OCR Computing

Work Book

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Form:\_\_\_\_\_\_\_\_\_\_

Teacher:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2014-15

## This Workbook

This workbook is very important, look after it carefully. When you do your controlled assessments you will be able to refer to it to help you.

## Help

**Ask** for help from **others**

Complete them in **sequence** and do not move on to the next challenge until you have successfully got the program running.

If possible try the extension tasks.

Make sure that you save your solutions to your Python Practice folder using the challenge title/topic as the file name.

Look at the programs that you have completed in previous lessons for help if you struggle

## Get Organised

Life will be much harder if you don’t save your work carefully.

Create the following folders:

My Work Folder

Computing GCSE

Python Practice

Programs

Theory

Contents

[This Workbook 2](#_Toc389819473)

[Help 2](#_Toc389819474)

[Get Organised 2](#_Toc389819475)

[Inputting information and strings 6](#_Toc389819476)

[Challenge 1 - Write a program that…. 6](#_Toc389819477)

[Challenge 2 - Write a program that… 6](#_Toc389819478)

[Working with INT and FLOAT to handle numbers in Python 7](#_Toc389819479)

[Challenge 3 - Write a program that…. 7](#_Toc389819480)

[Challenge 4 – Change the Program 7](file:///\\Server1b\users\ps36\Mywork\Computer%20Science\Programming\Learning%20python\Student%20Booklet%20for%20Sep%202014.docx#_Toc389819481)

[Challenge 5 – Change the Program 8](#_Toc389819482)

[Challenge 6 – Calorie Counter 10](#_Toc389819483)

[Making the program pause between messages 11](#_Toc389819484)

[Challenge 7 – Area Calculator 11](#_Toc389819485)

[Challenge 8 – Days Alive Calculator 11](#_Toc389819486)

[Some more about Strings 12](#_Toc389819487)

[Challenge 9 – I am an excellent programmer 12](#_Toc389819488)

[Challenge 10 – Concatenation 12](#_Toc389819489)

[How to use Python for math calculations 13](#_Toc389819490)

[Challenge 11 – Multiplication 13](#_Toc389819491)

[Challenge 12 – Dividing 13](#_Toc389819492)

[Challenge 13 – Modulus 13](#_Toc389819493)

[Challenge 14 – Addition 13](#_Toc389819494)

[Operands 14](#_Toc389819495)

[Challenge 15 - Formatting how numbers are printed on the screen 14](#_Toc389819496)

[Challenge 16 - Restaurant Tip 15](#_Toc389819497)

[Making decisions in program 16](#_Toc389819498)

[Challenge 17 – Magic 8 Ball 16](#_Toc389819499)

[Comparison Operators 17](#_Toc389819500)

[Challenge 18 – If 18](#_Toc389819501)

[Challenge 19 18](#_Toc389819502)

[Indenting 18](#_Toc389819503)

[More advanced ways to use an if 19](#_Toc389819504)

[Challenge 20 – Beauty products 19](#_Toc389819505)

[Not equal to: 19](#_Toc389819506)

[Challenge 21 – Mega Sale 21](#_Toc389819507)

[Challenge 22 – Happy Message 21](#_Toc389819508)

[Challenge 23 – Mobile Phone Costs 21](#_Toc389819509)

[Challenge 24 – Secret password 21](#_Toc389819510)

[Boolean or logical expression 22](#_Toc389819511)

[Challenge 25 – For loops 22](#_Toc389819512)

[Counting Loops 22](#_Toc389819513)

[22](#_Toc389819514)

[Challenge 26 – For loops 23](#_Toc389819515)

[Challenge 27 – Times Table using For loops 23](#_Toc389819516)

[For loops are *really* clever 23](#_Toc389819517)

[Challenge 28 – Reading a sentence 23](#_Toc389819518)

[Conditional Loops 24](#_Toc389819519)

[Challenge 29 – While loops 24](#_Toc389819520)

[Challenge 30 – Rumpelstiltskin 24](#_Toc389819521)

[Comments 25](#_Toc389819522)

[Challenge 31 – Rock, Paper, Scissors Game 26](#_Toc389819523)

[Extension: 26](#_Toc389819524)

[Further extension: 26](#_Toc389819525)

[Planning the Rock, Paper, Scissors Game 27](#_Toc389819526)

[Challenge 32 – Making a times table (using nested for loops) 29](#_Toc389819527)

[Extension: 29](#_Toc389819528)

[In Built Functions 30](#_Toc389819529)

[Functions 31](#_Toc389819530)

[Challenge 34 – Area or Perimeter 31](#_Toc389819531)

[Extension 32](#_Toc389819532)

[Passing Functions a Value by using Parameters 33](#_Toc389819533)

[Method 1 33](#_Toc389819534)

[Method 2 33](#_Toc389819535)

[Method 3 34](#_Toc389819536)

[Method 4 34](#_Toc389819537)

[Global variables 35](#_Toc389819538)

[Lists 36](#_Toc389819539)

[Challenge 35 36](#_Toc389819540)

[More lists 37](#_Toc389819541)

[Slicing lists 37](#_Toc389819542)

[Adding to a list: 38](#_Toc389819543)

[Deleting from a list: 38](#_Toc389819544)

[Random choice from a list 39](#_Toc389819545)

[Dictionaries 39](#_Toc389819546)

[Challenge 36 40](#_Toc389819547)

[Extension: 40](#_Toc389819548)

## C:\Users\jwardell\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\92R9WSNQ\MC900331536[1].wmfInputting information and strings

print() function

The print function prints information to the screen. Type this in:

print (“Hello World”)

Input() function

The input function allows the user to enter information into the computer. Type this in:

name = input("Hi. What's your name? ")

print ("Hello, ", name)

Variables

A variable is used in a program to store information. In the program above the variable is **name.** Because the variable is in quotes “” or ‘’, the variable type is called a **string**.

String

A string can contain letters, characters and numbers.

### Challenge 1 - Write a program that….

Use variable names:

firstName

surnameName

1. Asks for the first name and then a surname.
2. Print the First name and then the second name
3. On the next line print the second name and then the first name

Extension: The name may not have a space between first and second names,  
 can you work out how to add one?

Save as: Challenge 1 - Names

Strings can be joined together:

### Challenge 2 - Write a program that…

Use a new variable **fullName.** Add to your program:

1. fullName = firstName + surnameName
2. print (fullName)

Joining strings together is known as:

## Working with INT and FLOAT to handle numbers in Python

We have used the input command to ask the user to enter text. We’re going to use this again but with numbers. What do you think this code will do?

### Challenge 3 - Write a program that….

Type this in:

numberOne = input(“Enter a number between 0 and 10”)

numberTwo = input(“Enter a number between 0 and 10”)

print (numberOne + numberTwo)

What happened? Why do you think this happened?

### Challenge 4 – Change the Program

Type this in:

numberOne = int(input(“Enter a number between 0 and 10”))

numberTwo = int(input(“Enter a number between 0 and 10”))

print (numberOne + numberTwo)

What happened? Why do you think this happened?

Type in 5 and 5 as your numbers to test your program

Type in 5.5 and 4.5 to test your program again. You should receive an error message.

### Challenge 5 – Change the Program

Type this in:

numberOne = float(input(“Enter a number between 0 and 10”))

numberTwo = float(input(“Enter a number between 0 and 10”))

print (numberOne + numberTwo)

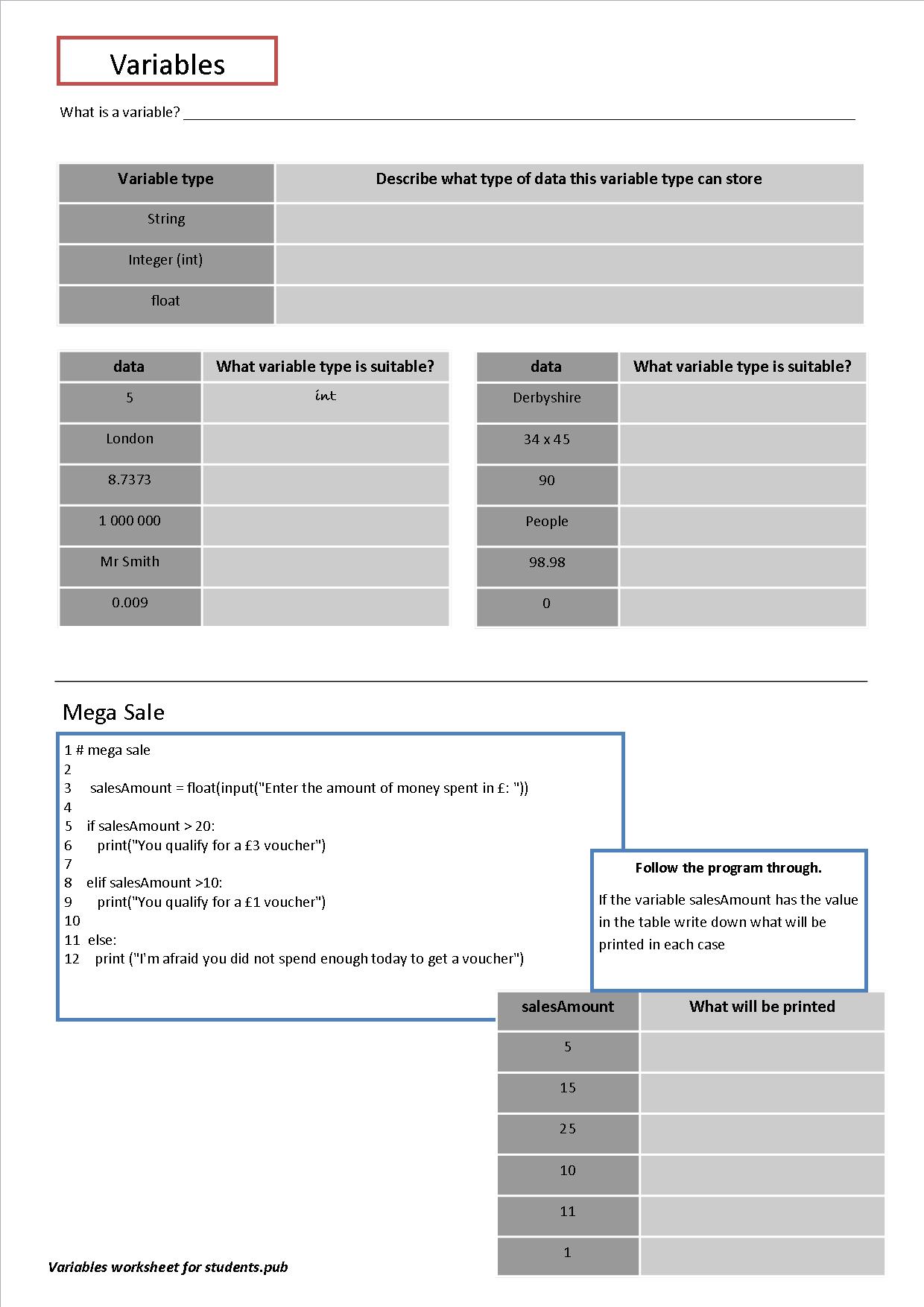
Type in 5 and 5 as your numbers to test your program

Type in 5.5 and 4.5 to test your program again. Did you get an error message?

What happened? Why do you think this happened?

**Update the**

**Log Book**



### Challenge 6 – Calorie Counter

Now let’s build a calorie counter. The NHS recommends that an adult male takes on board 2,500 calories per-day and an adult woman takes on 2,000 calories per-day. Build your program for a woman or a man.

Save your program as **CalorieCounter**

print("YOUR NAME’s calorie counter")

calories = int(input("How many calories have you eaten today?"))

s=2000-calories

print("You can eat", s, "more calories today")

**Challenge**

Write a program that asks the user to enter how much they have spent on their school dinner. It should subtract how much they’ve spent from the amount they had at the start of the day. Display the result on the screen.

Save your work as **SchoolDinner**

2. This asks the user to enter how many calories they have eaten as the **variable** c. It stores it as an integer (whole number)

1. Line 3 of the code subtracts how many calories they’ve eaten and stores it as the **variable** s

Line 4 then prints outs in between 2 ’strings’ of text

## Making the program pause between messages

The program works but it’s very fast so we can add some delays in to make it more user friendly.

Python doesn’t have all of the commands we need but we can add some like this:

**import time**

print("YOUR NAME’s calorie counter")

**time.sleep(3)**

calories = int(input("How many calories have you eaten today?"))

s=2000-calories

print("You can eat", s, "more calories today")

Now try these challenges. Think carefully – what type of variable will you need to use?

This tells the program to pause for 3 seconds before moving onto the next line of code.

**Try:** Adding your own delays into the program to see how it can make it a better program for the user.

When you do these next tasks remember that by ***default*** variables are strings.

### Challenge 7 – Area Calculator

Jennifer wants to carpet her new room with pink carpet. Create a program that will work out the area of any sized room, (length x width).

*Save as Square*

### Challenge 8 – Days Alive Calculator

Write a program to work out how many days you have been alive for (to the nearest year for the moment - there are 365 days in a year)

Get the program to ask for the person’s name and age.

Develop this to work out how many hours this is – 24 hours per day

Develop it further to work out how many minutes and seconds you have lived for – 60 minutes per hour / 60 seconds per minute.

Make sure the information is clearly displayed on the screen.

***Save as Age***

## Some more about Strings

Strings are variables that contain letters, numbers and symbols. There are lots of things that can be done with strings.

### Challenge 9 – I am an excellent programmer

Type this in:

Quote = “i am an excellent programmer”

print (Quote)

What is shown on the screen should be the same as you typed in.

Now try changing your program to this:

Type this in:

Quote = “i am an EXCELLENT programmer”

Print (Quote.lower())

This is known as a string **method**

Then instead of lower use these methods and write down what happens:

|  |  |
| --- | --- |
| **String method** | **Description of what it does** |
| lower() | **Update the**  **Log Book** |
| upper() |  |
| title() |  |
| swapcase() |  |
| capitalize() |  |

### Challenge 10 – Concatenation

Strings can be joined together – which is called concatenation.

Write a programme that asks for 3 gifts that someone would like for Christmas.

Add the variables (something like this) : giftTotal = gift1 + gift2 + gift3

Then print the variable giftTotal (make sure there are spaces between each word)

gift1 = input(“What gift would you like most”)

gift2 =

gift3 =

giftTotal = gift1+gift2+gift3

Save as **Christmas Gifts**

## How to use Python for math calculations

Let’s look at how python calculates using numbers.

### Challenge 11 – Multiplication

Let’s see how much a student spends on food at school. Do you remember why we have to use **int** or **float**?

food = float(input(“How much do you spend on lunchtime food in an average day?”))

food = food \*5

print (“This means that in a week you will spend “, food)

Save as Food spending

### Challenge 12 – Dividing

An Aunt wins on the lottery. She gives £1000 to you. You are thinking of sharing it with some of your family. How much would each person get (try different numbers of people to share it with). To get you started:

shares = int(input(“How many people do you want to share the money with?”))

money = 1000 / *????*

*finish this off*

Save as: sharing my money out

### Challenge 13 – Modulus

If we divide two numbers and they don’t divide evenly we get a remainder. Modulus gives us the remainder of a sum.

For example 7/2 = 3 with remainder 1. Try this:

Print (7 % 2)

You should get the answer 1. Try some other numbers to see what you get.

### Challenge 14 – Addition

Addition is easy. Try this:

length = 10

length = length + 20

What’s the value of the variable length:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Try it out to test it.

length = 10

length += 20

Another way of writing this in python is:

Test it out. It’s just quicker to write!

## Operands

The +, -, /, \* symbols are called operators. That’s because they “operate on” the numbers.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Example** | **Is Equivalent to** |
| \*= | K\*=5 | K = K \* 5  **Update the**  **Log Book** |
| /= | K/=5 |  |
| %= | K%=5 |  |
| += | K+=5 |  |
| -= | K-=5 |  |

### Challenge 15 - Formatting how numbers are printed on the screen

Printing information out accurately with the right number of digits is important.

Type this program in and see what happens to the number of numbers printed after the decimal point.

nOne = 3.123456789  
nTwo = 9.87654321

ans =(nOne\*nTwo)

print (ans)

#rounds the answer to 3 dps  
print (ans[:5])

#adding a zero to a decimal  
n3 =1.1  
print ("%0.2f"% (n3))

prints

prints

prints

**Update the**

**Log Book**

### Challenge 16 - Restaurant Tip

**The basics:**

Two people eat dinner at a restaurant and want to split the bill.

The total comes to £100 and they want to leave a 15% tip. How much should each person pay?

**TIP**. You can calculate an increase by multiplying the meal cost by 1.15.

**Harder**: Make the program ask how many people there are, what percentage the tip should be and how much the bill comes to.

**Harder still**: These diners would like to go home in a taxi. The taxi costs £0.45 per mile. Can you work out the taxi costs (the user should be able to input the distance they have to travel) and how much the complete evening has cost overall and per person?

Save this as Restaurant tip.

## Making decisions in program

Python can make decisions based on the input. To make decisions, programs check to see **if** a condition is true or not.

Python has a few ways to test something, and there are only two possible answers for each test: ***true* or *false***

### Challenge 17 – Magic 8 Ball

The code below will use Python’s **random number generator** to make a magic 8 ball game.

 Make this game and save it as **Magic8Ball.**

import random

answer1=(“Absolutely!”)

answer2=(“No way Pedro!”)

answer3=(“Go for it tiger.”)

print(“Welcome to the Magic 8 Ball game—use it to

answer your questions...”)

question = input(“Ask me for any advice and I’ll help you out. Type in your question and then press Enter for an answer.”)

print(“shaking.... \n” \* 4)

choice=random.randint(1,3)

if choice == 1:

answer=answer1

elif choice == 2:

answer=answer2

else:

answer=answer3

print(answer)

Can you modify this so that it chooses from 5 possible answers? Think of more elif’s.

Write into your Log Book what the lines of code are doing.

**Update the**

**Log Book**

For testing whether two things are equal Python uses a double equal sign (==) .

## Comparison Operators

|  |  |
| --- | --- |
| **Comparison Operators** | **What the symbols mean** |
| == | are two things equal? |
| != | are two things not equal? |
| < | less than |
| > | greater than |
| >= | greater than OR equal to |
| <= | less than OR equal to |

These comparison operators are important to learn and use correctly. Getting a symbol wrong will mean that the program does the wrong thing!

**Update the**

**Log Book**

### Challenge 18 – If

Let’s write a program that compares two numbers.

num1 = float(input("Enter the first number: "))

num2 = float(input("Enter the second number: "))

if num1 > num2:

print (num1, " is greater than ", num2)

if num1 < num2:

print (num2, " is greater than", num1)

if num1 == num2:

print (num1, "is equal to", num2)

Save this as: compare 2 numbers

This tests each combination and does something (prints a line out) if the test is **TRUE**

Python can do other things if the test is **FALSE**:

* *Do another test* – If the test is False we can get Python to another test: we use **elif** (short for “else if”)
* *Do something else* – If all the other tests come out false do something else: we use **else**

Challenge 19

We can re-write the code to use these features:

num1 = float(input("Enter the first number: "))

num2 = float(input("Enter the second number: "))

if num1 > num2:

print (num1, " is greater than ", num2)

elif num1 < num2:

print (num2, " is greater than", num1)

else:

print (num1, "is equal to", num2)

Why do we **not** need to test if num1==num2 but can print out :

print (num1, "is equal to", num2)

knowing it to be TRUE?

## Indenting

Indenting is very important in Python, it’s a necessary part of how you write the code. Indenting tells Python where blocks of code start and where they end. The Python IDE will do this for you – just be aware what it is doing.

**Update the**

**Log Book**

## More advanced ways to use an if

We might have a case where we want someone to type in an **M** or **F** to say if they are male or female. So we could write:

if choice == ”M” :

But what if they type in a lowercase **m** this wouldn’t work! We could use:

if choice == "M" or choice == "m":

This would work fine but the code is getting long and harder to read and understand. A better way is to see if the variable is in a list of possible options:

choice = input("Are you male or female")

if choice in ("M", "m"):

print ("We have a special offer of 10% off grooming products for men today")

### Challenge 20 – Beauty products

Develop the code above to give a message to men and women about special offers on sale in SuperDrug the chemists.

Men get 10% off grooming products today, women get a free manicure.

Men can type in male, M or m and women female, F or f.

Save as: beauty offer

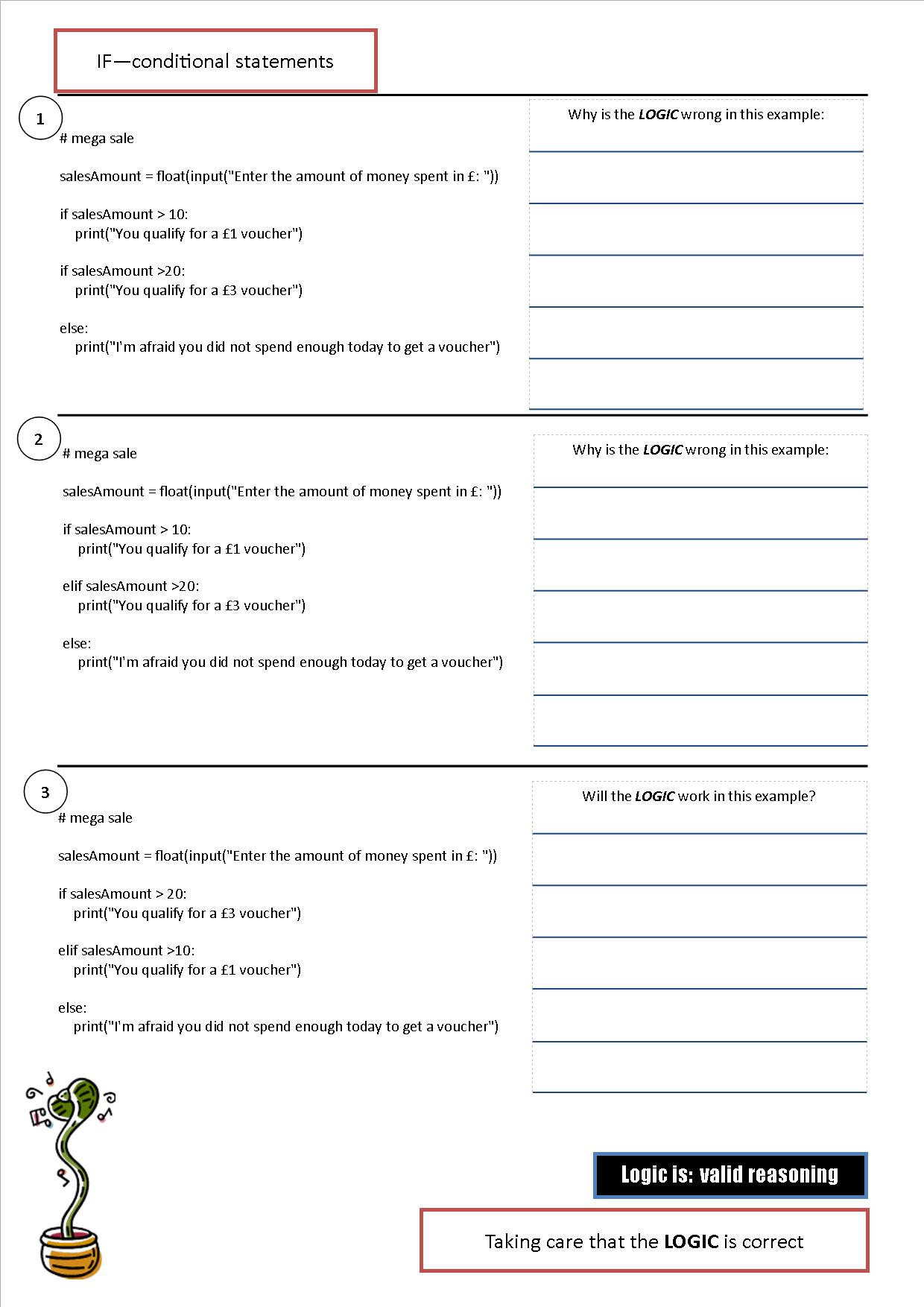
## Not equal to:

What if you want to do something when a value **isn’t** in a list?

choice = input("Are you male or female")

if choice **not** in ("M", "m", "male","Male","F", "f", "female", "Female"):

print ("Sorry you have not made a valid choice")



### Challenge 21 – Mega Sale

A local shop is having a promotion. If you spend over £10 you will get a £1 voucher to spend next time you come in the store. If you spend over £20 you get a £3 voucher.

Write a program to tell the sales assistant which voucher to give the customer.

Save this as: megasale

### Challenge 22 – Happy Message

Write a program that gives a message depending upon how happy they say they are.

You could get the user to rate how happy they feel on a scale between 1 and 10. If the reply is 3 or less it gives one message.

Between 4 and 7 (including these numbers) they get another message.

8 and above they get a different message.

Try to make the messages ones to make them happy all day long!

Save this as: Happy message

### Challenge 23 – Mobile Phone Costs

You want to see how much a mobile phone will cost. There are charges for sending pictures (£0.35), for texts (£0.10) and for data (£0.05 for 1MB).

1. Write a program that asks the user for how many pictures, texts and data they would use each month. It should then calculate a total bill for the month.
2. If the total comes to more than £10 they would be better on a contract. Get the program to give them this advice.

Save this as **Mobile Phone Costs**

**Harder**

Data is charged at £2.50 per 500MB. So even if only 2MB are used you still pay for £2.50. What we want to do is be able to round-up a number (the function *round ()* rounds up or down).

You will need to: **import maths** to get more maths features and use:

math.ceil(*number*) – to roundup a number to the next integer

Can you figure out what the calculation is to make this work?

### Challenge 24 – Secret password

Make a program where the user has to enter a secret password to use the program.

The program could be one you’ve already written, or make it display a message when they have got the password correct.

## Boolean or logical expression

Boolean is a type of arithmetic that only uses two values: true or false, yes or no, 1 or 0.

It was invented by an English mathematician George Boole.

We use Boolean expressions and can combine them with **and**, **or** and **not** to make decisions in our programs.

### Challenge 25 – For loops

Sometimes we want to do something a number of times.

We may know how many times we want to do it – and we use a **counting loop** OR

We may want to do it until something happens – and then we can use a **conditional loop.**

## Counting Loops

This is the simplest loop. We use these when we know how many times we want to repeat the code. Try these out and make a note in your log book of what they do:

for a in range (10):

print(a)

**Update the**

**Log Book**

for a in range (1, 11,2):

print(a)

for a in range (1, 11):

print(a)

for a in range (1, 10):

print(a)

Write into your log book what this for loop does:

### 

**Update the**

**Log Book**

for a in range (1, 11,2):

### Challenge 26 – For loops

Write a loop that displays numbers 10 to 100 in steps of 5.

### Challenge 27 – Times Table using For loops

Write a loop that displays the 5 times table

Here is the 5 times table:

1 x 5 = 5  
2 x 5 = 10  
etc to 12 x 5 = 60

Save as 5 times table.

## For loops are *really* clever

They can be used for even more things.

sentence = "Here is a sentence. How many letter 'e''s are there in it?"

numberE = 0

for letter in sentence:  
 if letter =="e":  
 numberE +=1

print ("The number of e's in the sentence is:", numberE)

**letter** is just a variable that stores the letter from the sentence – try changing the name to something else and check that it still works

### Challenge 28 – Reading a sentence

Change the program shown above so that the user can type in their own sentence and the computer will calculate the number of “e’s”.

Extension: count the number of vowels – a, e, i, o, u in the sentence (instead of: if letter == “e” try using the method on page 15: if letter in….)

## Conditional Loops

A conditional loop is one that repeats until a condition is met. We use these when we aren’t sure how many times we will repeat the code, we want to repeat it until a condition is **true**.

### Challenge 29 – While loops

All of this program should be familiar to you apart from the while loop. Think about what it does before typing it in.

The **while** loop repeats whist the conditions is **TRUE**.

number = 100

while number>10 or number<0:

number = int(input("Please enter a number between 1 and 10"))

print("Your number multiplied by 10 is:", number\*10)

**Update the**

**Log Book**

**Why is the int necessary?**

number = **int**(input("Please enter a number between 1 and 10"))

Save as: challenge 29

### Validation

Validation is defined as: checking that data entered is sensible (computers cannot check that the data is correct – the best they can do is spot obvious errors). Adding validation makes sure that our programs continue to run even when incorrect data is entered. We often know that we want a particular data type to be entered: float, integer, string and if the wrong one is entered then the program can end.

Here is a neat way to use a while loop to add validation to your program that stops these user errors from halting the program.

Enter the code and see how it works. Enter letters and decimals and numbers not in the range 1 to 10. Try using it as a starting point with the Rumplestiltskin program.

### 

|  |  |
| --- | --- |
| **Code** | **What it does** |
| # Enter a number between 1 and 10  number = 40  while number not in range (1,11):  try:  number = int(input("Please enter a number between 1 and 10"))  except:  print("ERROR invalid input. Out of range or wrong type of data.")  print("Thank you I have recorded your entry as :", number) | **Update the**  **Log Book** |

### Challenge 30 – Rumpelstiltskin

Rumpelstilitskin was a nasty character is a fairy tale (surely you know this). He wants to take a princesses’ first baby – and will only let her off if she guesses his name correctly. He gives her three guesses.

Make a program that asks for a first name. If the first name is the same as your own then the program should print a welcome message. If the name entered is not your own name then it should print a suitable (polite message) to say the guess is wrong.

It should repeat asking for the user’s first name until it enters your first name.

## Comments

Comments are added to a program to explain what the program does and what the section of code is doing. Programmers do it so that if the code is looked at in the future (either by themselves or others) they can work out what the code is doing.

Comments can easily be added using # at the start of the line of code. Python ignores anything on a line that begins with a #.

Load your last program: guess a number 1 to 10.

Add your own comments to the program to explain what the parts of the code are doing:

|  |
| --- |
| *# this program creates a random number between 1 and 10*  *# the user guesses the number. It ends when the user guesses the number correctly*  import random |
| guess ="" |
| print ("I've thought of a number between 1 and 10. Try to guess it.") |
| randomNumber = random.randrange (1,10) |
| *# this loop runs until the number guessed equals the randomNumber* |
| while guess != randomNumber: |
| guess = int(input("Guess the number")) |
| print("You got it wrong") |
|  |
| input ("Well done. Press any key to exit.") |

We can also add comments to many lines at a time by using three quotes together “”” at the beginning and end of our descriptions:

“” This program calculates how much savings you have after a number of months.

Written by: May Dup

Date: Created 23/11/12

“””

Some further challenges for you to have a go at. Use this workbook and the programs you have already used to help you solve the problems.

### Challenge 31 – Rock, Paper, Scissors Game

We’ve all played the rock, paper, scissors game. The computer will play against you. Get the computer to ask for the player’s name.

The game rules are simple: rock beats scissors, scissors beat paper, paper beats rock. Two of the same gives a draw.

You can start with pseudo code to help you identify the steps needed.

Add **comments** to the game so it’s clear what the code in the game is going.

Extension: Ask the user how many rounds they want to play, between 3 and 10.

Keep score and show this at the end of the game.

Further extension: Check to make sure the user can only enter a number between 3 and 10 and give them an error message.

### 

## Planning the Rock, Paper, Scissors Game

Get the computer to ask for the player’s name.

The game rules are simple: rock beats scissors, scissors beat paper, paper beats rock. Two of the same gives a draw.

The program should display on the screen what the computer choose, how many points are awarded and who has won.

**Simple**: play just one round

**Harder**: play 5 rounds

**Harder still**: ask the player how many rounds they want to play.

Plan you program using pseudocode.

Think what variables you will have and what the main sections of the code will need to do.

You will need to use a **FOR** loop and **IF / ELIF / ELSE.**

*Write in pencil so that you can rub out if needed.*

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

We can use a truth table (just like with gates) to see what the permutations are:

Rock = 1, Paper = 2, Scissors = 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User choice** | **Number** | **Computer Choice** | **Number** | **Scoring** |
| Rock | 1 | Rock | 1 | User +=1 / Computer +=1 |
| Rock | 1 |  |  |  |
| Rock | 1 |  |  |  |
| Paper | 2 |  |  |  |
| Paper | 2 |  |  |  |
| Paper | 2 |  |  |  |
| Scissors | 3 |  |  |  |
| Scissors | 3 |  |  |  |
| Scissors | 3 |  |  |  |

### Challenge 32 – Making a times table (using nested for loops)

With a times table we start with a number and then multiply this by a series of numbers.

For the 2 times table we multiply it by 1 to 12, before moving on to do the 3 times table, etc.

Computers are very good at this sort of task and we can do it using loops within loops to consider every combination – these are called **NESTED LOOPS.**

Type this code in (try to predict what you will get before you do it).

**for i in range(1,13):** # i is the first number we are going to multiply by

# print a title at the top of the times table to show which times table

**Update the**

**Log Book**

# we are working on

**print (i, "Times table\n")**

**for j in range (1,13):** # loop through the numbers we are multiplying

# i by and then print the results

**print (i, "times", j, " = ", i\*j)**

## Extension:

Maths Teachers will be impressed if you’ve worked out the 1345 x 1 to 12 times table - change the program to work out your own different times tables.

# In Built Functions

In built functions are functions that come with the python. New ones get added in new releases of the software to make it ass new functions. Here are some to try:

|  |  |  |
| --- | --- | --- |
| Function | What it does | Example |
| bin(x) | Where x is a decimal number the bin function converts the decimal to a binary number.  This prints the number with 0b at the start to show it’s in binary. To remove this add in [2:] to print only after the 2nd digit | print(bin (10))  print (bin(10)[2:]) |
|  |  |  |
|  |  |  |
|  |  |  |

Sometimes python doesn’t have a particular function and then people have written functions to perform specific tasks. These need to be imported at the start of the program using the import command. You’ve already used one called random for generating random numbers.

Here are some more to try.

|  |  |  |
| --- | --- | --- |
| **Import function** | **Code** | **Explanation** |
| import os | print("Hello", os.getlogin(), "and Welcome to the...") | Prints a welcome message getting the username from the operating system |
|  | os.system("exit") | Exits the program |
|  | os.system("cls") | Clears the screen |
|  |  |  |
| import time | time.sleep(1) | waits one second |
|  |  |  |
|  |  |  |

## Functions

Programming languages have pre-made functions for us to use. Python has some such as print() or random.

But we can make our own functions, these are called user-defined functions:

A function is a block of organised, **reusable** code that is used to perform a single action.

By using functions we can make our programs **modular** (made up of separate parts) and **simpler**, because we don’t have to write the same instructions many time but **reuse** the same code.

Using Turtle graphics we can draw shapes just like when we used Small Basic

# program that draws a square

import turtle # import turtle module

wn = turtle.Screen() # make a graphics windows to draw in

tom = turtle.Turtle() # create a turtle called tom

for a in range (4): # loop to repeat instructions 4 times

tom.forward(150) # move forward

tom.right(90) # turn right

wn.exitonclick() # when window clicked end

We can define our own function to draw a square:

import turtle

wn = turtle.Screen()

tom = turtle.Turtle()

def square(): # **def**ine our new function in this format

for a in range (4): # all code is indented to show its part of the for loop

tom.forward(150)

tom.right(90)

wn.exitonclick()

square() # call the function with this command

### Challenge 34 – Make shapes

Make other regular shapes: a triangle / hexagon / octagon using functions. Call the functions one after another.

Try using an ***iterative statement*** (a FOR loop) to draw multiple shapes in patterns using this method.

We can pass information to our functions, the data we transfer are called the function’s **parameters**.

import turtle

wn = turtle.Screen()

tom = turtle.Turtle()

def square(length):

for a in range (4):

tom.forward(length)

tom.right(90)

wn.exitonclick()

square(80)

The function square receives the **parameter**, and we have called it ***length.***

**NOTE**: The variable length is used – but it only exists within the function – **not** the rest of the program.

**2**

**1**

Here square passes a value 80 to the function

**Type this in and then develop your program as shown below.**

This time we get the user to enter the **size**, then we pass this to the function as a **parameter**.

import turtle

wn = turtle.Screen()

tom = turtle.Turtle()

def square(length):

for a in range (4):

tom.forward(length)

tom.right(90)

wn.exitonclick()

size = int(input("Enter the length of the square"))

square(size)

### Challenge 35 – Area or Perimeter

Let’s make a program to calculate the area **OR** the perimeter of a rectangle.

**Area = length \* width**

**Perimeter = length \*2 + width \*2**

width = 100

length = 200

We can define a function to calculate area when we need it and one to calculate the perimeter when we need it:

Just as before with the shapes we define a function using def and the function name:

**def area():**

def area():

shapeArea = length \* width

print("Area = ",shapeArea)

def perimeter():

shapePerimeter = length\*2 + width\*2

print ("Perimeter = ", shapePerimeter)

Here is the code to type in. Look at the code and think what it is doing as you enter it.

**Update the**

**Log Book**

Run the code and then update the log book explaining the code:

# function to calculate area

def area(length,width):

shapeArea = length \* width

print("Area = ",shapeArea)

def perimeter():

shapePerimeter = length\*2 + width\*2

print ("Perimeter = ", shapePerimeter)

response = None

while response not in ("a","p"):

response = input("Do you want to calculate area or perimeter? Enter a or p")

response = response.lower()

Here we get the user to enter the width and length.

These **PARAMETERS** are then sent to the area function and used to work out the area.

if response == "a":

length = int(input ("Enter the length of the rectangle"))

width = int(input ("Enter the width of the rectangle"))

area(length, width)

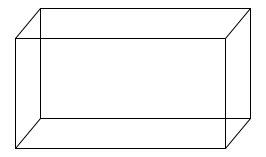
elif response == "p":

Enter this into Python and test it.

Then add your own code so it works similarly if you chose to calculate perimeter.

perimeter()

## Extension

[](http://upload.wikimedia.org/wikipedia/commons/3/32/Cuboid_simple.svg)Add a function to calculate the volume.

To work out the volume of a regular cuboid shape is:

length \* width \* height

Add an option to the program to calculate the volume defining a new function called volume()

# Passing Functions a Value by using Parameters

We have learnt how useful making our own functions can be. We can also pass values of variables around using them, which is a useful thing to be able to do.

Functions in other programming languages are known as sub-routines.

There are a few different ways of passing values between functions:

## Method 1

**Pass** the function a value and use this value within the function. The parameter in this code is:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| **Python Code** | **What the code is doing** |
| def display (message):  print (message)  display ("This is a message")  **Type this in and then develop your program as shown below.** | The parameter in this code is:\_\_\_\_\_\_\_\_\_\_\_\_\_\_   within the function\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Update the**  **Log Book** |

|  |  |
| --- | --- |
| **Python Code** | **What the code is doing** |
| def display (message):  print (message)  def main():  display ("This is a message")  display ( “This is a second message” )  main() | Often we create a function main() and call other functions from here. It makes our code more organised and is easier to see how the sections of code work. |

## Method 2

Pass a function a **value** and return a **different value.**

|  |  |
| --- | --- |
| **Python Code** | **What the code is doing** |
| def ask\_yes\_no(question):  response = None  while response not in ("y","n"):  response = input(question)  return response  def main():  answer = ask\_yes\_no("\nPlease enter y or n: ")  print ("Thanks for entering:", answer)  input ("\nPress the key to exit.")  main() | The parameter in this code is:\_\_\_\_\_\_\_\_\_\_\_\_\_\_   within the function\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   and the returned value is \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

## Method 4

|  |  |
| --- | --- |
| **Python Code** | **What the code is doing** |
| def favourites (subject, meal):  print("My favourite subject is ", subject, "and my favourite meal is ", meal)  def main():  favourites ("Computing", "Burgers and Chips\n")  favourites ("Spag Bol", "Computing\n")  favourites (meal = "Chicken wraps", subject="Computing")  main() | Positional parameter  Positional parameter  Keyword arguments (helps avoid errors and useful to make the code clear) |

# Global variables

Variables and their values only exist inside the functions themselves. Once the function has run and exited the values are no longer there.

If we want to use a variable throughout our program we must declare it as a global variable. This can then be alterable by the main program and functions as well.

To make a variable global define it as global in the function and in the main body of the program (at the start).

global name

def printName():

global name

This overcomes the need to pass parameters around.

## Lists

We have seen how we can use variables to hold information for us as we program and have seen how necessary it is to temporarily store data.

But what if we want to store our school timetable, at the moment we would need to have lots of new variables such as *mondayP1, mondayP2, etc. (or 40 for a week at Lady Manners).*

There is another way that uses lists (these are often know as Arrays in other languages).

Here is an example of a list:

mondayTT = ["French", "English", "PSHE", "ICT", "Maths"]

mondayTT is the list name and the contents of the list are in the following positions:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MondayTT =[ | “French” | “English” | “PSHE” | “ICT | “Maths” |
| Position | 0 | 1 | 2 | 3 | 4 |

Note: Lists always start with 0.

### Challenge 35

Type this code in. Run it and then explain what it does.

|  |  |
| --- | --- |
| **Code** | **Description** |
|  |  |
| print ("The list monday contains the following information:") |  |
| print (Monday[2]); | Try changing the 2 to another number and see what happens.  **Update the**  **Log Book** |
| print ("We can also just ask to print out the list. So here is monday:")  print(monday) |  |
| print (“We can also print them out one at a time”)  for i in monday:  print (i) |  |

We can also add to lists, very easily in fact!

Add this line after the first line that defines Monday: monday.append ("Geography")

What position is geography in?

### More lists

Lists are very cool! We can do lots with them because they can store information with them, adding information and reading information.

Type this simple list into a new Python module:

bars = [“Mars” , ”Bounty” , ”Twix”]

To add to the list we use **append**. Type this in:

bars.append(“KitKat”)

We’ve seen that we can print information from the list using its position:

print (bars[1]); can you remember what this will print??

## Slicing lists

We can slice a list. All this means is that we can find part of the list and make a copy of it. Type this in:

print (bars[0:2]);

Write down what prints:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

See how it prints differently, just like a list. And look at what it prints – position 0 and 1 – up to but not including 2.

Try typing these in and write down in your log book what Python is doing:

|  |  |
| --- | --- |
| **Instruction** | **What it does** |
| print (bars[:2]; | **Update the**  **Log Book** |
| print (bars[2:]); |  |
| print (bars[:]); |  |

## Adding to a list:

|  |  |
| --- | --- |
| **Instruction** | **What it does** |
| bars.append(“Snickers”);  print (bars) | **Update the**  **Log Book** |
| bars.insert(1,”Yorkie”);  print (bars) |  |
| bars.extend(["Picnic","Twirl"])  print (bars) |  |

**Note**: If you want to add more than one item to a list use extend.

Save this file as: **adding to a list**

## Deleting from a list:

Let’s start with a list. Enter this into a new program: bars = ["Mars" , "Bounty" , "Twix", “Yorkie”, ”KitKat”]

Then add the following instructions and see what they do (the **print(bars)** option is there just so you can see what has happened). You can add each command after the previous ones:

|  |  |
| --- | --- |
| **Instruction** | **What it does** |
| bars.remove("Twix")  print (bars) |  |
| del bars [0]  print (bars) | **Update the**  **Log Book** |
| myBar = bars.pop()  print (bars)  print (myBar) |  |
| myBar1 = bars.pop(0)  print (bars)  print (myBar1) |  |

## Random choice from a list

We have used the random function in our programs before. Python has an inbuilt way to select a random choice from a list which is really cool and saves lots of time:

**import random** is placed at the start of the program.

Try this out :

**Update the**

**Log Book**

letter = ['a', 'b', 'c', 'd', 'e']

from random import choice

print (choice(letter))

### Challenge 36

## Dictionaries

A dictionary in python is very much like a list. It stores lots of data. But with a dictionary you store data in pairs so that they are related to each other: so you have a piece of information – **the key** – and a piece of data – **a value**. The format of a dictionary is:

dictionaryName = {“key” : ”value }

To look up a value from a key:

dictionaryName[“key”] - will give us the value

In this example we have stored computer terms.

value

key

compTerms ={"HDD":"Hard disk drive. Secondary storage device.",   
"Mouse":"Device used for making selections, such as menu items on the monitor",  
"CPU":"Central Processing Unit, which performs the logic, calculations and controls inputs and outputs in a computer"}

print(compTerms["HDD"])

This will print:

>>>   
Hard disk drive. Secondary storage device  
>>>

### Challenge 37

This list is available on Myvle to save writing it in yourself:

Press Shift and Enter at the end of a line to create a new line

countryPop = { "Japan":"127000000", "Germany":"81000000", "Iran":"77000000",

"UK":"64000000","Canada":"33000000","Australia":"23000000","USA":"317000000",

"Bulgaria":"7000000","Sweden":"10000000"}

The program should:

Ask the user to choose a first country

Ask the user to choose a second country

Add together the populations of the two countries.

Make sure that you put in good messages to help the user and tell them what is going on

To test the program enter Japan and USA: they should add up to: 444000000

If you get the answer as: 127000000317000000 there is a small error. Look back at Challenge 4 and 5 to find the answer.

## Extension:

What if the user enters the word incorrectly? How could we deal with this?  
(countryPop is the dictionary name and country1 is the name of the key)

country1 = “”

while country1 not in countryPop:

country1 = input("Enter the name of the first country:")

print("Sorry can't find that information. Please try again")

print("I've found the information for you. The population of ", country1, " is: ",countryPop[country1], "\n")

This code should be familiar to you. While country 1 cannot be found in countryPop the computer keeps asking for a valid entry to be made. When country1 can be found in the dictionary the program prints out the population of the country.

**Change the program to only accept valid inputs of country1 and country2.**

## Writing to and reading from a file

So far the information in our programs has disappeared as soon as the Python program has stopped. Python can write to a file and can read from the file.

If you run the program you will get and error.

Save the python program and then run it.

The text file: example.txt will be made in the same folder as the program

Check to see if a file has been made. Double click it to open it in Notepad.

You should see the sentence you typed.

Run it again but change the words.

See what happens to the file.

myFile = open ("example.txt", "wt")

myFile.write("Here is the first file I have written!")

myFile.close()

Save the file as: write example

That’s a good start but what we need to be able to do is keep the information separate so that we can do more with the data.

One way is to write the data to separate lines in the file. We can do this by adding the newline instruction:

The \n is the instruction to create a newline.

Open the file in notepad to see the result.

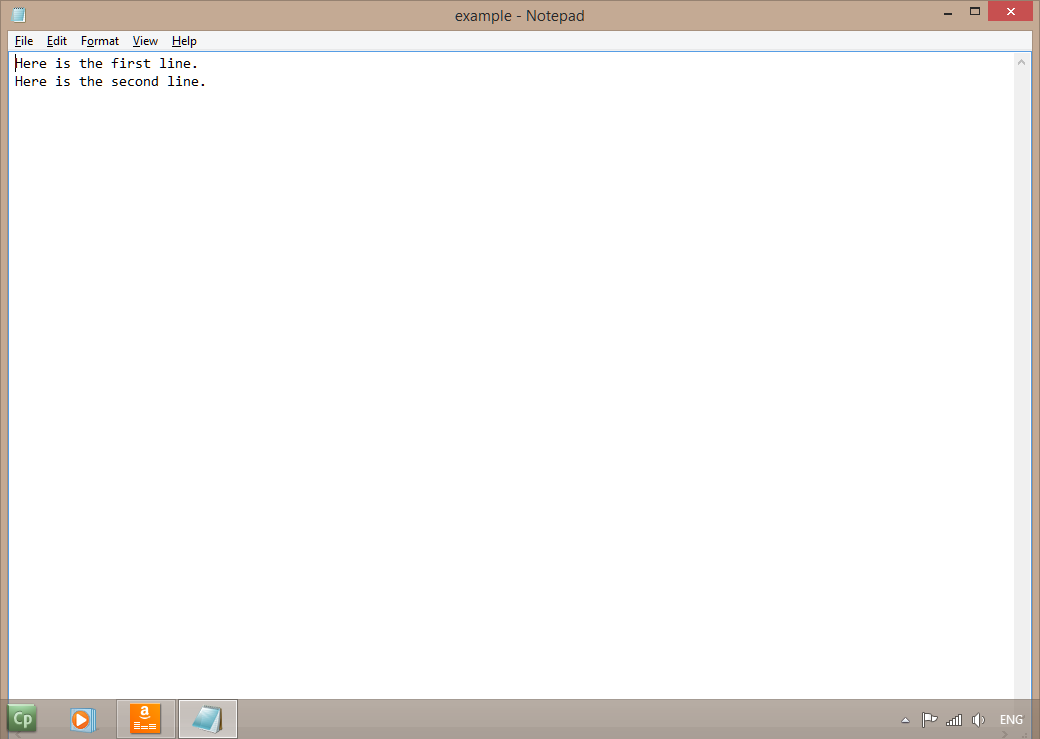
Close() is very important – it closes the file when we have finished.

mMyFile = open ("example.txt", "wt")

myFile.write("Here is the first line.\n")

myFile.write(“Here is the second line.\n”)

myFile.close()



## Writing a List to a File

This is all very exciting but how can we develop this and do more useful tasks? Writing a list to a file would be even more useful.

myFile = open ("example.txt", "wt")

myList = ["Lady Gaga", "Justin Timberlake", "ACDC"]

for item in myList:

myFile.write(item+"\n")*# writes the items in the list appending a \n to put it on a newline*

myFile.close()

Notice that the line myFile = open ("example.txt", "wt") has “wt” at the end. This tells Python to open a file for writing to (hence the ‘w’) and the ‘t’ means that the file is text.

An important thing to note is that using wt means that the contents of the file is deleted each time!

## Reading from a File

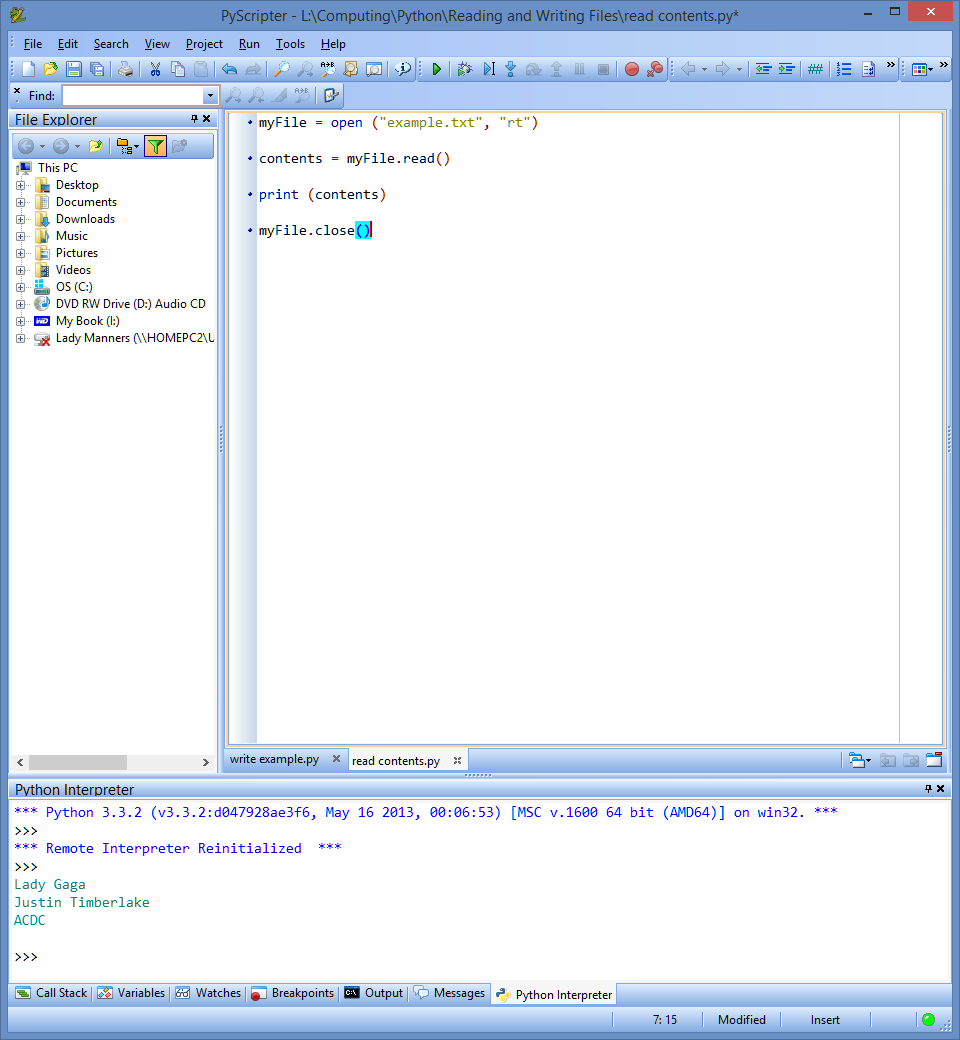
So how to do the other part – reading from a file?

This time we open the file with the ‘rt’ command. This means open it for reading. The contents of the file isn’t deleted this time!

myFile = open ("example.txt", "rt")

contents = myFile.read()

print (contents)

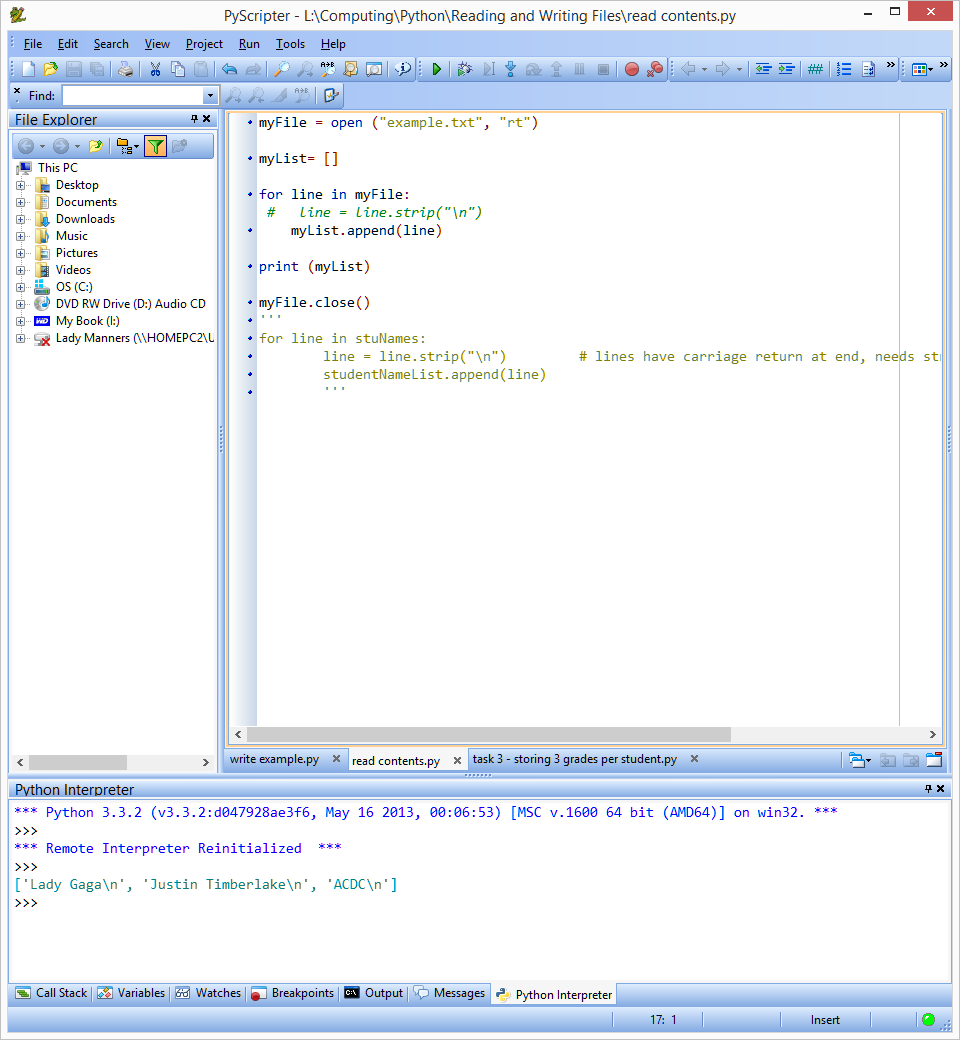
myFile.close()

That worked well, but what if we want to put the information from the file back into a list? Python does this easily..

myFile = open ("example.txt", "rt")

myList= [] *# define an empty list in which to store the information*

for line in myFile:

 myList.append(line)

print (myList)

myFile.close()

Well it’s OK but now we have newline commands in as text – so let’s get rid of them.

Add **after** the line with for line in myFile:

line = line.strip("\n")

and run it again.

### Challenge 38

**Step 1**. Write a program that asks the user to enter 6 numbers. Store these in a list.

Write these numbers to a file.

**Step 2**. Write another program that reads the file. Print out all of the numbers together with the total and average of the numbers.

**Extension**: Sort the numbers into ascending order (smallest to largest)

Step 3. Write a program that will allow you to **add** four new numbers to the list and write it to the file:

myFile = open("example.txt", "at") *# opens the file for appending (adding to)*

for a in range (4):

newNo = input("Please enter new number")

myFile.write(str(newNo)+"\n") *# converts the integer to a string & adds a newline instruction*

myFile.close()

Open the file in notepad and check it has written the information to the file.

### Challenge 38

This list has the names, heights and weights of 5 people. The height is a 2 digit whole number and the weight is a 3 digit decimal number.

heightandweight = [James, 73, 1.82, Peter, 78, 1.80, Jay, Beth, 65, 1.53, Mags, 66, 1.50, Joy, 62, 1.34]

Enter this list as it is.

Store the data in a file.

Write a program that reads the data.

Print out the person’s name and their height and weight.

**Extension:**

Work out the average height and average weight of all of the people.

Work out the average height and weight of the men and the women.

# Pickling and Shelving Lists

If we write a single list to a file using the commands we’ve already seen is straight-forward. It gets harder to keep a log of the position of information when there are multiple pieces of information.

Pickle allows information to be stored and read from a file easily.

This program saves a list to a file:

Write this program in Python and test it out. Save it as ***pickle\_data***.

import pickle *#imports the pickle module*

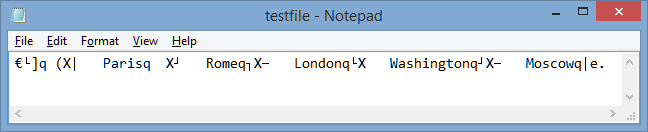
a = ["Paris", "Rome", "London","Washington", "Moscow"] *# a list to save to file*

myFile = open("testfile.dat","wb") *# opens the file for writing to*

pickle.dump(a,myFile) *# dumps the list (a) to the file*

myFile.close() *# closes the file*

Opening the file created shows the data has been written but it has extra characters.



The code below adds a second list and a second command to pickle the data:

import pickle *#imports the pickle module*

a = ["Paris", "Rome", "London","Washington", "Moscow"] *# a list to save to file*

**b = ["Jane","Janet","Jill", "Jean"] # a second list**

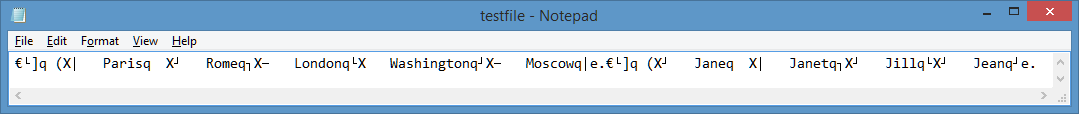
myFile = open("testfile.dat","wb") *# opens the file for writing to*

pickle.dump(a,myFile) *# dumps the list (a) to the file*

**pickle.dump(b,myFile) # dumps the list (b) to the file**

myFile.close() *# closes the file*

The file has been created adding the second list.



Now let’s try reading the information into a file – a very simple process:

import pickle *# import the pickle command*

myFile =open("testfile.dat","rb") *# name the file to use and open as "rb" read binary*

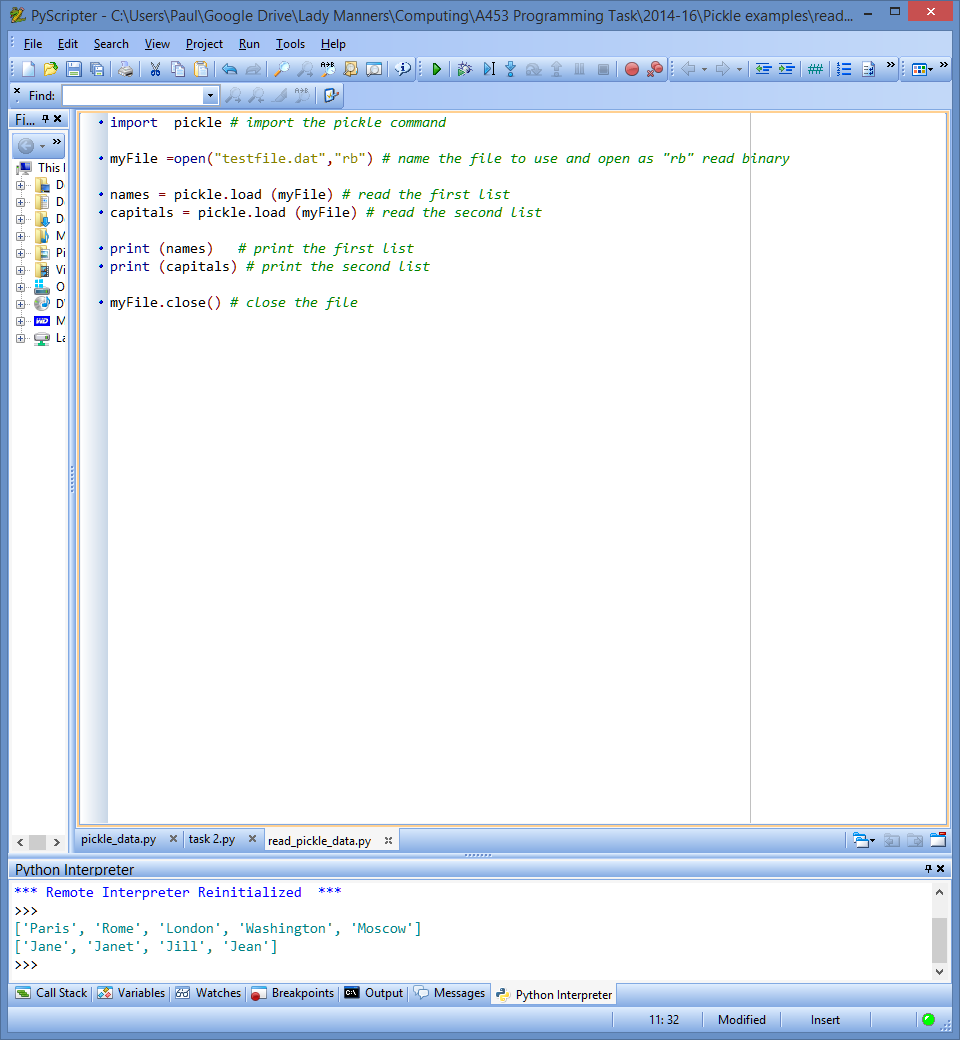
names = pickle.load (myFile) *# read the first list*

capitals = pickle.load (myFile) *# read the second list*

print (names) *# print the first list*

print (capitals) *# print the second list*

myFile.close() *# close the file*



The lists are loaded and printed out. The data is loaded one list at a time into the new list names (yes they are deliberately round the wrong way!)

Try it out with your own lists. Save it as ***read\_pickle***

Now try adding a dictionary with 3 sets of data to the pickle\_data file and run it.

Add an extra line to the read\_pickle file to read in a third set of data and print out the result.

You **should** see that the dictionary that you pickled was read into the program as a dictionary – neat!

### Challenge 39

Mandy is a tennis coach who tutors 6 young players who compete in local and national competitions.

Three of them play in the under 11s category and three in the 11-16 category.

She would like a program making that allows her to record the names of her students and the position they came in their last competition.

**Step 1**. Write a program that allows her to enter the names and positions and store the data so that she knows which age category they are in. Store the data in a file (try using pickle).

**Step 2.** Write a second program (or add this functionality to the first program) that reads in the file and prints it to the screen.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Under 11s** | |  | **11 to 16s** | |
| **Name** | **Last Competition Position** |  | **Name** | **Last Competition Position** |
| Jane | 14th |  | Henry | 23rd |
| Grant | 12th |  | Jane | 2nd |
| Lilly | 7th |  | Naomi | 18th |

**Extension:** Write into the program these options:

* Being able to select a list to delete
* Being able to change a player’s position
* Being able to store the position of the players in their last 3 tournaments