Learn to program with **Python** in 150 challenges

By Nichola Lacey

Learn to program with Python in 150 Challenges

Nichola Lacey

www.nicholawilkin.com

Terms and Conditions

PERMITTED USE, PLEASE READ CAREFULLY BEFORE USING

The material in this book including the accompanying material may only be used in the context for which it was intended and is only for the person or institution that has purchased the book.

You may not (without the written permission of Nichola Lacey):

1. Reproduce transmit or distribute any part of this book or the accompanying material in any form outside of the Purchasing Institution except for photocopying pages for students' personal use and putting content onto the Purchasing Institution's secure network or VLE

2. Reproduce, transmit or distribute any part of this book or the accompanying material to other schools, legal entities, individuals or teachers outside of the Purchasing Institution

3. Redistribute resell license sublicense lend rent lease assign or transfer any part of this book or the accompanying material or otherwise make it available for use or distribution to any other individual educational institution or legal entity

4. Upload any part of this book or the accompanying material onto the internet for use or consumption by the general public

5. Use or incorporate any of the images files challenges from any part of this book or the accompanying material in your own original work or to publish distribute or display or pass it off as your own work

The accompanying material has been tested for viruses at all stages of their production. However, we recommend that you run virus-checking software on your computer systems at all times. Nichola Lacey cannot accept responsibility for any loss, disruption or damage to your data or your computer systems that may occur as a result of using the files.

If you accept the above terms you may proceed to open the files attached in the zipped folder.

Contents

Introduction

This book helps people learn how to program with Python by using practical examples rather than pages and pages of explanation. Many programmers learn through experimentation and looking at others' code to work out how it works and that is exactly the method that this book uses. It is a much more hands-on approach than other programming guides and expects the user to work out how to solve the challenges, create the programs, experiment and look at the example solutions in order to learn how to think like a programmer.

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 simple exercises and then work through them 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 areas you need.

The book is split into two sections:

Part 1

In Part 1, each chapter takes you through some basic programming rules and challenges for you to complete. Each chapter 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 for each that 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 and each challenge should only take between a couple of minutes and 20 minutes to solve, with some of the more complex ones near the end of Part 1 taking longer as you build up the techniques you will be using.
- code containing **possible answers** for each challenge; there is often more than one answer available but we include just one possible answer.

Part 2

In Part 2, you are given some larger challenges which utilise the programming skills you learnt in Part 1 of the book 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 1 and it will take longer to solve each challenge. After each challenge you are given one possible answer which you may find useful if you are stuck on one particular aspect of the code. 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 using Python. It is an essential tool for teachers and students studying GCSE and A-level computing who need help and ready-made examples to practise programming techniques and build confidence. It can also be used to help with the new GCSE Computer Science non-exam assessment (NEA) resource bank, to be used to help pupils needing additional support or just a quick reminder when writing their programs under controlled conditions.

Part 1

Learning Python

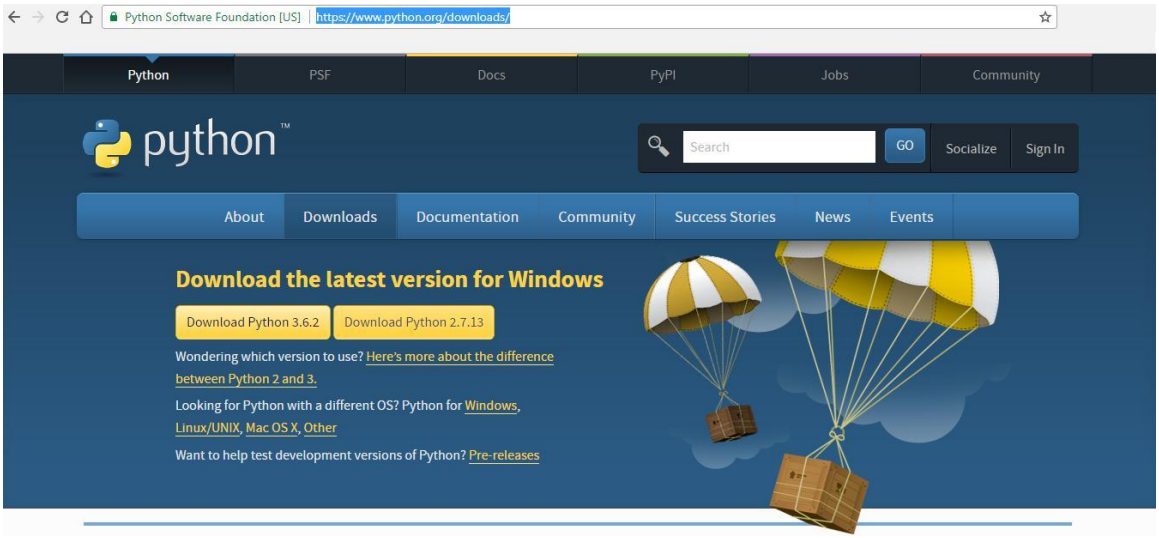
Using Python

In this first section we will be looking at how you can install Python and include some tips for getting you started. It is worth reading this section first but if you prefer you can just get stuck in and come back and read the relevant sections later.

Downloading and installing Python

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

https://www.python.org/downloads/



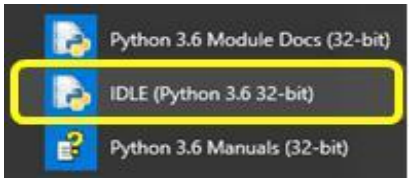
Click on the latest version (in the example above, click on the yellow Download Python 3.6.2 button) to start the installation.

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



File Location

The Python folder is usually found in the C:\ drive and will be named Python36 or similar and the files will automatically be saved there unless you save them specifically in another location.

Using Comments

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

- Add an explanation to how the program works
- Stopping parts from working temporarily so you can test other parts of the program

The first, and original, purpose of explaining how a program works is so that other programmers can make sense of your programs in case they need 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 three 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 and you would only add comments where necessary.

As Python knows to ignore anything after a # symbol, programmers soon started to use # to block out areas of code they do not want to run so they can test others.

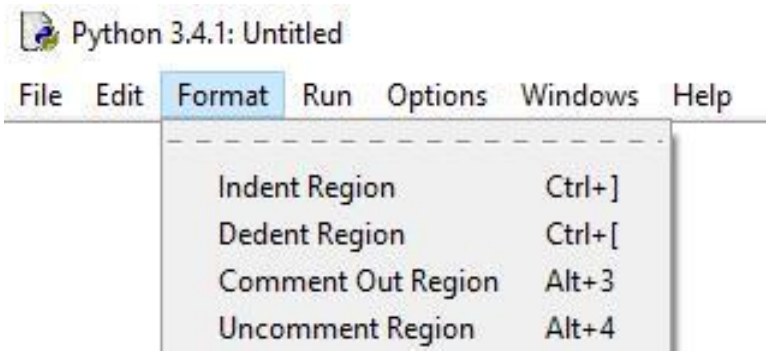
```
#print("This is a simple program")
print()
name = input("Please input your name: ")
print("Hello", name)
```

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!

A few Python IDLE tips

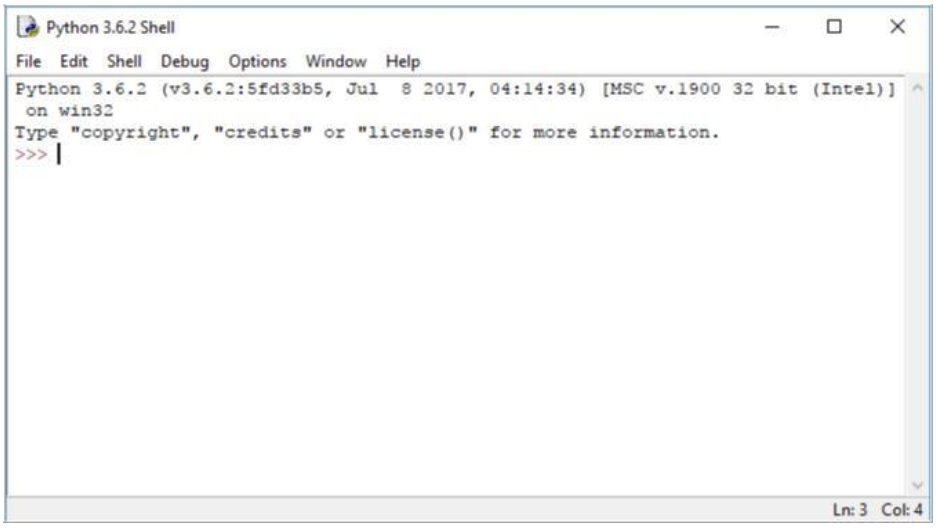
In most versions of Python IDLE it is possible to quickly add comments and indent code using the menus. This way, if you need to block out entire areas using a comment you simply highlight the lines and then select the Format menu and “Comment Out Region”. Similarly, if you need to indent a region (we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



The Basics

Explanation

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



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.

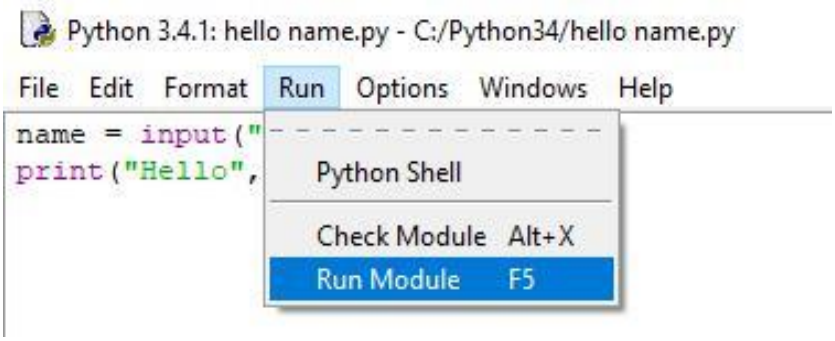
To create a new window in which to write your code select File and New. Once you enter your code in this new window you can save it and run it all in one go. This will then run the code in the shell window.

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 has 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 will be saved, Python will prompt you to name 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 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, insert 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, insert 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 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 will not be able to edit it.

Example code

Code	Explanation
`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 then the speech marks are not needed.
`textvalue = input("Enter a text value: ")`	Displays the question “Enter a text value:” and stores the value they enter in a variable called textvalue
`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
`num1 = 93`	Set the value of a variable called num1 to 93. If there is not a variable already created it will create one. A variable can be called whatever you want but it must be a single word with no spaces and it is not allowed to be one of Python’s dedicated words such as print or input.
`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("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.

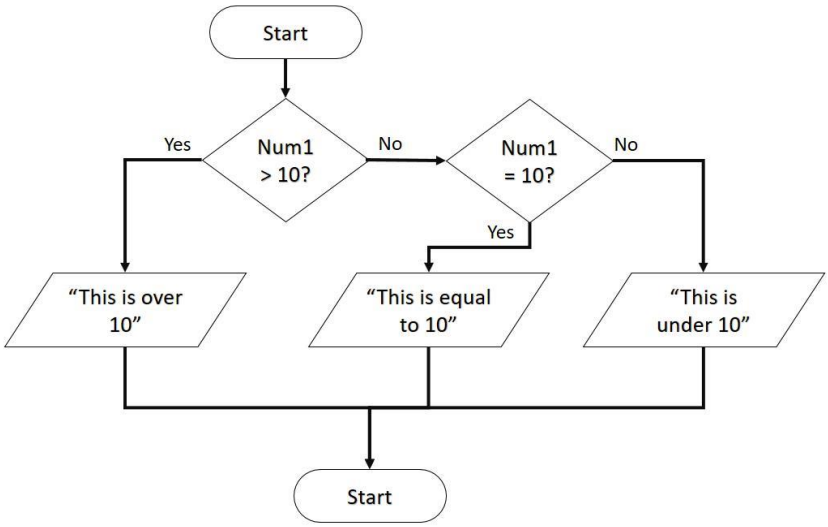
Challenges

Number	Challenge
1	Ask for their first name and display the output message “Hello [First Name]”
002	Ask for their first name and then ask for their surname and display the output message “Hello [First Name] [Surname]”
3	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 for how many sweets they used to have and ask how many sweets they have eaten. Work out how many sweets they have left and display the answer.
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 has to 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.
010	Use the internet to find out the latest conversion rate for GB Pounds (£) to Euros (€). Write a program to ask the user to enter the number of Pounds they want to take on holiday and convert this into the equivalent amount in Euros. NB to get the € symbol on a windows keyboard hold down the Ctrl and Alt keys and press 4.
011	Improve the above program so that it will tell you how many 50, 20, 10 and 5 Euro notes you would receive to make the correct number of GB Pounds.

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 like in Python. Indenting is very important in Python as it shows the lines that are dependent on others as shown in the example below. In order to indent text you can use your [tab] key. The [backspace] key will remove indents.

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

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 it will test the second condition (if there is one) and so on.

The comparison operators

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

Logical operators

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

Example code

Code	Explanation

`if num1 > 10:`
`  print("This is over 10")`
`else:`
`  print("This is under 10")`
	If num1 is over 10 then it will display the message “This is over 10”, otherwise it will display the message “This is under 10”.

`if num1 > 10:`
`  print("This is over 10")`
`elif num1 == 10:`
`  print("This is equal to 10")`
`else:`
`  print("This is under 10")`
	If num1 is over 10 then 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 num1 > 10:`
`  if num1 <= 20:`
`  print("This is between 10 and 20")`
`  else:`
`  print("This is over 20")`
`else:`
`  print("This is under 10")`
	If num1 is over 10 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 over 10 but 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.

`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 the conditions in the if statement. With and, both or all of the conditions must be met.

`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. With or, just one condition must be met.

Challenges

Number	Challenge
12	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.
13	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".
14	Ask the user to enter a number between 10 and 20. If they enter a number within this range display the message "Thank you" otherwise display the message "Incorrect answer"
15	Ask them 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".
16	Ask them if it is raining. 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"
17	Ask for their age. If they are 18 or over display the message "You can vote", if they are age 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".
18	Ask them 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"
19	Ask them 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".

Strings

Explanation

String is the technical name for text. This can include numbers and letters but it is all treated as text. Therefore, even if you only enter numbers into a string you cannot perform any calculations with it.

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 as long as you are consistent.

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

" ' \

That is because these symbols have special meaning 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 proceed it with a backslash symbol and then Python will know to ignore the symbol and will treat it as normal text which is to be displayed.

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

Example Code

Code	Explanation
`len(word)`	Finds the length of a word.
`word.upper()`	Changes a word into UPPER CASE.
`word.lower()`	Changes a word into lower case.
`phrase.capatalize()`	Changes a phrase so that only the first word has a capital letter at the beginning and everything else is in lower case.
`phrase.title()`	Changes a phrase so that every word has a capital letter at the beginning with the rest of the letters in lower case.
`strip(phrase)`	Removes extra spaces leaving only one space between each word and none at the start or end of the phrase.
`print("Hello world"[7:10])`	Each letter is assigned a number to identify its position in the phrase including the space. Python starts counting from 0 and not 1. Therefore, in this example, it would display the letters 7, 8 and 9 which is “orl”.
`name = firstname+surname`	Join the first name and surname together without a space between them.

Please note: In the above examples the terms word, phrase, name, firstname and surname are all variable names and are not dedicated keywords that Python understands.

Challenges

Number	Challenge
20	Ask the user to enter their first name and then display the length of their name.
21	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.
22	Ask them to enter their first name and surname in lower case. Change the case to title case and join them together. Display the finished result.
23	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).
24	Ask the user to type in a word and display it in upper case.
25	Ask the user to enter their first name. If the length of their name is under 5 characters ask them to enter their surname and join them together (without a space) and display the name in uppercase. If the length of the first name is 5 or more, display their first name in lowercase.
26	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 ananabay, 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.

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 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.

Example Code

Code	Explanation
`num=float(input("Enter number: "))`	Allows numbers with a decimal point dividing the integer and fraction part.
`print(round(num,2))`	Displays a number rounded to 2 decimal places.
`**`	To the power of (i.e. 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.
`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 (i.e. 15//2 gives the answer 7).
`x % y`	Finds the remainder (i.e. 15%2 gives the answer 1)

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 in **`import math`** as the first line of your program.

Challenges

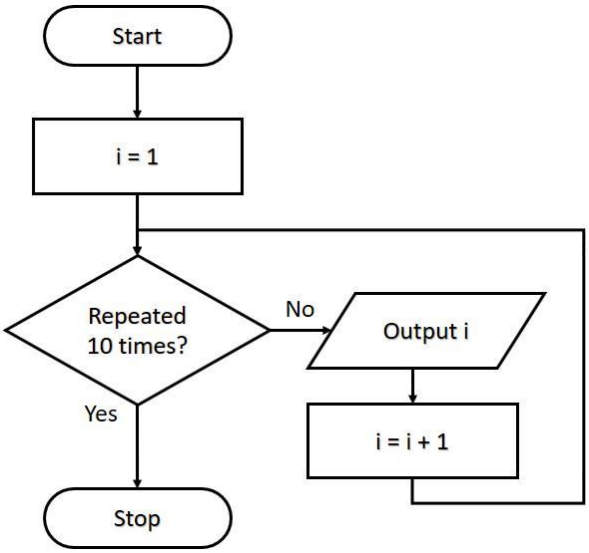
Number	Challenge
027	Ask them to enter a number with lots of decimal places. Multiply this number by two and display the answer.
028	Update the previous program so that it will display the answer to 2 decimal places.
029	Ask the user to enter an integer that is over 500. Work out the square root of that number and display it to 2 decimal places.
030	Display pi (π) to 5 decimal places.
31	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 (π*radius^2)
032	Ask for the radius and the depth of a cylinder and work out the total volume (circle area*depth) rounded to 3 decimal places.
33	Ask the user to enter 2 numbers. Display the answer when the first number is divided by the other. Show the answer for a whole number division and also the remainder from a whole number division.

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.

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 stop. This is how this For 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 (i.e. counting in 1's, 5's or any other value you wish) as the table below shows:

Range function	Output that would be produced
`for i in range(1,10):` `  print(i)`	1, 2, 3, 4, 5, 6, 7, 8, 9
`for i in range(1,10,2):` `  print(i)`	1, 3, 5, 7, 9
`for i in range(10,1,-3):` `  print(i)`	10, 7, 4
`for i in word:` `  print(i)`	This would display each character in the word or phrase as a separate output (i.e. on a separate line).

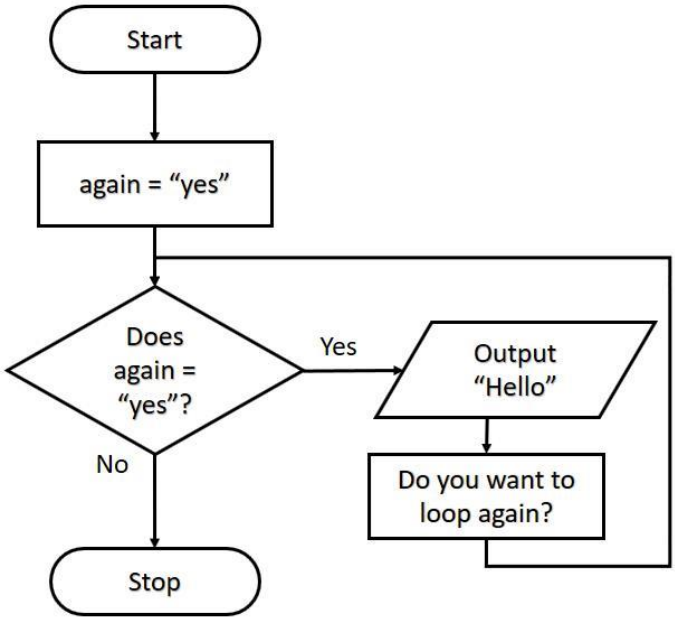
Challenges

Number	Challenge
35	Ask the user to enter their name and then display their name 3 times.
036	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.
038	Ask the user to enter a number and their name and then display their name (one letter at a time on each line) and repeat this for the number they entered.
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, including showing the number they entered in the output.
41	Ask them 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.
042	Set a variable called total to 0. Ask the user to enter 5 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 onto the total. After they have entered all 5 numbers display the total.
43	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 then 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 then 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".

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 a hundred times, just the once or even never. A large difference between a For loop and a While loop is when the condition is checked. In a While loop it 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 that 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 set a variable called total to 0. It will ask the user to enter a number, will add it to the total and will keep repeating this until the total equals 100 or more, when it will stop the loop and display the total.

The comparison operators

Operator	Description
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater 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 don't.

Challenges

Number	Challenge
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.
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.
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". At the end display the total.
048	Ask for the name of somebody they want 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.
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 then display the message "Well done you took [count] attempts".
50	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".
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 bottles should accidently fall". Then ask the question "how many green bottles will be hanging on the wall?" If they answer the correct amount 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.

Random

Explanation

Python can generate random values. In reality, it isn't completely random as no computer can cope with that; instead it uses a very complex algorithm which makes it virtually impossible for a human to accurately predict its outcome so, in effect, it acts like a random function.

There are two types of random value that can be generated:

- Random numbers within a specified range
- Make a random choice from a range of items that are inputted

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

Code	Explanation
`import random`	This must appear at the start of your program otherwise the random function will not work.
`num=random.randint(0,9)`	Selects a random whole number between 0 and 9 (inclusive) and stores it in a variable called "num".
`colour=random.choice( ["red", "black", "green"] )`	Picks a random value from the options "red", "black" or "green" and stores it as the variable called "colour".

Challenges

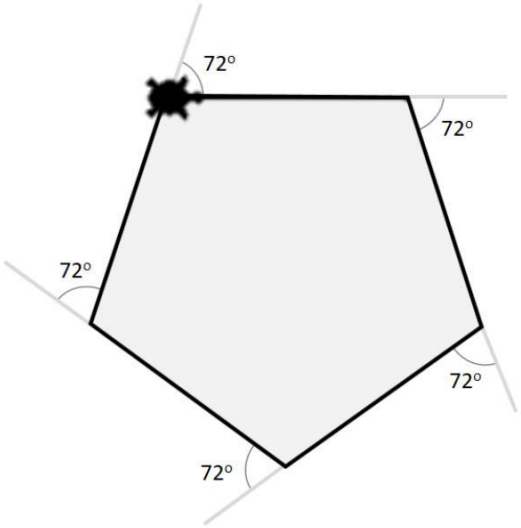
Number	Challenge
52	Print a random integer between 1 and 100 inclusive.
53	Print a random fruit from a list of 5 fruit.
54	Randomly choose either heads or tails ("h" or "t"). Ask the user to make their choice. If their choice is the same as the random selected value say "You win" otherwise display "Bad luck". At the end, display if it was 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 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 the challenge above so that it tells them if they are too high or too low before they pick again.
058	Make a maths quiz which will ask 5 questions by randomly generating two numbers to make the question (i.e. [num1] + [num2]). Ask the user to enter the answer. If they get it correct add a point to their score. At the end of the quiz tell them how many they got correct out of 5.
59	Display 5 colours and ask the user to pick 1. If they pick correctly say "Well done" otherwise say a witty answer which involves the correct colour i.e. "I bet you are GREEN with envy" or "You are probably feeling BLUE right now". Ask them to guess again and if they have still not got it right keep giving them the same clue and asking them to enter a colour until they do guess it correctly.

Turtle Graphics

Explanation

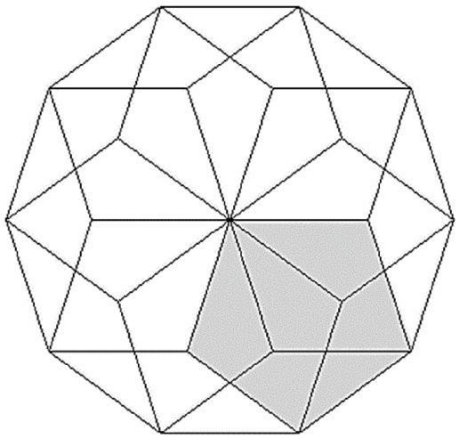
It is possible to draw using a turtle in Python. By typing in commands and using loops you can create intricate patterns. Here is how it works.

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 on the left you would type in the following code that is shown on the right.




```
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.

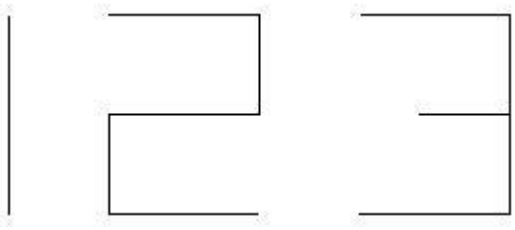
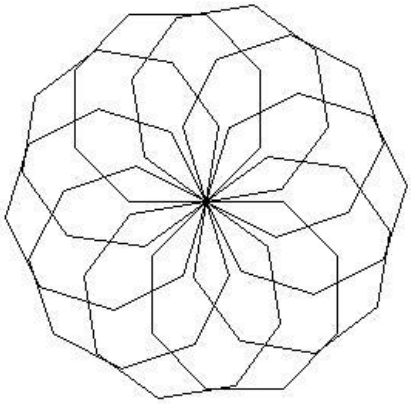


```
import turtle

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

turtle.exitonclick()
```

Example Code

Code	Explanation
`import turtle`	This line needs to be included at the beginning of your program to import the turtle library into Python so you can use the turtle functions.
`wn = turtle.Screen()` `wn.bgcolor("yellow")`	Defines the screen as being called "wn" and sets the screen background colour to yellow. By default this will be white unless it is changed.
`turtle.color("blue")`	Changes the turtle pen colour to blue. By default this will be black unless it is changed.
`turtle.pensize(3)`	Changes the turtle pen size (the thickness of the line that is drawn) to 3. By default this will be 1 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.forward(50)`	Moves the turtle forward 50 steps.
`turtle.left(120)`	Turns the turtle 120^{O} to the left (counter clockwise)
`turtle.right(90)`	Turns the turtle 90^{O} to the right (clockwise)
`turtle.showturtle()`	Shows the turtle on the screen. By default, the turtle is showing unless specified otherwise.
`turtle.hideturtle()`	Hides the turtle so it is not showing on the screen.
`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.exitonclick()`	When the user clicks on the turtle window it will automatically close.
`turtle.begin_fill()`	Entered *before* the code which draws a shape so it knows to fill in the shape it is drawing.
`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. This example will make the shape have a black outline and a red fill. This needs to be entered before the shape is drawn.

Challenges

Number	Challenge
060	Draw a square.
061	Draw a triangle.
062	Draw a circle.
063	Draw 3 squares in a line with a gap between each. Fill them using three different colours.
064	Draw a 5-pointed star.
065	Write the numbers as shown below starting at the bottom of the number one. 
066	Draw an octagon which uses a different colour (randomly selected from a list of 6 possible colours) for each line.
067	Create the following pattern: 
068	Draw a pattern which will change each time the program will run. Use the random function to pick the number of lines, the length of each line and the angle of each turn.

Tuples, Lists and Dictionaries

Explanation

So far, we have used variables which can store a single item of data in them. When you use the `random.choice(["red","blue","green"])` line of code you are picking a random item for a list of possible options which shows one item can hold several pieces of separate data.

There are several ways that groups of data can be stored as a single item and three of the simpler ones are:

- Tuples
- Lists
- Dictionaries

Tuples

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

Lists

The contents of a list can be changed while the program is running and this is one of the most common ways to store multiple sets of data in one item. 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.

Dictionaries

The contents of a dictionary can be changed while the program is running and 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.

Please note: In other programming languages the term “array” is often used and these work in a similar way to lists in Python. In Python, there is a data type called an array but these are only used to store numbers and we will look at these later on in the book.

Example Code

Code	Explanation
`fruit_tuple = ("apple","banana","strawberry","orange")`	This creates a variable name called "fruit_tuple" which stores the 4 separate pieces of fruit as separate items. The round brackets show it is a tuple and therefore the contents of this tuple cannot be altered while the program is running.
`print(fruit_tuple.index("strawberry"))`	This will display the index (i.e. the key) of the list of the item "strawberry". In this example it will return the number 2 as it starts counting the items from 0 and not 1.
`print(fruit_tuple[2])`	This will display item 2 from the "fruit_tuple", in this case "strawberry".
`names_list = ["John","Mark","Nicole","Fariah","Sam"]`	This will create a list of the names and stores them in the variable "names_list". The square brackets show it is a list and therefore the contents can be altered while the program is running.
`names_list.append(input("Add a name: "))`	This will ask the user to enter a name and will add that to the end of the "names_list".
`del names_list[2]`	This will delete item 2 from the "names_list". Remember it starts counting from 0 and not 1. In this case it will delete "Nicole" from the list.
`sorted_list = sorted(names_list)`	This will sort the "names_list" into alphabetical order and store it as a new variable called "sorted_list".
`colour_dictionary = {1:"red",2:"blue",3:"green"}`	This creates a dictionary called "colour_dictionary" where each item is assigned an index or key of your choosing. The first item in each block is the key, separated by a colon and then the item.
`colour_dictionary[2] = "yellow"`	This will change item 2. In this case it will change "blue" to "yellow".

More list example code

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

Code	Explanation
`x = ["red","blue",176,263,"green"]`	Creates a list called x which allows different data types to be added. This can cause problems and is not recommended if you wish to sort the list.
`print(x[1:4])`	This will display data in positions 1, 2 and 3. In this case "blue", 176 and 263. Remember Python starts counting from 0.
`x[2] = 472`	This changes the value at position 2.
`del x[2]`	Deletes the data from position 2.
`print(x.index("blue"))`	Displays the automatically assigned index number/position of an item in the list.
`print (len(x))`	Displays the length of the list (i.e. how many items are in the list).

`if "red" in x:`
`  print("It is in the list")`
`else:`
`  print("Not in the list")`
	Checks to see if the value is in the list and displays an appropriate message.

`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.
`x.append(372)`	Adds an item to the end of the list.
`x.insert(2,"pink")`	Inserts an item into a set position in the list and pushes everything else along to make space. This will change their index numbers according to their new position in the list.
`x.remove("red")`	Deletes an item from the list.
`x.sort()`	Sorts the list but does not cope if your list contains a mixture of data types (i.e. strings and integers in the same list).

Challenges

Number	Challenge
69	Store a tuple containing a list of 5 countries. Display the whole tuple. Ask the user to enter one of the countries from the tuple and then display the location of that item in the tuple.
070	Add to the program above to ask the user which number they want and display the country in that position.
071	Create a list of 2 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 6 school subjects. Ask the user which of these subjects they don't like. Delete the subject from the list and display the list again.
073	Ask the user to enter 4 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 it.
74	Enter a list of 10 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 inputted.
075	Create a list of 4 three-digit numbers. Display the list to the user (showing on separate lines) and 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. Ask them if they want to add another. If they do, allow them to add more names until they answer "no". At the end tell them how many people they have invited to the party.
77	Change the program for the above challenge. Once they have added their list of names, display the full list and ask them to type in one of the names on the list. Display that position on 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.
78	Display a list of 4 TV programs on separate lines. Ask the user to enter another show and a position they want it inserted into the list. Display the list again on separate lines
079	Create an empty list called "nums". Ask the user to enter numbers. After each number is entered add it to the nums list and display the list. Once they have entered 3 numbers ask them if they still want the last number they entered saved. If they say "no" remove the last item from the list.

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 position known as 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	!

Example code

Code	Explanation

`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 lowercase.

`msg = "Hello"`
`for letter in msg:`
`  print(letter,end="*")`
	Displays the message and between each character it will display a “*”.

`msg = "  This is some text  ".strip()`	Removes spaces at the beginning and end of a string but will not remove any spaces between words.

Please note: in the above examples “msg” is a variable name.

Challenges

Number	Challenge
080	Ask the user to enter their first name and 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 it. Finally display the length of their full name (including the space).
081	Ask them to type in their favourite school subject. Display it with "-" after each letter e.g. S-p-a-n-i-s-h-.
082	Show them a line of text from your favourite poem and ask for a starting and ending point. Display the characters between those two points.
83	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.
84	Ask the user to type in their postcode. Display the first two letters in uppercase.
085	Ask the user to type in their name and then tell them how many vowels are in their name.
086	Ask the user to enter a new password. Ask them to enter it again. If the two passwords match display "Thank you", otherwise 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 them to type in a word and then display it backwards. For instance, if they type in "Hello" it should display "olleH" on separate lines as shown below:
	`Enter a word: Hello`
	`o`
	`l`
	`l`
	`e`
	`H`
	`>>>`

Numeric Arrays

Explanation

Earlier in the book we looked at lists. Lists can store a jumble of different types of data at the same time however, this can cause problems when we try to sort the data. Arrays are very similar to lists but they are primarily 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 (see table below).

Here is a handy reference guide of the most common data types that an array in Python will allow:

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	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} (i.e. allows up to 308 numeric characters including a single decimal point anywhere in that number and can be negative or positive value)	8

When you create your array you need to define the type of data it will contain. It will not allow this to be altered or changed 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) then you cannot then 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 and lists allow the storage of any data type. If you want to create an array of characters (text), in Python you need to create a list rather than an array.

Example code

Code	Explanation
`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 5 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 a new number: "))` `nums.append(newvalue)`	Asks the user to enter a new number and it will add this to the end of the existing array.
`nums = sorted(nums)`	Sorts the array into ascending order.
`nums.reverse()`	Reverses the order of the array.

`newarray = array('i',[])`
`more = int(input("How many more items do you want to add:"))`
`for y in range(0,more):`
`  newvalue=int(input("Enter a new number: "))`
`  newarray.append(newvalue)`
`nums.extend(newarray)`
	Creates another array called "newarray" which also uses the integer data type. It asks the user how many more items they want to add. It will ask them to add the new values and store this in newarray. After all the items have been added it will join together the contents of the newarray onto the nums array.

`getrid = int(input("Enter the item you want to get rid of: "))` `nums.remove(getrid)`	Asks the user to enter the item they want to get rid of and then remove the first item that matches that value item from the array.
`print(nums.count(45))`	This will print out how many times the value "45" appears in the array.
`nums.pop()`	This will remove the last item from the array.

Challenges

Number	Challenge
088	Ask the user for a list of 5 whole numbers. Store them in an array. Sort the list and display it in reverse order.
89	Create an array which will store a list of numbers. Generate 5 random numbers and store them in the array. Print the array (displaying each item on a separate line).
90	Ask the user to enter numbers. If they enter a number between 10 and 20 then save it in the array, otherwise display the message "Outside the range". Once 5 numbers have been successfully added display the message "Thank you".
91	Create an array which contains 5 numbers (two of which should be repeated). Ask the user to enter one of the numbers and then display a message saying how many times that number appears in the list.
092	Create two arrays (one containing 3 numbers that the user enters and one containing a set of 5 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.
93	Ask the user to enter 5 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 5 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 do select a relevant item.
095	Create an array of 5 numbers between 10 and 100 which each contain 2 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 do 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.

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 many people would like to store strings, Python lists tend to be used by many Python programmers instead.

If you wanted to store 4 students' grades for 3 different subjects on paper, you may create a table 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 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

<table>
<tr><th>Code</th><th>Explanation</th></tr>
<tr><td>

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

</td><td>Creates a 2D list which uses standard Python indexing for the rows and columns.</td></tr>
<tr><td>

```
print(simple_array)
```

</td><td>Displays all the data in the 2D list.</td></tr>
<tr><td>

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

</td><td>Displays data from row 1 in this case [3, 7, 4].</td></tr>
<tr><td>

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

</td><td>Displays data from row 1 column 2, in this case 4.</td></tr>
<tr><td>

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

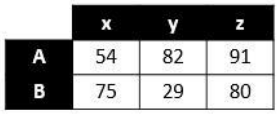
</td><td>Changes the data in row 1, column 2 to the value 5.</td></tr>
<tr><td>

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

</td><td>Adds the value 3 onto the end of the data in row 1 so in this case it becomes [3, 7, 5, 3].</td></tr>
<tr><td colspan="2">

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

Creates a 2D dictionary using user defines labels for the rows and columns.



	x	y	z
A	54	82	91
B	75	29	80

</td></tr>
<tr><td>

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

</td><td>Displays data from data set “A”.</td></tr>
<tr><td>

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

</td><td>Displays data from row “B”, column “y”.</td></tr>
<tr><td>

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

</td><td>Displays the “y” column from each row.</td></tr>
<tr><td>

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

</td><td>Changes the data in “B” “y”, to 53</td></tr>
<tr><td>

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

</td><td>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.</td></tr>
<tr><td>

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

</td><td>Displays only the name and the English grade for each student.</td></tr>
<tr><td>

```
del list[getrid]
```

</td><td>Removes a selected item.</td></tr>
</table>

Challenges

<table>
<tr><th>Number</th><th>Challenge</th></tr>
<tr><td>096</td><td>Create the following using a simple 2D list using the standard Python indexing:
<table><tr><th></th><th>0</th><th>1</th><th>2</th></tr><tr><td>0</td><td>2</td><td>5</td><td>8</td></tr><tr><td>1</td><td>3</td><td>7</td><td>4</td></tr><tr><td>2</td><td>1</td><td>6</td><td>9</td></tr><tr><td>3</td><td>4</td><td>2</td><td>0</td></tr></table></td></tr>
<tr><td>097</td><td>Using the 2D list from the example above, ask the user to select a row and a column and display that value.</td></tr>
<tr><td>98</td><td>Using the 2D list in the first example, 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.</td></tr>
<tr><td>099</td><td>Ask the user which row they want displayed. Display that row. Ask the user 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 they whole row again.</td></tr>
<tr><td>100</td><td>Create the following using a 2D dictionary showing the sales each person has made in the different regions:
<table><tr><th></th><th>N</th><th>S</th><th>E</th><th>W</th></tr><tr><td>John</td><td>3056</td><td>8463</td><td>8441</td><td>2694</td></tr><tr><td>Tom</td><td>4832</td><td>6786</td><td>4737</td><td>3612</td></tr><tr><td>Anne</td><td>5239</td><td>4802</td><td>5820</td><td>1859</td></tr><tr><td>Fiona</td><td>3904</td><td>3645</td><td>8821</td><td>2451</td></tr></table></td></tr>
<tr><td>101</td><td>Ask the user for a Sales person and a region. Display the relevant data. Ask the user for the Sales person 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 Sales person they identified.</td></tr>
<tr><td>102</td><td>Ask the user to enter data for 4 people. They need to enter the name, age and shoe size. Ask for the name of the person and display their age and shoe size.</td></tr>
<tr><td>103</td><td>For the data you gathered in the above example, display the names and ages of all the people in the list but do not show their shoe size.</td></tr>
<tr><td>104</td><td>After gathering the 4 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.</td></tr>
</table>

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 necessary to save the data outside of the program and this way the data can be saved, 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 to use you must specify how that file will work. 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.

Example code

Code	Explanation

`file = open("Countries.txt","w")`
`file.write("Italy\n")`
`file.write("Germany\n")`
`file.write("Spain\n")`
`file.close()`
	Creates a file called “Countries.text”. If one already exists then it will be overwritten with this new 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.

`file = open("Countries.txt", "r")` `print (file.read())`	This will open the Countries.txt file in “read” mode. It will 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. It will add another line and then close the file.

Challenges

Number	Challenge
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 check that the file has been created properly. The easiest way to do this using a Windows system is to load Notepad. Find the file and open it to see the contents.
106	Write 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 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 out 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.

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 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 like this:

	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 below:

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

Code	Explanation
`import csv`	This needs to be the first line 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". If there is already one there it will overwrite the original. It will also add a new record.

`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 and 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 and will display all rows that contain that data.

`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 altered using the skills from the list chapter we looked at earlier.

`file = open("NewStars.csv","w")`
`x = 0`
`for row in tmp:`
`  newrecord = tmp[x][0]+", "+tmp[x][1]+", "+tmp[x][2]+"\n"`
`  file.write(newrecord)`
`  x = x + 1`
`file.close()`
	Writes from a list into a new .csv file called "NewStars.csv"

Challenges

<table>
<tr><th>Number</th><th>Challenge</th></tr>
<tr><td>111</td><td>Create a .csv file that will store the following data. Call it “Books.csv”
<table>
<tr><th></th><th>Book</th><th>Author</th><th>Year Released</th></tr>
<tr><td>0</td><td>To Kill A Mockingbird</td><td>Harper Lee</td><td>1960</td></tr>
<tr><td>1</td><td>A Brief History Of Time</td><td>Stephen Hawking</td><td>1988</td></tr>
<tr><td>2</td><td>The Great Gatsby</td><td>F. Scott Fitzgerald</td><td>1922</td></tr>
<tr><td>3</td><td>The Man Who Mistook His Wife for a Hat</td><td>Oliver Sacks</td><td>1985</td></tr>
<tr><td>4</td><td>Pride and Prejudice</td><td>Jane Austen</td><td>1813</td></tr>
</table></td></tr>
<tr><td>112</td><td>Using the Books.csv file from the first challenge, 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.</td></tr>
<tr><td>113</td><td>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 books in the list by that author. If there are no books by that author in the list display a suitable message.</td></tr>
<tr><td>114</td><td>Ask the user to enter a starting year and an ending year and display all books released between those two years.</td></tr>
<tr><td>115</td><td>Display the data in the file along with the row number of each.</td></tr>
<tr><td>116</td><td>Import the data from the .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 into the original .csv file.</td></tr>
<tr><td>117</td><td>Create a simple maths quiz which 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.</td></tr>
</table>

Functions

Explanation

Functions 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 once 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 function and passing variables between functions

Below is a simple program that we could create without functions but have written it with functions 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 functions get_name(), print_Msg() and main().

The **get_name()** function 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 function. This is very important. If you do not return the values then the content of whatever happened in that function cannot be used elsewhere in your program.

The **print_Msg()** function will display the message "Hello" and then the user name. The variable "user_name" appears in the brackets as the current value of the variable is being imported into the function so it can be used.

The **main()** function will get the user_name from the get_name() function (using the variable user_name) as this was returned from the get_name () function. It will then use that user_name variable in the print_Msg() function.

The last line "main()" is the actual program itself and all this will do is start the main() function running.

Obviously, there is no need to create such a convoluted way of performing what is in fact a very simple program but this is only used as an example of how functions are laid out and variables can be used and passed between the functions.

Please note: When calling a function it must be written ABOVE the section of code you use to call it. Python will read from the top down and *run* the first line it comes across which has not been indented and does not start with the word def. In the program above this would be main().

However, in order to call a function it must have previously read it (i.e. it must have come across the def line with the correct function name in order to know to go back to that section). In this example, main() must be the last line in the program so it can call the other three functions which have been read first by Python before it can run them.

Example code

Code	Explanation

`def get_data():`
`  user_name = input("Enter your name: ")`
`  user_age = int(input("Enter age: "))`
`  data_tuple = (user_name, user_age)`
`  return data_tuple`
	Defines a function called "get_data()" which will ask the user for their name and age. As we want to send more than one piece of data back to the main program we have combined them into a tuple and returned the tuple. The return line can only return one value which is why we combined the user_name and user_age variables into a tuple called data_tuple and returned that.

`def message(user_name, user_age):`
`  if user_age <=10:`
`  print("Hi", user_name)`
`  else:`
`  print("Hello", user_name)`
	Defines a function called message() which uses two variables that have previously been defined (user_name and user_age).

`def main():`
`  user_name,user_age = get_data()`
`  message(user_name, user_age)`
	Defines a function called main() which obtains the two variables from the get_data() function which were saved in the tuple. Make sure these are retrieved in the same order as they were defined. It then calls the message() function to run with the two variables.

`main()`	Runs the main() function.

Challenges

Number	Challenge
118	Define a function that will ask the user to enter a number and save it as the variable "num". Define another function that will use "num" and count from 1 to that number.
119	Define a function that will ask the user to pick a low and a high number and then it will generate a random number between those two values and store it in a variable called "comp_num". Define another function that will give the instruction "I am thinking of a number…" and then ask the user to guess the number they are thinking of. Define a third function which will check to see if the comp_num is the same as the user's guess. If it is, it should display the message "Correct, you win" otherwise it will keep looping, telling the user if they are too low or too high and asking them to guess again until they guess correctly.

120	Display the following menu to the user:
	
	`1) Addition`
	`2) Subtraction`
	`Enter 1 or 2:`
	
	If they enter a 1 then it will run a function which will generate two random numbers between 5 and 20 and ask the user to add them together. Work out the correct answer and return both the user's answer and the correct answer.
	
	If they entered 2 as their selection on the menu, then it will run a function which will generate one number between 25 and 50 and another number between 1 and 25 and ask them to work out num1 minus num2. This way they should not have to worry about negative answers. Return both the user's answer and the correct answer.
	
	Create another function that will check if the user's answer matches the actual answer. If it does, display "Correct", otherwise display a message that will say "Incorrect, the answer is" and display the real answer.
	
	If they do not select a relevant option on the first menu you should display a suitable message.

121	Create a program that will allow the user to easily manage a list of names. You should display a menu which will allow them to easily add a name to the list, change a name in the list, delete a name from the list or view all the names in the list. There should also be a menu option to end the program. If they select an option which is not relevant then it should show them a suitable message. After they have made a selection to either add a name, change a name, delete a name or view all the names they should see the menu again without having to restart the program. The program should be made as easy to use as possible.

Number	Challenge
122	Create the following menu: 1) Add to file 2) View all records 3) Quit program Enter the number of your selection: If they select 1 allow them to add to a.csv file called Salaries.csv which will store their name and salary. If they select 2 it should display all records in the Salaries file. If they select 3 it should stop the program. If they select an incorrect option they should see an error message. They should keep returning to the menu until they select option 3.
123	In Python, it is not technically possible to directly delete a record from a .csv file. Instead you need to save the file to a temporary list in python, make the changes to the list and then overwrite the original file with the temporary list. Change the previous program to allow you to do this. Your menu should now look like this: 1) Add to file 2) View all records 3) Delete a record 4) Quit program Enter the number of your selection:

TKinter GUI (Graphical User Interface)

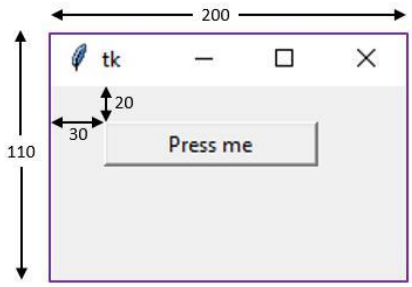
Explanation

A GUI (Graphical User Interface) makes the program easier to use for the user. It allows you, as the programmer, to create screens, text boxes and buttons to help the user navigate through the program in a more user-friendly way. TKinter is a library of features in Python that allows you to do this.

Look at the code below and in particular the measurements that are used in the window.geometry and button.place lines.

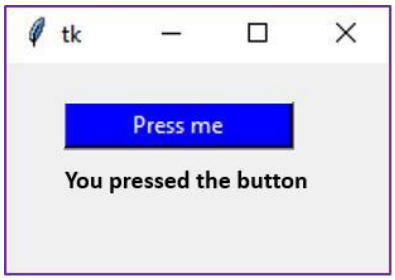
```
from tkinter import *

def Call():
        msg= Label(window, text = "You pressed the button")
        msg.place(x = 30, y = 50)
        button["bg"] = "blue"
        button["fg"] = "white"

window = Tk()
window.geometry("200x110")
button = Button(text = "Press me", command = Call)
button.place(x = 30, y = 20, width=120, height=25)
window.mainloop()
```

Now look at the window that this code will produce:



The geometry line in the code determines the size of the window and the place line in the code determines the position of the individual item on the window.

Once the button is pressed it will run the Call function and change the window to look as follows:



Example code

Code	Explanation
`from tkinter import *`	This line must go at the beginning of the program to import the TKinter library.

`window = Tk()`
`window.title("Add title here")`
`window.geometry("450x100")`
	Creates a window, referred to as "window", adds a title and defines the size of the window.

`label1 = Label(text = "Enter a number:")`	Creates a label on the screen displaying the message shown.
`entry_box = Entry(text = 0)`	Creates a blank entry box. Entry boxes can be used by the user to input data or used to display output.
`output_box = Message(text = 0)`	Creates a message box.
`list_box = Listbox()`	Creates a list box which is only used for output.
`button1 = Button(text = "Click here", command = click)`	Creates a button which will run the function click().
`label1.place(x = 50, y = 20, width=100, height=25)`	Specifies the position the object will appear in the window.
`output_box["bg"] = "yellow"`	Specifies the background colour.
`output_box["fg"] = "blue"`	Specifies the font colour.
`output_box["relief"] = "sunken"`	Specifies the style of the box. This can be flat, raised, sunken, groove and ridge.
`entry_box["justify"] = "center"`	Specifies the justification of the text in an entry box but does not work on message boxes.
`entry_box.delete(0, END)`	Deletes the contents of an entry or list box
`textbox2.insert(END, message)`	Appends a message to the end of the text in an entry box or a list box but it will not overwrite what is already there.
`num = entry_box.get()`	Obtains the contents of an entry box and stores it in a variable called num. This does not work with message boxes.
`answer = output_txt["text"]`	Obtains the contents of a message box and stores it in a variable called answer. This does not work with an entry box.
`output_txt["text"] = total`	Changes the content of a message box to display a new message.
`window.mainloop()`	This must be at the end of the program to make sure it keeps working.

Challenges

Number	Challenge
124	Create a window that will ask the user to enter their name. They should click on a button and it should display the message “Hello” and their name and change the background colour and font colour of the message box.
125	Write a program which can be used instead of rolling a 6-sided dice in a board game. When the user clicks a button it should display a random number between 1 to 6 (inclusive).
126	Create a program which will ask the user to enter a number in a box. When they click on a button it will add that number to a total and display it in another box. This can be repeated as many times as they want and keep adding to the total. There should be another button that resets the total back to 0 and empties the original text box ready for them to start again.
127	Create a window which will ask the user to enter a name in a text box and when they click on a button it will add it to the end of the list that is displayed on the screen. Add another button which will clear the list.
128	1 kilometre = 0.6214 mile and 1 mile = 1.6093 kilometres. Using these figures make a program which will allow the user to convert between miles and kilometres.
129	Create a window which will ask the user to enter a number in a text box and when they click on a button it will use the code `variable.isdigit()` to check to see if it is a number. If it is a number then add it to a list box, otherwise clear the entry box. Add another button which will clear the list.
130	Alter the program above to add a third button which will save the list to a .csv file. `tmp_list=num_list.get(0,END)` can be used to save a list as a tuple called tmp_list.
131	Create a program that will allow the user to create a new .csv file. It should ask them to enter the name and age of a person and then allow them to add this to the end of the file they have just created.
132	Using the .csv file you created for the last challenge, create a program which will allow people to add names and ages to the list and create a button which will display the contents of the .csv file by importing it to a list box.

More GUI interfaces with TKinter

Explanation

Here we will look at creating a GUI which includes more features and builds on the knowledge from the previous chapter.

On this screen we have:

- Changed the icon on the title bar
- Changed the background colour of the main window
- Added an image of the logo to the top left which will not change
- Created a label which at the moment displays “Hello”
- Added a Click Me button
- Added a drop down option entitled Select Name which will display three names “Bob”, “Sue” and “Tim” to the user
- Added a second image in the lower half of the window which will change to show the photograph of the person selected from the drop down list when the user clicks on the Click Me button

All the code to create this window can be created using the code we looked at in the previous section and the example code you will be looking at on the next page.

When using images in your program it is easier if they are stored in the same folder as the program, otherwise you would have to include the entire directory location of the file as follows:

```
logo = PhotoImage(file="c:\\Python34\images\logo.gif")
```

If you store the image in the same folder as the program you only need to include the file name as shown below:

```
logo = PhotoImage(file="logo.gif")
```

Please note: it is only possible to use GIF or PGM/PPM file types of images in TKinter as other file types are not supported.

Example code

Code	Explanation
`window.wm_iconbitmap("MyIcon.ico")`	Changes the window icon. You can design your own free icon image using the website **http://www.rw-designer.com/online_icon_maker.php** *(correct as of July 2017)*
`window.configure(background="light green")`	Changes the background colour of the window, in this case to light green.

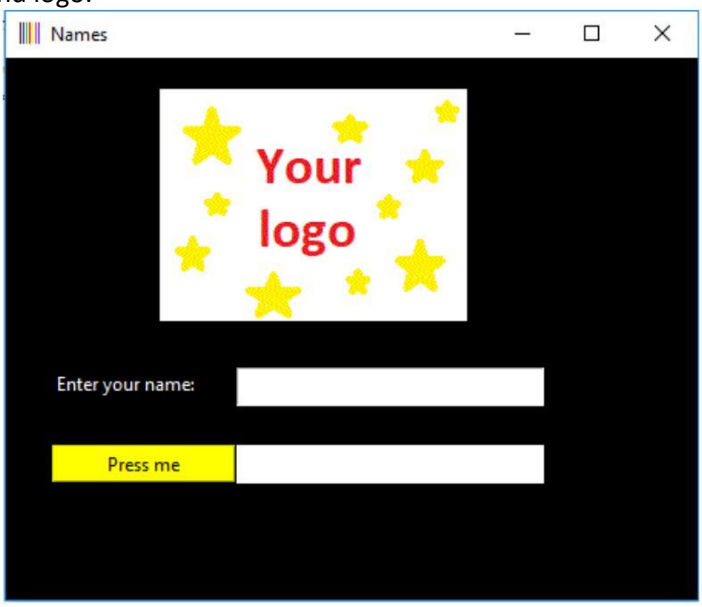
`logo = PhotoImage(file="logo.gif")`
`logoimage = Label(image=logo)`
`logoimage.place(x = 30, y = 20, width=200, height=120)`
	Displays an image in a label widget. This image will not change while the program is running.

`photo = PhotoImage(file="logo.gif")`
`photobox = Label(window, image=photo)`
`photobox.image = photo`
`photobox.place(x = 200, y = 200, width=200, height=120)`
	This is similar to the block above but as we want the image to change as we update the data we need to add the line `photobox.image=photo` otherwise it will not be updateable and we will only see a blank box.

`selectname = StringVar(window)`
`selectname.set("Select Name")`
 	
`nameslist = OptionMenu(window, selectname, "Bob","Sue","Tim")`
`nameslist.place(x = 30, y = 250)`
	Creates a variable called "selectname" which will store a string. The starting value of the variable is "Select Name". It will then create a drop down option menu which stores the value the user selects in the variable "selectname" and displays the values in the list Bob, Sue and Tim.

`def clicked():`
`  sel = selectname.get()`
`  mesg = "Hello " + sel`
`  mlable["text"]=mesg`
`  if sel == "Bob":`
`  photo=PhotoImage(file="Bob.gif")`
`  photobox.image = photo`
`  elif sel == "Sue":`
`  photo=PhotoImage(file="Sue.gif")`
`  photobox.image = photo`
`  elif sel == "Tim":`
`  photo=PhotoImage(file="Tim.gif")`
`  photobox.image = photo`
`  else:`
`  photo=PhotoImage(file="logo.gif")`
`  photobox.image = photo`
`  photobox["image"]=photo`
`  photobox.update()`
	In this example when a button is clicked it will run the "clicked" function. This will obtain the value from the "selectname" variable and create a message that will be displayed in a label. It will then check to see which option has been selected and change the picture to the correct image which is displayed in another label. If no name is selected it will simply show the logo.

Challenges

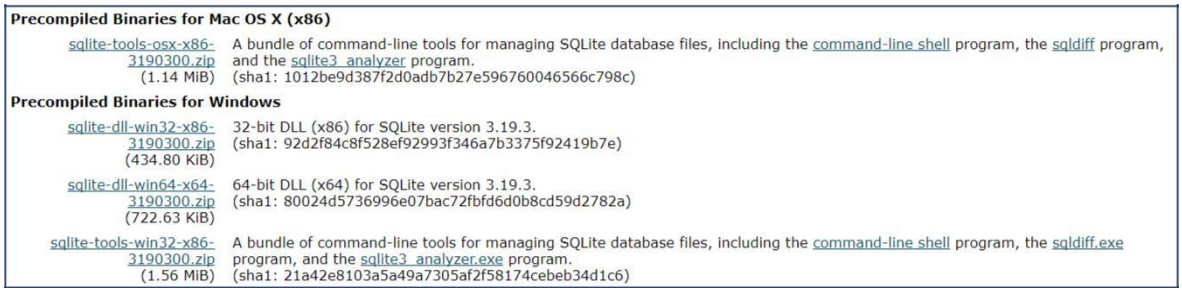
Number	Challenge
133	Create your own icon called stripes.ico which consists of several vertical multi-coloured lines. Create a logo using Paint or another graphics package which measures 200 x 150 called Mylogo.gif. Create the following window using your own icon and logo.  When they enter their name and click on the Press Me button it should display "Hello" and their name in the second box.
134	Create a new program that will generate two random numbers between 10 and 50. It should ask the user to add the numbers together and type in the answer. If they get the question correct display a suitable image, if they get the answer wrong display another suitable image. They should click on a Next button to get another question.
135	Create a simple program which shows a drop-down list containing several colours and a Click Me button. When the user selects a colour from the list and clicks the button it should change the background of the window to that colour. For an extra challenge, try to avoid using an if statement to do this.
136	Create a program that will ask the user to enter a name and then select the gender for that person from a drop-down list. It should then add the name and the gender (separated by a comma) to a list box when the user clicks on a button.
137	Change the previous program so that when a new name and gender is added to the list box it is also written to a text file. Add another button that will display the entire text file in the main Python shell window.
138	Save several images in the same folder as your program and call them 1.GIF, 2.GIF, 3.GIF etc. Make sure they are all GIF files. Display one in a window and ask the user to enter a number. It should then use that number to make the correct file name and display the correct image.

SQLite

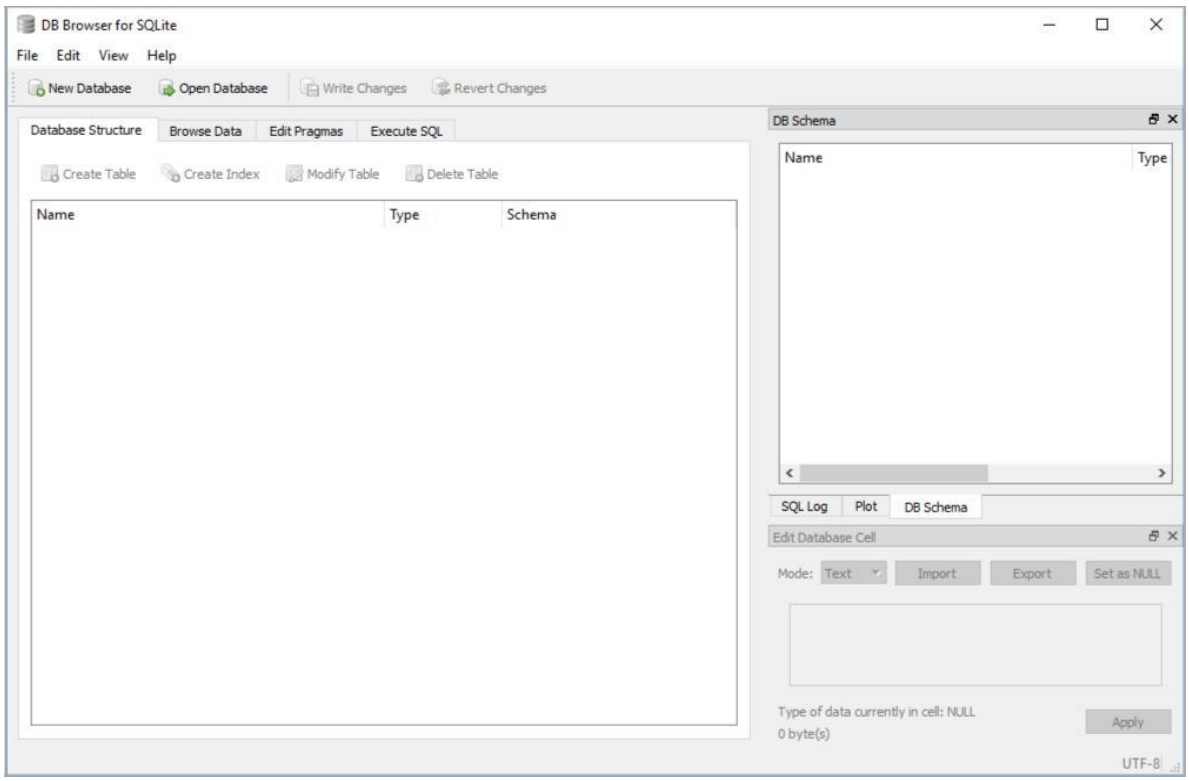
Explanation

SQL stands for "Structured Query Language" and is the main language that the large database packages use. SQLite is free software which can be used as a SQL database. You can download the software from ***https://www.sqlite.org/*** which will allow you to download the latest version.

From the download page you need to select one of the "Precompiled Binaries" option for either Mac or Windows.



To use SQL you need to load the "DB Browser for SQLite"



Understanding a relational database

We will use the example of a small manufacturing company which stores the details of their employees in an SQL database.

Below is an example of the Employees table which stores the details of all the employees in the company.

Table: Employees

	ID	Name	Dept	Salary
	Filter	Filter	Filter	Filter
1	1	Bob	Sales	25000
2	2	Sue	IT	28500
3	3	Tim	Sales	25000
4	4	Anne	Admin	18500
5	5	Paul	IT	28500
6	6	Simon	Sales	22000
7	7	Karen	Manufacturing	18500
8	8	Mark	Manufacturing	19000
9	9	George	Manufacturing	18500
10	10	Keith	Manufacturing	15000

It has 4 fields (ID, Name, Dept and Salary) and 10 records in it (one for each employee). Look at the employees list and you will notice that more than one employee has the same department. In most databases you would find repetitive data such as this and to make the database work more efficiently the repeated data is often linked to a separate table. In this case there is a department table which would store all the information about each department.

Table: Departments

	Dept	Manager
	Filter	Filter
1	Manufacturing	Kenith
2	Sales	James
3	IT	Connor
4	Admin	Sally

Here we can see who the manager is for each department and if we need to update the manager it only needs to be updated in one place. This is known as a one-to-many relationship as one department can have many employees in it.

A Primary Key is the field (usually the first one) in each table which stores the unique identifier for that record. Therefore, in the Employees table the Primary Key will be the ID column and in the Department table the Primary Key will be Dept.

When creating a table, you need to identify the following for each field:

- The name of the field (field names cannot contain spaces and follow the same rules as variable names)
- If it is a Primary Key (PK)
- The data type for that field

The data types are as follows:

- **Integer** - the value is an integer value
- **Real** - the value is a floating-point value
- **Text** - the value is a text string
- **Blob** - the value is stored exactly as it was inputted

You can also specify if the field cannot be left blank by adding NOT NULL to the end of the field when you create it.

Example Code

Code	Explanation
`import sqlite3`	This must be the first line of the program to allow Python to use the SQLite3 library.
`with sqlite3.connect("company.db") as db:` `  cursor=db.cursor()`	Connects to the company database. If no such database exists it will create one. The file will be stored in the same folder as the program.

`cursor.execute(""" CREATE TABLE IF NOT EXISTS employees(`
`id integer PRIMARY KEY,`
`name text NOT NULL,`
`dept text NOT NULL,`
`salary integer); """)`
	Creates a table called employees which has 4 fields (id, name, dept and salary). It specifies the data type for each field, defines which field is the Primary Key and which files cannot be left blank.

`cursor.execute(""" INSERT INTO employees(id,name,dept,salary)`
`VALUES("1","Bob","Sales","25000")`
`""")`
`db.commit()`
	Inserts data into the employees table.

`newid=input("Enter ID number: ")`
`newname=input("Enter name: ")`
`newdept=input("Enter dept: ")`
`newsalary=int(input("Enter salary: "))`
`cursor.execute("""INSERT INTO employees (id,name,dept,salary)`
`VALUES (?,?,?,?)""",(newid,newname,newdept,newsalary))`
`db.commit()`
	Allows a user to enter new data which is then inserted into the table.

`cursor.execute ("SELECT * FROM employees")` `print(cursor.fetchall())`	Displays all the data from the employees table.

`cursor.execute ("SELECT * FROM employees")`
`for x in cursor.fetchall():`
`  print(x)`
	Displays all the data from the employees table and displays each record on a separate line.

`cursor.execute ("SELECT * FROM employees ORDER BY name")`	Selects all the data from the employees table and sorts the names into alphabetical order.
`cursor.execute("SELECT * FROM employees WHERE salary>20000")`	Selects all the data from the employees table where the salary is over 20000.

Code	Explanation
`cursor.execute("SELECT * FROM employees WHERE dept='Sales'")`	Selects all the data from the employees table where the department is “Sales”.

`seldept=input("Enter a department: ")`
`cursor.execute("SELECT * FROM employees WHERE dept = ?", [seldept])`
`for x in cursor.fetchall():`
`  print(x)`
	Allows the user to type in a department and it displays the records of all the employees in that department.

`cursor.execute("SELECT id,name,salary FROM employees")`	Selects the id, name and salary fields from the employees table.
`cursor.execute("""SELECT employees.id,employees.name,dept.manager` `FROM employees,dept WHERE employees.dept=dept.dept""")`	Selects the id and name from the employees table and the manager from the department table, using the dept field in each table to link the data. If you do not specify how the tables are linked Python will assume every employee works in every department and you will not get the results you were expecting.

`cursor.execute("""SELECT employees.id,employees.name,dept.manager`
`  FROM employees,dept WHERE employees.dept=dept.dept`
`  AND employees.dept='Sales'""")`
`for x in cursor.fetchall():`
`  print(x)`
	Selects the id and name from the employees table and the manager from the department table if the department is “Sales”.

`cursor.execute("UPDATE employees SET name='Tony' WHERE id=1")` `db.commit()`	Updates the data in the table (overwriting the original data) to change the name to “Tony” for employee id 1.
`cursor.execute("DELETE FROM employees WHERE id=1")` `db.commit()`	Deletes the record for id 1
`db.close()`	This must be the last line in the program to close the database.

Challenges

<table>
<tr><th>Number</th><th>Challenge</th></tr>
<tr><td>139</td><td>Create an SQL database called PhoneBook which contains a table called Names with the following data:
<table>
<tr><th>ID</th><th>First Name</th><th>Surname</th><th>Phone Number</th></tr>
<tr><td>1</td><td>Simon</td><td>Howels</td><td>01223 349752</td></tr>
<tr><td>2</td><td>Karen</td><td>Phillips</td><td>01954 295773</td></tr>
<tr><td>3</td><td>Darren</td><td>Smith</td><td>01583 749012</td></tr>
<tr><td>4</td><td>Anne</td><td>Jones</td><td>01323 567322</td></tr>
<tr><td>5</td><td>Mark</td><td>Smith</td><td>01223 855534</td></tr>
</table></td></tr>
<tr><td>140</td><td>Using the database from above, write a program which will display the following menu.
<pre>
Main Menu

1) View phone book
2) Add to phone book
3) Search for surname
4) Delete person from phone book
5) Quit

Enter your selection:
</pre>
If the user selects 1, they should be able to view the entire phone book. If they select 2, it should allow them to add a new person to the phonebook. If they select 3, it should ask them for a surname and then display only the records of people with the same surname. If they select 4, it should ask for an id and then delete that record from the table. If they select 5, it should stop the program. Finally, it should display a suitable message if they enter an incorrect selection from the menu. They should return to the menu after each action, until they select 5.</td></tr>
</table>

<table>
<tr><th>Number</th><th>Challenge</th></tr>
<tr><td>141</td><td>
Create a new SQL database called BookInfo which will store a list of Authors and the books they wrote.

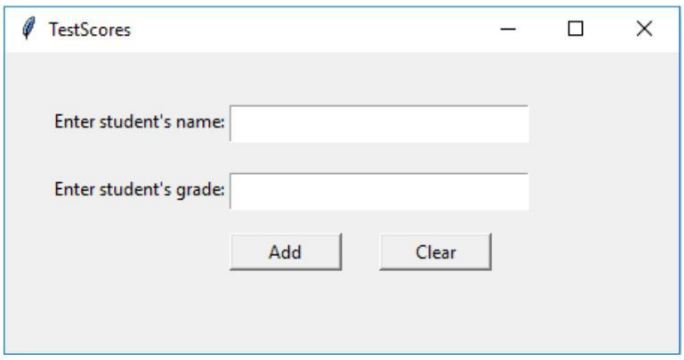
It will have two tables: the first one should be called Authors and contain the following data:

<table>
<tr><th>Name</th><th>PlaceofBirth</th></tr>
<tr><td>Agatha Christie</td><td>Torquay</td></tr>
<tr><td>Cecelia Ahern</td><td>Dublin</td></tr>
<tr><td>J.K. Rowling</td><td>Bristol</td></tr>
<tr><td>Oscar Wilde</td><td>Dublin</td></tr>
</table>

The second should be called Books and contain the following data:

<table>
<tr><th>ID</th><th>Title</th><th>Author</th><th>DatePublished</th></tr>
<tr><td>1</td><td>De Profundis</td><td>Oscar Wilde</td><td>1905</td></tr>
<tr><td>2</td><td>Harry Potter and the chamber of secrets</td><td>J.K. Rowling</td><td>1998</td></tr>
<tr><td>3</td><td>Harry Potter and the prisoner of Azkaban</td><td>J.K. Rowling</td><td>1999</td></tr>
<tr><td>4</td><td>Lyrebird</td><td>Cecelia Ahern</td><td>2017</td></tr>
<tr><td>5</td><td>Murder on the Orient Express</td><td>Agatha Christie</td><td>1934</td></tr>
<tr><td>6</td><td>Perfect</td><td>Cecelia Ahern</td><td>2017</td></tr>
<tr><td>7</td><td>The marble collector</td><td>Cecelia Ahern</td><td>2016</td></tr>
<tr><td>8</td><td>The murder on the links</td><td>Agatha Christie</td><td>1923</td></tr>
<tr><td>9</td><td>The picture of Dorian Gray</td><td>Oscar Wilde</td><td>1890</td></tr>
<tr><td>10</td><td>The secret adversary</td><td>Agatha Christie</td><td>1921</td></tr>
<tr><td>11</td><td>The seven dials mystery</td><td>Agatha Christie</td><td>1929</td></tr>
<tr><td>12</td><td>The year I met you</td><td>Cecelia Ahern</td><td>2014</td></tr>
</table>
</td></tr>
<tr><td>142</td><td>Using the database from the challenge above, display the list of authors and their place of birth. Ask the user to enter one place of birth and then show the title, date published and authors name for all the books for by authors who were born in that location.</td></tr>
<tr><td>143</td><td>Using the BookInfo database ask the user to enter a year and it will then display all the books published after that year, sorted by the year it was published.</td></tr>
<tr><td>144</td><td>Using the BookInfo database, ask the user for an Author's name and then save all the books by that author to a text file with each field separated by dashes so it looks as follows:

<pre>1 - De Profundis - Oscar Wilde - 1905
9 - The picture of Dorian Gray - Oscar Wilde - 1890</pre>
</td></tr>
</table>

Number	Challenge
145	Create a program that displays the following screen:  It should save the data to a SQL database called TestScores when the Add button is clicked. The Clear button should clear the window.

Part 2

Chunky Challenges

Introduction

In this section you are given some large programming challenges to work through. These will take longer and you may need to refer back through the book in order to remind yourself of some of the key skills you have covered so far. Don't feel bad if you need to refer back to previous sections; even experienced programmers look for help when they come across a tricky piece of code with which they are not familiar.

Each challenge contains a list of the skills that will be needed so you can decide if you feel ready to attempt the challenge. It also includes a description of the challenge and a section outlining problems you will have to overcome. The solutions in this section are much larger and may need to be split over several pages but should be read as a continuous single program for that challenge. If a program does need to be split onto separate pages we will try to split them between the functions if at all possible.

146 - Shift Code

In this challenge you will need to use the following skills:

- Input and display data
- Lists
- Splitting and joining strings
- If Statements
- Loops (while and for)
- Functions

The challenge

A shift code is where a message can be easily encoded and is one of the simplest codes to use. Each letter is moved forwards through the alphabet a set number of letters to be represented by a new letter. For instance, “abc” becomes “bcd” when the code is shifted by 1 (i.e. each letter in the alphabet is moved forward 1 character.

You need to create a program which will display the following menu:

```
1) Make a code
2) Decode a message
3) Quit

Enter your selection:
```

If the user selects 1, they should be able to type in a message (including spaces) and then enter a number. Python should then present them with the message once the shift code has been applied.

If the user selects 2, they should enter an encoded message and the correct number and it should display the decoded message (i.e. move the correct number of letters backwards through the alphabet).

If they select 3 it should stop the program from running.

After they have encoded or decoded a message the menu should be displayed to them again until they select quit.

Problems you will have to overcome:

Decide if you want to allow both upper and lowercase letters or if you want to convert all the data into one case.

Decide if you are allowing punctuation.

If the shift makes the letter go past the end of the alphabet it should start again. i.e. if the user enters “xyz” and 5 is entered as the shift number it should display “bcd”, for example. This should work the opposite way for decoding a message so if the value gets to “a” it will go back to “z”.

Make sure that suitable messages are displayed if the user selects an inappropriate option on the menu selects or an inappropriate number to make the shift code.

Test out your decode option by decoding the message “we ovugjohsslunl” which was created with the number 7.

147 - Mastermind

In this challenge you will need to use the following skills:

- Input and display data
- Lists
- Random choice from a list
- If statements
- Loops (while and for)
- Functions

The challenge

You are going to make an on-screen version of the board game “Mastermind”. The computer will automatically generate 4 colours from a list of possible colours (it should be possible for the computer to randomly select the same colour more than once from the list). For instance, the computer may choose “red”, “blue”, “red”, “green”. This sequence should not be displayed to the user.

After this is done the user should enter their choices for the 4 colours from the same list the computer used. For instance, they may choose “pink”, “blue”, “yellow” and “red”

After the user has made their selection, the program should display how many colours they got right in the correct position and how many colours they got right but in the wrong position. In the example above, it should display the message “Correct colour in the correct place: 1” and “Correct colour but in the wrong place: 1”.

The user continues guessing until they correctly enter the 4 colours in the order they should be in. At the end of the game it should display a suitable message and tell them how many guesses they took.

Problems you will have to overcome:

The hardest part of this game is working out the logic for checking how many they have correct and how many are in the wrong place. “Using the example above if the user enters “blue”, “blue”, “blue”, “blue” they should see the message “Correct colour in the correct place: 1” and “Correct colour but in the wrong place: 0”.

Decide if there is an easier way of allowing the user to enter their section (i.e. using a code or a letter to represent the colour). If using the first letter, make sure you only use colours which have a unique first letter (i.e. avoid using blue, black and brown as options and select just one of these as a possibility). Make your instructions clear to the user.

Decide if you want to allow upper and lowercase or if it is easier to convert everything to the same case.

Make sure you build in validation checks to make sure the user is only entering valid data and display a suitable message if they make an incorrect selection. If they do make an incorrect selection you may want to allow them to enter the data again.

148 - Passwords

In this challenge you will need to use the following skills:

- Input and display data
- Lists
- If statements
- Loops (while and for)
- Functions
- Saving to and reading from a .csv file

The challenge

You need to create a program that will store the user ID and passwords for the users of a system. It should display the following menu:

```
1) Create a new User ID
2) Change a password
3) Display all User IDs
4) Quit

Enter Selection:
```

If the user selects 1 it should ask them to enter a user ID. It should check if the user ID is already in the list. If it is, the program should display a suitable message and ask them to select another user ID. Once a suitable user ID has been entered it should ask for a password. Passwords should be scored with 1 point for each of the following: It should have at least 8 characters, it should include uppercase letters, it should include lowercase letters, it should include numbers and it should include at least one special character such as !, £, $, %, &, <, * and @. If the password scores only 1 or 2 it should be rejected saying it is a weak password; if it scores 3 or 4 tell them that "This password could be improved." And ask them if they want to try again. If it scores 5 tell them they have selected a strong password. Only acceptable user IDs and passwords should be added to the end of the .csv file.

If they select 2 from the menu they will need to enter a user ID, check to see if the user ID exists in the list and if it does, allow the user to change the password and save the changes to the .csv. Make sure the program only alters the existing password and does not create a new record.

If the user selects 3 display all the user IDs but not the passwords.

If the user selects 4 it should stop the program.

Problems you will have to overcome:

As existing data in .csv files cannot be edited and can only be read or added to you will need to find a way of importing the data as a temporary list into Python so you can make the changes before the data is written to the .csv file afresh.

Make sure only passwords belonging to an existing user ID can be altered.

Use suitable messages to guide the user easily through the system.

Repeat the menu until they quit the program.

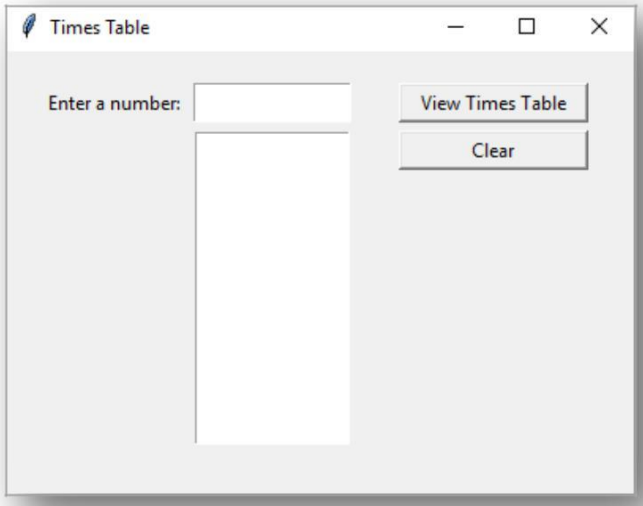
149 - Times Tables with a GUI Interface

In this challenge you will need to use the following skills:

- Loops
- Functions
- TKinter library

The challenge

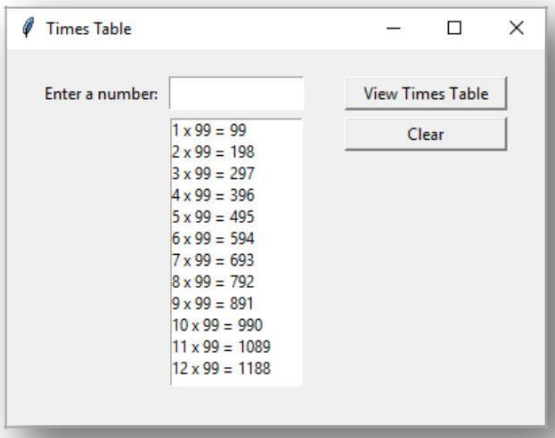
Create a program that will display the following screen:



When the user enters a number in the first box and clicks on the "View Times Table" button it should show the times table in the list area.

For instance, if the user enters 99 they would see the following list as shown in the example on the right.

The "Clear" button should clear both boxes.



Problems you will have to overcome:

You want to display the number sentence in the list rather than just the answers. The following line of code may help you do this:

```
num_list.insert(END,(i, "x", num, "=", answer))
```

Make sure it is as easy to use as possible by making sure the focus is in the correct location.

150 – Art Gallery

In this challenge you will need to use the following skills:

- TKinter
- SQLite 3

The challenge

A small art gallery is selling works from different artists and wants to keep track of the paintings using an SQL database. You need to create a user-friendly system to keep track of the art. This should include using a GUI. Below is the current data that needs to be stored in a database.

Artists Contact Details:

ArtistID	Name	Address	Town	County	Postcode
1	Martin Leighton	5 Park Place	Peterborough	Cambridge	PE32 5LP
2	Eva Czarniecka	77 Warner Close	Chelmsford	Essex	CM22 5FT
3	Roxy Parkin	90 Hindhead Road		London	SE12 6WM
4	Nigel Farnworth	41 Whitby Road	Huntly	Aberdeenshire	AB54 5PN
5	Teresa Tanner	70 Guild Street		London	NW7 1SP

Pieces of Art

PieceID	ArtistID	Title	Medium	Price
1	5	Woman with black Labrador	Oil	220
2	5	Bees & thistles	Watercolour	85
3	2	A stroll to Westminster	Ink	190
4	1	African giant	Oil	800
5	3	Water daemon	Acrylic	1700
6	4	A seagull	Watercolour	35
7	1	Three friends	Oil	1800
8	2	Summer breeze 1	Acrylic	1350
9	4	Mr Hamster	Watercolour	35
10	1	Pulpit Rock, Dorset	Oil	600
11	5	Trawler Dungeness beach	Oil	195
12	2	Dance in the snow	Oil	250
13	4	St Tropez port	Ink	45
14	3	Pirate assassin	Acrylic	420
15	1	Morning walk	Oil	800
16	4	A baby barn swallow	Watercolour	35
17	4	The old working mills	Ink	395

Problems you will have to overcome:

The user must be able to add new artists and pieces of art.

Once a piece has been sold the data should be stored in a text file and should be removed from the main SQL database

Users should be able to search by either artist, media or price.

What next?

If you have worked through all of the examples in this book you should have a good understanding of the basics of programming with Python. You will have become familiar with the syntax of the language and started to think like a programmer by breaking down larger problems into small manageable chunks you know how to solve.

The skills you have learnt in this book will allow you to create powerful programs but now is not the time to sit back and relax. You need to go out into the big programming world and find out how other programmers work. Search the internet, find new challenges. As you explore programming more you will see code that is unfamiliar to you as there are several variations that can be used. For instance, with TKinter there is another method called “pack” which many programmers prefer. It allows you to use a grid method for designing your screens but does not allow you to fine tune the position of an object that the place method we have been using allows. Be careful as some techniques do not work well with others. If you want to use the pack method then please don’t try to mix it with place method in the same program. Python doesn’t like working with two different systems simultaneously and will crash.

The best way to learn more advanced techniques it to try them out. Look at other people’s code and visits some chat forums. Programmers are very helpful and as long as you are not asking a question that has been already answered on the forum they are generally willing to help you out.

Whether you feel satisfied with your knowledge or want to explore it further I hope you have enjoyed your venture into programming and this book has proved useful.