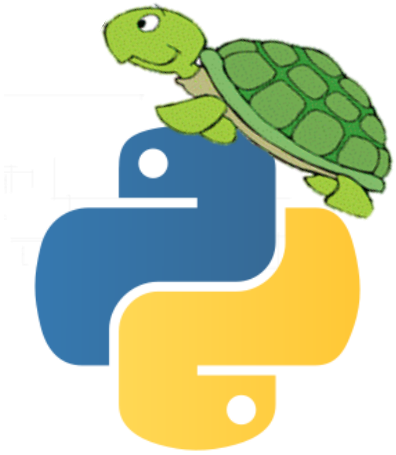
Python 3



KS3 Programming Workbook

"Taking you Parseltongue further"

Name_____

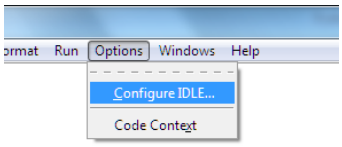
ICT Teacher _____Form_____

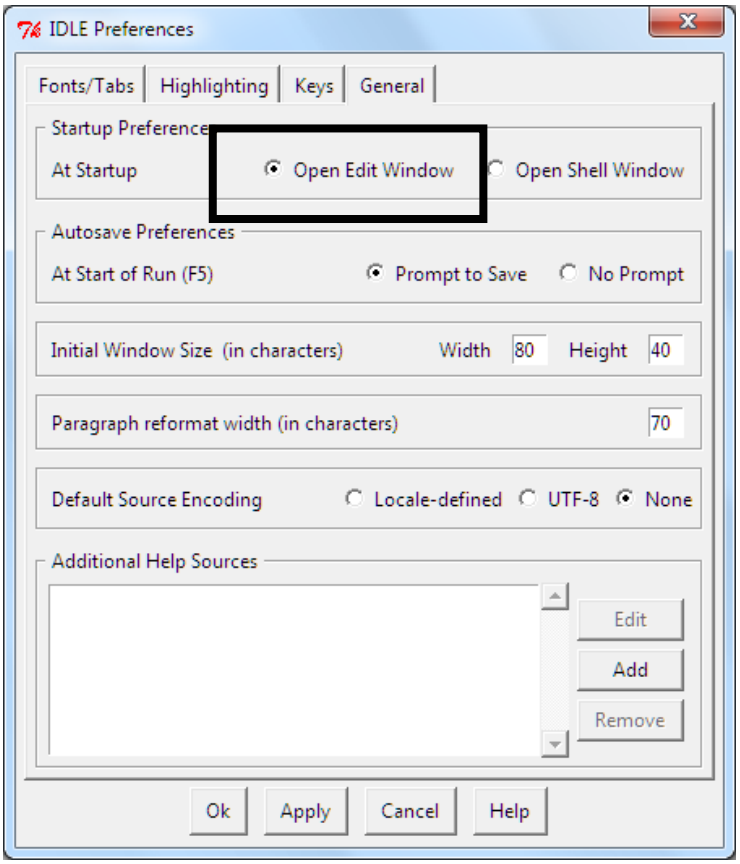
To Execute the program code press F5

Welcome to Python

The python software has two windows that we will use. The main window is called the Python Shell and allows you to directly write in program lines and then a return will execute them. This is good for testing one line of code.
The second window is the "Edit Window" which allows you to enter several lines of code, or a create a program, save it and then execute the code.

Open the python software





Python and the Raspberry Pi

To create a new python program on the Pi, you can use a text editor called "Joe's Text Editor"

Type: joe (the name of your program).py

To run or execute the program type: python (the name of your program).py

KEYWORDS to complete

Time.sleep		
\		

Making the Program Exe

In the exercises s in this workbook you may wish to run your code outside the Python shell. This is a simple process. Save your program with the file extension .py. When you run the file it will execute the program in a command box style window. Once the program has executed it will close the window. You may wish to build a delay feature into the program to allow time to view the output results of the program. Use the **import time;** function and the **time.sleep(5).**

- ❒ Create a simple program and save as an executable file, then run, Save as *simple*
- ❒ Create a program, add the time function to delay the closing of the window. Save as *simple2*

Introductions

This Python Module is called **Turtle.** Basically the snake becomes a turtle; you can name your turtle and then command it to draw shapes and structures as you desire. It's a little like a snake charmer, except that it's a Turtle not a Snake, more like a Turtle Charmer!

The first instruction imports the turtle function, use the simple code below,

import turtle

then create and name the turtle,

yourname = turtle.Turtle()

Next the turtle needs to appear in a window of its own using the command **wn**

wn = turtle.Screen()

Movements

The turtle has some basic moves, **forward, left, backward** and **right.** Command your turtle by using its name, in the direction you want to move it and then the number of spaces you want your turtle to move by. The **left** turns the turtle by the number of degrees you state,

START A NEW PROGRAM AND TYPE IN THIS CODE:

```
bob.forward(150)
bob.left(90)
bob.forward(75)
```

TRY THESE

- ❒ Create a simple program to draw a square, Save as *square*
- ❒ Create a simple program to draw a triangle Save as *triangle*
- ❒ Create a simple program to draw your own shape Save as *Own*

What does the command Backward(100) do?

What shape do you think this code will make? Try it, can you improve it?

```
import turtle
wn = turtle.Screen()
bob = turtle.Turtle()
bob.forward(150)
bob.left(145)
bob.forward(75)
bob.right(90)
bob.forward(75)
bob.left(125)
bob.forward(145)
bob.left(145)
bob.forward(75)
bob.right(90)
bob.forward(75)
```

When You Have Had Enough And You Want To Leave

An important feature of the program is to allow the user to close the window once the program has executed. There are two simple commands to do this,

exit()	closes the window after the program has run
wn.exitonclick()	closes the window when the user clicks on the window

exit will close the window straight away so you may not see the results of the program output.

Fun with Colours and Pen Size

It is possible to 'pimp' your turtle, well not quite but you can add colour features to the program. The first example allows you to change the colour of your turtle. To do this you name your turtle and then set the turtle colour using the command ***nameofyourturtle*.color("green")**

For example if your turtle is called Bob,

```
Bob = turtle.Turtle()
Bob.color("green")
```

Note that the spelling of colour is the American spelling, color

- ❒ Create a simple program to draw a pink rectangle Save as *rectangle*

Once you have created your shape or polygon you may wish to fill the shape with a colour

To get the program to fill a colour you need to define the colour and tell the program when to begin the fill and when to end the fill. This is achieved with the two commands, **begin_fill()** and **end_fill()**.

Note the use of the **_** underscore and also the **()** brackets.

Using the command, **bob.color("black", "red")** you define the colour of the line and the fill colour of the shape. Then tell bob, our turtle to begin the fill with the command, **bob.begin_fill().** Then create the instructions for the shape and then tell the program to end the fill, **bob.end_fill().**

TRY THE PROGRAM BELOW.

```
import turtle
wn = turtle.Screen()
bob = turtle.Turtle()

bob.color("black", "red")
bob.begin_fill()
bob.circle(80)
bob.end_fill()

wn.exitonclick()
```

TRY THESE

- ❒ Create a simple program to draw a square, fill it green and with a red line, Save as *redsquare*
- ❒ Create the same program using repeat functions, Save as *resquare*
- ❒ Create a simple program to draw your own shape Save as *Own*
- ❒ Create your own shape, with a fill colour of your choice Save as *Ownshape*

FINAL COLOUR CALL

The final colour change you can make is to the background colour of the window. This uses the simple command **bgcolor ("enter the color that you want)** for example **wn.bgcolor("red")**

- ❒ Add the code to one of your programs, **wn.bgcolor("red")** Save as *window*

What difference does adding the code at the beginning or the end of the program make to the colour of the window?

PEN WIDTH

Next you may wish to increase the width of the pen. Call the turtle and use the command .pensize(*enter the size of the pen width*). For example **Bob.pensize(5)**

- ❒ Create a simple program to draw a red rectangle, width 5 Save as *redrectangles*

PEN UP AND PEN DOWN

Sometimes you may be required to move the turtle without it drawing onto the canvas, this is a simple as lifting the pen off the page, use the command ***nameofyourturtle*.penup()** and then move the turtle forward and the place the pen down again using the command ***nameofyourturtle*.pendown()**

- ❒ Create a simple program to draw a square with two sides missing Save as S*quare2*
- ❒ Create a program to draw your name in orange, green background, width 6 *rectangle* Save as *MyName*

Going Loopy for a WHILE

Remember the **looping** technique from book one? We used the **while** function (Book 1 p.7, while loops). Looping allows you to repeat the same code, say four times without having to type in the same code four times. You first create a variable with the value of four, we will call the variable ***turns*** and define its value by typing ***turns = 4***

To draw a line you use the code ***nameoftheturtle.forward(100)***

If you wanted to draw a line of 400 you could retype this code another three times. (A waste of time!)
However, if you define the number of turns using the variable above and then use the **while** command to state what you want the command to do, the code looks like this.

Can you right this program? (Don't forget the indents?)

Did you get this answer?

```
import turtle
wn = turtle.Screen()
dan = turtle.Turtle()
turns = 4
while turns > 0:
   dan.forward(100)
   turns = turns -1
wn.exitonclick()
```

- ❒ Create a program to draw a square using **looping** and a **while** statement, and the forward and right command **only once**? Save your program as *Loop Square*

- ❒ Create a program to draw a Hexagon using **looping** and a **while** statement, and the forward and right command **only once**? Save your program as? *Loop Hexogan*

Write your answer here.

GOING LOOPY FOR A WHILE

As an alternative to the **while** function you can use the **for** function. (book one? (Book 1 p.8, For Loops). The **for** function requires a variable and a list. If you want to repeat the instruction four times first create a list of 4, remember that 0 is included as a value. Create a variable call I and the list values of 0,1,2,3, **for i in [0,1,2,3]:**

The **for** command tells the program to carry out the instructions for all the number in the list, in the example above this would be four times, once for 0 a second time for 1, a third time for 2 and a final forth time for 3.

TRY THIS CODE

```
import turtle
wn = turtle.Screen()
bob = turtle.Turtle()
for i in [0,1,2,3]:
   bob.forward(50)
   bob.left(90)
wn.exitonclick()
```

- ❒ Can you create a program to draw a rectangle using **looping** and a **for** statement and the forward and right command only once? Save your program as? *Loop Rectangle*

What angle is required to create an Octagon,

- ❒ Can you create a program to draw an Octagon using **looping** and a **for** statement, and the forward and left command only once? Save your program as? *Loop Octagon*

COMBINING LOOPS AND COLOURS FOR AND COLOUR

It is also possible to use looping and the **while** and **for** statements with colour. We set up a list of colours, which include yellow, red, purple, blue and we call this variable colors. Because there are four colours the instructions will repeat four times. This will draw a square; each side will be a color from the list.

TRY THIS CODE

```
import turtle
wn = turtle.Screen()
bob= turtle.Turtle()

for Colors in ["yellow", "red", "purple", "blue"]:
  terry.color(Colors)
  terry.forward(50)
  terry.left(90)

wn.exitonclick()
```

- ❒ Can you create a program to draw a triangle with three different colours? *Save as ColorTri*
- ❒ Create your own shape and colours, write your code below.

More Than One Turtle

Now you have had time to practice and master the basics we can introduce another Turtle into the screen, use the same previous commands to create another turtle and call the new turtle with a different name for example,

```
import turtle
Bob= turtle.Turtle()
Bob.color("yellow")
tess.pensize(5)
Dave = turtle.Turtle()
```

This creates two new Turtles called Bob and Dave

- ❒ Create a program where **Dave** draws a **Triangle** and **Bob** draws a **Square** Save as *Justthetwoof us*
- ❒ Create the same program using looping and Save as *Slender Code*

Change the Shape of your Turtle

The Turtle is a versatile creature and can be magically transformed into other shapes. The following shapes are available, 'arrow', 'classic', 'turtle', or 'circle' and more!

To change the shape of the Turtle you call the turtle with **.shape** and the shape that your desire in brackets. For example if you want an arrow type, ***thenameofyourturtle*.shape("arrow")**

- ❒ Create a program that draws any shape using the "circle" Save as *Circle*

What does the "Turtle" shape do?

Write the code below to make the Turtle bigger.

BONUS CAN YOU CREATE CHRISTMAS TREE?

Summary of Turtle Methods

Method	Parameters	Description
Turtle	None	Creates and returns a new turtle object
forward	distance	Moves the turtle forward
backward	distance	Moves the turle backward
right	angle	Turns the turtle clockwise
left	angle	Turns the turtle counter clockwise
up	None	Picks up the turtles tail
down	None	Puts down the turtles tail
color	color name	Changes the color of the turtle's tail
fillcolor	color name	Changes the color of the turtle will use to fill a polygon
heading	None	Returns the current heading
position	None	Returns the current position
goto	x,y	Move the turtle to position x,y
begin_fill	None	Remember the starting point for a filled polygon
end_fill	None	Close the polygon and fill with the current fill color
dot	None	Leave a dot at the current position
stamp	None	Leaves an impression of a turtle shape at the current location
shape	shapename	Should be 'arrow', 'classic', 'turtle', or 'circle'

What have you enjoyed so far?

What have you found difficult?

What would help?

What do you still need to improve?

What will you do next?