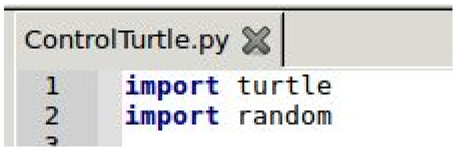
Python - Turtle Power

In this project we will be learning how to respond to key presses when using the Turtle module in Python.

Create Turtle

We must import the turtle library, we will also be using the random library so we can generate random movement of non player controlled turtles.

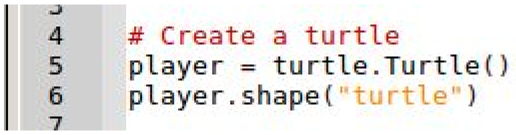
Add the following imports to a new python file.



ControlTurtle.py

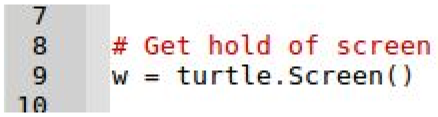
```python
import turtle
import random
```

We will be creating an instance of the turtle to be controlled by the player. For this create a variable named **player** and assign a new Turtle object to it.



```python
# Create a turtle
player = turtle.Turtle()
player.shape("turtle")
```

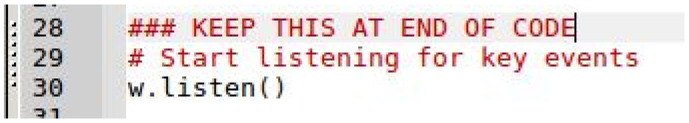
We also need access to the screen so we can set-up key listening events and the game loop. Add the following code to get the Screen in a variable called **w**.



```python
# Get hold of screen
w = turtle.Screen()
```

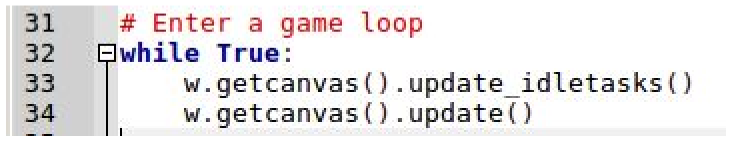
Set-up Listening and Game Loop

We will now write the last section of our code, first we will turn on key listening.



```python
### KEEP THIS AT END OF CODE
# Start listening for key events
w.listen()
```

We will now write our Game Loop. This will go round forever (or at least until the user quits) and handle any tasks we may wish to run in addition to the key bindings

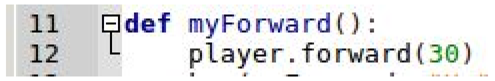


```python
# Enter a game loop
while True:
    w.getcanvas().update_idletasks()
    w.getcanvas().update()
```

This code must stay *AT THE END* of your python file. We are about to start putting in event handlers, these will go between the setup and listening code. Watch out for that!

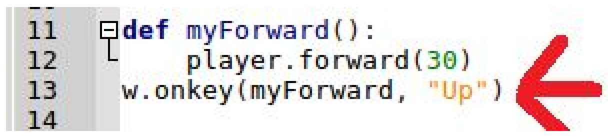
Up Arrow Goes Forward

We will be using the **onkey** function on the Screen to listen for a keypress. We must first write a function to call when the **Up Arrow** is pressed. Pay attention to *indentation*!



```python
def myForward():
    player.forward(30)
```

Now we can tell the screen object about our key handler using the code in line 13 shown:

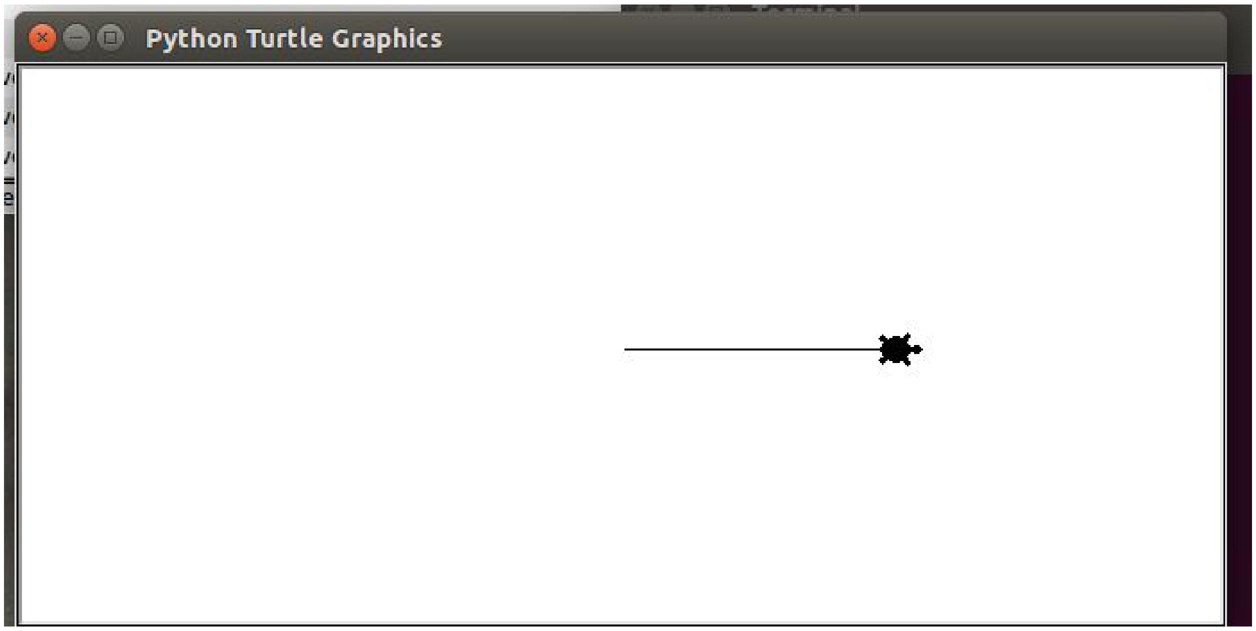


```python
11  def myForward():
12      player.forward(30)
13  w.onkey(myForward, "Up")
14
```

The “Up” refers to the up arrow, a complete list of available key presses can be found here https://www.tcl.tk/man/tcl8.4/TkCmd/keysyms.htm

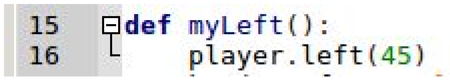
Test

Press F5 (assume using Geany) to play your new game, you should be able to go forward by repeatedly pressing the Up arrow.



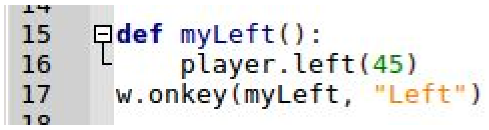
Handle Turning

Now that we know how to respond to key presses, write a function to turn left as shown:



```
15  def myLeft():
16      player.left(45)
```

Attach a key handler to this by adding line 17 shown



```
15  def myLeft():
16      player.left(45)
17  w.onkey(myLeft, "Left")
```

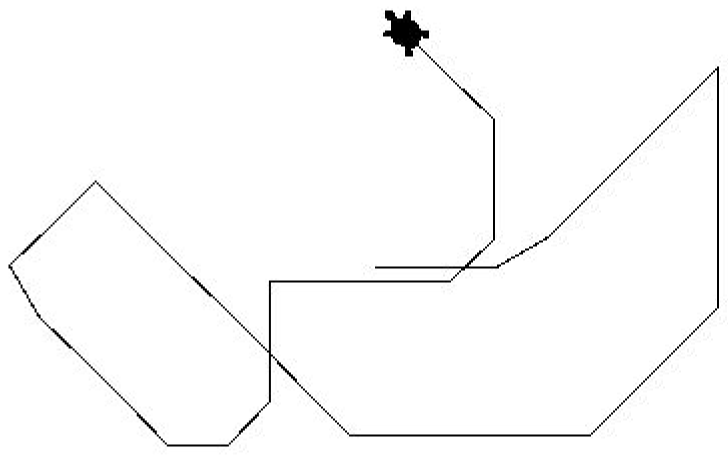
Test your code, can you turn left?

For working out yourself

Can you write the function and key binding for turning right?

Test again

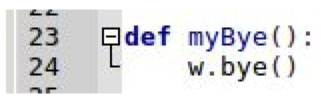
Now that you have turning left and right working, play with the turtle to check it all works. Here is my turtle after a quick spin round.



Handle Quitting

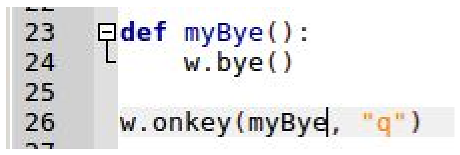
At the moment you have to close the window yourself to finish the program. It would be better if we provided a key handler for quitting. I am going to use the letter **q** to force a quit.

Add the following function to handle the quit command.



```
23  def myBye():
24      w.bye()
```

To call this function when **q** is pressed, add line 26 shortly after the new function.



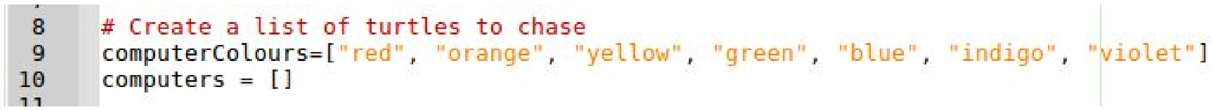
```
23  def myBye():
24      w.bye()
25
26  w.onkey(myBye, "q")
```

Test Quitting
Play with the turtle again, press **q** to see if your program exits correctly.

Computer Controlled Turtles

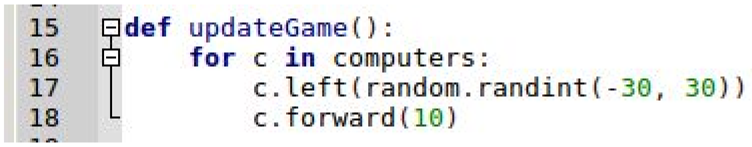
Now we are going to add the ability to spawn computer controlled turtles when the player presses the **spacebar**. Just for extra credit we will make the computer turtles random colours.

Create a list of turtles by adding lines 9 and 10, these go after the player creation.



```
 8  # Create a list of turtles to chase
 9  computerColours=["red", "orange", "yellow", "green", "blue", "indigo", "violet"]
10  computers = []
```

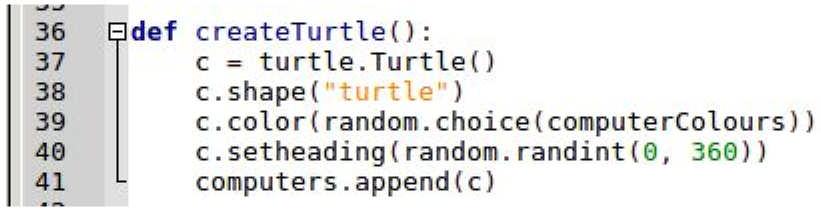
We will now create a function that gets called for every frame of the game. I will call it **updateGame** and it looks like this. Put it just after the definition of the computers list.



```
def updateGame():
    for c in computers:
        c.left(random.randint(-30, 30))
        c.forward(10)
```

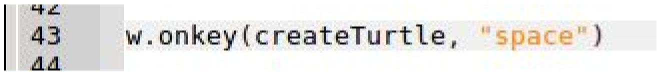
This will loop through all the turtles in the **computers** list and change their direction (randomly) and then move them forward a bit.

How to we create these turtles? We will add a function called **createTurtle** and call it whenever the user presses the **spacebar**. The **createTurtle** function looks like this.



```
def createTurtle():
    c = turtle.Turtle()
    c.shape("turtle")
    c.color(random.choice(computerColours))
    c.setheading(random.randint(0, 360))
    computers.append(c)
```

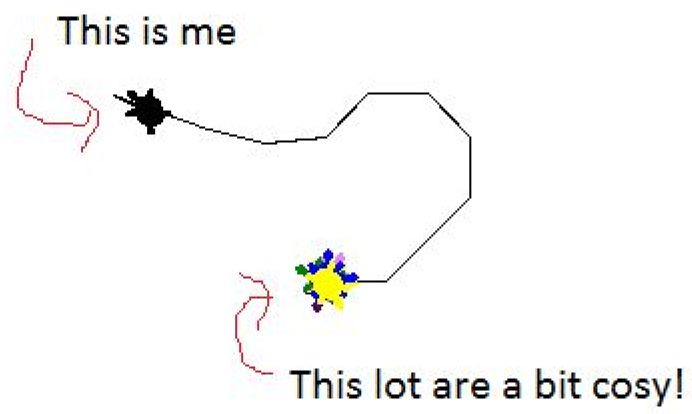
Then add the key handler as shown.



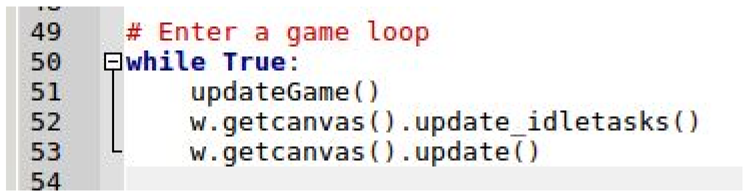
```
w.onkey(createTurtle, "space")
```

Test Code - Turtles don't move?

Run the code now and press **spacebar** a few times. You should find the program creates turtles in the centre, points them in a random direction but *they do not move*.



We need to invoke the **updateGame** function in our core game loop. The game loop should be at the end of the code, so add the code shown here as line 51.



```
49  # Enter a game loop
50  while True:
51      updateGame()
52      w.getcanvas().update_idletasks()
53      w.getcanvas().update()
54
```

Test Again - Randomly Moving Coloured Turtles?

Run the code again and press **spacebar** a few times. You should find that the **updateGame** function is being called in our loop and the computer controlled turtles will spider out from the centre.

You now know how to respond to user key presses and also run computer controlled code during the game loop. What else can you add to this program?

CONGRATULATIONS - YOU HAVE FINISHED THE TURTLE POWER TUTORIAL