PROGRAMMING MICROBITS

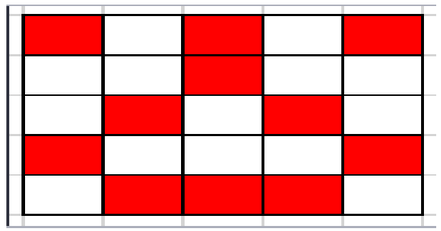
# KNOWLEDGE ORGANISER

Fulwood academy

## KEY SUBJECT TERMONOLGY

| Tier 2 Vocab: | Tier 3 Vocab: |
|---|---|
| Program, block, algorithm, refactor, define, code, conditional, statements, calculate, absolute, variable, random, repetition datatypes, develop, selection, implement, instructions, sequence, solutions, decomposition, distinct, succinct, processing,. | Loop, conditionals, execute, pseudo code, Boolean, string, float, integer, operators, nested, syntax. |

## REPRESENTING IMAGES

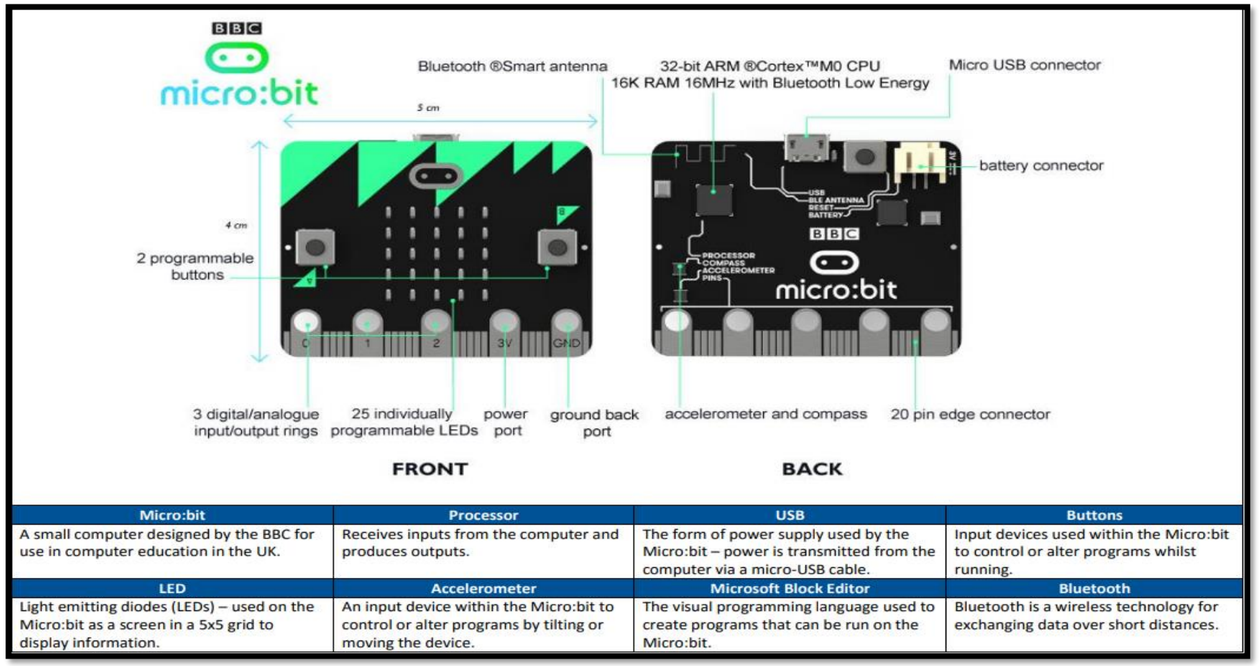


pattern = Image("10101:00100:01010:10001:01110")

## REFACTORING CODE

Loops are often used in code – they are **used to repeat certain sections of your program.**

Benefits of refactoring code to use a loop? > **Allows your code to be repeated without the need of typing it out over and over, this avoids errors and keeps the code short.**




| Micro:bit | Processor | USB | Buttons |
|---|---|---|---|
| A small computer designed by the BBC for use in computer education in the UK. | Receives inputs from the computer and produces outputs. | The form of power supply used by the Micro:bit – power is transmitted from the computer via a micro-USB cable. | Input devices used within the Micro:bit to control or alter programs whilst running. |
| **LED** | **Accelerometer** | **Microsoft Block Editor** | **Bluetooth** |
| Light emitting diodes (LEDs) – used on the Micro:bit as a screen in a 5x5 grid to display information. | An input device within the Micro:bit to control or alter programs by tilting or moving the device. | The visual programming language used to create programs that can be run on the Micro:bit. | Bluetooth is a wireless technology for exchanging data over short distances. |

## COMMENTING CODE

Commenting on your code is an important way to learn and show your understanding of what the code does.

Start all comments with # so that the program knows it is a comment.

## CONDITIONAL STATEMENTS

Conditional statement **(IF THEN ELSE)**
Tests a condition and makes a decision what to do next

## VARIABLES

Variables are like **a box that you can name and store data for you**

**Different types of data can be stored in these**

## BRACKETS

Brackets in programming are important as they identify and retrieve items from a collection

Using them correctly is important to avoid syntax errors.

**[ ] brackets used when defining a list**

**( ) brackets are used when we want to display text or images**

## CODE THAT WILL BE USED IN THIS UNIT

| CODE | PURPOSE |
|---|---|
| display.scroll | used to scroll text |
| display.show | use to show images or patterns |
| sleep(1000) | used to rest or delay code |
| Loop = True | used to repeat code whilst the condition is true |
| display.show (random.choice(variablenamehere)) | use to choose a random from your variable list |
| If accelerometer.was_gesture(“shake”) | used to use the accelerometer feature |
| If gesture = = “left” or “right” | Used to test the condition of the gesture |
| If button_a.is_pressed(): | Used to test the condition of button A |
| Elif button_b.is_pressed(): | Used to test the condition of button B |
| Name = (“Miss W”) | Defining a variable |
| Name = [name, name1, name2] | Defining a variable as a list |

WE CARE • WE CHALLENGE • WE COMMIT

# KNOWLEDGE ORGANISER



## KEY SUBJECT TERMONOLGY

**Tier 2 Vocab:**

CODE / CODING

LOOP / REPEATING

CONDITION

STATEMENTS

BLOCKS / TEXT / RULES / INSTRUCTIONS

**Tier 3 Vocab:**

PROGRAMS / PROGRAMMING LANGUAGE

ITERATION / REFRACTION

CONDITIONAL STATEMENTS

ALGORITHMS

## EduBlocks Setup

**Go to app.edublocks.org**

**Select Python**

**You're all setup!**

**Predict** Be able to make educated guesses about what you believe a particular code will do when executed

**Run** Be able to put/write code into Python IDE and run the code

**Investigate** Be able to read, identify errors and understand what each line of code is intended for

**Modify** Be able to fix errors and or improve code to make is better or change the outcome

**Make** Be able to use the skills learnt to create your own program or for a given scenario

REMEMBER ME!!

## Every language uses the same key coding concepts!

**Sequence** The instructions for our code

**Selection** Using logical tests to change the flow of the sequence

**Iteration** Using loops to repeat sequences of code

# KNOWLEDGE ORGANISER



## PRINT FUNCTION

EduBlocks


Hello!

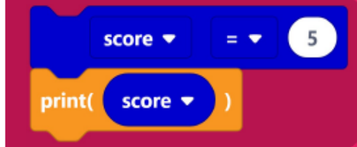
Python

```
import time
print("Hello!")
time.sleep(2)
```

Hello!

## COUNTING

EduBlocks




5

Python

```
score = 5
print(score)
```

5

## LIBRARIES

EduBlocks


2

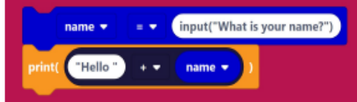
Python

```
import random
print(random.randint(1,6))
```

3

## USER INPUT

EduBlocks




What is your name? George
Hello George

Python

```
name = input("What is your name?")
print("Hello " + name)
```

What is your name? George
Hello George

## LOOPING

EduBlocks


Hello World!
Hello World!
...

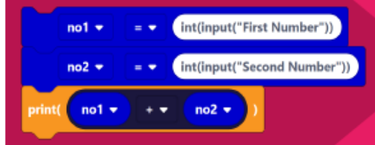
Python

```
import time
while True:
  print("Hello World!")
  time.sleep(1)
```

Hello World!
Hello World!
...

## CONCATENATING VARIABLES

EduBlocks




7

Python

```
no1 = int(input("First Number"))
no2 = int(input("Second Number"))
print(no1 + no2)
```

7

## OPERATORS

# KNOWLEDGE ORGANISER



## SUB FUNCTIONS

**EduBlocks**


Hello!

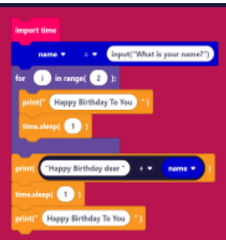
**Python**

```
import time
print("Hello!")
time.sleep(2)
```

Hello!

## MAKE

**EduBlocks**



What is your name? George
Happy Birthday To You
Happy Birthday To You
Happy Birthday Dear George
Happy Birthday To You

**Python**

```
import time
name = input("What is your name?")
for i in range(2):
  print("Happy Birthday To You")
  time.sleep(1)
print("Happy Birthday dear " + name)
time.sleep(1)
print("Happy Birthday To You")
```

What is your name? George
Happy Birthday To You
Happy Birthday To You
Happy Birthday Dear George
Happy Birthday To You

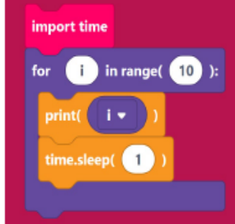
## DATA TYPES

At the basic level there are three data types.

- String: A sequence of characters, including punctuation, numbers and letters.
- Integer: A number with no decimal place.
- Float: A number with a decimal place.
- Character: (or char). Used for single letters
- Boolean (or bool). Used where data is restricted to True/False or yes/no options.

## TIMER

**EduBlocks**




0 5
1 6
2 7
3 8
4 9

**Python**

```
import time
for i in range(10):
  print(i)
  time.sleep(1)
```

0 5
1 6
2 7
3 8
4 9

## RUN

**How to run your code:**

| Scratch | EduBlocks | Python |
|---|---|---|
| Click the first block | Click the Run Button | Press Play |

REMEMBER ME!!

Sequencing is the specific order in which instructions are performed in an algorithm.

## ALGORITHM

REMEMBER ME!!

You use code to tell a computer what to do. Before you write code you need an algorithm. An algorithm is a list of rules to follow in order to solve a problem. They are at the heart of all computer programs.

# PYTHON PROGRAMMING

# KNOWLEDGE ORGANISER

Fulwood academy

## KEY SUBJECT TERMONOLGY

Tier 2 Vocab:

Programming, Code, Language, Sequence, Statements, Print, Run, programs, applications, distinct, succinct, processing, Absolute, Define, Random, Calculate, Selection, Implement, Instructions.

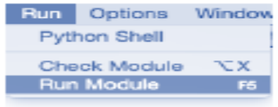
Tier 3 Vocab:

Coding, Algorithm, Logic, Binary, Data types, Conditionals, Variables, Abstraction, Decomposition, Execute, Syntax, Refactor, Nested, Loops, Boolean, comparative operator, syntax, string, integer, float, char, pseudo code.

## EXECUTING CODE

### Executing a program

In order to run or test a program written in Python the user needs to go to **Run** and then **Run Module.**




Alternatively, you could press the F5 button on the keyboard.

## SYNTAX

### Syntax

Syntax is what we call the format that the code needs to be in, in order to be processed correctly.
If it is not in the correct format then the code will not work.

```
Traceback (most recent call last):
  File "C:/Python33/a.py", line 2, in <module>
    prin (greeting)
NameError: name 'prin' is not defined
>>>
```

Python tells us where the error is and what type it is.
Here it says the line the error is on
Here it says what type of error.

## BASICS

### What is a Python?

Python is a **text based programming language** that can be used to create small programs, web applications, games and even search engines like Google and YouTube!

Python is easy to learn and is a great beginner language.

### Print statements

In order to display text in the **shell** you need to use a **Print** statement.

```
print ("Hello World")
print ("I am a programmer")
```

This is the output:

```
Hello World
I am a programmer
```

### Input statements

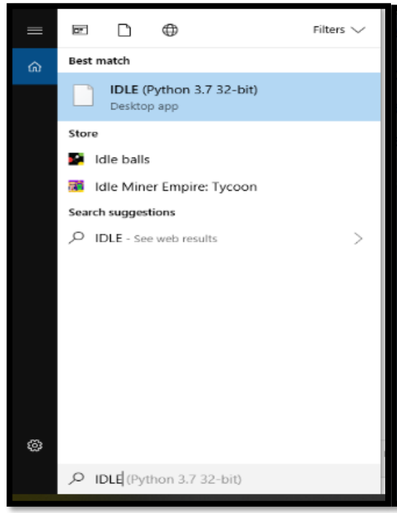
Using **var = input ()** we can ask a user to input some information.

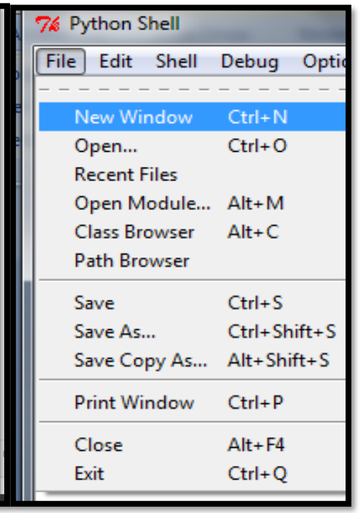
We can then **print** this back to the console window.

```
userName = input("what is your name?")
print ("Welcome ", userName)
```

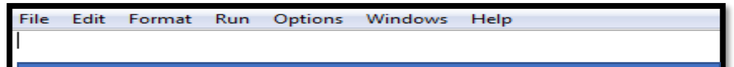
**userName** is a **variable.** This means we can change the information stored. We can also name it whatever we want.

## SETTING UP PYTHON

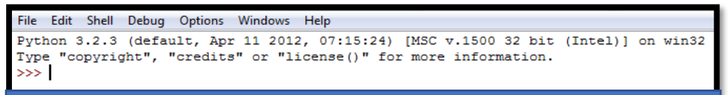



## INPUT WINDOW (CODE IS WRITTEN HERE)




## OUTPUT WINDOW (OUTPUT ONCE CODE IS RAN)




## IDE COLOUR CODING

| Colour | Used for | Examples |
|---|---|---|
| **Black** | Data and Variables | **23.6 area** |
| **Green** | Strings | "Hello World" |
| **Purple** | Functions | len() print() |
| **Orange** | Commands | if for else |
| **Blue** | User Function | get_area() |
| **Dark Red** | Comments | #Remember VAT |
| **Light Red** | Error Message | SyntaxError |

## VARIABLE / DATA TYPES

### Variables

A variable is something that can be **used to store information.** The information that is stored can be changed.

### Data types

Different types of data are stored in variables as different data types. There are three main data types:
**String, Integer & Float**

**String**
A type of variable for storing text "strings" e.g. "Hello World"

```
string = str("This is a string")
```

**Integer**
A type of variable for storing **whole numbers**
e.g. 10, 182, -44

```
integer = int("This is an integer")
```

**Float**
A type of variable for storing **decimal numbers**. Also known as a real number
e.g. 2.5, 5.05, 3.14

```
decimal = float("This is a decimal")
```

PYTHON PROGRAMMING

# KNOWLEDGE ORGANISER

Fulwood academy

## KEY SYNTAX

Python functions use brackets. Examples of how they are used – (“Hello World”) or exit()

## CONVENTIONAL NAMING OF VARIABLES

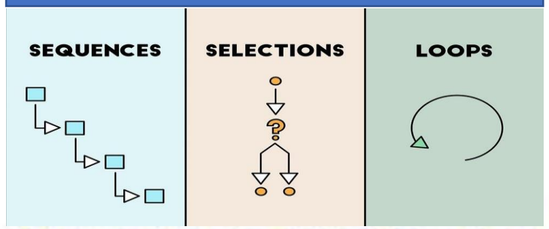
When naming a variable, keep it short and it should identify its purpose. For example, id the data stored in a variable is a player’s score in a game, you would name the variable score!

You must not use spaces or symbols when naming a variable. For example: Syb@l s

If a space must be used, then replace it with an underscore _ Or capital letter.

play_name  playerName

## PROGRAMMING CONCEPTS




## PYTHON TO ENGLISH

| Python -> English | |
|---|---|
| `print('hello!')` | Prints a value on screen (in this case, hello!) |
| `input('')` | Inputs a value into the computer. |
| `x=input('')` | Inputs a value and stores it into the variable x. |
| `x=int(input(''))` | Inputs a value into x, whilst also making it into an integer. |
| `print(str(x))` | Prints the variable x, but converts it into a string first. |
| `if name == "Fred":` | Decides whether the variable ‘name’ ha a value which is equal to ‘Fred’. |
| `else:` | The other option if the conditions for an if statement are not met (eg. name = ‘Bob’ when it should be Fred) |
| `elif name == "Tim"` | elif (short for else if) is for when the first if condition is not met, but you want to specify another option. |
| `#` | # is used to make comments in code – any line which starts with a # will be ignored when the program runs. |

## ALGORITHMS

An algorithm is a set of step-by-step instructions.

Computer programmers write algorithms to tell the computers what to do!

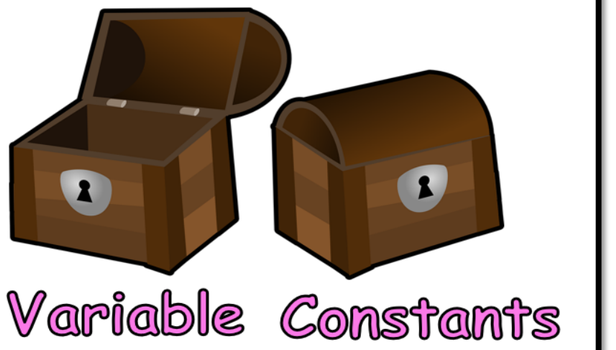
Computers need order and sequence – programs must have a logical sequence to get a desired result.


## VARIABLES / CONSTANTS

### Variables

**Understanding variables**

Sometimes we need computers to remember the  we give it and that it calculates during programs. **And can be thought of as a box that the computer can use to store a value**. The value held in that box can change or ‘vary’.
A  can use as many variables as it needs it to.
All variables are made up of three parts:
- a name
- a type
- a value




## VARIABLES / DATA TYPES

| Variable Name | Example Data | Data Type |
|---|---|---|
| Player Name | “John Smith” | String |
| Best Score | 5000 | Integer |
| Worst Score | 109 | Integer |
| Average Score | 3437.5 | Float |
| Game Completed | False | Boolean |
| Difficulty | “M” | Character |

## MORE THAN ONE WAY TO WRITE A PROGRAM

```
>>> print("Hello World!")
Hello World!
>>>
```

```
1   msg = "Hello World"
2   print(msg)
3
```

WE CARE • WE CHALLENGE • WE COMMIT

PYTHON PROGRAMMING

# KNOWLEDGE ORGANISER

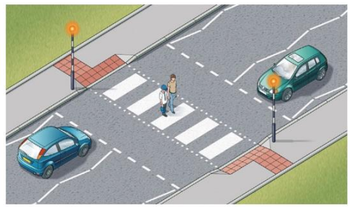
Fulwood academy

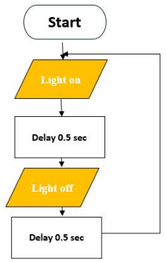
## FLOWCHARTS

A **flowchart** is a graphical representation of an algorithm.

Each step in the algorithm is represented by a symbol.

Symbols are linked together with arrows showing the order in which steps are executed.

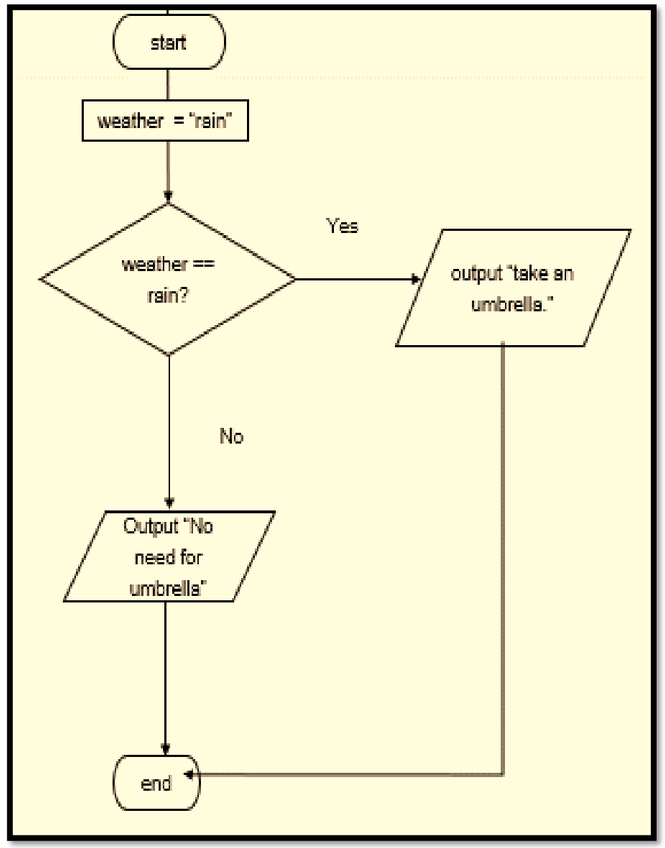





## OPERATORS

| Operator | Name | Example |
|---|---|---|
| + | Addition | x + y |
| - | Subtraction | x - y |
| * | Multiplication | x * y |
| / | Division | x / y |

## MAKING DECISIONS (CONDITIONAL STATEMENTS) – IF




```
if weather == "rain":
    print("Take an umbrella")
else:
    print("No need for the umbrella today.")
```

## KS3 FLOW CHART SYMBOLS

These are the symbols that you will need to be able to use in your flowcharts.

These are not the only symbols you will use, but we'll introduce the rest later.

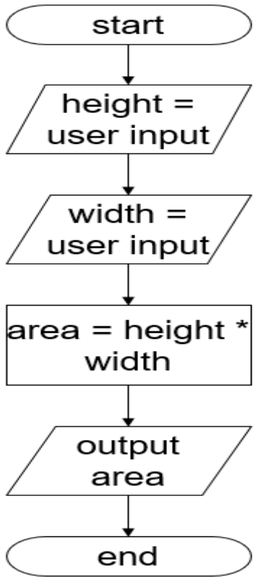

This is the terminator symbol. It goes at the beginning and the end of a flowchart.

This is the process symbol. It is for any action that needs to be carried out such as a calculation or variable declaration.

This is the data symbol. It is used any time you need to take an input or output a value.

The arrow is used to join symbols together. The arrow shows the flow of the algorithm.

## KS3 FLOW CHART EXAMPLE




This is the terminator symbol. It goes at the beginning and the end of a flowchart.

This is the data symbol. It is used any time you need to take an input or output a value.

This is the data symbol. It is used any time you need to take an input or output a value.

This is the process symbol. It is for any action that needs to be carried out such as a calculation or variable declaration.

This is the data symbol. It is used any time you need to take an input or output a value.

This is the terminator symbol. It goes at the beginning and the end of a flowchart.

## CONDITIONAL STATEMENTS

```
IF statements
IF statements can be used to select different
options in a program depending on a condition.
Also known as selection.
question = input("Are you revising?")
if question == "yes":
    print ("well done!")
elif question == "no":
    print("Oh dear!")
else:
    print("I don't understand")
```

WE CARE • WE CHALLENGE • WE COMMIT

PYTHON PROGRAMMING

# KNOWLEDGE ORGANISER

Fulwood academy

## MORE ABOUT VARIABLES

**Variables** can contain all kinds of information. Let's look at a few examples:

```
studentAge = 14
heightInM = 1.61
orderPaid = True
gender = 'F'
```

| Data type | Used for | DATA Example | Example use in code |
|---|---|---|---|
| integer | whole numbers | 42 | number = int( ) |
| Real or float | numbers with a decimal place | 3.14 | number = float( ) |
| Boolean | true or false only | False | decision = "y" |
| character | single characters | d | name = str( ) |

## INPUT FUNCTION

This is the variable where the data will be stored. Remember to make its name useful.

This is the input **prompt**. You should use it to give instructions to the user.

```
name = input("Enter your name: ")
```

The = is the **assignment** symbol. It means 'is given the value of'. You've used it before.

This is the function. Be careful with spelling and use lower case letters.

## RELATIONAL OPERATORS

Selection relies on **conditions**. A condition is a statement which compares one value with another. The result of the comparison can be true or false.

There are several conditions that you need to be able to use.

| | |
|---|---|
| == | Equal to |
| < | Less than |
| > | Greater than |
| != | Not equal to |
| <= | Less than or equal to |
| >= | Greater than or equal to |

**Note** the double equals symbol here. Single = is used to assign values to variables. Confusing these two symbols is a common bug.

## USER INPUTS

Let's have a look at how **input()** works using an example.

Here is a program we have used previously:

```
name = "Eric"
print(name)
```

To make it more useful, we want the user to type in their name.

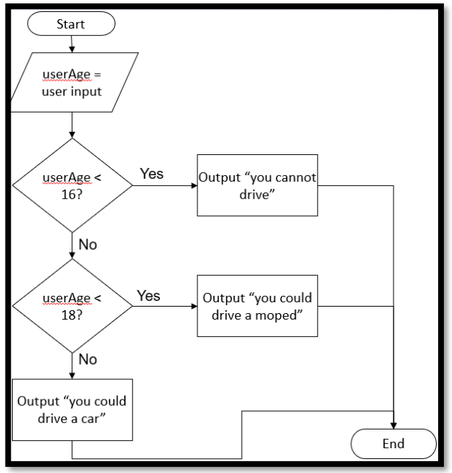
To do this, we use the input function like this:

```
name = input("Enter your name: ")
print(name)
```

## TYPE CONVERSION

| | |
|---|---|
| int() | Convert to integer |
| str() | Convert to string |
| float() | Convert to floating point number (real) |
| bool() | Convert to Boolean |

## MORE ABOUT CONDITIONAL STATEMENTS - IF AND ELIF




```
userAge = int(input("Enter your age: "))
if userAge < 16:
    print("You cannot drive")
elif userAge < 18:
    print("you could drive a moped")
else:
    print("You could drive a car")
```

WE CARE • WE CHALLENGE • WE COMMIT

# KNOWLEDGE ORGANISER

## ITERATION

say it!

### Iteration

**Iteration is the process of repeating steps.**

***Iteration allows algorithms to be simplified by stating that certain steps will repeat until told otherwise.***

*Are we nearly there yet?*
*Are we nearly there yet?*
*Are we nearly there yet?*
*Are we nearly there yet?*

## ITERATION – WHILE LOOP

First create a variable that will count how many times the code has looped.

The 'while' keyword together with a condition is used to control the loop.

This reads 'while the value in the variable count is less than 10…'

```
count = 0
while count < 10:
    print("Hello world")
    count = count + 1
```

Notice the colon – just as in 'if' statements.

These lines are indented. This means they happen inside the loop. They will be repeated over and over.

This line adds one to the counter variable, indicating that a pass through the loop has been completed.

## PYTHON TURTLE

| | |
|---|---|
| `import turtle` | Make all the turtle commands available to your program |
| `turtle.mode('logo')` | Set the mode |
| `turtle.speed(integer)` | Set the animation speed of the turtle. *1* = slowest, *10* = fastest. *0* turns off animation completely |
| `turtle.shape('turtle')` | Set the shape. You can also choose from: *arrow*, *square*, *circle*, *triangle* and *classic* |

| | |
|---|---|
| `turtle.forward(distance)` | Go forwards by amount *distance* |
| `turtle.backward(distance)` | Go backwards by amount *distance* |
| `turtle.right(angle)` | Turn right by *angle* degrees |
| `turtle.left(angle)` | Turn left by *angle* degrees |
| `turtle.home()` | Go home (0, 0) and face north |
| `turtle.goto(x, y)` | Go to position *x*, *y* |
| `turtle.setheading(degrees)` | Point in compass direction *degrees*. 0 is north, 90 is east, 180 is south, 270 is west |

| | |
|---|---|
| `turtle.begin_fill()` | Use this command before you start drawing the shape you want to be filled |
| `turtle.end_fill()` | Use this command when you have finished drawing the shape to be filled. |
| `turtle.pendown()` | Put the pen down to draw |
| `turtle.penup()` | 'Lift' the pen from the screen |
| `turtle.pensize(integer)` | Set the size of the pen to the given integer. |
| `turtle.pencolor(string)` | Set the pen colour to the string given. Note the American spelling. |
| `turtle.fillcolor(string)` | Set the fill colour to the string given. See list of colours below. |
| `turtle.color(string)` | Set **both** the fill colour and pen colour to given string. |
| `turtle.color(string1, string2)` | Set the pen and fill colour at the same time. *String1* should be the name of the pen colour, and *string2* is the fill colour. |

## ITERATION – FOR LOOPS

You need a program that can count from 1 to 100 and output that value.

Using a **for** loop it would look like this:

```
for num in range(1, 101):
    print(num)
```

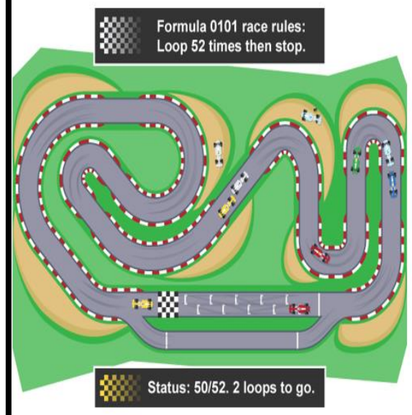
Start at 1 and go up to 100. Remember that the stop is the number after the one you really want.

You may see the letters 'i', 'j', or 'k' used in a for loop. Experienced programmers tend to use one-letter identifiers for variables in situations where the variable will not be used again.

However, you should use an identifier that has meaning in the context of your code, as this will make your code more readable and your logic clear.

## CONDITION AND COUNT CONTROLLED

### COUNT Controlled LOOP




### CONDITION Controlled LOOP