Name:

Mulberry Academy Shoreditch

Sixth Form

Summer Assignment

A Level Computer Science

If you did GCSE CS…

1. Ensure you can do challenges 1-50 below for non-GCSE students

PyGame Programming

In order to be successful in A Level Computer Science, you MUST program in your spare time at home. If you don't do this, you will not have the level of programming needed to do the project or the exams. Programming is a much bigger part of the course than it was at GCSE.

With this in mind, over summer you should start gaining the skills needed to independently learn to program. Your summer assignment is to create a game using the PyGame library.

Your game must meet the following criteria:

- At least one player controlled by arrow keys
- Items to collect
- An end of the game (e.g. when all coins are collected)

Extension tasks:

- Multi-player
- Enemies
- Multiple levels
- Introduction and end screen
- Use your creativity!

I recommend using one of the following resources:

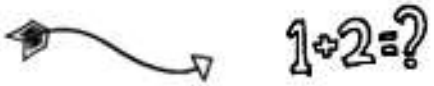
1. http://programarcadegames.com/
    a. This is an excellent website (how I learned PyGame!). To get the most from it, you need to read and run each bit of code as you go in order to fully understand what you're doing.
2. https://youtube.com/playlist?list=PLxPGuI-jRTkyqubyUX4GTh2UJQYbhQyyK
    a. Search on youtube 'Lauren Gillott' → go to my channel → find the playlist called **PyGame Tutorials (Procedural)**
    b. These are tutorials from Ms Gillott showing you the basics of PyGame before talking you through making full games.
    c. Note: these are **procedural** programming. There is also a playlist on **Object Oriented** Programming. You haven't learned this yet, but will in Y12.

If you did not do GCSE CS…

1. Create a login for repl.it using your Teams email.
2. Complete 1-50 of the below challenges on repl.
    a. To create a new file, click the + and choose Python as the language
    b. To run your program, click the Run button. The output will be on the right hand side.
3. Ask a peer who did the GCSE/ email Ms Gillott if you need help

Example Code

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



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

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

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

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

answer = num1 // num2
A whole number division (i.e. 9//4 = 2) and stores the answer in a variable called answer.

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

print ("First line\nSecond line")
"\n" is used as a line break.

print ("The answer is", answer)
Displays the text "The answer is" and the value of the variable answer.

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

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

Challenges

001

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

002

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

003

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

004

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

005

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

006

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

007

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

008

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

009

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

010

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

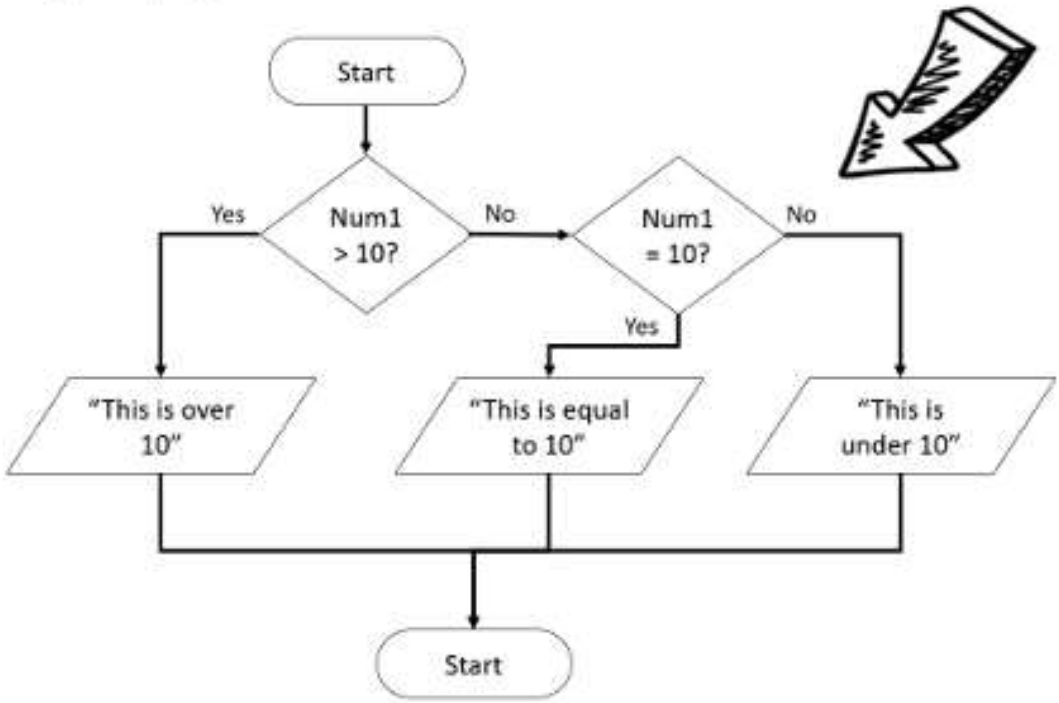
011

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

If Statements

Explanation

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



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

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

Indenting Lines of Code

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

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

Comparison Operators

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

Logical Operators

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

Example Code

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

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

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

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

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

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

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

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

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

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

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

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

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

Challenges

012

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

013

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

014

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

015

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

016

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

017

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

018

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

019

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

Strings

Explanation

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

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

" ' \

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

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

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

Strings and Numbers as Variables

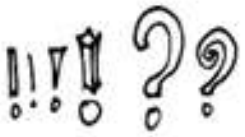
If you define a variable as a string, even if it only contains numbers, you cannot later use that string as part of a calculation. If you want to use a variable that has been defined as a string in a calculation, you will have to convert the string to a number before it can be used.

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

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

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

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



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

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

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

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

The same can happen with strings.

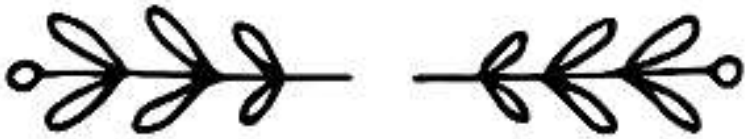
```
name = input("Enter a name: ")
num = int(input("Enter a number: "))
ID = name + num
print(ID)
```

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

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

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

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



Multiple-Line Strings

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

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

Example Code

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

`len(word)`
Finds the length of the variable called word.

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

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

`word.upper()`
Changes the string into upper case.

`word.lower()`
Changes the string into lower case.

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

```
text = " This is some text. "
print(text.strip(" "))
```
Removes extra characters (in this case spaces) from the start and end of a string.

`print ("Hello world"[7:10])`
Each letter is assigned an index number to identify its position in the phrase, including the space. Python starts counting from 0, not 1.

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

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

Don't forget that you can reuse previous programs to save time when you are making new programs. Simply use SAVE AS and give it a new name.

Challenges

020

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

021

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

022

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

023

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

024

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

025

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

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

026

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

Maths

Explanation

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

Example Code

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

`print(round(num,2))`
Displays a number rounded to two decimal places.

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

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

`num=float(input("Enter number: "))`
Allows numbers with a decimal point dividing the integer and fraction part.

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

`x // y`
Whole number division (e.g.15//2 gives the answer 7).

`x % y`
Finds the remainder (e.g. 15%2 gives the answer 1).

Challenges

027

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

028

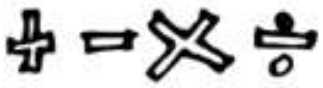
Update program 027 so that it will display the answer to two decimal places.

029

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

030

Display pi (π) to five decimal places.



031

Ask the user to enter the radius of a circle (measurement from the centre point to the edge). Work out the area of the circle (π*radius^2).

032

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

033

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

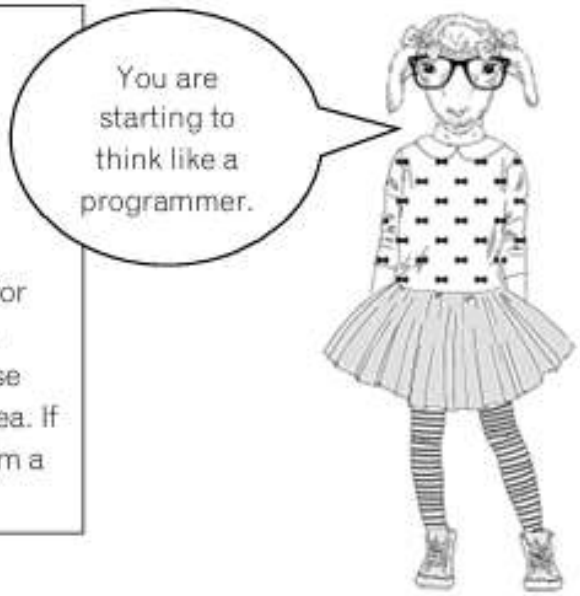
034

Display the following message:

```
1) Square
2) Triangle

Enter a number:
```

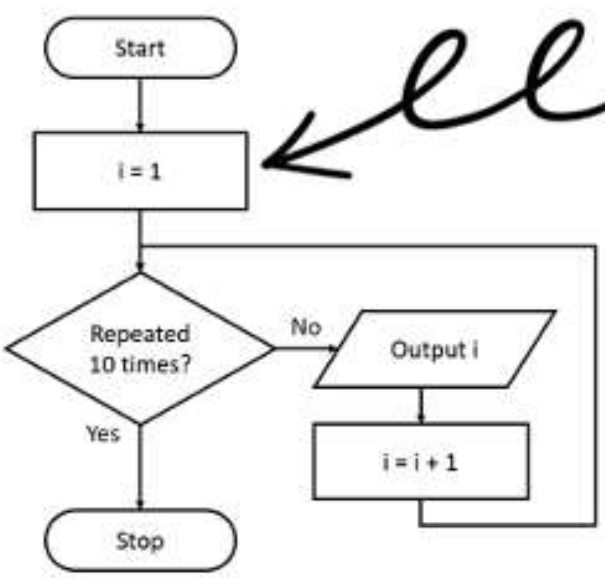
If the user enters 1, then it should ask them for the length of one of its sides and display the area. If they select 2, it should ask for the base and height of the triangle and display the area. If they type in anything else, it should give them a suitable error message.



For Loop

Explanation

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



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

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

In this example, the outputs would be 1, 2, 3, 4, 5, 6, 7, 8 and 9. **When it gets to 10 the loop would stop so 10 would not be shown in the output.**

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

Example Code

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

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

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

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

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

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

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

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

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


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

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

Challenges

035

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

036

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

037

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

038

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

039

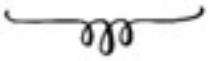
Ask the user to enter a number between 1 and 12 and then display the times table for that number.

040

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

041

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



042

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

043

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

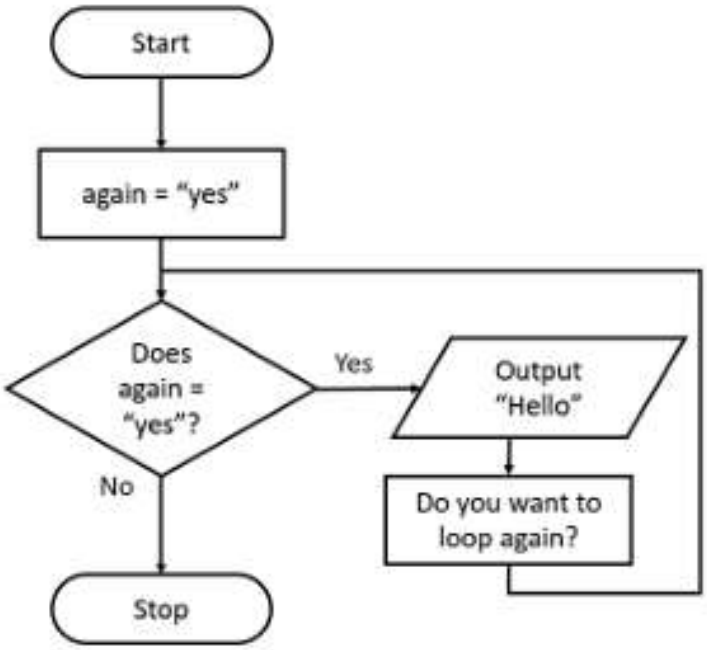
044

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

While Loop

Explanation

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



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

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

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

Example Code

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

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

Comparison Operators

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

Logical Operators

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

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

Challenges

045

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

046

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

047

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

048

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

049

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

050

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

051

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