Copperfield College MHR, updated JMB

Unit 1 Applied Computing

Python Workbook

Name: Anthony

2025

Help

Ask for help from **others** in the class or GitHub

Complete the challenges 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.

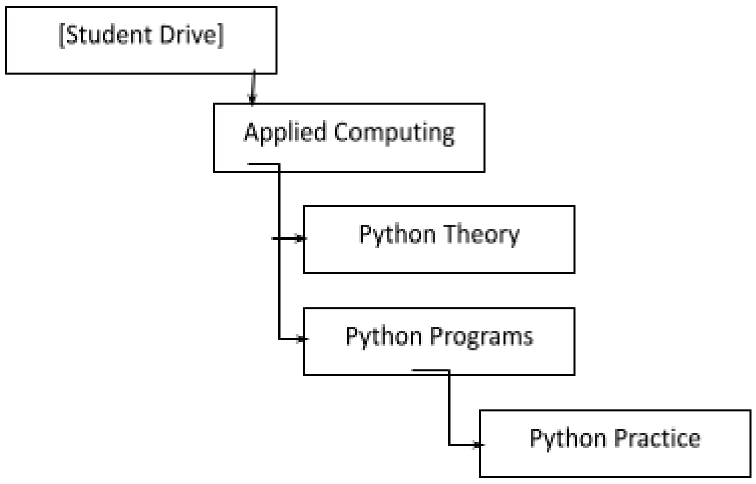
Make sure you save your solution programs to an organised folder in your student drive and GitHub repository.

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

Get Organised

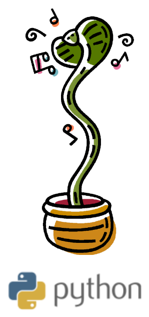
Life will be much harder if you don't save your work carefully.

Create the following folders:



Contents

Inputting 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….

Asks for the first name and then a last name.

Prints the first name and then the last name

On the next line print the last name and then the first name

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

Use first_Name and last_Name as the variables.

Strings can be joined together:

Challenge 2 - Write a program that…

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

```
Full_Name = first_Name  +  last_Name
print (full_Name)
```

Save as: **Names.py**

Joining strings together is known as:

concatenate

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?

It asked for a number between 0-10 I entered 5 than it asked the same question and I put 5 and the result was combining the numbers into 55. Because the code isn't asking for integer its just asking to input both numbers togather.

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)
```

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.

What happened? Why do you think this happened?

When the Integer input was added the code added number one and two together for example I imputed 5 and 5 and the result was 10. Because with the integer it adds both numbers together making the result of 5 and 5 10.

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?

When I wrote 5 and 5 the result was 10.0. When 5.5 and 4.5 was wrote the result is 10.0. This is due to float being used, float is basically the computer understanding decimal numbers in coding.

Update the

Log Book

Variables

What is a variable? ____

Variable type	Describe what type of data this variable type can store
String	
Integer (int)	
float	

data	What variable type is suitable?
5	int
London	
8.7373	
1 000 000	
Mr Smith	
0.009	

data	What variable type is suitable?
Derbyshire	
34 x 45	
90	
People	
98.98	
0	

Mega Sale

```
# mega sale

salesAmount = float(input("Enter the amount of money spent in £: "))

if salesAmount > 20:
   print("You qualify for a £3 voucher")

elif salesAmount >10:
   print("You qualify for a £1 voucher")

else:
   print ("I'm afraid you did not spend enough today to get a voucher")
```

Follow the program through.

If the variable salesAmount has the value in the table write down what will be printed in each case

salesAmount	What will be printed
5	
15	
25	
10	
11	
1	

Challenge 6 – Calorie Counter

Now let's build a calorie counter. Nutrition Australia 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")
```

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

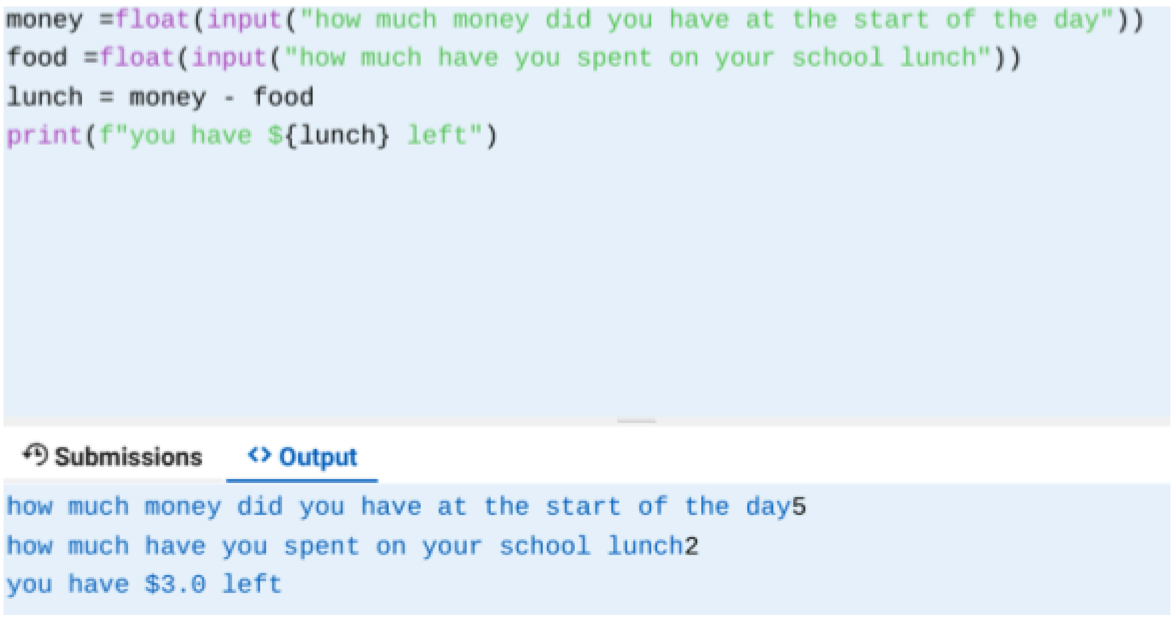
Line 4 then prints outs in between 2 'strings' of text

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)

Challenge

Write a program that asks the user to enter how much they have spent on their school lunch 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 **SchoolLunch**



```
money =float(input("how much money did you have at the start of the day"))
food =float(input("how much have you spent on your school lunch"))
lunch = money - food
print(f"you have ${lunch} left")
```

Submissions   Output

```
how much money did you have at the start of the day5
how much have you spent on your school lunch2
you have $3.0 left
```

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 built in but we can add external functions using **import**:

```
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?

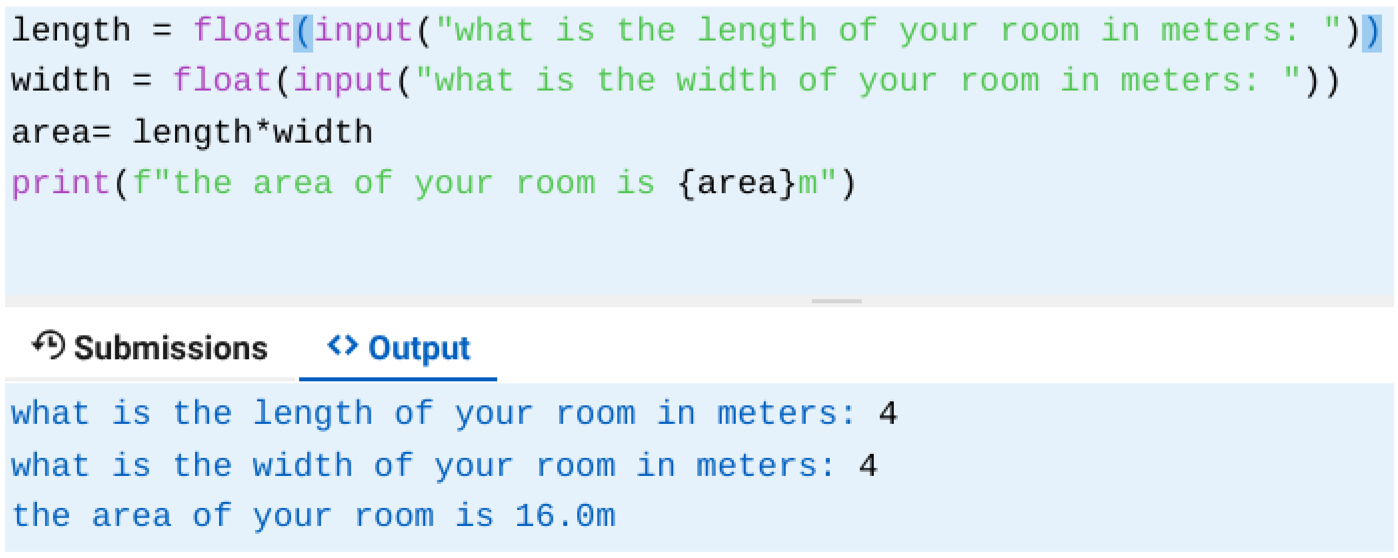
This tells the program to pause for 3 seconds before moving onto the next line of code.

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).



```
length = float(input("what is the length of your room in meters: "))
width = float(input("what is the width of your room in meters: "))
area= length*width
print(f"the area of your room is {area}m")
```

```
what is the length of your room in meters: 4
what is the width of your room in meters: 4
the area of your room is 16.0m
```

Save as Square.py

Challenge 8 – Days Alive Calculator

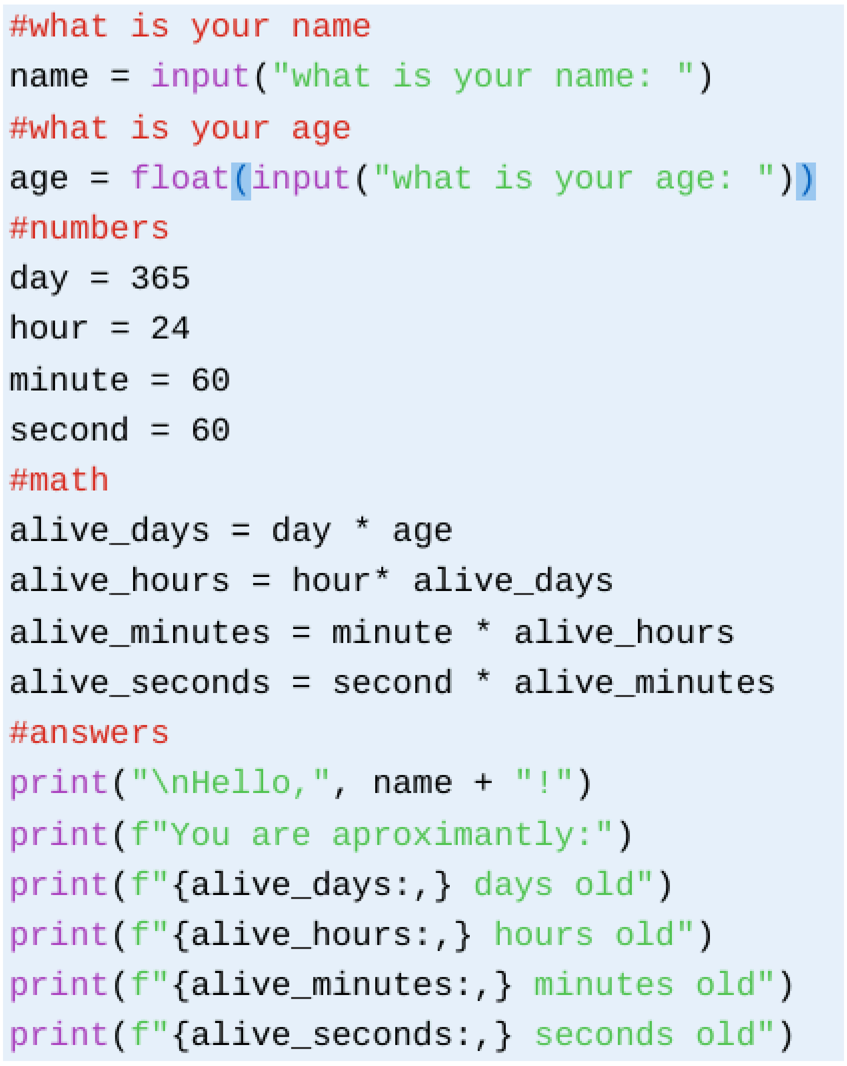
Write a program to work out how many days you have been alive for (to the nearest year for the moment - there are 365 days in a year – ignore leap years for now)

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.



```python
#what is your name
name = input("what is your name: ")
#what is your age
age = float(input("what is your age: "))
#numbers
day = 365
hour = 24
minute = 60
second = 60
#math
alive_days = day * age
alive_hours = hour* alive_days
alive_minutes = minute * alive_hours
alive_seconds = second * alive_minutes
#answers
print("\nHello,", name + "!")
print(f"You are aproximantly:")
print(f"{alive_days:,} days old")
print(f"{alive_hours:,} hours old")
print(f"{alive_minutes:,} minutes old")
print(f"{alive_seconds:,} seconds old")
```

Save as Age.py

Copperfield College MHR, updated JMB

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()	it makes the sentence in lower case
upper()	it makes the sentence in upper case
title()	it makes the first letter of every word uppercase
swapcase()	every word that was upper case is know lowercase and every word that was lower case is not uppercase
capitalize()	makes the first letter capitalized

Update the Log Book

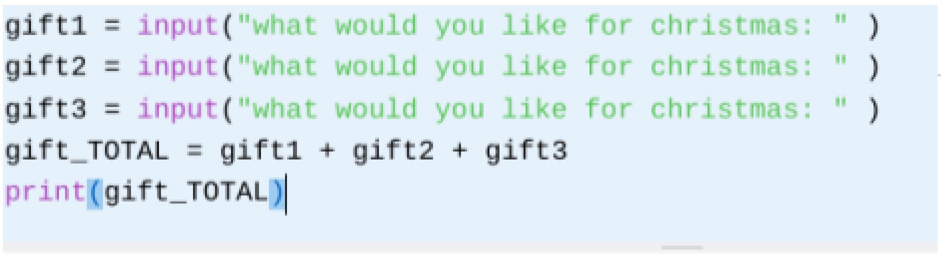
Challenge 10 – Concatenation

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) : gift_Total = gift1 + gift2 + gift3

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



```
gift1 = input("what would you like for christmas: " )
gift2 = input("what would you like for christmas: " )
gift3 = input("what would you like for christmas: " )
gift_TOTAL = gift1 + gift2 + gift3
print(gift_TOTAL)
```

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

gift2 =

gift3 =

gift_Total = gift1+gift2+gift3

How to use Python for math calculations

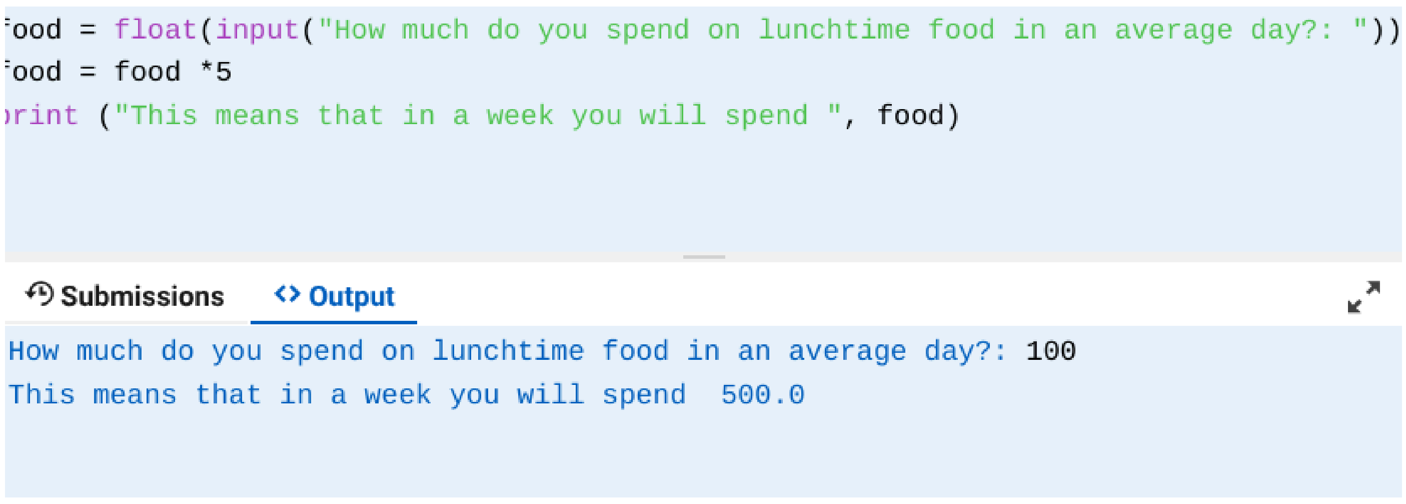
Let’s look at how python calculates using numbers.

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 FoodSpending.py



```
ood = float(input("How much do you spend on lunchtime food in an average day?: "))
ood = food *5
rint ("This means that in a week you will spend ", food)
```

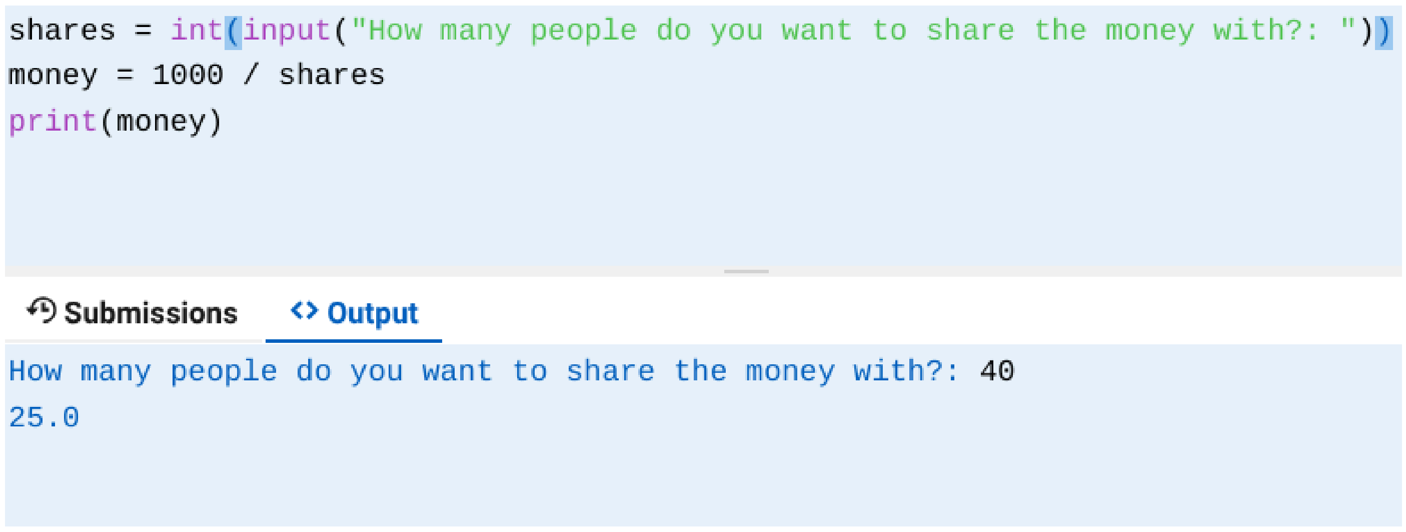
```
How much do you spend on lunchtime food in an average day?: 100
This means that in a week you will spend  500.0
```

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: **SharingMoney.py**



```
shares = int(input("How many people do you want to share the money with?: "))
money = 1000 / shares
print(money)
```

Submissions Output

```
How many people do you want to share the money with?: 40
25.0
```

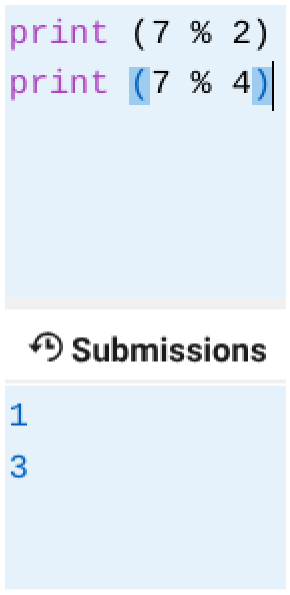
Challenge 13 – Modulus %

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

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.



```
print (7 % 2)
print (7 % 4)
```

Submissions

```
1
3
```

Challenge 14 – Addition

Addition is easy. Try this:

```
length = 10

length = length + 20
```

What's the value of the variable length:_____30____________ Try it out to test it.

Another way of writing this in python is:

```
length = 10

length += 20
```

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

Operators

The +, -, /, * symbols are called operators. That's because they "operate on" the numbers.

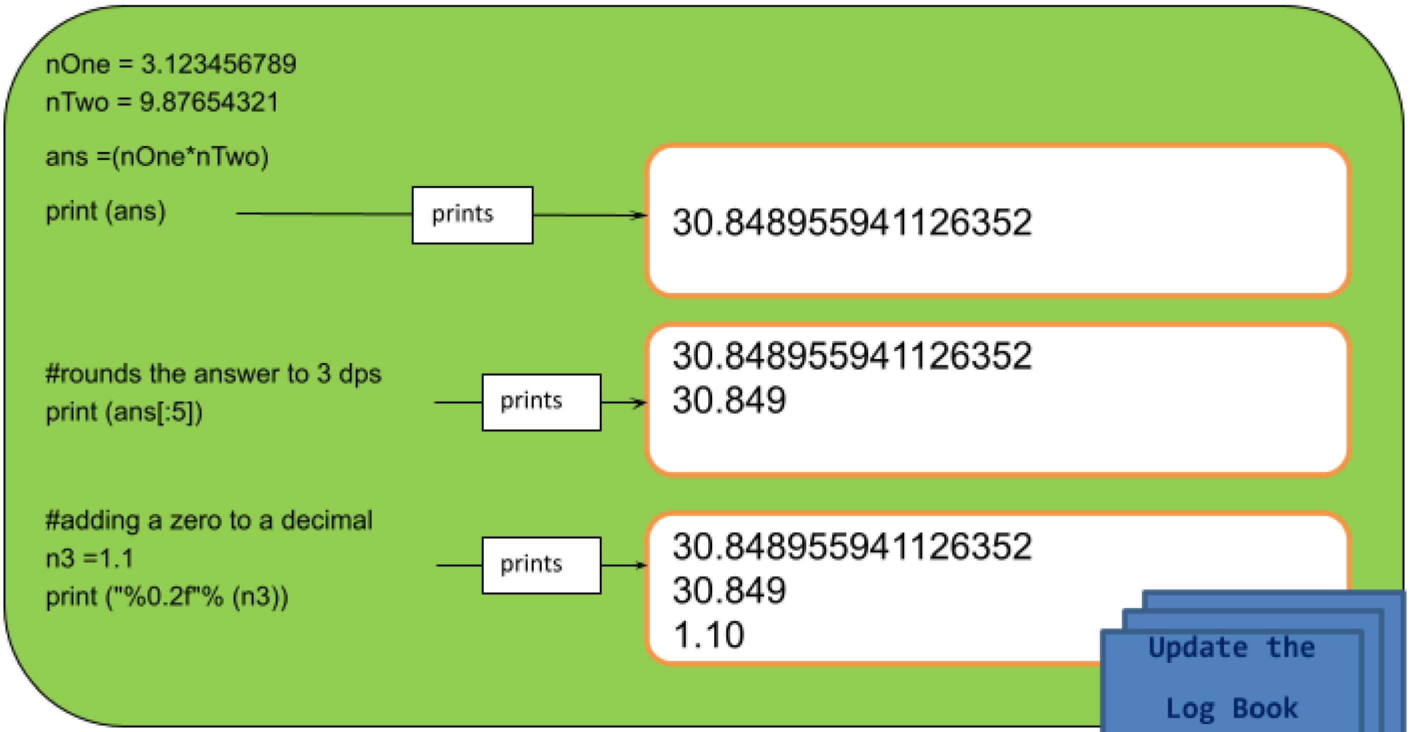
Operator	Example	Is Equivalent to
*=	K*=5	K = K * 5
/=	K/=5	K = K / 5
%=	K%=5	K = K % 5
+=	K+=5	K = K + 5
-=	K-=5	K = K - 5
a55556		

Update the Log Book

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)
```

```
30.848955941126352
```

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

```
30.848955941126352
30.849
```

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

```
30.848955941126352
30.849
1.10
```

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 **RestaurantTip.py**

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

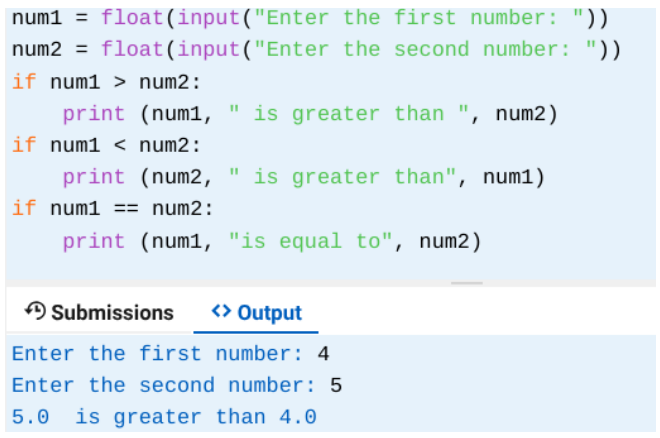
Log Book

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: compare2numbers



```
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)
```

```
Enter the first number: 4
Enter the second number: 5
5.0  is greater than 4.0
```

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

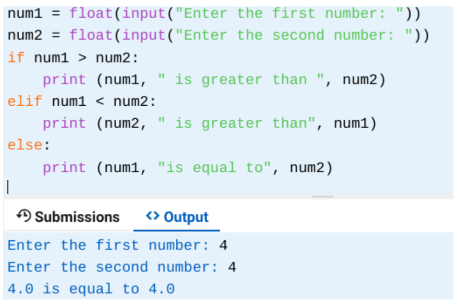
We can re-write the code to use these features:

```
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?

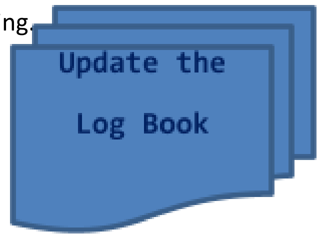


```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
if num1 > num2:
    print (num1, " is greater than ", num2)
elif num1 < num2:
    print (num2, " is greater than", num1)
else:
    print (num1, "is equal to", num2)
```

```
Enter the first number: 4
Enter the second number: 4
4.0 is equal to 4.0
```

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. Visual Studio Code will do this for you – just be aware what it is doing.



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("Which team are you in?")


if choice in ("K", "B"):
    print ("We have a special offer of 10% off ferrets for students in B or K team")
```

Modify the program so it asks which home group you are in and offers a discount if you are in XB1, XK2, or XK3.

Save as: ferret offer

Not equal to:

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

```
choice = input("Which home group are you in? ")


if choice not in ("XB1", "XB2", "XB3","XB4","XK1", "XK2", "XK3", "XK4"):
    print ("Sorry you have not made a valid choice")
```

IF—conditional statements

1

```
# mega sale

salesAmount = float(input("Enter the amount of money spent in £: "))

if salesAmount > 10:
    print("You qualify for a £1 voucher")

if salesAmount >20:
    print("You qualify for a £3 voucher")

else:
    print("I'm afraid you did not spend enough today to get a voucher")
```

Why is the ***LOGIC*** wrong in this example:

2

```
# mega sale

salesAmount = float(input("Enter the amount of money spent in £: "))

if salesAmount > 10:
    print("You qualify for a £1 voucher")

elif salesAmount >20:
    print("You qualify for a £3 voucher")

else:
    print("I'm afraid you did not spend enough today to get a voucher")
```

Why is the ***LOGIC*** wrong in this example:

3

```
# mega sale

salesAmount = float(input("Enter the amount of money spent in £: "))

if salesAmount > 20:
    print("You qualify for a £3 voucher")

elif salesAmount >10:
    print("You qualify for a £1 voucher")

else:
    print("I'm afraid you did not spend enough today to get a voucher")
```

Will the ***LOGIC*** work in this example?

Logic is: valid reasoning

Taking care that the **LOGIC** is correct

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.

```
#Amount spent from customer
amount_spent = float(input("Enter the total amount spent: "))
#voucher to give
if amount_spent > 20:
    voucher = "$3 voucher"
elif amount_spent > 10:
    voucher = "$1 voucher"
else:
    voucher = "No voucher"
#results
print(f"give the customer: {voucher}")
```

Save this as: megasale.py

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!

```
happy = int(input("How happy are you on a scale from 1-10: "))

if happy <= 3:
    voucher = "You're sad"
elif happy > 8:
    voucher = "You're happy"
else:
    voucher = "You're not that sad"

print(voucher)
```

Save this as: Happy_message.py

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.py**

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?

```
Picture = int(input("How many pictures do you take a month: "))
Text = int(input("How many texts do you make each month: "))
Data = int(input("How much data do you use a month: "))
#prices
picture_cost = 0.10
text_cost = 0.05
data_cost = 1.00
#calculate
total = (Picture * picture_cost) + (Text * text_cost) + (Data * data_cost)
#total
print(f"Your total bill for the month would be: ${total:.2f}")
#advice
if total > 10:
    print("You would be better off on a contract.")
else:
    print("Pay-as-you-go is the better option for you.")
```

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.

```python
#secret password
Secret_password = "opensesame"
#enter password
entered_password = input("Enter the secret password to continue: ")
if entered_password == Secret_password:
    print("Access granted! Welcome to the program.")
else:
    print("Access denied. Wrong password.")
```

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

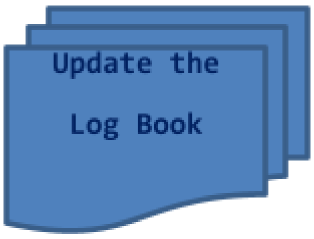
We may want to do it until something happens – and then we can use a **conditional loop.**

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)
```

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

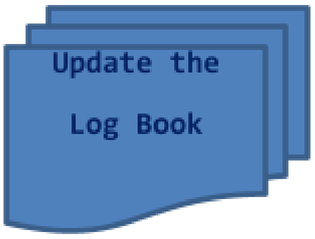


```
for a in range (1, 11):
  print(a)
```

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

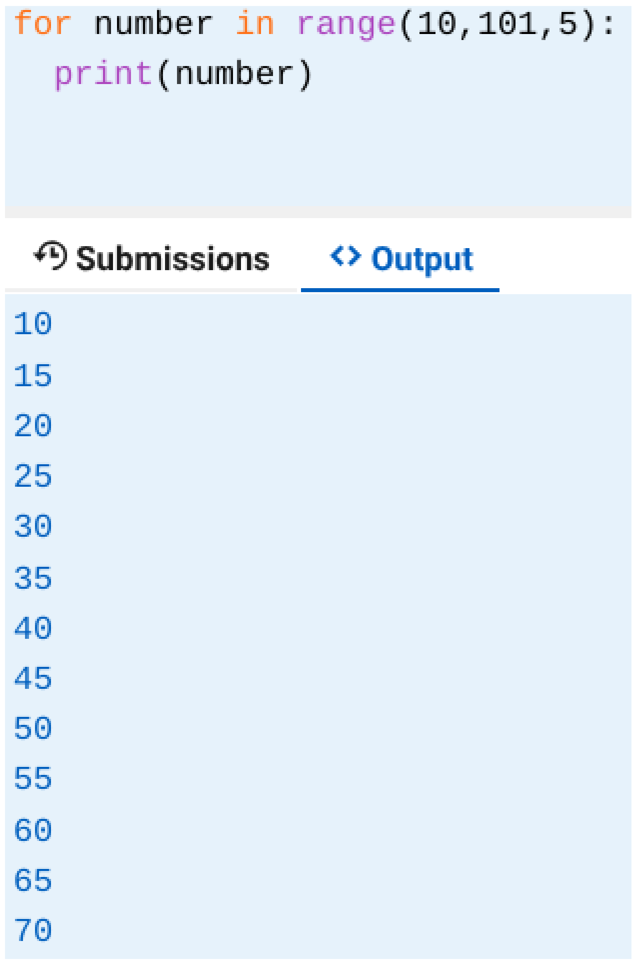
Write into your log book what this for loop does:

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



Challenge 26 – For loops

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



```
for number in range(10,101,5):
  print(number)
```

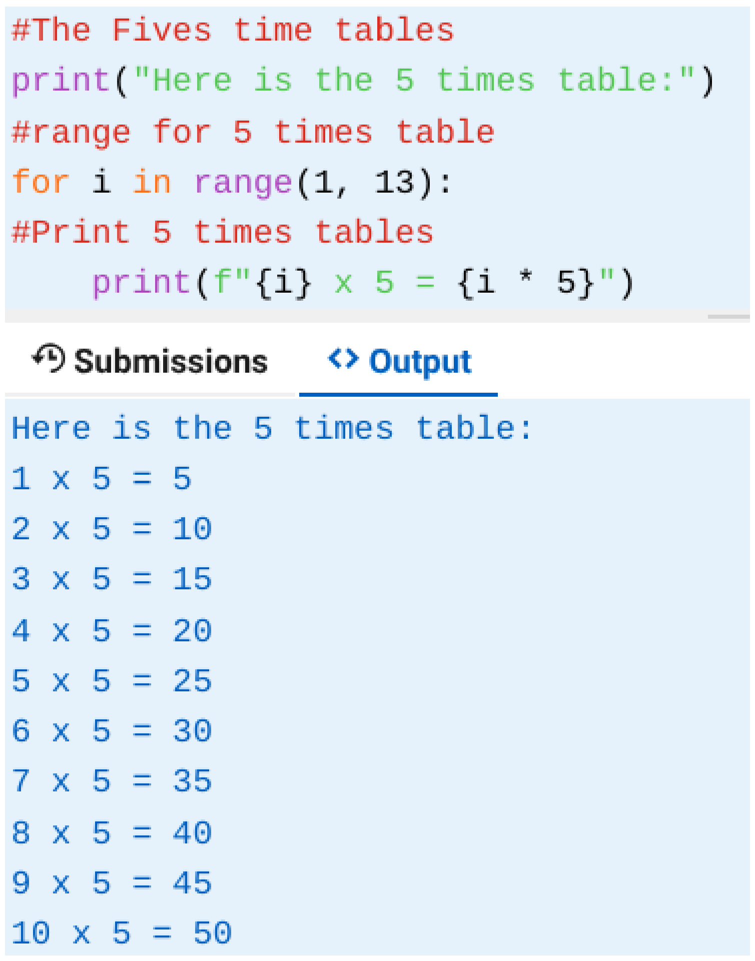
```
10
15
20
25
30
35
40
45
50
55
60
65
70
```

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



```python
#The Fives time tables
print("Here is the 5 times table:")
#range for 5 times table
for i in range(1, 13):
#Print 5 times tables
    print(f"{i} x 5 = {i * 5}")
```

```
Here is the 5 times table:
1 x 5 = 5
2 x 5 = 10
3 x 5 = 15
4 x 5 = 20
5 x 5 = 25
6 x 5 = 30
7 x 5 = 35
8 x 5 = 40
9 x 5 = 45
10 x 5 = 50
```

Save as 5 times table.

For loops are *really* clever

They can be used for even more things.

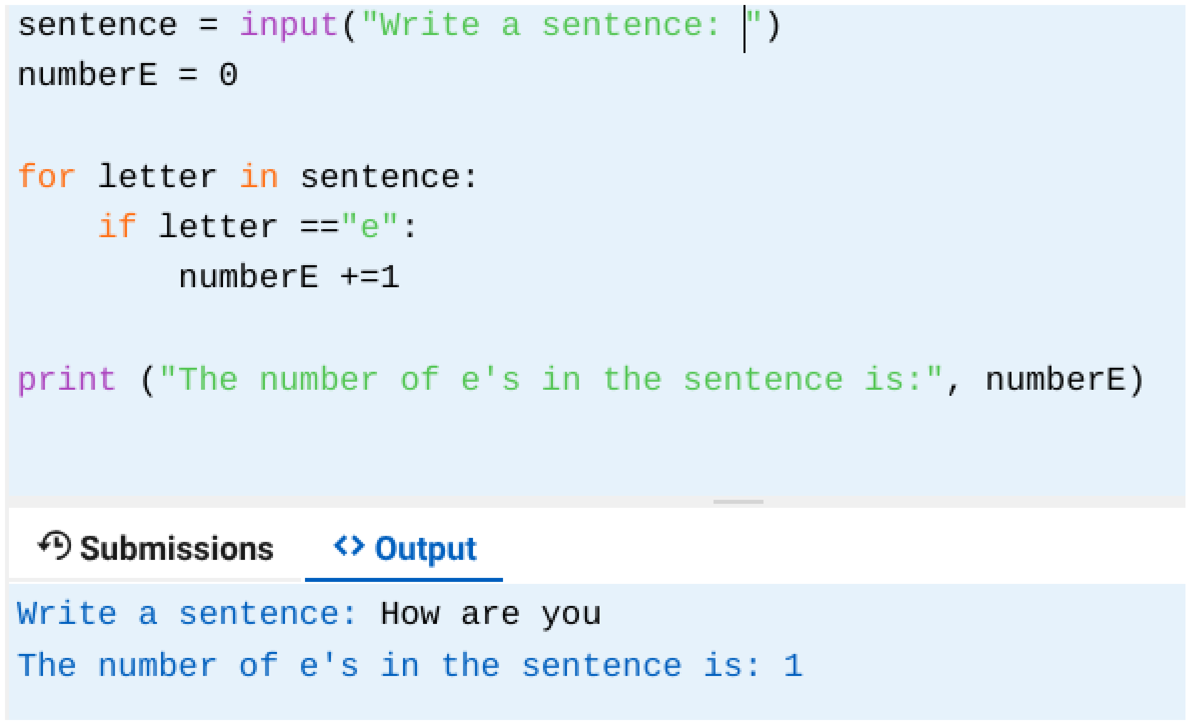
```python
sentence = "Here is a sentence. How many letter 'e''s are there in it?"
numberE = 0


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....`)



```
sentence = input("Write a sentence: ")
numberE = 0

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

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

```
Write a sentence: How are you
The number of e's in the sentence is: 1
```

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

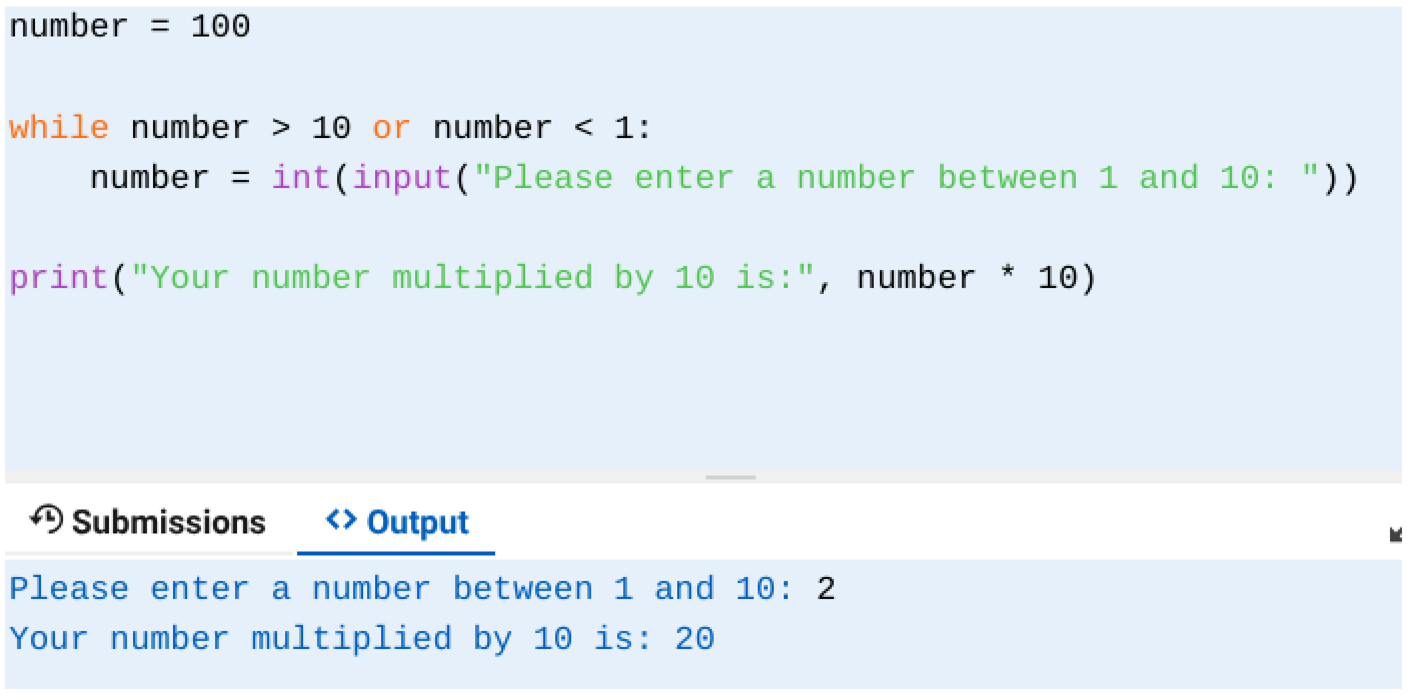
All of this program should be familiar to you apart from the while loop. Think about what it does before typing it in.

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



```python
number = 100

while number > 10 or number < 1:
    number = int(input("Please enter a number between 1 and 10: "))

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

```
Please enter a number between 1 and 10: 2
Your number multiplied by 10 is: 20
```

Save as: challenge 29

Why is the int necessary?

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

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.

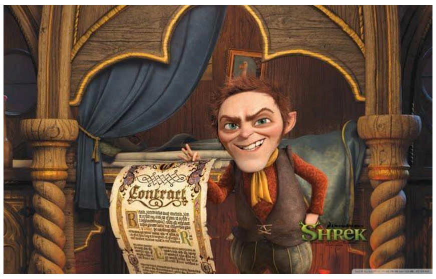
Enter the code and see how it works. Enter letters and decimals and numbers not in the range 1 to 10. Try using it as a starting point with the Rumplestiltskin program below.

Code	What it does
(see code below)	Update the Log Book

```
# 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)
```

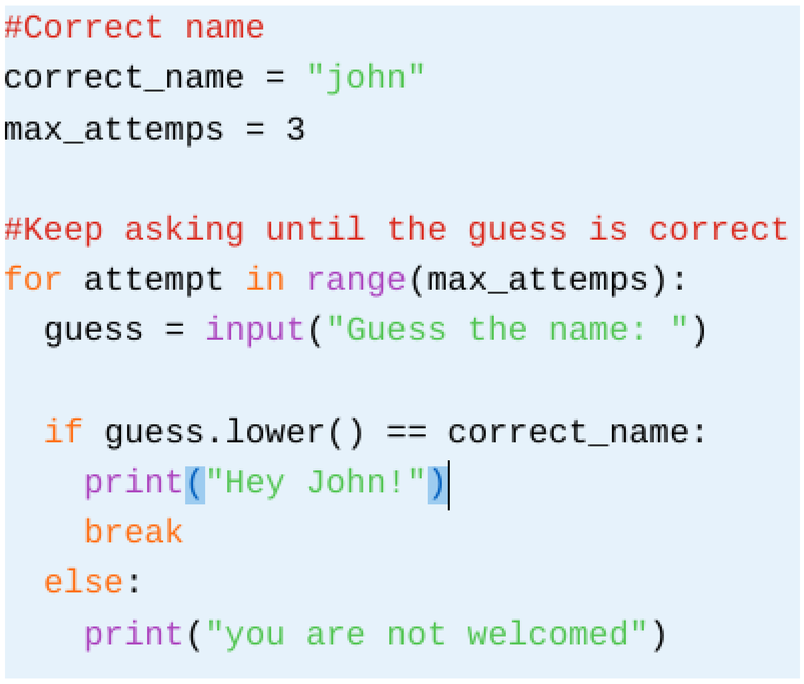
Challenge 30 – Rumpelstiltskin

Rumpelstilitskin was a nasty character in 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.



```
#Correct name
correct_name = "john"
max_attemps = 3

#Keep asking until the guess is correct
for attempt in range(max_attemps):
  guess = input("Guess the name: ")

  if guess.lower() == correct_name:
    print("Hey John!")
    break
  else:
    print("you are not welcomed")
```

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.

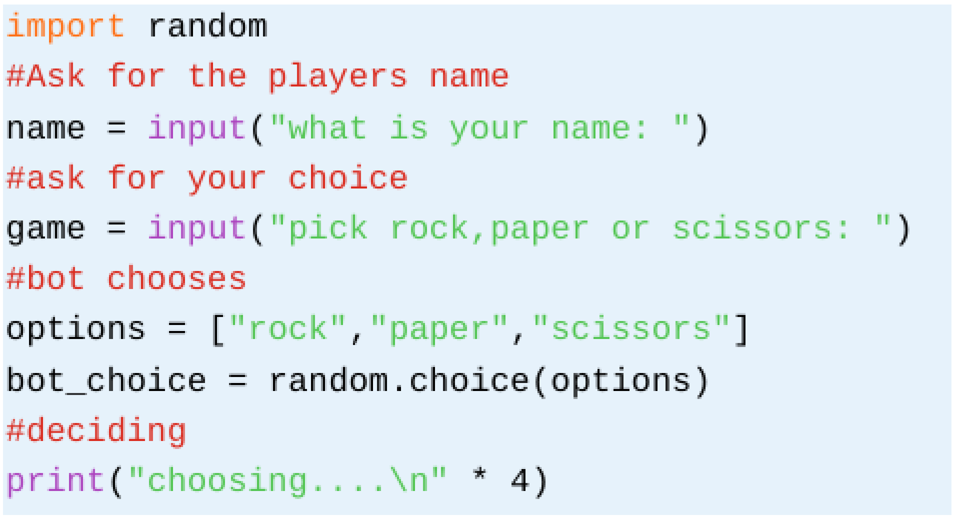
Challenge 31 – Rock, Paper, Scissors Game

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.

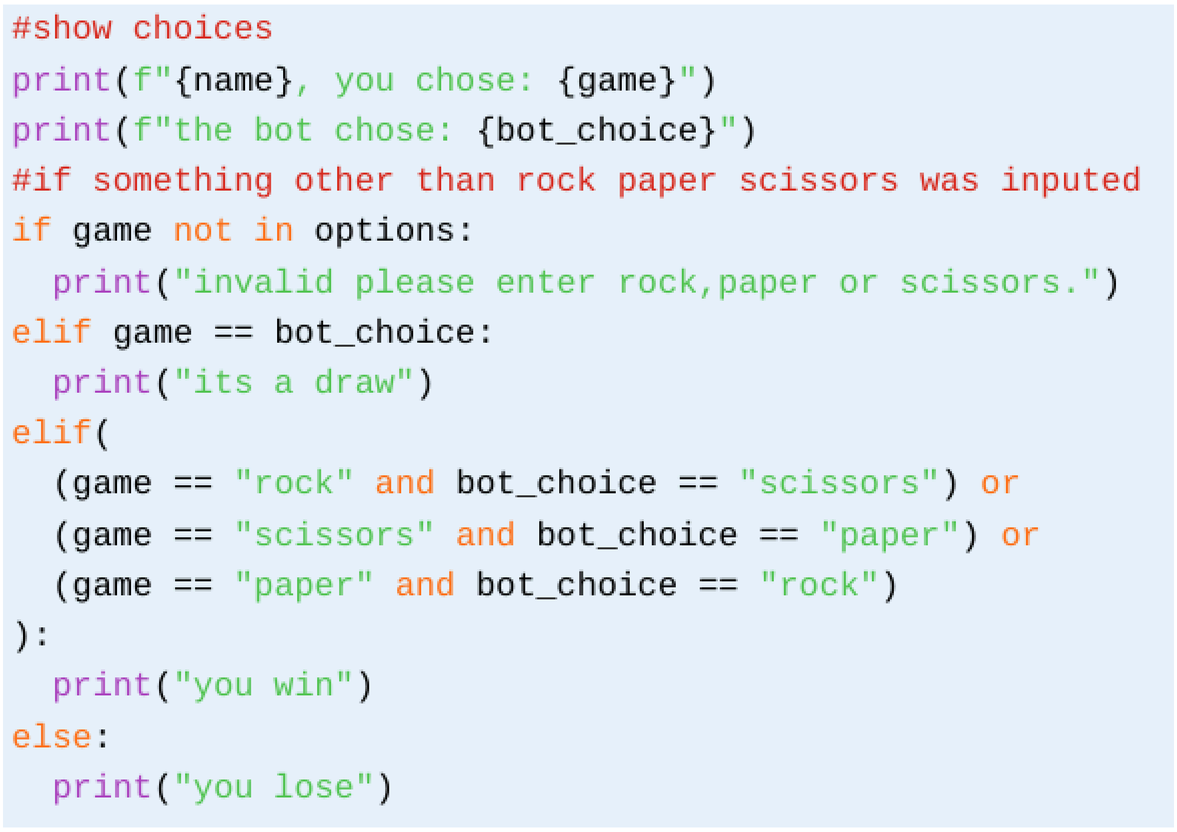
The game rules are simple: rock beats scissors, scissors beat paper, paper beats rock. Two of the same gives a draw.

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.



```
import random
#Ask for the players name
name = input("what is your name: ")
#ask for your choice
game = input("pick rock,paper or scissors: ")
#bot chooses
options = ["rock","paper","scissors"]
bot_choice = random.choice(options)
#deciding
print("choosing....\n" * 4)
```



```
#show choices
print(f"{name}, you chose: {game}")
print(f"the bot chose: {bot_choice}")
#if something other than rock paper scissors was inputed
if game not in options:
  print("invalid please enter rock,paper or scissors.")
elif game == bot_choice:
  print("its a draw")
elif(
  (game == "rock" and bot_choice == "scissors") or
  (game == "scissors" and bot_choice == "paper") or
  (game == "paper" and bot_choice == "rock")
):
  print("you win")
else:
  print("you lose")
```

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	Paper	2	User +=0 / Computer +=1
Rock	1	Scissors	3	User +=1 / Computer +=0
Paper	2	Rock	1	User +=1 / Computer +=0
Paper	2	Paper	2	User +=1 / Computer +=1
Paper	2	Scissors	3	User +=0 / Computer +=1
Scissors	3	Rock	1	User +=10 / Computer +=1
Scissors	3	Paper	2	User +=1 / Computer +=0
Scissors	3	Scissors	3	User +=1 / Computer +=1

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

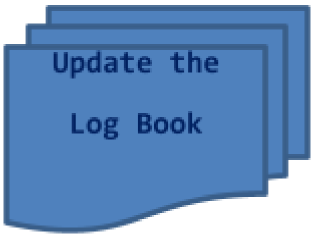
With a times table we start with a number and then multiply this by a series of numbers.

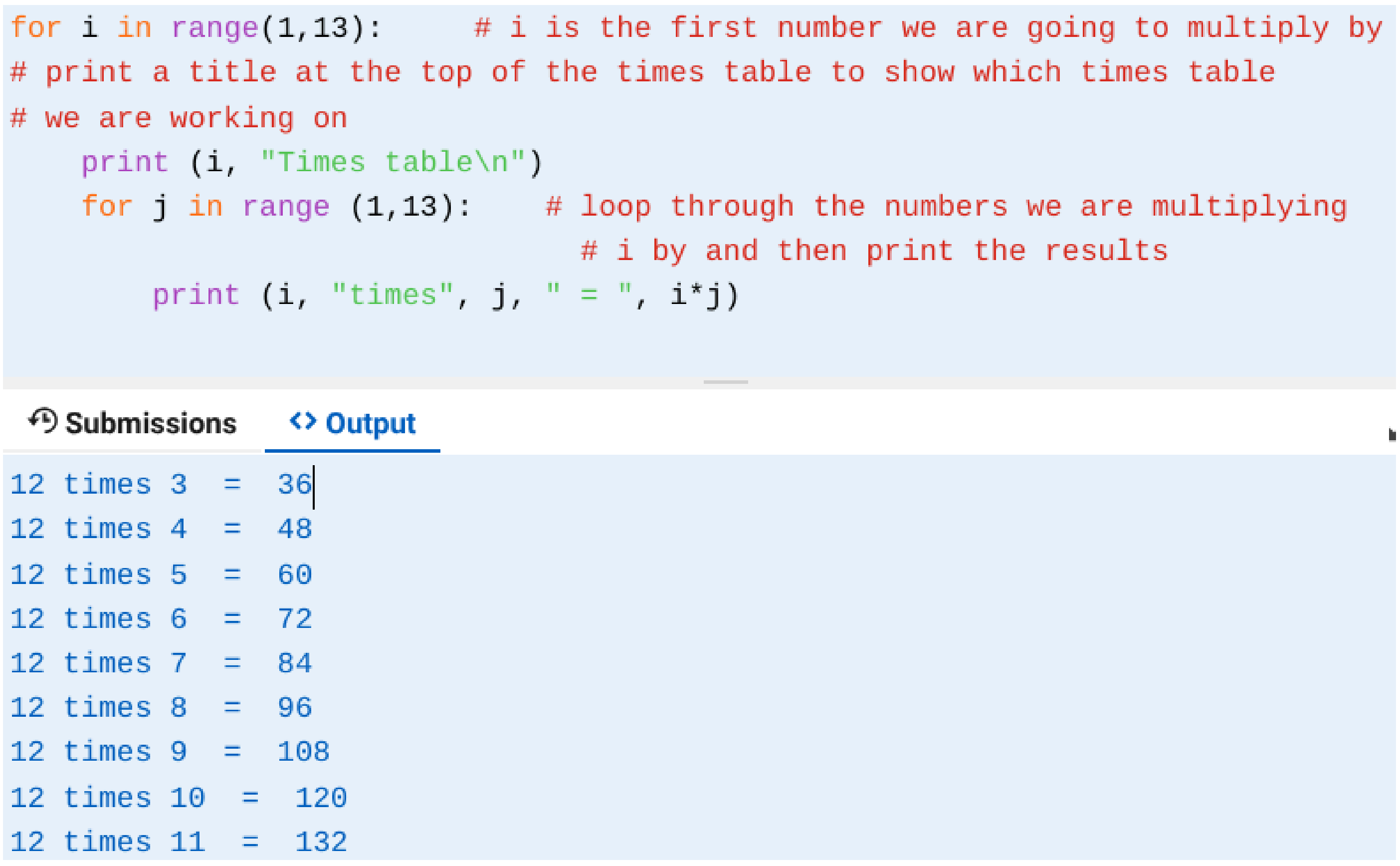
For the 2 times table we multiply it by 1 to 12, before moving on to do the 3 times table, etc.

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
# 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)
```





```
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
# 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)
```

```
12 times 3  =  36
12 times 4  =  48
12 times 5  =  60
12 times 6  =  72
12 times 7  =  84
12 times 8  =  96
12 times 9  =  108
12 times 10  =  120
12 times 11  =  132
```

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 as 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.

```
# 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
```

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

We can define our own function to draw a square:

```
import turtle
def square():               # define 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()

wn = turtle.Screen()
tom = turtle.Turtle()

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)
```

1

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

Here square passes a value 80 to the function

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

```
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)
```

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

Challenge 35 – Area or Perimeter

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

width = 100

length = 200

Area = length * width

Perimeter = length *2 + width *2

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

```
def area():
   shapeArea = length * width
   print("Area = ",shapeArea)

def perimeter():
   shapePerimeter = length*2 + width*2
   print ("Perimeter = ", shapePerimeter)
```

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

def area():

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

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

Update the

Log Book

```
# 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()

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":
    perimeter()
```

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.

Enter this into Python and test it.

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

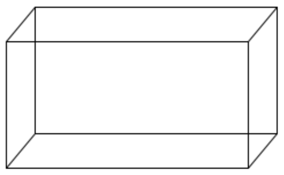
Extension

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")
```
	The parameter in this code is:_______________
	
	within the function_________________

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

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():
```
<pre> favourites ("Computing", "Burgers and Chips\n")</pre>	Positional parameter
<pre> favourites ("Spag Bol", "Computing\n")</pre>	Positional parameter

```
    favourites  (meal = "Chicken wraps",
subject="Computing")

main()
```
	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]);	it doesn’t work
print ("We can also just ask to print out the list. So here is monday:") print(monday)	it doesn’t work
print (“We can also print them out one at a time”) for i in monday: print (i)	it doesn’t work

Update the Log Book

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];	
print (bars[2:]);	
print (bars[:]);	

Update the Log Book

Adding to a list:

Instruction	What it does
bars.append("Snickers"); print (bars)	
bars.insert(1,"Horn"); print (bars)	
bars.extend(["Picnic","Twirl"]) print (bars)	Update the Log Book

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", "Horn", "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)	
myBar = bars.pop() print (bars) print (myBar)	
myBar1 = bars.pop(0) print (bars) print (myBar1)	

Update the Log Book

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 :

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

from random import choice

print (choice(letter))
```

Update the

Log Book

Challenge 36

A Teacher is to award a prize to a random student from a list of students who have qualified by being given a positive recognition.

Help them by writing a program that will randomly pick a student from the following list.

Andrew , Bobby , Renato , Tracey, Adam , Michael , Nicci , Mary , Lance , Soula

Save as **Positive_St.py**

Dictionaries

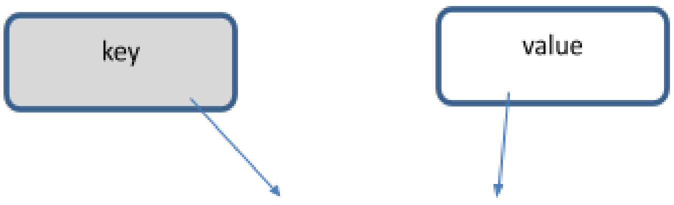
A dictionary in python is very much like a list. It stores lots of data. But with a dictionary you store data in pairs so that they are related to each other: so you have a piece of information – **the key** – and a piece of data – **a value**. The format of a dictionary is:

```
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.



```
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

Create the following dictionary:

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

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

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

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

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.

```
#countries and their population
countryPop = {
    "Japan": "127000000",
    "Germany": "81000000",
    "Iran": "77000000",
    "UK": "64000000",
    "Canada": "33000000",
    "Australia": "23000000",
    "USA": "317000000",
    "Bulgaria": "7000000",
    "Sweden": "10000000"
}

#greet the user and explain what this program is
print("Welcome! This program will add the populations of two countries.")
print("Here are the available countries:")
print(", ".join(countryPop.keys()))
```

```
#ask for two countries
country1 = input("Enter the name of the first country: ")
country2 = input("Enter the name of the second country: ")

#convert populations from strings to integers before adding
pop1 = int(countryPop[country1])
pop2 = int(countryPop[country2])
total = pop1 + pop2

#result
print(f"The population of {country1} is {pop1:,}")
print(f"The population of {country2} is {pop2:,}")
print(f"The combined population is: {total:,}")
```

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.

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

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

myFile.close()
```

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.

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.

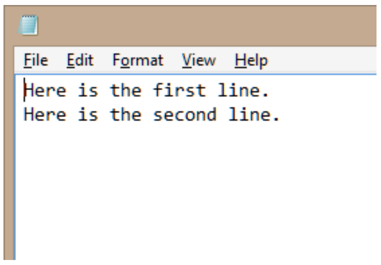
One way is to write the data to separate lines in the file. We can do this by adding the newline instruction:

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

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

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

myFile.close()
```

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.



```
Here is the first line.
Here is the second line.
```

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.

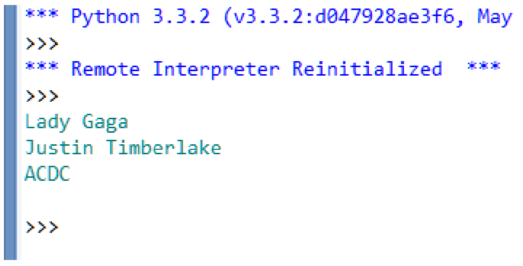
An important thing to note is that using wt means that the contents of the file is deleted each time!

Reading from a File

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

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

contents = myFile.read()

print (contents)

myFile.close()
```

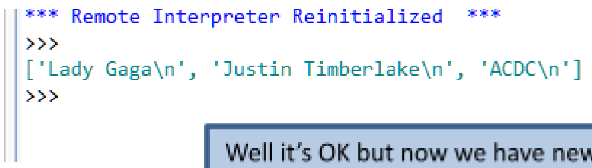
This time we open the file with the ‘rt’ command. This means open it for reading. The contents of the file isn’t deleted this time!



```
*** Python 3.3.2 (v3.3.2:d047928ae3f6, May
>>>
*** Remote Interpreter Reinitialized  ***
>>>
Lady Gaga
Justin Timberlake
ACDC

>>>
```

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()
```



```
*** Remote Interpreter Reinitialized  ***
>>>
['Lady Gaga\n', 'Justin Timberlake\n', 'ACDC\n']
>>>
```

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.

```
#ask for 6 numbers and write them to a file
numbers = []

print("Enter 6 numbers:")

for i in range(6):
    num = float(input(f"Enter number {i+1}: "))
    numbers.append(num)

#write the numbers to a file
with open("numbers.txt", "w") as file:
    for num in numbers:
        file.write(str(num) + "\n")

print("Numbers saved to 'numbers.txt'.")
```

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

```
#read numbers from file, print all numbers, total, and average
numbers = []

#read the numbers from the file
with open("numbers.txt", "r") as file:
    for line in file:
        numbers.append(float(line.strip()))

#print each number
print("Numbers read from file:")
for num in numbers:
    print(num)

#calculate total and average
total = sum(numbers)
average = total / len(numbers)
```

```
print(f"Total: {total}")
print(f"Average: {average}")
```

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 39

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.

```
heightandweight = [
    "James", 73, 1.82,
    "Peter", 78, 1.80,
    "Beth", 65, 1.53,
    "Mags", 66, 1.50,
    "Joy", 62, 1.34
]
#write the data to a file
heightandweight = [
    "James", 73, 1.82,
    "Peter", 78, 1.80,
    "Beth", 65, 1.53,
    "Mags", 66, 1.50,
    "Joy", 62, 1.34
]
```

```
with open("peopledata.txt", "w") as file:
    for i in range(0, len(heightandweight), 3):
        name = heightandweight[i]
        height = heightandweight[i+1]
        weight = heightandweight[i+2]
        file.write(f"{name},{height},{weight}\n")
```

```
print("Data has been written to 'peopledata.txt'.")
#read the data and print it
with open("peopledata.txt", "r") as file:
    print("People's height and weight data:")
    for line in file:
        name, height, weight = line.strip().split(",")
        print(f"Name: {name}, Height: {height}, Weight: {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.