Python & Java {4} Teachers

Making a Monster

Level 2 - Python

Funded by the Erasmus+ Programme of the European Union

Python & Java 4 Teachers

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

Introduction

You probably think that art and programming couldn’t be more unalike, but actually they might be more similar than you think!

There are many modern artists whose main medium is code. For example: Mark Dorf, Josh Davis and Kyle McDonald.

Here are a few websites that merge together are and programming:

- Silk – Interactive Generative Art (weavesilk.com)
- Dream by WOMBO

Task

- For this project you are going to use your own knowledge of the shapes to create your very own monster face!

Extension:

- Try using different shapes and colour's to create a completely different monster face!
- For example: one giant diamond eye, three square noses or even a triangular head, it’s all up to you.

Funded by the Erasmus+ Programme of the European Union

Python & Java 4 Teachers

Process

This program should:

✓Import the python turtle graphics library,

✓Use for loops and subroutines to create regular shapes to create the face,

✓Apply previous knowledge of angles to navigate the turtle around the screen.

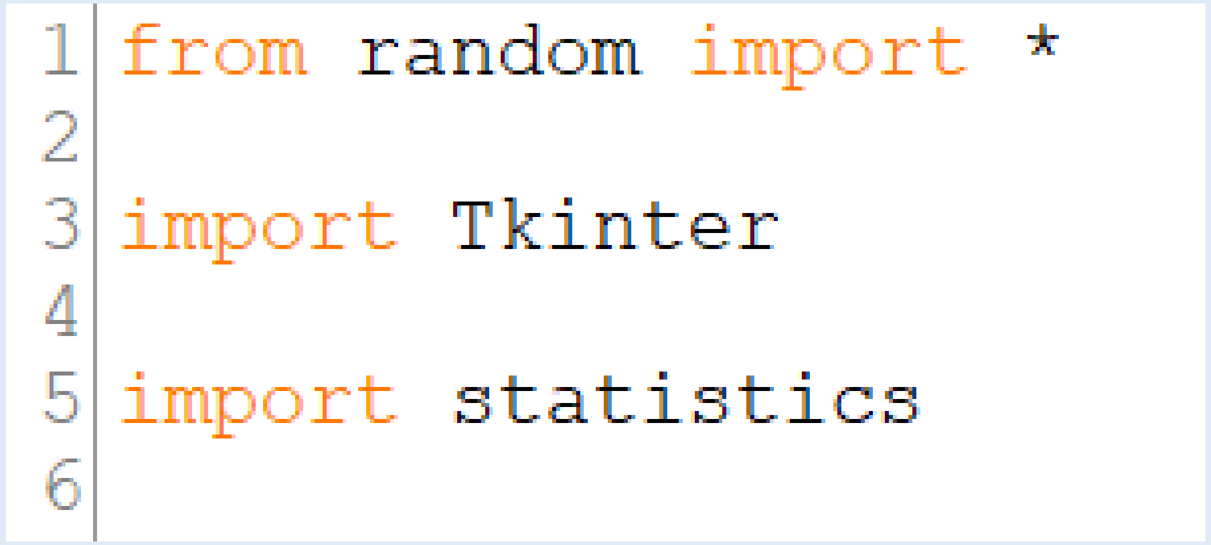
Funded by the Erasmus+ Programme of the European Union

Python & Java 4 Teachers

Python Libraries

Funded by the Erasmus+ Programme of the European Union

Python Libraries are a set of useful functions that eliminate the need for writing codes from scratch.



```
from random import *

import Tkinter

import statistics

```

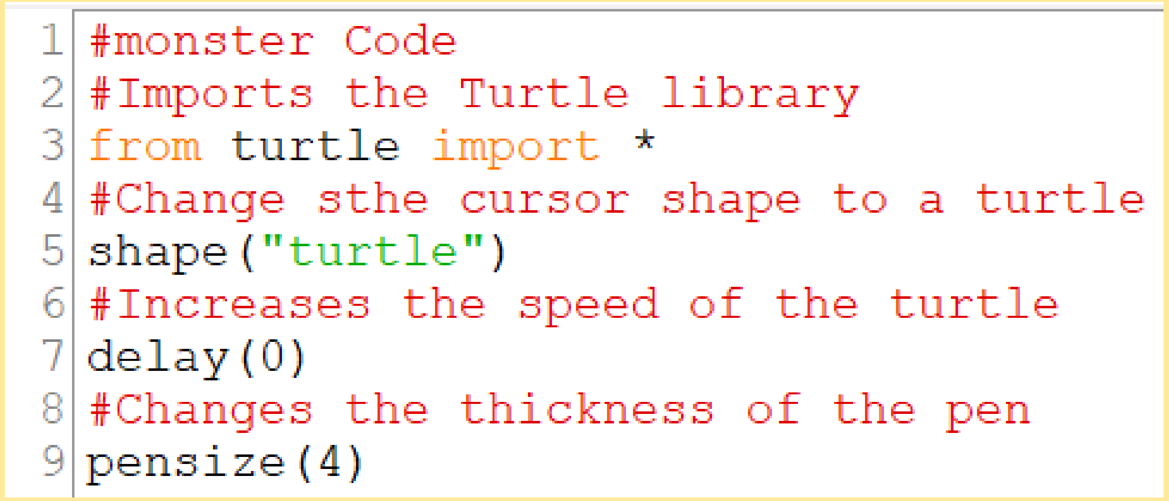
They can be brought into the program using the ‘import’ keyword and can save valuable time when writing complex programs.

One common example of a python library is the ‘random’ module, used often for generating pseudo-random numbers

Step 1

Importing the turtle library

The turtle graphics library has been imported into the program which means we can now draw in python.



```
#monster Code
#Imports the Turtle library
from turtle import *
#Change sthe cursor shape to a turtle
shape("turtle")
#Increases the speed of the turtle
delay(0)
#Changes the thickness of the pen
pensize(4)
```

The cursor shape has been changed to look like a turtle, and the speed of the turtle has ben increased using ‘delay()’. The line thickness of the pen has also been increased to 4 so it becomes more visible when the program is run.

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

Simple Turtle Commands

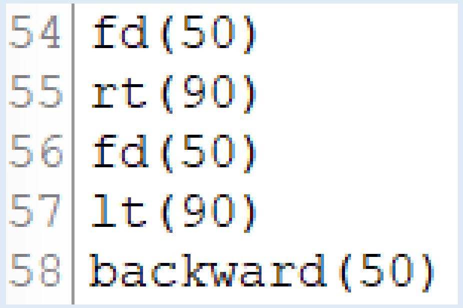
Funded by the Erasmus+ Programme of the European Union

The turtle understands simple commands such as left turns, right turns, move forwards and move backwards.

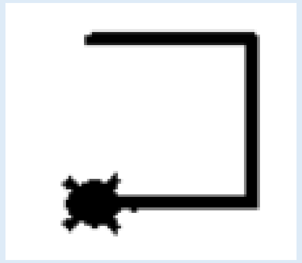
Simple Turtle commands:

- fd(10) – moves the turtle forward 10 pixels,
- backward(5) – moves the turtle backward by 5 pixels,
- right(35) – moves the turtle clockwise by an angle of 35 degrees,
- left(55) – moves the turtle counter-clockwise by an angle of 55 degrees;
- goto(x, y) – moves the turtle to the position x, y.

Another command turtle uses is pu() and pd(), which stands for pen up and pen down. This means that the turtle can move without leaving a linen behind it.



```
fd(50)
rt(90)
fd(50)
lt(90)
backward(50)
```



Funded by the Erasmus+ Programme of the European Union

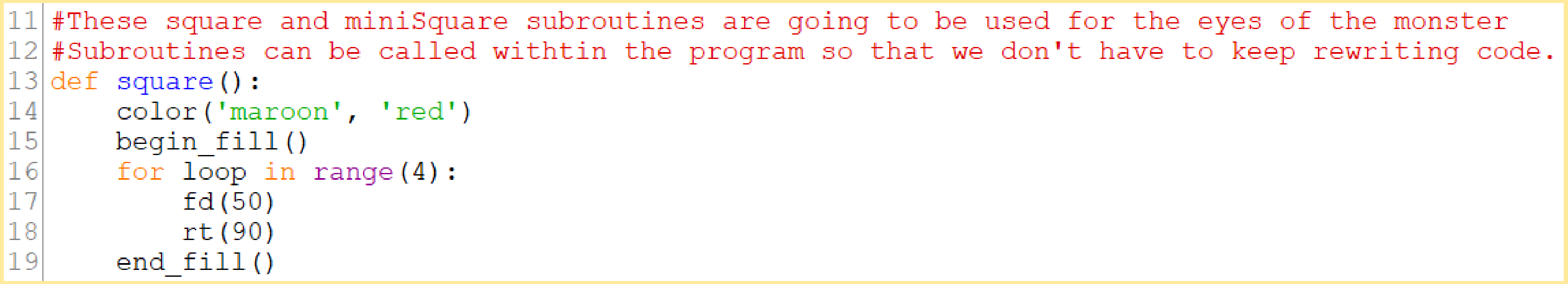
{4}

Python & Java 4 Teachers

Step 2

Defining the subroutine

From line 13 the first subroutine in the program has been defined using the def() function and the subroutine name square().



```
11 #These square and miniSquare subroutines are going to be used for the eyes of the monster
12 #Subroutines can be called withtin the program so that we don't have to keep rewriting code.
13 def square():
14     color('maroon', 'red')
15     begin_fill()
16     for loop in range(4):
17         fd(50)
18         rt(90)
19     end_fill()
```

The colour is set to fill in red with a maroon outline. As a square is a regular shape a for loop is coded in to reduce unnecessary code within the subroutine.

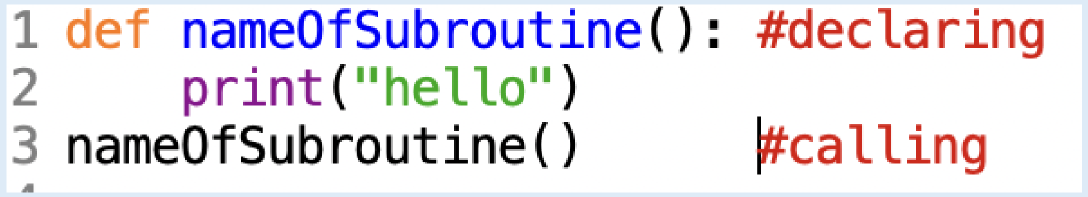
Funded by the Erasmus+ Programme of the European Union

Subroutines

Subroutines are sequences of instructions that perform a specific task.

- It may be easier to think of them as mini-programs within a large program.
- Subroutines consist of modules of code that perform different tasks.
- If these tasks are repeated throughout the program, they can be written as subroutines.
- Each subroutine is given a unique name so that it can be called and executed quickly throughout the program, without having to write the code again.
- This reduces the size of the code, making the program more logical and easier to read and maintain.

```
#Defining a subroutine:
def nameOfSubroutine():
       code goes here
#Calling a subroutine:
nameOfSubroutine()
```



```
def nameOfSubroutine(): #declaring
    print("hello")
nameOfSubroutine()      #calling
```

Loops

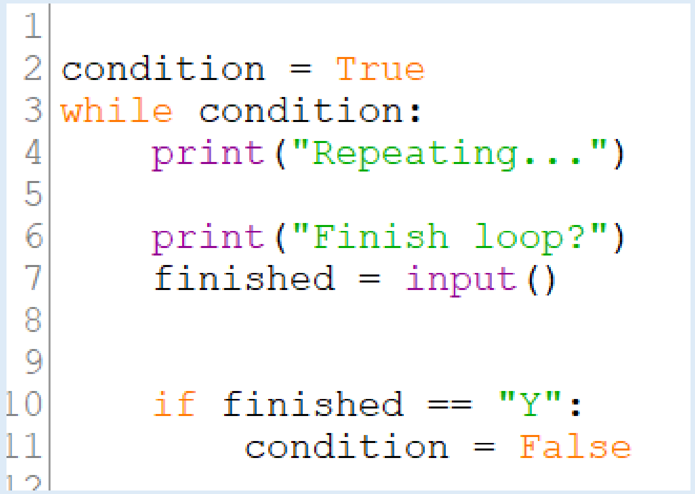
Funded by the
Erasmus+ Programme
of the European Union

A loop is a sequence of instructions that is continually repeated until a certain condition is reached.

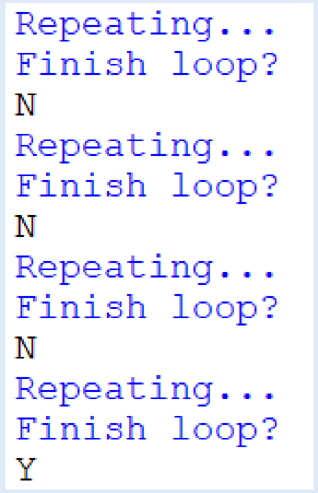
In Python there are two main loops: ‘FOR Loops’ and ‘WHILE Loops’

While Loops are condition controlled and will repeat until their condition is false.

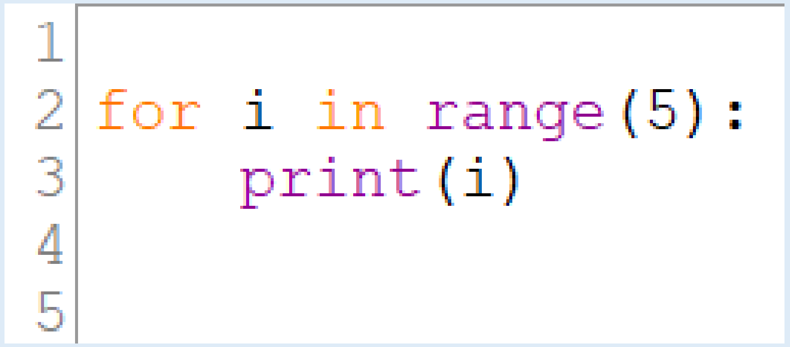
For loops are count controlled and will repeat a set number of times.



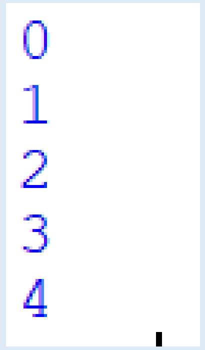
```python

condition = True
while condition:
    print("Repeating...")

    print("Finish loop?")
    finished = input()


    if finished == "Y":
        condition = False
```



```
Repeating...
Finish loop?
N
Repeating...
Finish loop?
N
Repeating...
Finish loop?
N
Repeating...
Finish loop?
Y
```



```python

for i in range(5):
    print(i)
```

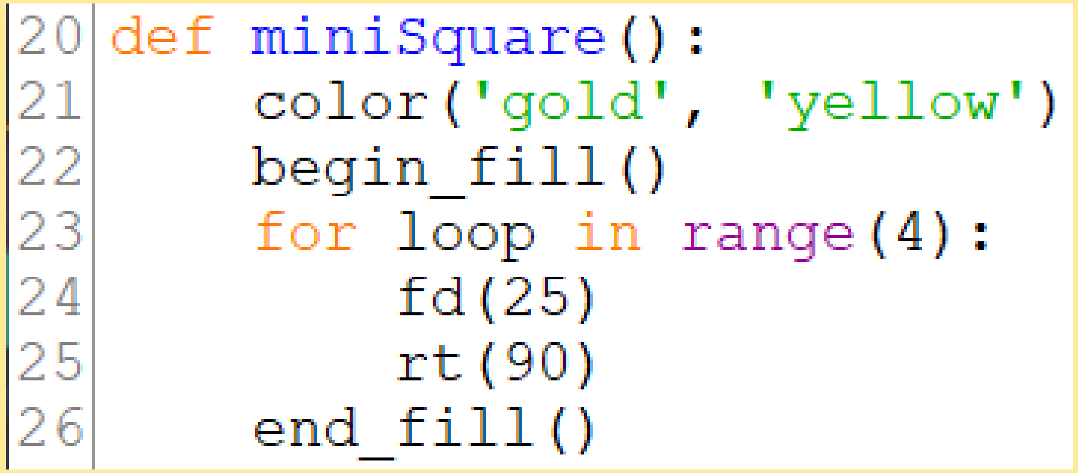


```
0
1
2
3
4
```

Step 3

Defining the second subroutine

The next subroutine is coded to create the pupils for the square eyes of the monster face. The code is again defined using the def() function and the subroutine title.



```
def miniSquare():
    color('gold', 'yellow')
    begin_fill()
    for loop in range(4):
        fd(25)
        rt(90)
    end_fill()
```

This is the same shape as the other subroutine, however this square is yellow with a gold outline and the dimensions for the shape itself are a lot smaller.

Funded by the Erasmus+ Programme of the European Union

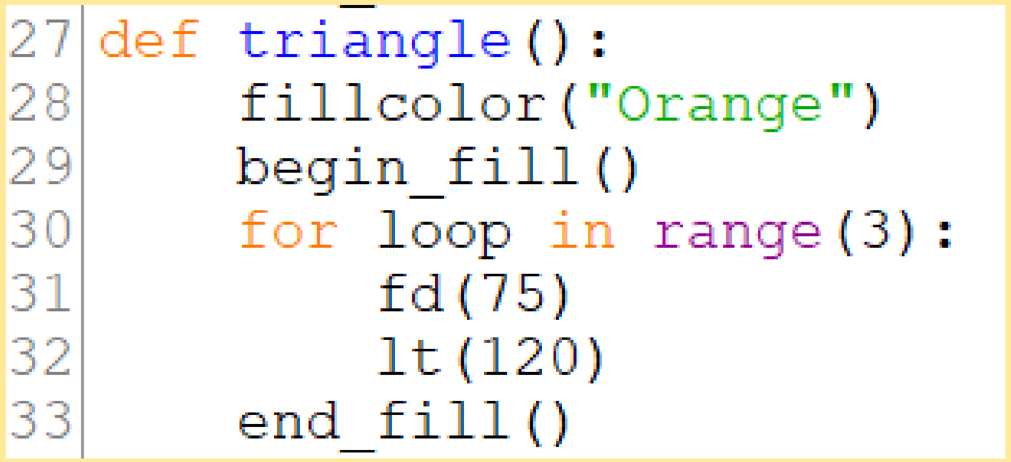
{4}

Python & Java 4 Teachers

Step 4

Defining the third subroutine

This is the last subroutine that's needs to be defined for this project. The other two subroutines were to create the eyes of the monster, this subroutine is to create the nose.



```
def triangle():
    fillcolor("Orange")
    begin_fill()
    for loop in range(3):
        fd(75)
        lt(120)
    end_fill()
```

The fill-colour for the nose is orange, and due to the nose being a triangular shape the angle inside the for loop() has also changed.

Funded by the Erasmus+ Programme of the European Union

Python & Java 4 Teachers

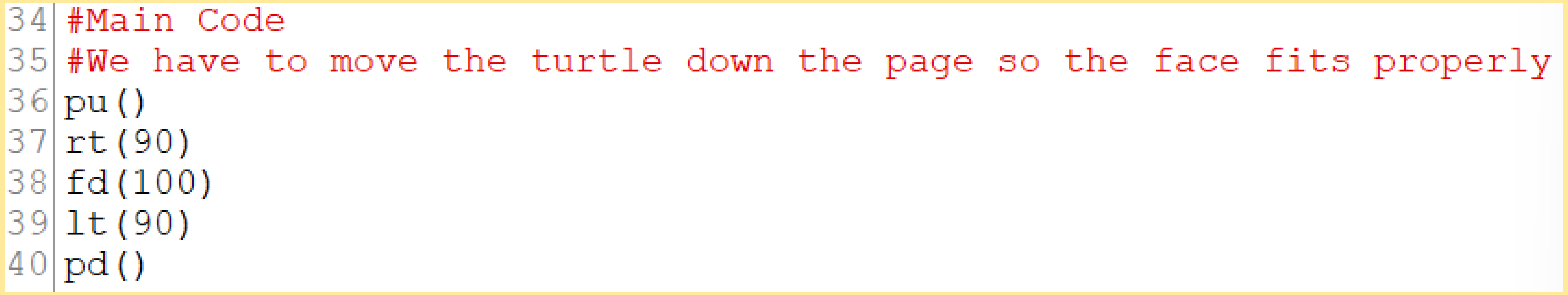
Funded by the Erasmus+ Programme of the European Union

Python & Java 4 Teachers

Step 5

Placement of the turtle

Now that all of the subroutines have been created we can focus on the placement of the turtle before any other shapes need to be drawn.



```
#Main Code
#We have to move the turtle down the page so the face fits properly
pu()
rt(90)
fd(100)
lt(90)
pd()
```

Here the pen up() function is used to make sure we don’t draw any excessive lines when giving the turtle its initial placement. The turtle needs to be moved down so that when the face is drawn it doesn’t cut off the ends of the screen.

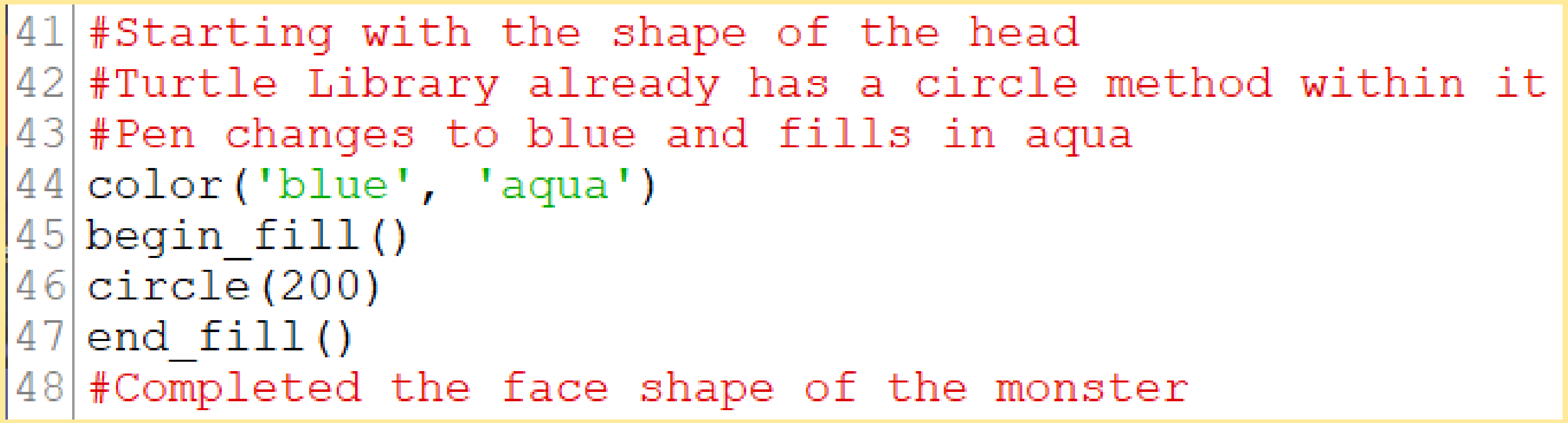
Funded by the
Erasmus+ Programme
of the European Union

Python & Java 4 Teachers

Step 6

Creating the face

The placement of the turtle is now set so we can begin actually creating the face of our monster. Turtle library comes with a built in circle function that we can use, and we can change the size of the circle by changing the number in between the brackets.



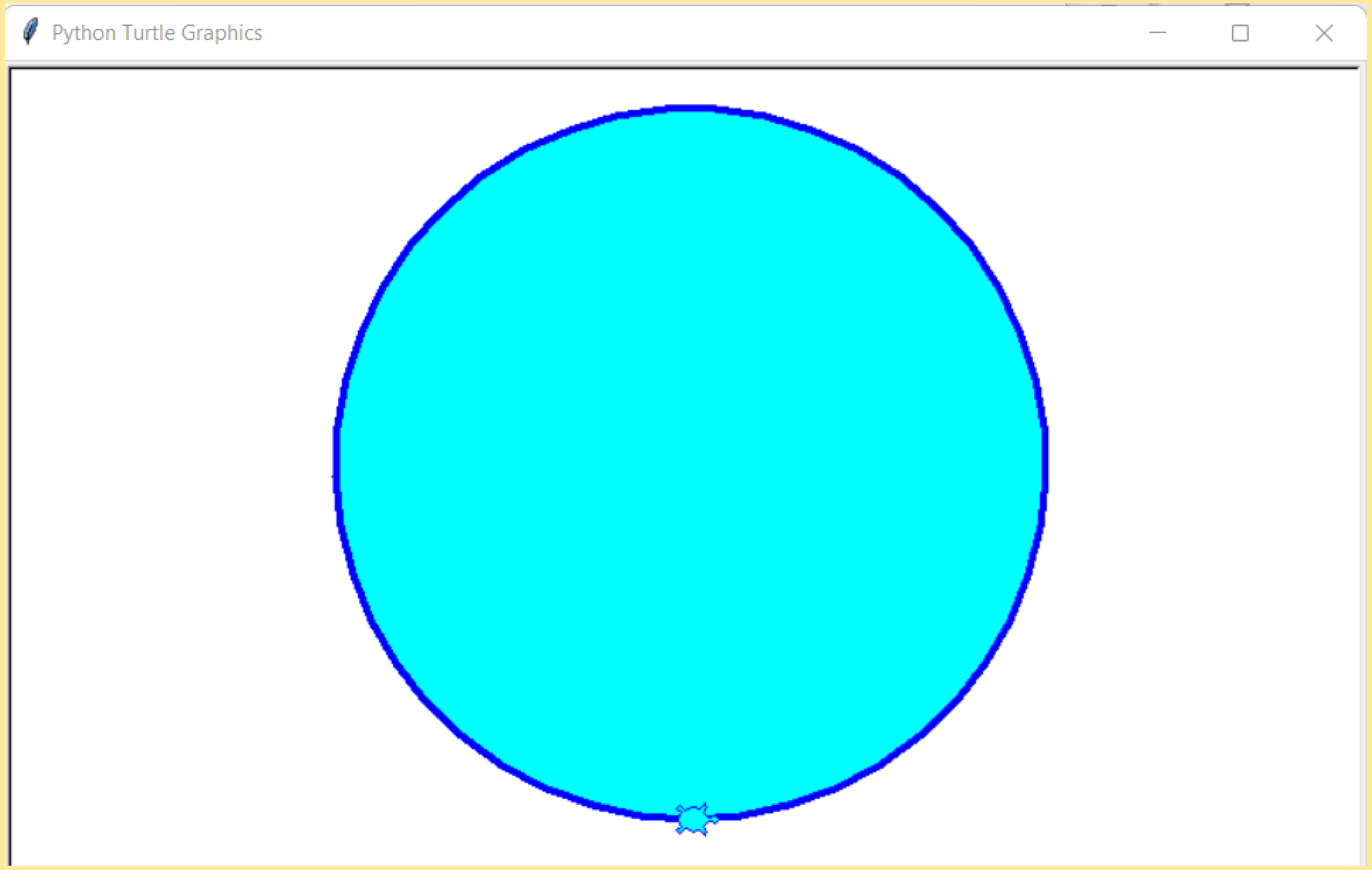
```
#Starting with the shape of the head
#Turtle Library already has a circle method within it
#Pen changes to blue and fills in aqua
color('blue', 'aqua')
begin_fill()
circle(200)
end_fill()
#Completed the face shape of the monster
```

The circle has been filled in with the colour aqua and the pen line blue.

What the code looks like…

Funded by the Erasmus+ Programme of the European Union

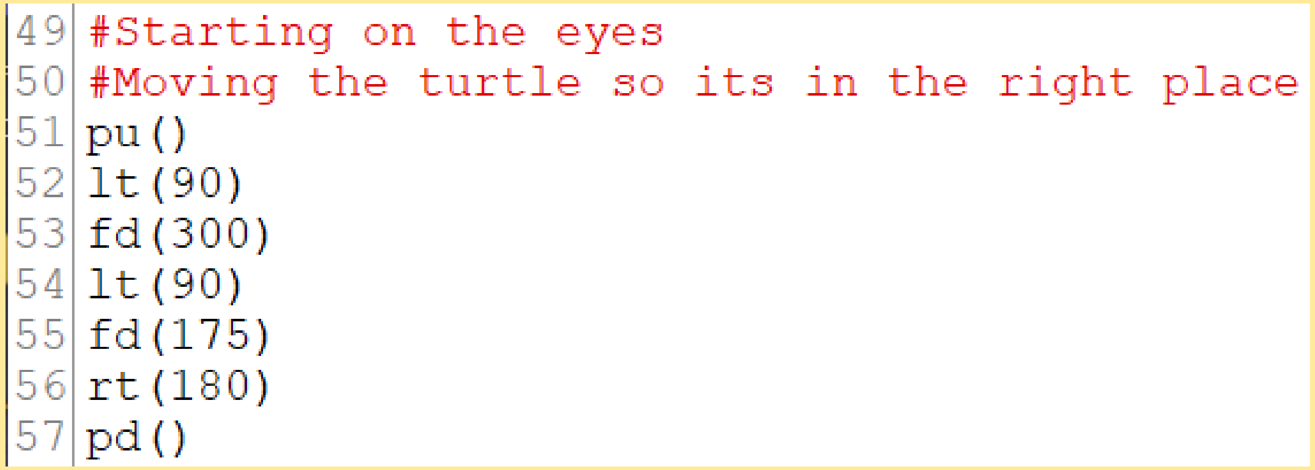
Python & Java 4 Teachers



Step 7

Creating the eyes pt1

The turtle finished the shape of the face near the bottom of the screen. In order to draw the eyes in the correct place the turtle needs to be moved again.



```
#Starting on the eyes
#Moving the turtle so its in the right place
pu()
lt(90)
fd(300)
lt(90)
fd(175)
rt(180)
pd()
```

This code moves the turtle up and to the left, so that when we call the subroutine for the eyes they sit in the right place.

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

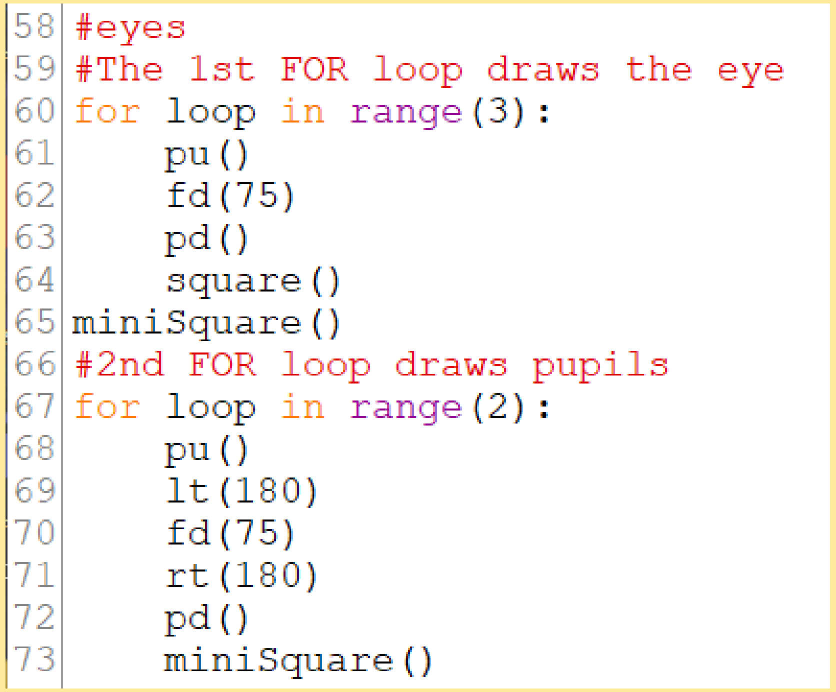
Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

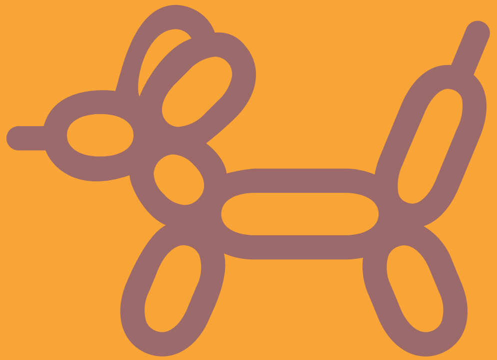
Step 8

Creating the eyes pt2



```
#eyes
#The 1st FOR loop draws the eye
for loop in range(3):
    pu()
    fd(75)
    pd()
    square()
miniSquare()
#2nd FOR loop draws pupils
for loop in range(2):
    pu()
    lt(180)
    fd(75)
    rt(180)
    pd()
    miniSquare()
```

The first for loop() moves the turtle forwards and draws the larger red square for the eye, and repeats 3 times. The yellow pupil is then drawn inside the red eye and the second for loop() moves the turtle back and draws two more pupils for the eyes.

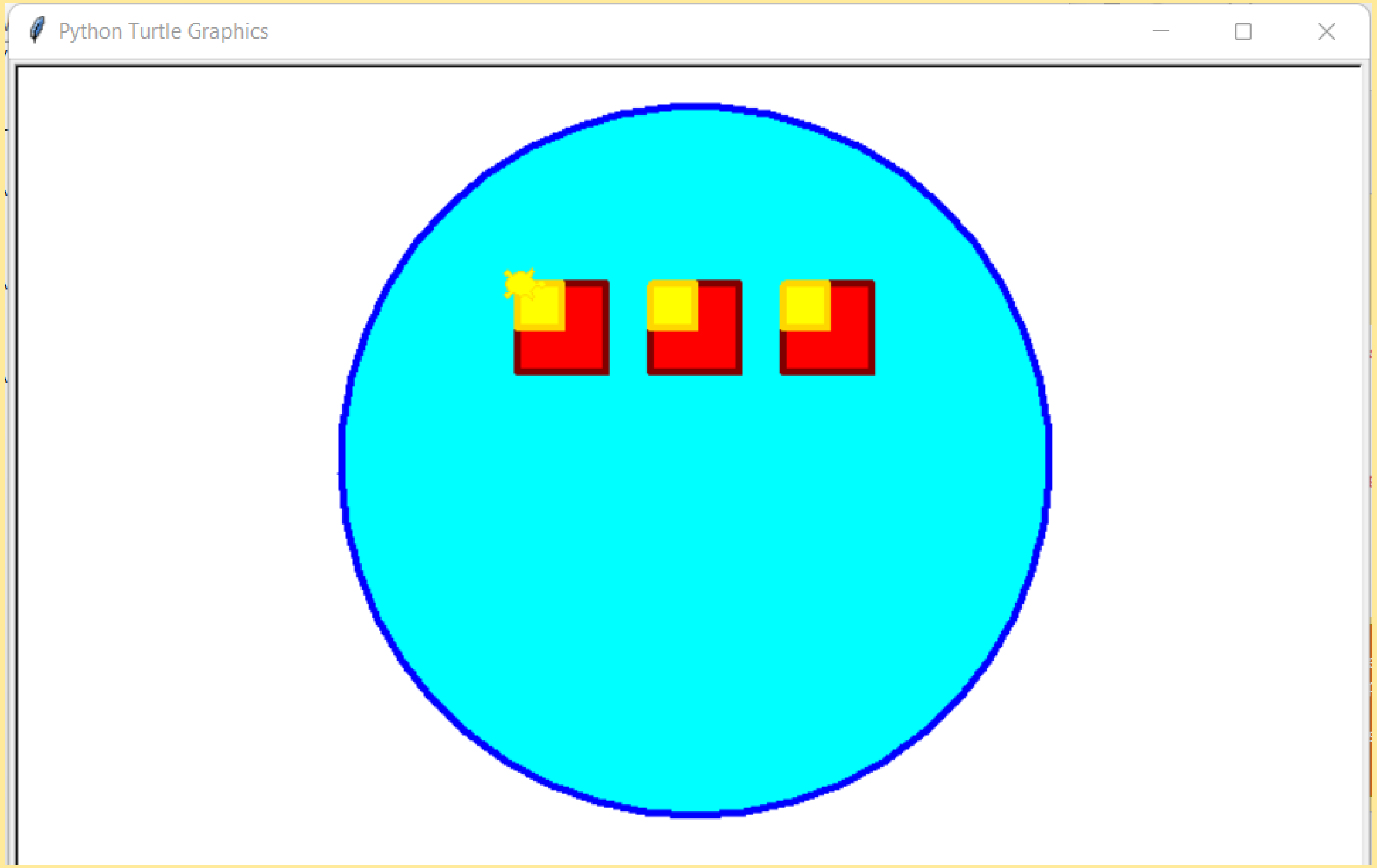


What the code looks like…

Funded by the Erasmus+ Programme of the European Union

{4}

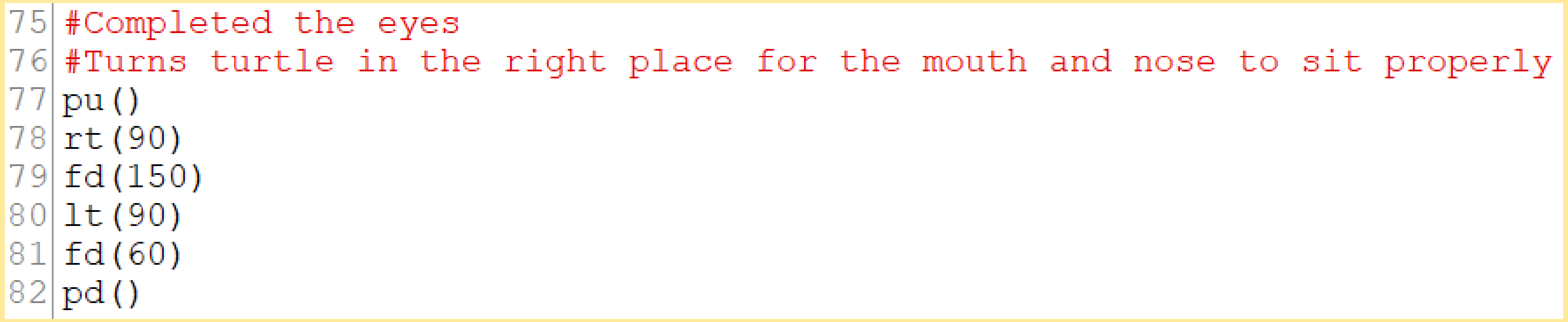
Python & Java 4 Teachers



Step 9

Moving the turtle

The eyes are now complete. The turtle needs the pen up() function again so that when we move the turtle no excessive lines show up. The turtle moves towards the middle of the face ready to draw the nose.



```
#Completed the eyes
#Turns turtle in the right place for the mouth and nose to sit properly
pu()
rt(90)
fd(150)
lt(90)
fd(60)
pd()
```

Funded by the Erasmus+ Programme of the European Union

Funded by the Erasmus+ Programme of the European Union

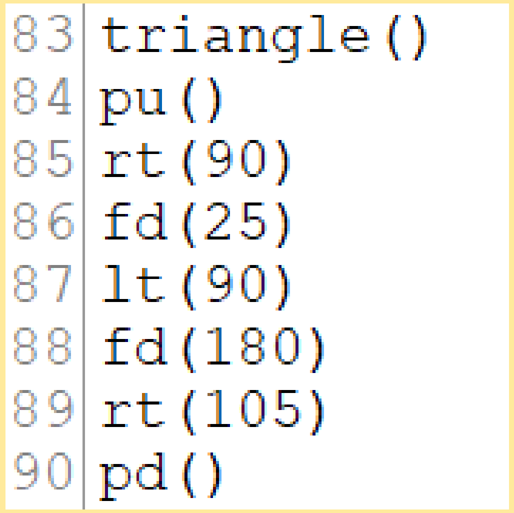
{4}

Python & Java 4 Teachers

Step 10

Drawing the nose

The triangle() subroutine is called from earlier on in the program – line 27. This draws the triangle as the nose for the monster.



```
triangle()
pu()
rt(90)
fd(25)
lt(90)
fd(180)
rt(105)
pd()
```

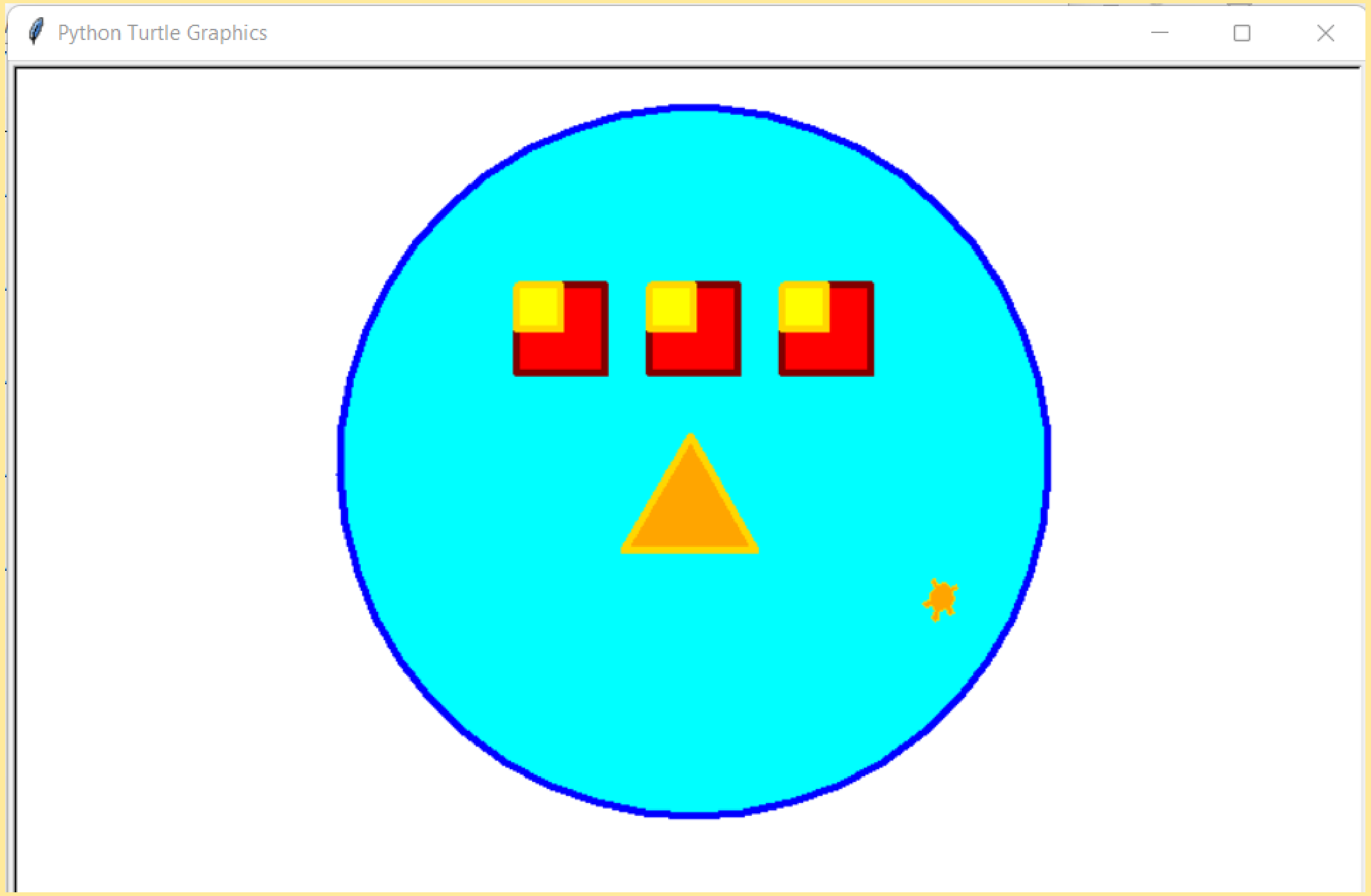
The following code allows for the turtle to move further down ready to draw in the monster's mouth and thus, complete the face.

What the code looks like…

Funded by the Erasmus+ Programme of the European Union

{4}

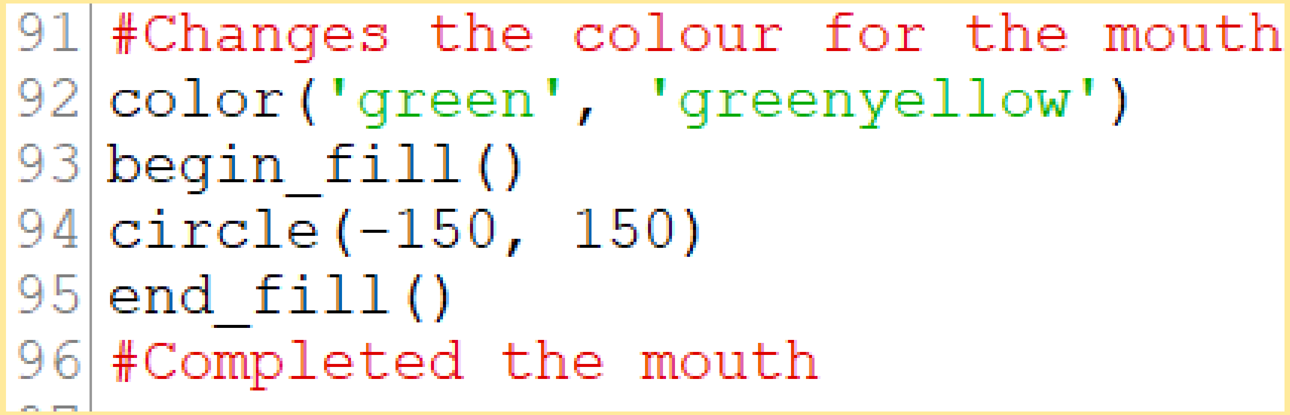
Python & Java 4 Teachers



Step 11

Completing the mouth

The colour of the pen is changed to ‘green’ with the colour fill becoming ‘greenyellow’.



```
91 #Changes the colour for the mouth
92 color('green', 'greenyellow')
93 begin_fill()
94 circle(-150, 150)
95 end_fill()
96 #Completed the mouth
```

Line 94 creates a semi circle which becomes the smile for our monster.

Funded by the Erasmus+ Programme of the European Union

{4}

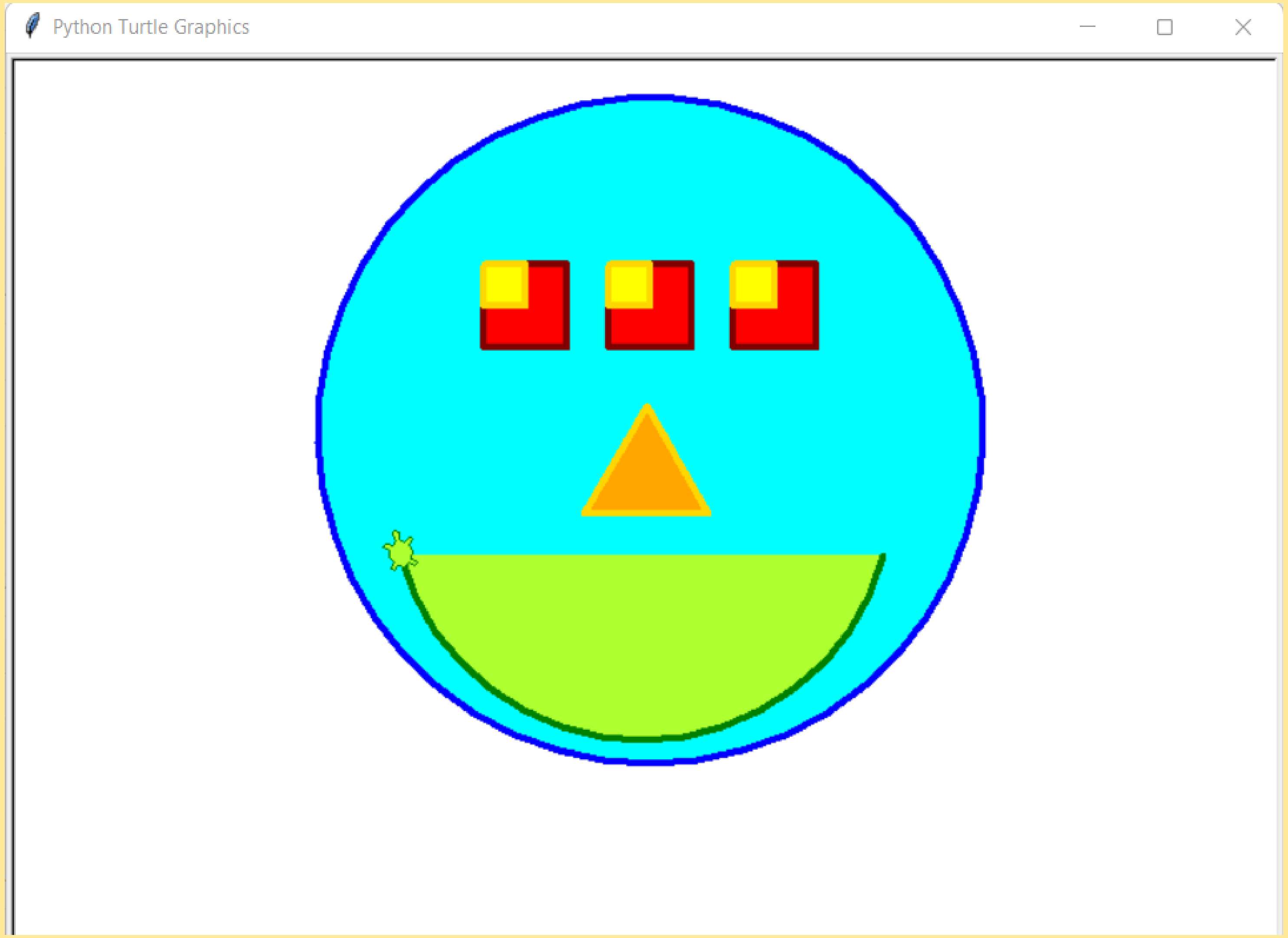
Python & Java 4 Teachers

What the code looks like…

Funded by the
Erasmus+ Programme
of the European Union

{4}

Python & Java 4 Teachers



Conclusion

This program should:

- ✓You should confidently be able to import the turtle library,
- ✓You should be confident in using subroutines to create regular shapes,
- ✓You should be comfortable using angles to move the turtle,
- ✓You should be able to manipulate the speed and colour within the turtle graphics program.

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

Congratulations!

You have created your own monster face

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers