no computer can cope with that; instead it uses an incredibly complex algorithm that makes it virtually impossible to accurately predict its outcome so, in effect, it acts like a random function.



There are two types of random value that we will be looking at:

- Random numbers within a specified range;
- A random choice from a range of items that are input.



To use these two options, you will need to import the random library. You do this by typing **import random** at the start of your program.



Example Code

```
import random
```

This must appear at the start of your program otherwise the random function will not work.

```
num = random.random()
```

Selects a random floating-point number between 0 and 1 and stores it in a variable called "num". If you want to obtain a larger number, you can multiply it as shown below:

```
import random  
num = random.random()  
num = num * 100  
print(num)
```

```
num = random.randint(0,9)
```

Selects a random whole number between 0 and 9 (inclusive).



You
are
doing
great!

```
num1 = random.randint(0,1000)  
num2 = random.randint(0,1000)  
newrand = num1/num2  
print(newrand)
```

Creates a random floating-point number by creating two random integers within two large ranges (in this case between 0 and 1000) and dividing one by the other.

```
num = random.randrange(0,100,5)
```

Picks a random number between the numbers 0 and 100 (inclusive) in steps of five, i.e. it will only pick from 0, 5, 10, 15, 20, etc.

```
colour = random.choice(["red", "black", "green"])
```

Picks a random value from the options "red", "black" or "green" and stores it as the variable "colour". Remember: strings need to include speech marks but numeric data does not.



Challenges

052

Display a random integer between 1 and 100 inclusive.

053

Display a random fruit from a list of five fruits.

054

Randomly choose either heads or tails ("h" or "t"). Ask the user to make their choice. If their choice is the same as the randomly selected value, display the message "You win", otherwise display "Bad luck". At the end, tell the user if the computer selected heads or tails.



055

Randomly choose a number between 1 and 5. Ask the user to pick a number. If they guess correctly, display the message "Well done", otherwise tell them if they are too high or too low and ask them to pick a second number. If they guess correctly on their second guess, display "Correct", otherwise display "You lose".

056

Randomly pick a whole number between 1 and 10. Ask the user to enter a number and keep entering numbers until they enter the number that was randomly picked.



057

Update program 056 so that it tells the user if they are too high or too low before they pick again.

058

Make a maths quiz that asks five questions by randomly generating two whole numbers to make the question (e.g. $[num1] + [num2]$). Ask the user to enter the answer. If they get it right add a point to their score. At the end of the quiz, tell them how many they got correct out of five.



059

Display five colours and ask the user to pick one. If they pick the same as the program has chosen, say "Well done", otherwise display a witty answer which involves the correct colour, e.g. "I bet you are GREEN with envy" or "You are probably feeling BLUE right now". Ask them to guess again; if they have still not got it right, keep giving them the same clue and ask the user to enter a colour until they guess it correctly.



Answers

052

```
import random
num = random.randint(1,100)
print(num)
```

053

```
import random
fruit = random.choice( ['apple', 'orange', 'grape', 'banana', 'strawberry'] )
print(fruit)
```

054

```
import random
coin = random.choice( ["h", "t"] )
guess = input("Enter (h)eads or (t)ails: ")
if guess == coin:
    print("You win")
else:
    print("Bad luck")
if coin == "h":
    print("It was heads")
else:
    print("It was tails")
```

055

```
import random
num = random.randint(1,5)
guess = int(input("Enter a number: "))
if guess == num:
    print("Well done")
elif guess > num:
    print("Too high")
    guess = int(input("Guess again: "))
    if guess == num:
        print("Correct")
    else:
        print("You lose")
elif guess < num:
    print("Too low")
    guess = int(input("Guess again: "))
    if guess == num:
        print("Correct")
    else:
        print("You lose")
```


056

```
import random
num = random.randint(1,10)
correct = False
while correct == False:
    guess = int(input("Enter a number: "))
    if guess == num:
        correct = True
```

057

```
import random
num = random.randint(1,10)
correct = False
while correct == False:
    guess = int(input("Enter a number: "))
    if guess == num:
        correct = True
    elif guess > num:
        print("Too high")
    else:
        print("Too low")
```

058

```
import random
score = 0
for i in range(1,6):
    num1 = random.randint(1,50)
    num2 = random.randint(1,50)
    correct = num1 + num2
    print(num1, "+", num2, "= ")
    answer = int(input("Your answer: "))
    print()
    if answer == correct:
        score = score + 1
print("You scored", score, "out of 5")
```

059

```
import random

colour = random.choice(["red", "blue", "green", "white", "pink"])
print("Select from red, blue, green, white or pink")
tryagain = True
while tryagain == True:
    theirchoice = input("Enter a colour: ")
    theirchoice = theirchoice.lower()
    if colour == theirchoice:
        print("Well done")
        tryagain = False
    else:
        if colour == "red":
            print("I bet you are seeing RED right now!")
        elif colour == "blue":
            print("Don't feel BLUE.")
        elif colour == "green":
            print("I bet you are GREEN with envy right now.")
        elif colour == "white":
            print("Are you WHITE as a sheet, as you didn't guess correctly?")
        elif colour == "pink":
            print("Shame you are not feeling in the PINK, as you got it wrong!")
```

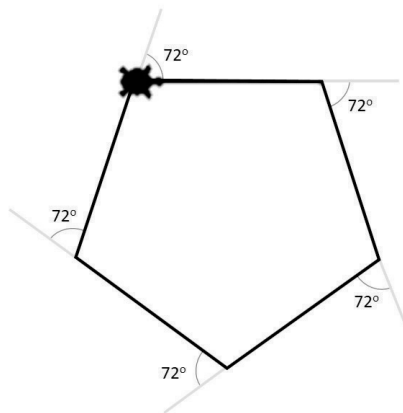
Turtle Graphics

Explanation

It is possible to draw using a **turtle** in Python. By typing in commands and using loops, you can create intricate patterns. Here is how it works.



A turtle will travel along a path that you define, leaving a pen mark behind it. As you control the turtle, the pattern that is left is revealed. To draw the pentagon shown below you would type in the following code.



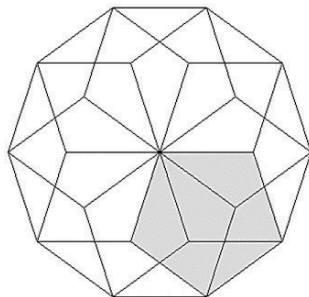
```
import turtle

turtle.shape("turtle")

for i in range(0,5):
    turtle.forward(100)
    turtle.right(72)

turtle.exitonclick()
```

By combining these simple shapes and using **nested** loops (i.e. loops inside other loops) it is possible to create beautiful patterns very easily.



```
import turtle

for i in range(0,10):
    turtle.right(36)
    for i in range(0,5):
        turtle.forward(100)
        turtle.right(72)

turtle.exitonclick()
```

In the above pattern, one pentagon has been repeatedly drawn 10 times, rotating 36 degrees around a central point. **Please note:** we have highlighted one of the pentagons to help you identify it within the pattern, but it would not usually be highlighted.

Example Code

`import turtle`

This line needs to be included at the beginning of your program to import the turtle library into Python, allowing you to use the turtle functions.

`scr = turtle.Screen()`

Defines the window as being called "scr". This means we can use the shorthand "scr", rather than having to refer to the window by its full name each time.



`scr.bgcolor("yellow")`

Sets the screen background colour to yellow. By default, the background colour will be white unless it is changed.

`turtle.penup()`

Removes the pen from the page so that as the turtle moves it does not leave a trail behind it.

`turtle.pendown()`

Places the pen on the page so that when the turtle moves it will leave a trail behind it. By default, the pen is down unless specified otherwise.

`turtle.pensize(3)`

Changes the turtle pen size (the thickness of the line that is drawn) to 3. By default, this is 1 unless it is changed.

`turtle.left(120)`

Turns the turtle 120° to the left (counter clockwise).

`turtle.right(90)`

Turns the turtle 90° to the right (clockwise).

`turtle.forward(50)`

Moves the turtle forward 50 steps.



`turtle.shape("turtle")`

Changes the shape of the turtle to look like a turtle 🐢. By default, the turtle will look like a small arrow.

`turtle.hideturtle()`

Hides the turtle so it is not showing on the screen.

`turtle.begin_fill()`

Entered before the code that draws a shape so it knows to fill in the shape it is drawing.

`turtle.showturtle()`

Shows the turtle on the screen. By default, the turtle is showing unless specified otherwise.

`turtle.end_fill()`

Entered after the code that is drawing the shape to tell Python to stop filling in the shape.

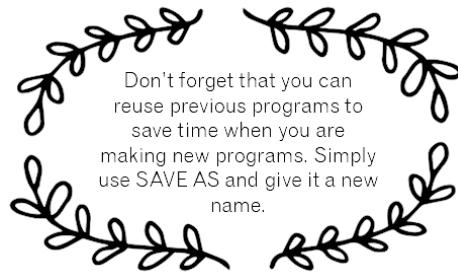
`turtle.color("black", "red")`

Defines the colours filling in the shape. This example will make the shape have a black outline and a red fill. This needs to be entered before the shape is drawn.

`turtle.exitonclick()`

When the user clicks on the turtle window it will automatically close.

Challenges



060
Draw a square.

061
Draw a triangle.

062
Draw a circle.

063
Draw three squares in a row with a gap between each. Fill them using three different colours.



065
Write the numbers as shown below, starting at the bottom of the number one.

064
Draw a five-pointed star.



066
Draw an octagon that uses a different colour (randomly selected from a list of six possible colours) for each line.

067
Create the following pattern:

068
Draw a pattern that will change each time the program is run. Use the random function to pick the number of lines, the length of each line and the angle of each turn.



Answers

060

```
import turtle

for i in range(0,4):
    turtle.forward(100)
    turtle.right(90)

turtle.exitonclick()
```

061

```
import turtle

for i in range(0,3):
    turtle.forward(100)
    turtle.left(120)

turtle.exitonclick()
```

062

```
import turtle

for i in range(0,360):
    turtle.forward(1)
    turtle.right(1)

turtle.exitonclick()
```

063

```
import turtle

turtle.color("black", "red")
turtle.begin_fill()
for i in range (0,4):
    turtle.forward(70)
    turtle.right(90)
turtle.penup()
turtle.end_fill()
turtle.forward(100)

turtle.pendown()
turtle.color("black", "yellow")
turtle.begin_fill()
for i in range (0,4):
    turtle.forward(70)
    turtle.right(90)
turtle.penup()
turtle.end_fill()
turtle.forward(100)

turtle.pendown()
turtle.color("black", "green")
turtle.begin_fill()
for i in range (0,4):
    turtle.forward(70)
    turtle.right(90)
turtle.end_fill()

turtle.exitonclick()
```

064

```
import turtle

for i in range (0,5):
    turtle.forward(100)
    turtle.right(144)

turtle.exitonclick()
```

065

```
import turtle

turtle.left(90)
turtle.forward(100)
turtle.right(90)
turtle.penup()
turtle.forward(50)
turtle.pendown()
turtle.forward(75)
turtle.right(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(75)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
turtle.forward(75)
turtle.penup()
turtle.forward(50)
turtle.pendown()
turtle.forward(75)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
turtle.forward(45)
turtle.left(180)
turtle.forward(45)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
turtle.forward(75)

turtle.hideturtle()

turtle.exitonclick()
```


066

```
import turtle
import random

turtle.pensize(3)

for i in range(0,8):
    turtle.color(random.choice(["red","blue","yellow","green","pink","orange"]))
    turtle.forward(50)
    turtle.right(45)

turtle.exitonclick()
```

067

```
import turtle
import random

for x in range(0,10):
    for i in range(0,8):
        turtle.forward(50)
        turtle.right(45)
    turtle.right(36)

turtle.hideturtle()

turtle.exitonclick()
```

068

```
import turtle
import random

lines = random.randint(5,20)

for x in range(0,lines):
    length = random.randint(25,100)
    rotate = random.randint(1,365)
    turtle.forward(length)
    turtle.right(rotate)

turtle.exitonclick()
```

Tuples, Lists and Dictionaries

Explanation

So far, we have used variables that can store a single item of data in them. When you used the `random.choice(["red", "blue", "green"])` line of code you are picking a random item from a list of possible options. This demonstrates that one item can hold several pieces of separate data, in this case a collection of colours.

There are several ways that collections of data can be stored as a single item. Three of the simpler ones are:

- tuples
- lists
- dictionaries



Tuples

Once a **tuple** is defined you cannot change what is stored in it. This means that when you write the program you must state what the data is that is being stored in the tuple and the data cannot be altered while the program is running. Tuples are usually used for menu items that would not need to be changed.

Lists

The contents of a **list** can be changed while the program is running and lists are one of the most common ways to store a collection of data under one variable name in Python. The data in a list does not all have to be of the same data type. For example, the same list can store both strings and integers; however, this can cause problems later and is therefore not recommended.



Please note: In other programming languages the term **array** is often used to describe a variable that contains a collection of data, and these work in a similar way to

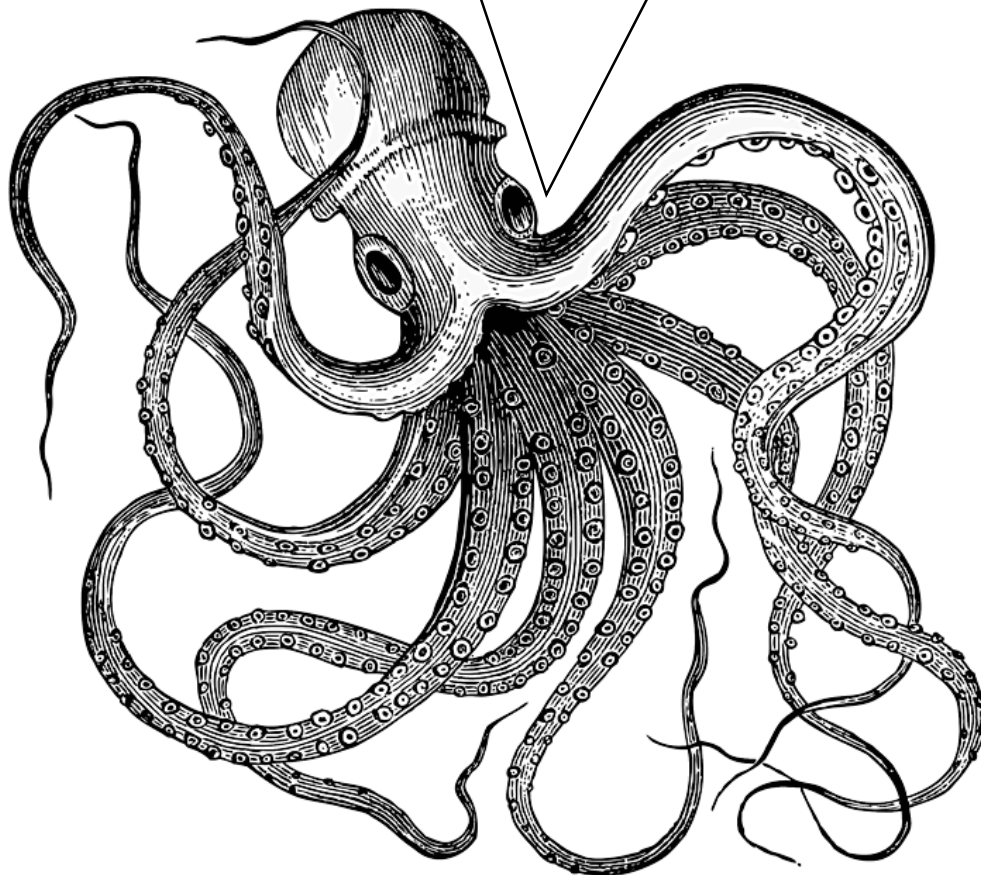
lists in Python. There is a data type called an array in Python, but this is only used to store numbers and we will look at Python numeric arrays on page 72.

Dictionaries

The contents of a **dictionary** can also be changed while the program is running. Each value is given an index or key you can define to help identify each piece of data. This index will not change if other rows of data are added or deleted, unlike lists where the position of the items can change and therefore their index number will also change.



Don't get yourself in a tangle, take each program and break it into the parts you already know from previous programs and build in the new skills you are learning.



Example Code

```
fruit_tuple = ("apple", "banana", "strawberry", "orange")
```

Creates a variable name called "fruit_tuple" which stores four pieces of fruit within it. The round brackets define this group as a tuple and therefore the contents of this collection of data cannot be altered while the program is running.

```
print(fruit_tuple.index("strawberry"))
```

Displays the index (i.e. the numeric key) of the item "strawberry". In this example it will return the number 2 as Python starts counting the items from 0, not 1.



```
print(fruit_tuple[2])
```

Displays item 2 from "fruit_tuple", in this case "strawberry".

```
names_list = ["John", "Tim", "Sam"]
```

Creates a list of the names and stores them in the variable "names_list". The square brackets define this group of data as a list and therefore the contents can be altered while the program is running.

```
del names_list[1]
```

Deletes item 1 from "names_list". Remember it starts counting from 0 and not 1. In this case it will delete "Tim" from the list.

```
names_list.append(input("Add a name: "))
```

Asks the user to enter a name and will add that to the end of "names_list".

```
print(sorted(names_list))
```

Displays names_list in alphabetical order but does not change the order of the original list, which is still saved in the original order. This does not work if the list is storing data of different types, such as strings and numeric data in the same list.



```
names_list.sort()
```

Sorts name_list into alphabetical order and saves the list in the new order. This does not work if the list is storing data of different types, such as strings and numeric data in the same list.

```
colours = {1:"red", 2:"blue", 3:"green"}
```

Creates a dictionary called "colours", where each item is assigned an index of your choosing. The first item in each block is the index, separated by a colon and then the colour.



```
colours[2] = "yellow"
```

Makes a change to the data stored in position 2 of the colours dictionary. In this case it will change "blue" to "yellow".

As lists are one of the most common data structures we include more example code just for lists.

```
x = [154, 634, 892, 345, 341, 43]
```

Here we have created a list that contains numbers. **Please note:** as it contains numeric data, no speech marks are required.

```
print(len(x))
```

Displays the length of the list (i.e. how many items are in the list).

```
print(x[1:4])
```

This will display data in positions 1, 2 and 3. In this case 634, 892 and 345. Remember, Python starts counting from 0 and will stop when it gets to the last position, without showing the final value.

```
for i in x:
```

```
    print(i)
```

Uses the items in the list in a for loop, useful if you want to print the items in a list on separate lines.

```
num = int(input("Enter number: "))  
if num in x:  
    print(num, "is in the list")  
else:  
    print("Not in the list")
```

Asks the user to enter a number and checks whether the number is in the list and displays an appropriate message.

```
x.insert(2, 420)
```

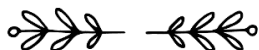
Inserts the number 420 into position 2 and pushes everything else along to make space. This will change the index numbers of the items in the list.

```
x.remove(892)
```

Deletes an item from the list. This is useful if you do not know the index of that item. If there is more than one instance of the data it will only delete the first instance.

```
x.append(993)
```

Adds the number 993 to the end of the list.



Challenges

069

Create a tuple containing the names of five countries and display the whole tuple. Ask the user to enter one of the countries that have been shown to them and then display the index number (i.e. position in the list) of that item in the tuple.

070

Add to program 069 to ask the user to enter a number and display the country in that position.

071

Create a list of two sports. Ask the user what their favourite sport is and add this to the end of the list. Sort the list and display it.



072

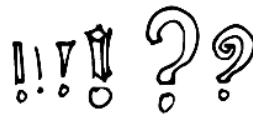
Create a list of six school subjects. Ask the user which of these subjects they don't like. Delete the subject they have chosen from the list before you display the list again.

073

Ask the user to enter four of their favourite foods and store them in a dictionary so that they are indexed with numbers starting from 1. Display the dictionary in full, showing the index number and the item. Ask them which they want to get rid of and remove it from the list. Sort the remaining data and display the dictionary.

074

Enter a list of ten colours. Ask the user for a starting number between 0 and 4 and an end number between 5 and 9. Display the list for those colours between the start and end numbers the user input.



075

Create a list of four three-digit numbers. Display the list to the user, showing each item from the list on a separate line. Ask the user to enter a three-digit number. If the number they have typed in matches one in the list, display the position of that number in the list, otherwise display the message "That is not in the list".

076

Ask the user to enter the names of three people they want to invite to a party and store them in a list. After they have entered all three names, ask them if they want to add another. If they do, allow them to add more names until they answer "no". When they answer "no", display how many people they have invited to the party.

077

Change program 076 so that once the user has completed their list of names, display the full list and ask them to type in one of the names on the list. Display the position of that name in the list. Ask the user if they still want that person to come to the party. If they answer "no", delete that entry from the list and display the list again.

You are over halfway there. Keep going, you have already learnt so much.

**078**

Create a list containing the titles of four TV programmes and display them on separate lines. Ask the user to enter another show and a position they want it inserted into the list. Display the list again, showing all five TV programmes in their new positions.

079

Create an empty list called "nums". Ask the user to enter numbers. After each number is entered, add it to the end of the nums list and display the list. Once they have entered three numbers, ask them if they still want the last number they entered saved. If they say "no", remove the last item from the list. Display the list of numbers.



Answers

069

```
country_tuple = ("France", "England", "Spain", "Germany", "Australia")
print(country_tuple)
print()
country = input("Please enter one of the countries from above: ")
print(country, "has index number", country_tuple.index(country))
```

070

```
country_tuple = ("France", "England", "Spain", "Germany", "Australia")
print(country_tuple)
print()
country = input("Please enter one of the countries from above: ")
print(country, "has index number", country_tuple.index(country))
print()
num = int(input("Enter a number between 0 and 4: "))
print(country_tuple[num])
```

071

```
sports_list = ["tennis", "football"]
sports_list.append(input("What is your favourite sport? "))
sports_list.sort()
print(sports_list)
```

072

```
subject_list = ["maths", "english", "computing", "history", "science", "spanish"]
print(subject_list)
dislike = input("Which of these subjects do you dislike? ")
getrid = subject_list.index(dislike)
del subject_list[getrid]
print(subject_list)
```

073

```
food_dictionary = {}
food1 = input("Enter a food you like: ")
food_dictionary[1] = food1
food2 = input("Enter another food you like: ")
food_dictionary[2] = food2
food3 = input("Enter a third food you like: ")
food_dictionary[3] = food3
food4 = input("Enter one last food you like: ")
food_dictionary[4] = food4
print(food_dictionary)
dislike = int(input("Which of these do you want to get rid of? "))
del food_dictionary[dislike]
print(sorted(food_dictionary.values()))
```


074

```
colours = ["red", "blue", "green", "black", "white", "pink", "grey", "purple", "yellow", "brown"]
start = int(input("Enter a starting number (0-4): "))
end = int(input("Enter an end number (5-9): "))
print(colours[start:end])
```

075

```
nums = [123, 345, 234, 765]
for i in nums:
    print(i)
selection = int(input("Enter a number from the list: "))
if selection in nums:
    print(selection, "is in position", nums.index(selection))
else:
    print("That is not in the list")
```

076

```
name1 = input("Enter a name of somebody you want to invite to your party: ")
name2 = input("Enter another name: ")
name3 = input("Enter a third name: ")
party = [name1, name2, name3]
another = input("Do you want to invite another (y/n): ")
while another == "y":
    newname = party.append(input("Enter another name: "))
    another = input("Do you want to invite another (y/n): ")
print("You have", len(party), "people coming to your party")
```

077

```
name1 = input("Enter a name of somebody you want to invite to your party: ")
name2 = input("Enter another name: ")
name3 = input("Enter a third name: ")
party = [name1, name2, name3]
another = input("Do you want to invite another (y/n): ")
while another == "y":
    newname = party.append(input("Enter another name: "))
    another = input("Do you want to invite another (y/n): ")
print("You have", len(party), "people coming to your party")
print(party)
selection = input("Enter one of the names: ")
print(selection, "is in position", party.index(selection), "on the list")
stillcome = input("Do you still want them to come (y/n): ")
if stillcome == "n":
    party.remove(selection)
print(party)
```

078

```
tv = ["Task Master", "Top Gear", "The Big Bang Theory", "How I Met Your Mother"]
for i in tv:
    print(i)
print()
newtv = input("Enter another TV show: ")
position = int(input("Enter a number between 0 and 3: "))
tv.insert(position, newtv)
for i in tv:
    print(i)
```

079

```
nums = []
count = 0
while count < 3:
    num = int(input("Enter a number: "))
    nums.append(num)
    print(nums)
    count = count + 1
lastnum = input("Do you want the last number saved (y/n): ")
if lastnum == "n":
    nums.remove(num)
print(nums)
```

More String Manipulation

Explanation

A **string** is the technical name for a group of characters that you *do not need to perform calculations with*. "Hello" would be an example of a string, as would "7B".

Here we have a variable called **name** which is assigned the value "Simon".

```
name = "Simon"
```

"Simon" can be thought of as a sequence of individual characters and each character in that string can be identified by its index.

Index	0	1	2	3	4
Value	S	i	m	o	n

Note how strings start indexing from 0 and not 1, just as lists do. If the string had a space in it, the space would also be counted as a character, as would any punctuation in the string.

Index	0	1	2	3	4	5	6	7	8	9	10	11
Value	H	e	l	l	o		W	o	r	l	d	!

Now you are familiar with dealing with lists, strings should hold no problems for you as they use the same methods you have used with lists. However, I have included some additional code which may prove useful.

Example Code

Please note: in the examples below, "msg" is a variable name containing a string.

```
if msg.isupper():  
    print("Uppercase")  
else:
```

```
    print("This is not in uppercase")
```

If the message is in uppercase it will display the message "Uppercase", otherwise it will display the message "This is not in uppercase".

```
msg.islower()
```

Can be used in place of the `isupper()` function to check if the variable contains lower case letters.



```
msg="Hello"
```

```
for letter in msg:  
    print(letter, end="*")
```

Displays the message, and between each character it will display a "*".

The output in this example will be: **H*e*l*l*o***

Remember, you can always look back on previous programs to remind yourself of the skills learnt earlier.



Challenges

080

Ask the user to enter their first name and then display the length of their first name. Then ask for their surname and display the length of their surname. Join their first name and surname together with a space between and display the result. Finally, display the length of their full name (including the space).

084

Ask the user to type in their postcode. Display the first two letters in uppercase.

086

Ask the user to enter a new password. Ask them to enter it again. If the two passwords match, display "Thank you". If the letters are correct but in the wrong case, display the message "They must be in the same case", otherwise display the message "Incorrect".

087

Ask the user to type in a word and then display it backwards on separate lines. For instance, if they type in "Hello" it should display as shown below:

```
Enter a word: Hello
o
l
l
e
H
>>>
```

081

Ask the user to type in their favourite school subject. Display it with "-" after each letter, e.g. S-p-a-n-i-s-h-.

082

Show the user a line of text from your favourite poem and ask for a starting and ending point. Display the characters between those two points.

083

Ask the user to type in a word in upper case. If they type it in lower case, ask them to try again. Keep repeating this until they type in a message all in uppercase.

085

Ask the user to type in their name and then tell them how many vowels are in their name.



Answers

080

```
fname = input("Enter your first name: ")
print("That has", len(fname), "characters in it")
sname = input("Enter your surname: ")
print("That has", len(sname), "characters in it")
name = fname + " " + sname
print("Your full name is", name)
print("That has", len(name), "characters in it")
```

081

```
subject = input("Enter your favourite school subject: ")
for letter in subject:
    print(letter, end = "-")
```

082

```
poem = "Oh, I wish I'd looked after me teeth,"
print(poem)
start = int(input("Enter a starting number: "))
end = int(input("Enter an end number: "))
print(poem[start:end])
```

083

```
msg = input("Enter a message in uppercase: ")
tryagain = False
while tryagain == False:
    if msg.isupper():
        print("Thank you")
        tryagain = True
    else:
        print("Try again")
        msg = input("Enter a message in uppercase: ")
```

084

```
postcode = input("Enter your postcode: ")
start = postcode[0:2]
print(start.upper())
```

085

```
name = input("Enter your name: ")
count = 0
name = name.lower()
for x in name:
    if x == "a" or x == "e" or x == "i" or x == "o" or x == "u":
        count = count + 1
print("Vowels =", count)
```

086

```
pswd1 = input("Enter a password: ")
pswd2 = input("Enter it again: ")
if pswd1 == pswd2:
    print("Thank you")
elif pswd1.lower() == pswd2.lower():
    print("They must be the same case")
else:
    print("Incorrect")
```

087

```
word = input("Enter a word: ")
length = len(word)
num = 1
for x in word:
    position = length - num
    letter = word[position]
    print(letter)
    num = num + 1
```

Numeric Arrays



Explanation

Earlier in the book we looked at lists (see page 58). Lists can store a jumble of different types of data at the same time, including strings and numbers. Python **arrays** are similar to lists, but they are **only used to store numbers**. Numbers can have varying ranges, but in an array all pieces of data in that array **must have the same data type**, as outlined in the table below.

Type code	Common name	Description	Size in bytes
'i'	Integer	Whole number between -32,768 and 32,767	2
'l'	Long	Whole number between -2,147,483,648 and 2,147,483,647	4
'f'	Floating-point	Allows decimal places with numbers ranging from -10^{38} to 10^{38} (i.e. allows up to 38 numeric characters including a single decimal point anywhere in that number and can be negative or positive value)	4
'd'	Double	Allows decimal places with numbers ranging from -10^{308} to 10^{308}	8

When you create your array you need to define the type of data it will contain. You cannot alter or change this while the program is running. Therefore, if you define an array as an 'i' type (this allows whole numbers between the values $-32,768$ and $32,767$) you cannot add a decimal point to a number in that array later as it will cause an error message and crash the program.



Please note: Other programming languages use the term array to allow the storage of any data type, but in Python arrays only store numbers whereas lists allow the storage of any data type. If you want to create a variable that stores multiple strings, in Python you need to create a list rather than an array.



Example Code

```
from array import *
```

This needs to be the first line of your program so that Python can use the array library.



```
nums = array ('i', [45, 324, 654, 45, 264])
```

```
print(nums)
```

Creates an array called "nums". It uses the integer data type and has five items in the array. It will display the following as the output:

```
array('i', [45, 324, 654, 45, 264])
```

```
for x in nums:
```

```
    print(x)
```

Displays the array with each item appearing on a separate line.

```
newValue = int(input("Enter number: "))
```

```
nums.append(newValue)
```

Asks the user to enter a new number which it will add to the end of the existing array.

```
nums.reverse()
```

Reverses the order of the array.

```
nums = sorted(nums)
```

Sorts the array into ascending order.

```
nums.pop()
```

This will remove the last item from the array.



```
newArray = array('i', [])
```

```
more = int(input("How many items: "))
```

```
for y in range(0, more):
```

```
    newValue = int(input("Enter num: "))
```

```
    newArray.append(newValue)
```

```
nums.extend(newArray)
```

Creates a blank array called "newArray" which uses the integer data type. It asks the user how many items they want to add and then appends these new items to newArray. After all the items have been added it will join together the contents of newArray and the nums array.

```
getRid = int(input("Enter item index: "))
```

```
nums.remove(getRid)
```

Asks the user to enter the item they want to get rid of and then removes the first item that matches that value from the array.

```
print(nums.count(45))
```

This will display how many times the value "45" appears in the array.

Challenges

088

Ask the user for a list of five integers. Store them in an array. Sort the list and display it in reverse order.

089

Create an array which will store a list of integers. Generate five random numbers and store them in the array. Display the array (showing each item on a separate line).

090

Ask the user to enter numbers. If they enter a number between 10 and 20, save it in the array, otherwise display the message "Outside the range". Once five numbers have been successfully added, display the message "Thank you" and display the array with each item shown on a separate line.

091

Create an array which contains five numbers (two of which should be repeated). Display the whole array to the user. Ask the user to enter one of the numbers from the array and then display a message saying how many times that number appears in the list.

092

Create two arrays (one containing three numbers that the user enters and one containing a set of five random numbers). Join these two arrays together into one large array. Sort this large array and display it so that each number appears on a separate line.

Keep going!



093

Ask the user to enter five numbers. Sort them into order and present them to the user. Ask them to select one of the numbers. Remove it from the original array and save it in a new array.



094

Display an array of five numbers. Ask the user to select one of the numbers. Once they have selected a number, display the position of that item in the array. If they enter something that is not in the array, ask them to try again until they select a relevant item.

095

Create an array of five numbers between 10 and 100 which each have two decimal places. Ask the user to enter a whole number between 2 and 5. If they enter something outside of that range, display a suitable error message and ask them to try again until they enter a valid amount. Divide each of the numbers in the array by the number the user entered and display the answers shown to two decimal places.

Answers

088

```
from array import *

nums = array('i', [])

for i in range (0,5):
    num = int(input("Enter a number: "))
    nums.append(num)

nums = sorted(nums)
nums.reverse()

print(nums)
```

089

```
from array import *
import random

nums = array('i', [])

for i in range (0,5):
    num = random.randint(1,100)
    nums.append(num)

for i in nums:
    print(i)
```

090

```
from array import *

nums = array('i', [])

while len(nums) < 5:
    num = int(input("Enter a number between 10 and 20: "))
    if num >= 10 and num <= 20:
        nums.append(num)
    else:
        print("Outside the range")

for i in nums:
    print(i)
```

091

```
from array import *

nums = array('i', [5, 7, 9, 2, 9])

for i in nums:
    print(i)

num = int(input("Enter a number: "))

if nums.count(num) == 1:
    print(num, "is in the list once")
else:
    print(num, "is in the list", nums.count(num), "times")
```

092

```
from array import *
import random

num1 = array('i', [])
num2 = array('i', [])

for i in range(0, 3):
    num = int(input("Enter a number: "))
    num1.append(num)

for i in range(0, 5):
    num = random.randint(1, 100)
    num2.append(num)

num1.extend(num2)

num1 = sorted(num1)

for i in num1:
    print(i)
```

093

```
from array import *

nums = array('i', [])

for i in range(0,5):
    num = int(input("Enter a number: "))
    nums.append(num)

nums = sorted(nums)

for i in nums:
    print(i)

num = int(input("Select a number from the array: "))
if num in nums:
    nums.remove(num)
    num2 = array('i', [])
    num2.append(num)
    print(nums)
    print(num2)
else:
    print("That is not a value in the array")
```

094

```
from array import *

nums = array('i', [4,6,8,2,5])

for i in nums:
    print(i)

num = int(input("Select one of the numbers: "))

tryagain = True
while tryagain == True:
    if num in nums:
        print("This is in position",nums.index(num))
        tryagain = False
    else:
        print("Not in array")
        num = int(input("Select one of the numbers: "))
```

095

```
from array import *
import math

nums = array('f', [34.75, 27.23, 99.58, 45.26, 28.65])
tryagain = True
while tryagain == True:
    num = int(input("Enter a number between 2 and 5: "))
    if num < 2 or num > 5:
        print("Incorrect value, try again.")
    else:
        tryagain = False
for i in range(0, 5):
    ans = nums[i]/num
    print(round(ans, 2))
```

2D Lists and Dictionaries

Explanation

Technically it is possible to create a two-dimensional array in Python, but as Python arrays are limited to storing numbers and most Python programmers feel more comfortable with working with lists, 2D arrays are rarely used and **2D lists** are far more common.



Imagine, for one terrifying moment, you are a teacher. Scary I know! Also imagine you have four students and you teach those same students across three different subjects. You may, if you are a conscientious teacher, need to keep records of those students' grades for each of their subjects. It is possible to create a simple chart on paper to do this as follows:



	Maths	English	French
Susan	45	37	54
Peter	62	58	59
Mark	49	47	60
Andy	78	83	62

Two-dimensional lists work in a similar way.

	0	1	2
0	45	37	54
1	62	58	59
2	49	47	60
3	78	83	62

In Python, this two-dimensional list would be coded as follows:

```
grades = [[45, 37, 54], [62, 58, 59], [49, 47, 60], [78, 83, 62]]
```

Alternatively, if you do not want to use the standard Python column index numbers you can use a dictionary as follows:

```
grades = [{"Ma": 45, "En": 37, "Fr": 54}, {"Ma": 62, "En": 58, "Fr": 59}, {"Ma": 49, "En": 47, "Fr": 60}]  
print(grades[0]["En"])
```

This program will produce the output 37 (the English grade for the pupil with the index number 0) and can make the data easier to understand.

You can even go further and add a row index as follows:

```
grades = {"Susan": {"Ma": 45, "En": 37, "Fr": 54}, "Peter": {"Ma": 62, "En": 58, "Fr": 59}}  
print(grades["Peter"]["En"])
```

This will give the output 58, the grade for Peter's English exam.



Example Code

```
simple_array = [[2,5,8],[3,7,4],[1,6,9]]
```

Creates a 2D list (as shown on the right) which uses standard Python indexing for the rows and columns.

	0	1	2
0	2	5	8
1	3	7	4
2	1	6	9

```
print(simple_array)
```

Displays all the data in the 2D list.

```
simple_array[2][1]= 5
```

Changes the data in row 2, column 1 to the value 5.

```
simple_array[1].append(3)
```

Adds the value 3 onto the end of the data in row 1 so in this case it becomes [3, 7, 4, 3].



```
print(simple_array[1])
```

Displays data from row 1, in this case [3, 7, 4].

```
print(simple_array[1][2])
```

Displays data from row 1, column 2, in this case 4.

	x	y	z
A	54	82	91
B	75	29	80



```
data_set = {"A":{"x":54,"y":82,"z":91}, "B":{"x":75,"y":29,"z":80}}
```

Creates a 2D dictionary using user-defined labels for the rows and columns (as shown above).

```
print(data_set["A"])
```

Displays data from data set "A".

```
print(data_set["B"]["y"])
```

Displays data from row "B", column "y".

```
for i in data_set:  
    print(data_set[i]["y"])
```

Displays the "y" column from each row.

```
data_set["B"]["y"] = 53
```

Changes the data in "B", "y", to 53.

```
grades[name]={ "Maths":mscore, "English":escore}
```

Adds another row of data to a 2D dictionary. In this case, name would be the row index and Maths and English would be the column indexes.

```
for name in grades:  
    print((name),grades[name]["English"])
```

Displays only the name and the English grade for each student.

```
del list[getRid]
```

Removes a selected item.

Challenges

096

Create the following using a simple 2D list using the standard Python indexing:

	0	1	2
0	2	5	8
1	3	7	4
2	1	6	9
3	4	2	0

097

Using the 2D list from program 096, ask the user to select a row and a column and display that value.

098

Using the 2D list from program 096, ask the user which row they would like displayed and display just that row. Ask them to enter a new value and add it to the end of the row and display the row again.

099

Change your previous program to ask the user which row they want displayed. Display that row. Ask which column in that row they want displayed and display the value that is held there. Ask the user if they want to change the value. If they do, ask for a new value and change the data. Finally, display the whole row again.

100

Create the following using a 2D dictionary showing the sales each person has made in the different geographical regions:

	N	S	E	W
John	3056	8463	8441	2694
Tom	4832	6786	4737	3612
Anne	5239	4802	5820	1859
Fiona	3904	3645	8821	2451

101

Using program 100, ask the user for a name and a region. Display the relevant data. Ask the user for the name and region of data they want to change and allow them to make the alteration to the sales figure. Display the sales for all regions for the name they choose.

102

Ask the user to enter the name, age and shoe size for four people. Ask for the name of one of the people in the list and display their age and shoe size.

104

After gathering the four names, ages and shoe sizes, ask the user to enter the name of the person they want to remove from the list. Delete this row from the data and display the other rows on separate lines.

103

Adapt program 102 to display the names and ages of all the people in the list but do not show their shoe size.



Answers

096

```
list = [[2,5,8],[3,7,4],[1,6,9],[4,2,0]]
```

097

```
list = [[2,5,8],[3,7,4],[1,6,9],[4,2,0]]
row = int(input("Select a row: "))
col = int(input("Select a column: "))
print(list[row][col])
```

098

```
list = [[2,5,8],[3,7,4],[1,6,9],[4,2,0]]
row = int(input("Select a row: "))
print(list[row])
newvalue = int(input("Enter a new number: "))
list[row].append(newvalue)
print(list[row])
```

099

```
list = [[2,5,8],[3,7,4],[1,6,9],[4,2,0]]
row = int(input("Select a row: "))
print(list[row])
col = int(input("Select a column: "))
print(list[row][col])
change = input("Do you want to change the value? (y/n) ")
if change == "y":
    newvalue = int(input("Enter new value: "))
    list[row][col] = newvalue
print(list[row])
```

100

Please note the data has been split onto separate rows to make it easier to read the code. This is possible, as long as the breaks are where the rows will natural break and are contained within the curly brackets.

```
sales = {"John":{"N":3056, "S":8463, "E":8441, "W":2694},
"Tom":{"N":4832, "S":6786, "E":4737, "W":3612},
"Anne":{"N":5239, "S":4802, "E":5820, "W":1859},
"Fiona":{"N":3904, "S":3645, "E":8821, "W":2451}}
```

101

```
sales = {"John":{"N":3056, "S":8463, "E":8441, "W":2694},
"Tom":{"N":4832, "S":6786, "E":4737, "W":3612},
"Anne":{"N":5239, "S":4802, "E":5820, "W":1859},
"Fiona":{"N":3904, "S":3645, "E":8821, "W":2451}}
person = input("Enter sales person's name: ")
region = input("Select region: ")
print(sales[person][region])
newdata = int(input("Enter new data: "))
sales[person][region] = newdata
print(sales[person])
```

102

```
list = {}
for i in range (0,4):
    name = input("Enter name: ")
    age = int(input("Enter age: "))
    shoe = int(input("Enter shoe size: "))
    list[name] = {"Age":age,"Shoe size":shoe}

ask = input("Enter a name: ")
print(list[ask])
```

103

```
list = {}
for i in range (0,4):
    name = input("Enter name: ")
    age = int(input("Enter age: "))
    shoe = int(input("Enter shoe size: "))
    list[name] = {"Age":age,"Shoe size":shoe}

for name in list:
    print((name),list[name]["Age"])
```

104

```
list = {}
for i in range (0,4):
    name = input("Enter name: ")
    age = int(input("Enter age: "))
    shoe = int(input("Enter shoe size: "))
    list[name] = {"Age":age, "Shoe size":shoe}

getrid = input("Who do you want to remove from the list? ")
del list[getrid]

for name in list:
    print((name), list[name] ["Age"], list[name] ["Shoe size"])
```

Reading and Writing to a Text File



Explanation

It is all very well being able to define a list, make changes and add new data, but if the next time the program is run it returns to the original data and your changes are lost then it is not a lot of use. Therefore, it is sometimes necessary to save data outside of the program and this way the data can be stored, along with any changes that are made.

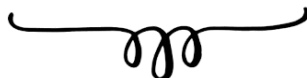


The easiest place to start learning about writing and reading from an external file is with a **text** file.

When opening an external file you must specify how that file will be used within the program. The options are below.

Code	Description
w	Write mode: used to create a new file. Any existing files with the same name will be erased and a new one created in its place.
r	Read mode: used when an existing file is only being read and not being written to.
a	Append mode: used to add new data to the end of the file.

Text files are only used to write, read and append data. By the very nature of how they work it is not easy to remove or alter individual elements of data once it is written to the file, unless you want to overwrite the entire file or create a new file to store the new data. If you want to be able to alter the individual elements once the file has been created it is better to use a .csv file (see page 91) or an SQL database (see page 134).



Example Code

```
file = open("Countries.txt", "w")
file.write("Italy\n")
file.write("Germany\n")
file.write("Spain\n")
file.close()
```

Creates a file called "Countries.txt". If one already exists then it will be overwritten with a new blank file. It will add three lines of data to the file (the \n forces a new line after each entry). It will then close the file, allowing the changes to the text file to be saved.

```
file = open("Countries.txt", "r")
print(file.read())
```

This will open the Countries.txt file in "read" mode and display the entire file.

```
file = open("Countries.txt", "a")
file.write("France\n")
file.close()
```

This will open the Countries.txt file in "append" mode, add another line and then close the file. If the **file.close()** line is not included, the changes will not be saved to the text file.



Challenges

105

Write a new file called "Numbers.txt". Add five numbers to the document which are stored on the same line and only separated by a comma. Once you have run the program, look in the location where your program is stored and you should see that the file has been created. The easiest way to view the contents of the new text file on a Windows system is to read it using Notepad.

106

Create a new file called "Names.txt". Add five names to the document, which are stored on separate lines. Once you have run the program, look in the location where your program is stored and check that the file has been created properly.

107

Open the Names.txt file and display the data in Python.



108

Open the Names.txt file. Ask the user to input a new name. Add this to the end of the file and display the entire file.

109

Display the following menu to the user:

- 1) Create a new file
 - 2) Display the file
 - 3) Add a new item to the file
- Make a selection 1, 2 or 3:

Ask the user to enter 1, 2 or 3. If they select anything other than 1, 2 or 3 it should display a suitable error message.

If they select 1, ask the user to enter a school subject and save it to a new file called "Subject.txt". It should overwrite any existing file with a new file.

If they select 2, display the contents of the "Subject.txt" file.

If they select 3, ask the user to enter a new subject and save it to the file and then display the entire contents of the file.

Run the program several times to test the options.

110

Using the Names.txt file you created earlier, display the list of names in Python. Ask the user to type in one of the names and then save all the names except the one they entered into a new file called Names2.txt.



Fantastic work, saving data to external files is an important programming skill.

Answers

105

```
file = open("Numbers.txt", "w")
file.write("4, ")
file.write("6, ")
file.write("10, ")
file.write("8, ")
file.write("5, ")
file.close()
```

106

```
file = open("Names.txt", "w")
file.write("Bob\n")
file.write("Tom\n")
file.write("Gemma\n")
file.write("Sarah\n")
file.write("Timothy\n")
file.close()
```

107

```
file = open("Names.txt", "r")
print(file.read())
file.close()
```

108

```
file = open("Names.txt", "a")
newname = input("Enter a new name: ")
file.write(newname + "\n")
file.close
```

```
file = open("Names.txt", "r")
print (file.read())
file.close
```

109

```
print ("1) Create a new file")
print ("2) Display the file")
print ("3) Add a new item to the file")
selection = int(input("Make a selection 1, 2 or 3: "))
if selection == 1:
    subject = input("Enter a school subject: ")
    file = open("Subject.txt", "w")
    file.write(subject + "\n")
    file.close()
elif selection == 2:
    file = open("Subject.txt", "r")
    print(file.read())
elif selection == 3:
    file = open("Subject.txt", "a")
    subject = input("Enter a school subject: ")
    file.write(subject + "\n")
    file.close()
    file = open("Subject.txt", "r")
    print(file.read())
else:
    print("Invalid option")
```

110

```
file = open("Names.txt", "r")
print(file.read())
file.close()

file = open("Names.txt", "r")
selectedname = input("Enter a name from the list: ")
selectedname = selectedname + "\n"
for row in file:
    if row != selectedname:
        file = open("Names2.txt", "a")
        newrecord = row
        file.write(newrecord)
        file.close()
file.close()
```

Reading and Writing to a .csv File

Explanation

CSV stands for **Comma Separated Values** and is a format usually associated with importing and exporting from spreadsheets and databases. It allows greater control over the data than a simple text file, as each row is split up into identifiable columns. Below is an example of data you may want to store.

Name	Age	Star sign
Brian	73	Taurus
Sandra	48	Virgo
Zoe	25	Scorpio
Keith	43	Leo



A .csv file would store the above data as follows:



```
Brian, 73, Taurus  
Sandra, 48, Virgo  
Zoe, 25, Scorpio  
Keith, 43, Leo
```

However, it may be easier to think of it as being separated into columns and rows that use an index number to identify them.

	0	1	2
0	Brian	73	Taurus
1	Sandra	48	Virgo
2	Zoe	25	Scorpio
3	Keith	43	Leo

When opening a .csv file to use, you must specify how that file will be used. The options are:

Code	Description
w	Creates a new file and writes to that file. If the file already exists, a new file will be created, overwriting the existing file.
x	Creates a new file and writes to that file. If the file already exists, the program will crash rather than overwrite it.
r	Opens for reading only and will not allow you to make changes.
a	Opens for writing, appending to the end of the file.



Example Code

```
import csv
```

This must be at the top of your program to allow Python to use the .csv library of commands.

```
file = open ("Stars.csv", "w")  
newRecord = "Brian,73,Taurus\n"  
file.write(str(newRecord))  
file.close()
```

This will create a new file called "Stars.csv", overwriting any previous files of the same name. It will add a new record and then close and save the changes to the file.

```
file = open ("Stars.csv", "a")  
name = input("Enter name: ")  
age = input("Enter age: ")  
star = input("Enter star sign: ")  
newRecord = name + "," + age + "," + star + "\n"  
file.write(str(newRecord))  
file.close()
```

This will open the Stars.csv file, ask the user to enter the name, age and star sign, and will append this to the end of the file.



```
file = open("Stars.csv", "r")  
for row in file:  
    print(row)
```

This will open the Stars.csv file in read mode and display the records one row at a time.

```
file = open("Stars.csv", "r")  
reader = csv.reader(file)  
rows = list(reader)  
print(rows[1])
```

This will open the Stars.csv file and display only row 1. Remember, Python starts counting from 0.

```
file = open ("Stars.csv", "r")  
search = input("Enter the data you are searching for: ")  
reader = csv.reader(file)  
for row in file:  
    if search in str(row):  
        print(row)
```

Asks the user to enter the data they are searching for. It will display all rows that contain that data anywhere in that row.

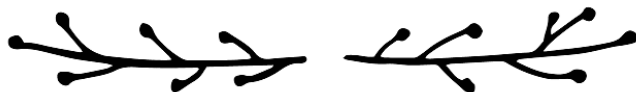


```
import csv
file = list(csv.reader(open("Stars.csv")))
tmp = []
for row in file:
    tmp.append(row)
```

A .csv file cannot be altered, only added to. If you need to alter the file you need to write it to a temporary list. This block of code will read the original .csv file and write it to a list called "tmp". This can then be used and altered as a list (see page 58).

```
file = open("NewStars.csv", "w")
x = 0
for row in tmp:
    newRec = tmp[x][0] + "," + tmp[x][1] + "," + tmp[x][2] + "\n"
    file.write(newRec)
    x = x + 1
file.close()
```

Writes from a list into a new .csv file called "NewStars.csv".



Challenges

111

Create a .csv file that will store the following data. Call it "Books.csv".

	Book	Author	Year Released
0	To Kill A Mockingbird	Harper Lee	1960
1	A Brief History of Time	Stephen Hawking	1988
2	The Great Gatsby	F. Scott Fitzgerald	1922
3	The Man Who Mistook His Wife for a Hat	Oliver Sacks	1985
4	Pride and Prejudice	Jane Austen	1813

112

Using the Books.csv file from program 111, ask the user to enter another record and add it to the end of the file. Display each row of the .csv file on a separate line.

113

Using the Books.csv file, ask the user how many records they want to add to the list and then allow them to add that many. After all the data has been added, ask for an author and display all the books in the list by that author. If there are no books by that author in the list, display a suitable message.

114

Using the Books.csv file, ask the user to enter a starting year and an end year. Display all books released between those two years.

115

Using the Books.csv file, display the data in the file along with the row number of each.



116

Import the data from the Books.csv file into a list. Display the list to the user. Ask them to select which row from the list they want to delete and remove it from the list. Ask the user which data they want to change and allow them to change it. Write the data back to the original .csv file, overwriting the existing data with the amended data.

117

Create a simple maths quiz that will ask the user for their name and then generate two random questions. Store their name, the questions they were asked, their answers and their final score in a .csv file. Whenever the program is run it should add to the .csv file and not overwrite anything.

Answers

111

```
import csv

file = open("Books.csv", "w")
newrecord = "To Kill A Mockingbird, Harper Lee, 1960\n"
file.write(str(newrecord))
newrecord = "A Brief History of Time, Stephen Hawking, 1988\n"
file.write(str(newrecord))
newrecord = "The Great Gatsby, F. Scott Fitzgerald, 1922\n"
file.write(str(newrecord))
newrecord = "The Man Who Mistook His Wife for a Hat, Oliver Sacks, 1985\n"
file.write(str(newrecord))
newrecord = "Pride and Prejudice, Jane Austen, 1813\n"
file.write(str(newrecord))
file.close()
```

112

```
import csv

file = open("Books.csv", "a")
title = input("Enter a title: ")
author = input("Enter author: ")
year = input("Enter the year it was released: ")
newrecord = title + "," + author + ", " + year + "\n"
file.write(str(newrecord))
file.close()

file = open("Books.csv", "r")
for row in file:
    print(row)
file.close()
```


113

```
import csv

num = int(input("How many books do you want to add to the list? "))
file = open("Books.csv","a")
for x in range(0,num):
    title = input("Enter a title: ")
    author = input("Enter author: ")
    year = input("Enter the year it was released: ")
    newrecord = title + "," + author + ", " + year + "\n"
    file.write(str(newrecord))
file.close()

searchauthor = input("Enter an authors name to search for: ")

file = open("Books.csv","r")
count = 0
for row in file:
    if searchauthor in str(row):
        print(row)
        count = count + 1
if count == 0:
    print ("There are no books by that author in this list.")
file.close()
```

114

```
import csv

start = int(input("Enter a starting year: "))
end = int(input("Enter an end year: "))

file = list(csv.reader(open("Books.csv")))
tmp = []
for row in file:
    tmp.append(row)

x = 0
for row in tmp:
    if int(tmp[x][2]) >= start and int(tmp[x][2]) <=end:
        print(tmp[x])
    x = x+1
```

115

```
import csv

file = open("Books.csv","r")
x = 0
for row in file:
    display = "Row: " + str(x) + " - " + row
    print(display)
    x = x + 1
```

116

```

import csv

file = list(csv.reader(open("Books.csv")))
Booklist = []
for row in file:
    Booklist.append(row)

x = 0
for row in Booklist:
    display = x,Booklist[x]
    print(display)
    x = x + 1
getrid = int(input("Enter a row number to delete: "))
del Booklist[getrid]

x = 0
for row in Booklist:
    display = x,Booklist[x]
    print(display)
    x = x + 1
alter = int(input("Enter a row number to alter: "))
x = 0
for row in Booklist[alter]:
    display = x,Booklist[alter][x]
    print(display)
    x = x + 1
part = int(input("Which part do you want to change? "))
newdata = input("Enter new data: ")
Booklist[alter][part] = newdata
print(Booklist[alter])

file = open("Books.csv", "w")
x = 0
for row in Booklist:
    newrecord = Booklist[x][0] + ", " + Booklist[x][1] + ", " + Booklist[x][2] + "\n"
    file.write(newrecord)
    x = x+1
file.close()

```

117

```

import csv
import random

score = 0
name = input("What is your name: ")
q1_num1 = random.randint(10,50)
q1_num2 = random.randint(10,50)
question1 = str(q1_num1) + " + " + str(q1_num2) + " = "
ans1 = int(input(question1))
realans1 = q1_num1+q1_num2
if ans1 == realans1:
    score = score + 1
q2_num1 = random.randint(10,50)
q2_num2 = random.randint(10,50)
question2 = str(q2_num1) + " + " + str(q2_num2) + " = "
ans2 = int(input(question2))
realans2 = q2_num1+q2_num2
if ans2 == realans2:
    score = score + 1

file = open("QuizScore.csv", "a")
newrecord = name+", "+question1+", "+str(ans1)+", "+question2+", "+str(ans2)+", "+str(score)+"\n"
file.write(str(newrecord))

file.close()

```

Subprograms

Explanation

Subprograms are blocks of code which perform specific tasks and can be called upon at any time in the program to run that code.

Advantages

- You can write a block of code and it can be used and re-used at different times during the program.
- It makes the program simpler to understand as the code is grouped together into chunks.

Defining a subprogram and passing variables between subprograms

Below is a simple program that we would normally create without subprograms but have written it with subprograms so you can see how they work:

```
def get_name():
    user_name = input("Enter your name: ")
    return user_name

def print_Msg(user_name):
    print("Hello", user_name)

def main():
    user_name = get_name()
    print_Msg(user_name)

main()
```

This program uses three subprograms `get_name()`, `print_Msg()` and `main()`.

The `get_name()` subprogram will ask the user to input their name and then it will return the value of the variable "user_name" so that it can be used in another subprogram. This is very important. If you do not return the values, then the values of any variables that were created or altered in that subprogram cannot be used elsewhere in your program.

