[the academy_of_code]

Grade 3
Unit 4

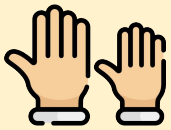
Python Programming

www.theacademyofcode.com/handouts

Lesson 1 & 2 - Python Turtle Instructions

Learning Outcomes:

- Learning some python basics
- Learning how for loops work by making shapes
- Making a shape making app



REMEMBER: Put up your hand. We love to help!

Python Turtle - Simple Instructions

We're going to get things going by experimenting with a few turtle functions to draw shapes. Pictured right are the functions we'll be using and what they mean.

turtle.forward	Go forward
turtle.backward	Go backward
turtle.right	Turn right
turtle.left	Turn left
turtle.penup()	Lifts the pen up so you can move the turtle without drawing
turtle.pendown()	Drops the pen back onto the screen so you can draw
turtle.fillcolor("Brown")	Changes the fill colour to brown(or other colour)
turtle.pencolor("Red")	Changes the pen colour to red (or other colour)
turtle.begin_fill()	Begins to fill the shape
turtle.end_fill()	End the filling sequence

trinket ▶ Run ? Modules

main.py

```
#!/bin/python3
import turtle
turtle.fillcolor("Brown")
turtle.begin_fill()
turtle.pencolor("Red")
turtle.forward (100)
turtle.right (90)
turtle.forward (100)
turtle.right (90)
turtle.forward (100)
turtle.right (90)
turtle.forward (100)
turtle.end_fill()
```

1. CTRL + Click this link www.taoc.ie/py1
2. Complete tasks 1-7 making all shapes as required

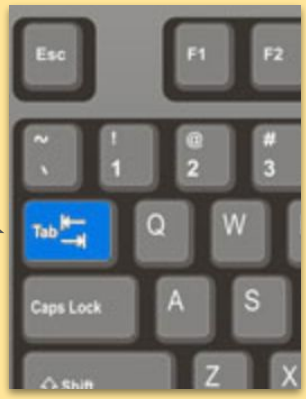
For Loops I

Writing out the same instructions many times is not very efficient. We're going to use **for loops** to write instruction in the form of a loop. The code on the right must be **indented** – this is important – it tells the program that these lines are to be repeated inside the loop.

```
#!/bin/python3
import turtle
for i in range(4):
  turtle.forward(100)
  turtle.right(90)
```

This indentation can be done in two ways:

- With two spaces
- By pressing the **Tab** button (shown here in blue)



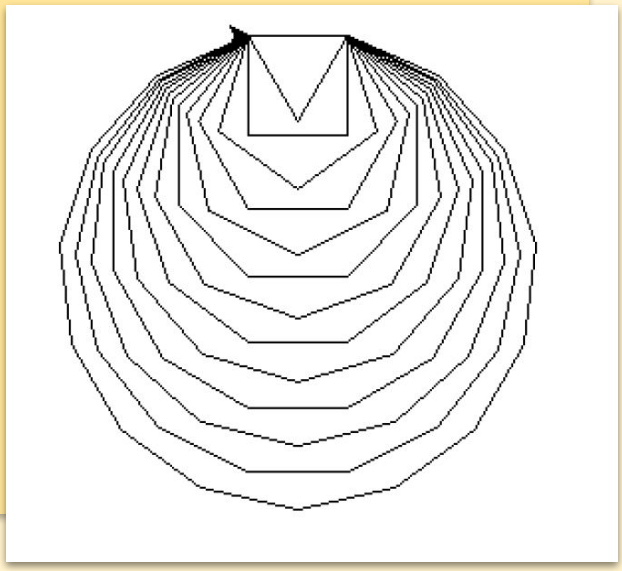
Complete the rest of the tasks on:

www.taoc.ie/py1

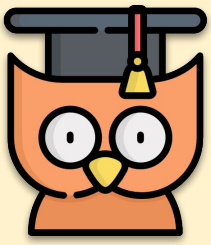
Lesson 2 - For Loops II

We're going to get more practice with For Loops by making a 'Shape Maker' drawing app. Users will be asked what features they would like to see on their shapes and the turtle will the draw out the shape.

Complete all of the tasks on: www.taoc.ie/py2

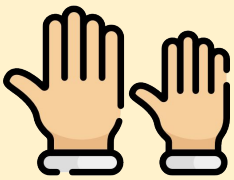


Lesson 3 - Python Project III - “Py” Charts



Learning Outcomes:

- Creating pie charts and bar graph from collected data (depending on venue)

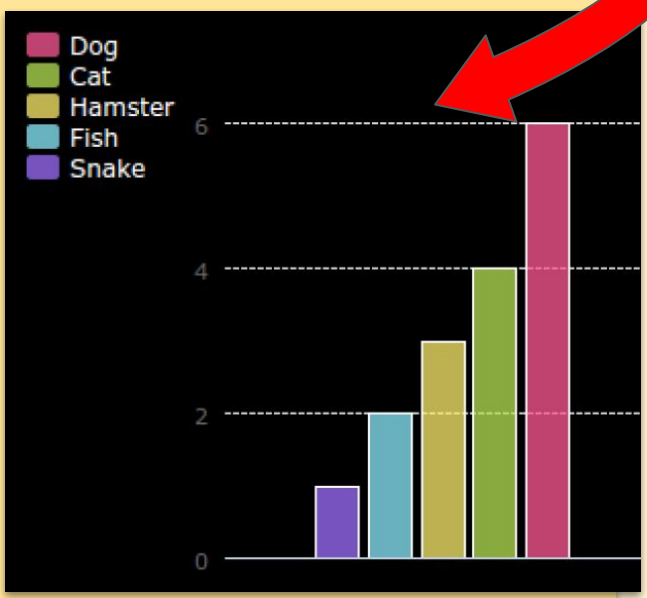


REMEMBER: Put up your hand. We love to help!

Collect Your Data

We’re going to create Pie and Bar charts based on data that you collect. Take 2 minutes to make a list (with a partner if you wish) of data about people's favourite pets, bands, games consoles or whatever you wish. Write out a list and make sure everyone’s favourite is included! If you don’t have a chance to collect data, you can use this example on the right.

```
Dog 6
Cat 4
Hamster 3
Fish 2
Snake 1
```



1. Open up this blank template: www.taoc.ie/pychart

2. We’re going to import a library called PyGal which will do a lot of the hard work for us. **Import** the library by putting the following code in your document:

```
import pygal
```

Using PyGal

3 Let's make a pie chart and display it. Type in the code on the right, but don't worry, it will get more interesting!

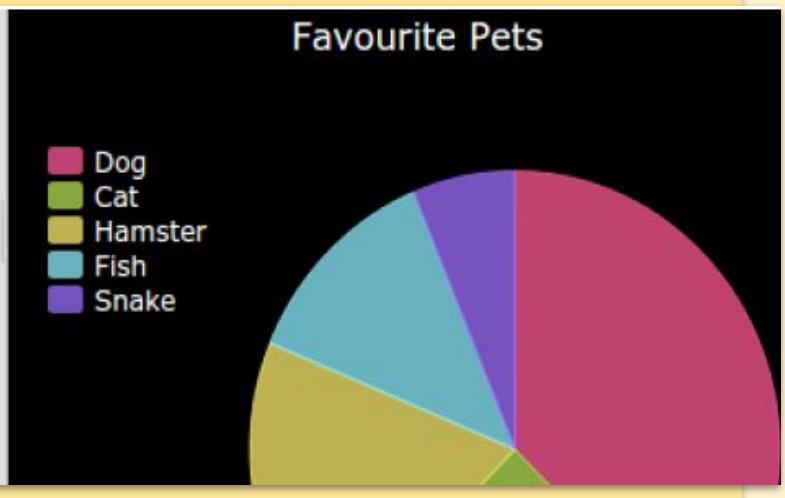
```
import pygal

piechart = pygal.Pie()
piechart.render()
```

4 Next, we're going to add the data for all of our animals, games or whatever you collected. Remember, if you don't have any data, use the example on the previous page.

```
import pygal

piechart = pygal.Pie()
piechart.add('Dog', 6)
piechart.add('Cat', 4)
piechart.add('Hamster', 3)
piechart.add('Fish', 2)
piechart.add('Snake', 1)
piechart.render()
```

5 Lasty, add a title to your code:

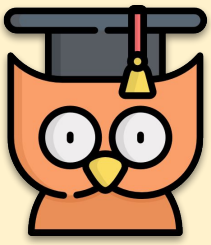
```
piechart = pygal.Pie()
piechart.title = 'Favourite Pets'
piechart.add('Dog', 6)
piechart.add('Cat', 4)
piechart.add('Hamster', 3)
piechart.add('Fish', 2)
piechart.add('Snake', 1)
piechart.render()
```



Extra Challenge - Bar Charts

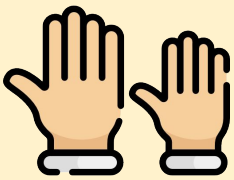
Collect more data and create a bar chart using the line `barchart = pygal.Bar()`. Follow the same pattern as above to make this. Consider asking people their favourite sport, how they get to The Academy of Code, what month their birthday is, if they like Fortnite (yes/no). The decision is yours!

Lesson 4 - Python Project I - Rock/Paper/Scissors



Learning Outcomes:

- Creating a rock/paper/ scissors game
- Using simple if statements
- Announcing the winner



REMEMBER: Put up your hand. We love to help!

Let's Get Coding

1. **CTRL+CLICK** www.bit.ly/rpstrinket

   This project contains these words.

   ```
   from random import randint
   ```

   We'll use them later to generate random numbers.

2. First, we're going to let the player choose Rock, Paper or Scissors by typing **r, p or s.** Add the blue lines of code. The second will print the player's choice.

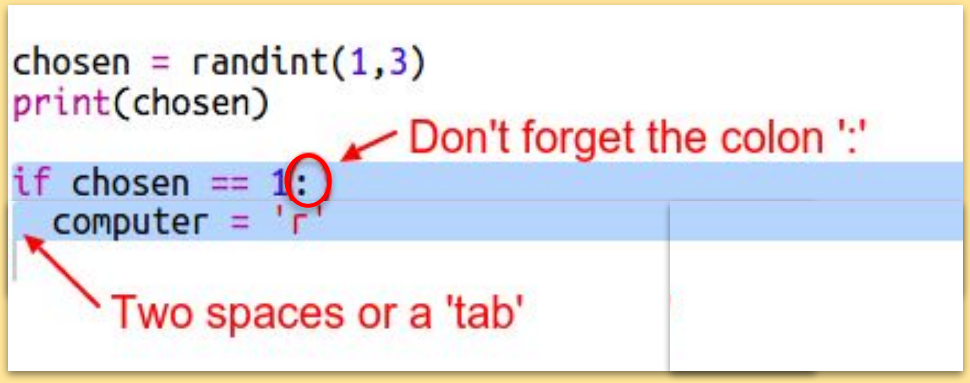
   ```
   from random import randint

   player = input('rock (r), paper (p) or scissors (s)?')

   print(player, 'vs')
   ```

3. Now to code the computers guess. randit will generate a random number to decide between rock, paper and scissors. Add in the blue lines and test it a few times to see that you're getting a random number.

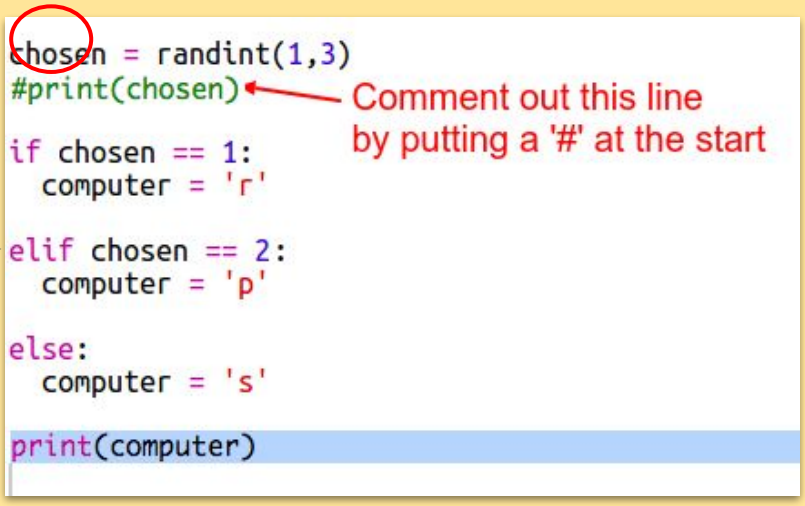
   ```
   player = input('rock (r), paper (p) or scissors (s)?')
   print(player, 'vs', end=' ')

   chosen = randint(1,3)
   #print(chosen)
   ```

Coding the Computer

In our code, **1 = rock (r), 2 = paper (p)** and **3 = scissors(s).** We're going to use an **if statement** to check **if** the number 1 (for example) is randomly chosen, the computer will see that as a rock. Add the lines in blue making sure to **indent** (move to the right). We can use two spaces or a 'tab' for this. Tab is usually above the **Caps Lock** button on your keyboard.



Short for "else if"



Add the rest of the code on the left, but first, *what does it do?* When finished, test your code to see if it works.

Announcing the Winner

Lets look at a case where the player choses ‘r’ (rock). If the computer chose ‘s’ (scissors) then the player wins (rock beats scissors). If the computer chose ‘p’ (paper) then the computer wins (paper beats rock). We can check the player choice *and* the computer choice using the word **and**.

```
if player == computer:
  print('DRAW!')

elif player == 'r' and computer == 's':
  print('Player wins!')

elif player == 'r' and computer == 'p':
  print('Computer wins!')
```

Add in more elif’s so that every ‘p’ and ‘s’ combination is covered.

Challenge: ASCII Art

Instead of using the letters r, p and s to represent rock, paper and scissors, can you use ASCII art? For example:

```
rock (r), paper (p) or scissors
(s)? s
>8 vs ___
Player wins!
```

Challenge: Create a new game

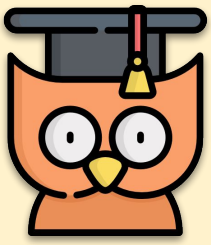
Can you create your own game like Rock, Paper, Scissors with different objects?

Click the ‘Duplicate’ button to make a copy of your Rock, Paper Scissors project to start from.

This example uses Fire, Logs and Water:

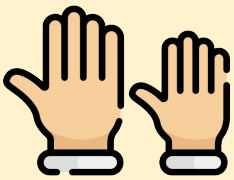
```
Fire, Logs, Water
Fire burns Logs
Logs make a bridge over Water.
Water puts out Fire
fire (f), logs (l) or water (w)? l
@@@ vs ~~~
Player wins!
```

Lesson 5 - Python Project II - Turtle Race



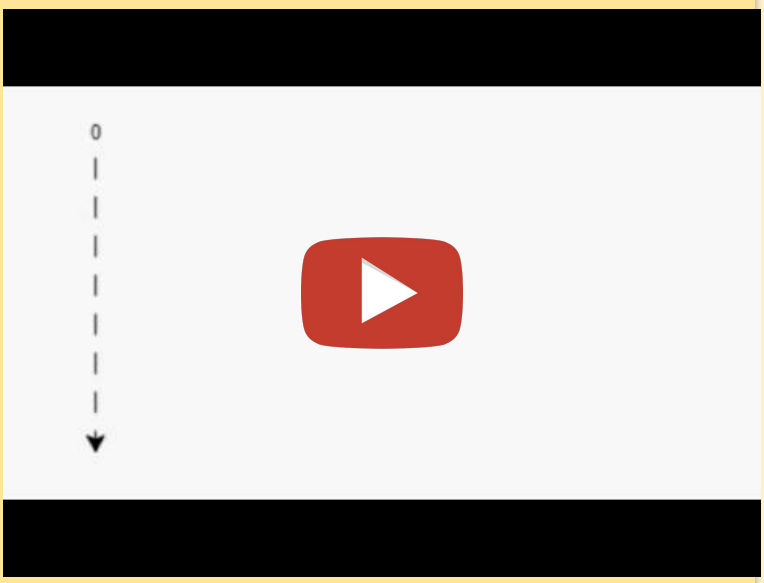
Learning Outcomes:

- Make a random race which races turtles for fun
- Learning about for loops
- Customising your game



REMEMBER: Put up your hand. We love to help!

Let's Get Coding



CTRL+CLICK this video to see the end result. The game will be similar to a horse racing game you may have seen before.

The code on the left below is very repetitive. There is an easier way to code this in python called a **for** loop. Change your code so it looks like the code on the right. We need to give it a **range of 6** because there are **6** numbers between 0 and 5.

```
1  #!/bin/python3
2  from turtle import *
3
4  write(0)
5  forward(20)
6  write(1)
7  forward(20)
8  write(2)
9  forward(20)
10 write(3)
11
```

```
#!/bin/python3
from turtle import *

for step in range(6):
  write(step)
  forward(20)
```

For Loops - Making it easy

1 **CTRL+CLICK** this link:

https://trinket.io/embed/python/49ff05172b

```
#!/bin/python3
from turtle import *
from random import randint

penup()
goto(-140,140)

for step in range(15):
  write(step)
  forward(20)
```

This code will make a list from 1-14. We then need to draw the lines

2 To draw the lines, we're going to have to turn the turtle to the right, before putting a pen down, drawing a line of 150, picking the penup and moving backward to the start of the line. Add the code in blue below and see what happens.

```
for step in range(15):
  write(step, align='center')
  right(90)
  forward(10)
  pendown()
  forward(150)
  penup()
  backward(160)
  left(90)
  forward(20)
```

2

`right(90)` makes the turtle turn right 90 degrees (a right angle.) Moving **`forward(10)`** before putting the pen down leaves a small gap between the number and the start of the line. After drawing the line you lift up the pen and go **`backward(160)`** the length of the line plus the gap.

Racing Turtles

3 When you use commands like **`forward(20)`** you are using a single turtle. But you can create more turtles. Add the following code to the end of your script (but make sure it's not indented)!

```
redTurtle = Turtle()
redTurtle.color('red')
redTurtle.shape('turtle')
redTurtle.penup()
redTurtle.goto(-160,100)
redTurtle.pendown()
```

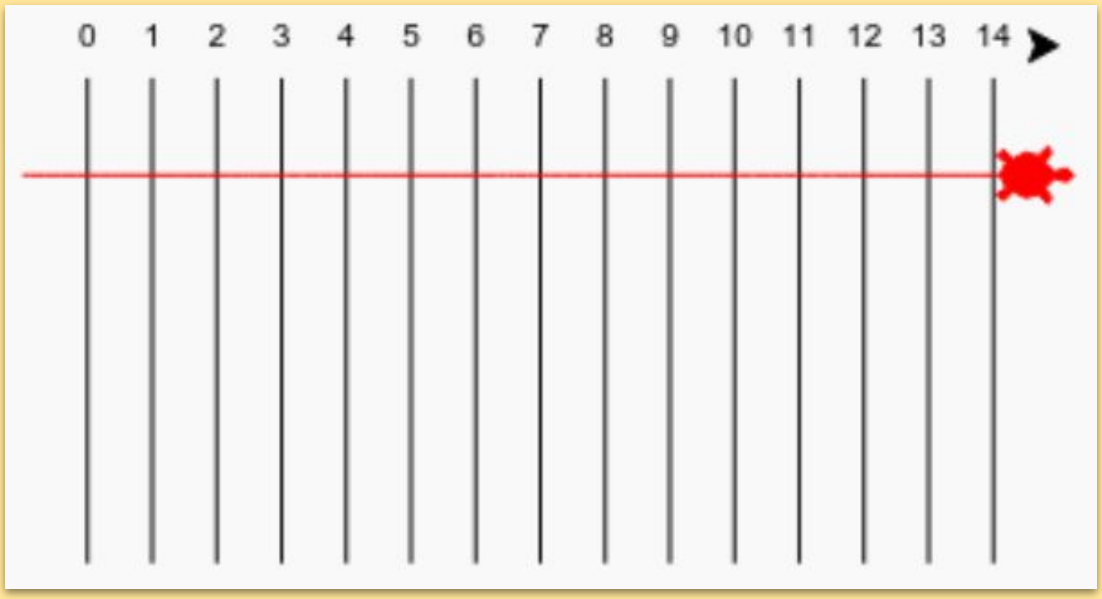
3

The first line creates a turtle called 'red'. The next lines set the colour and shape of the turtle. Now it really looks like a turtle!

4 Test the code and you should get a one-turtle race! We'll add more next. The code below will get it to move in a random movement

```
redTurtle.goto(-160,100)
redTurtle.pendown()

for turn in range(100):
  redTurtle.forward(randint(1,5))
```

4



Adding more Turtles

```
redTurtle = Turtle()
redTurtle.color('red')
redTurtle.shape('turtle')
redTurtle.penup()
redTurtle.goto(-160,100)
redTurtle.pendown()

blueTurtle = Turtle()
blueTurtle.color('blue')
blueTurtle.shape('turtle')
blueTurtle.penup()
blueTurtle.goto(-160,70)
blueTurtle.pendown()

greenTurtle = Turtle()
greenTurtle.color('green')
greenTurtle.shape('turtle')
greenTurtle.penup()
greenTurtle.goto(-160,40)
greenTurtle.pendown()

yellowTurtle = Turtle()
yellowTurtle.color('yellow')
yellowTurtle.shape('turtle')
yellowTurtle.penup()
yellowTurtle.goto(-160,10)
yellowTurtle.pendown()

for turn in range(100):
  redTurtle.forward(randint(1,5))
  blueTurtle.forward(randint(1,5))
  greenTurtle.forward(randint(1,5))
  yellowTurtle.forward(randint(1,5))
```

Copy

5

Paste

Paste

Paste

Copy

Paste

Paste

Paste

5 A one turtle race isn't much of a game! Add the code on the left to get the remaining turtles going.

You should be able to see a pattern here so don't forget...

REMEMBER TO:

COPY (*CTRL + C*)

AND

PASTE (*CTRL + V*)

Copy and Paste are a coders best friends!

Challenge: Do a twirl

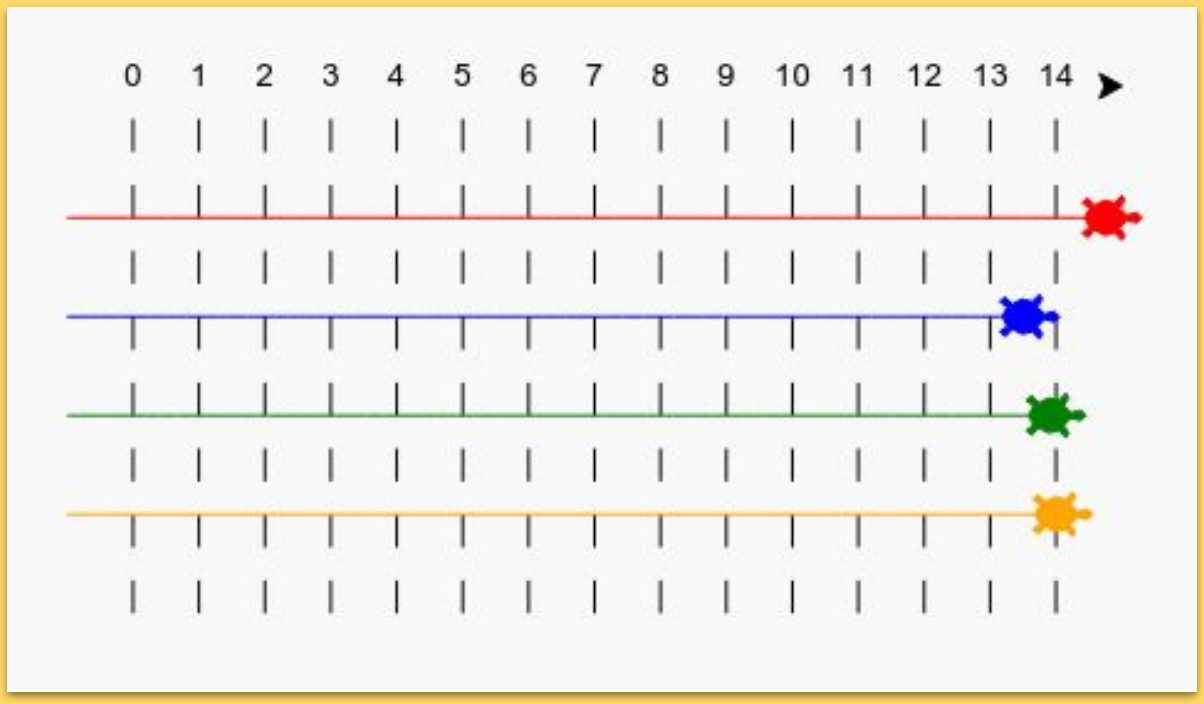
Can you use a **`for turn in range():`** loop to make each turtle do a 360 degree twirl after they get to the starting line? You'll need to make sure they are facing in the right direction at the start of the race!

`redTurtle.right(36)` will turn the red turtle right by 36 degrees.

Hint: A full turn is 360 degrees. A turtle could turn right 10 degrees 36 times, or left 5 degrees 72 times, or any other numbers make 360!

Challenge: Dashed lines

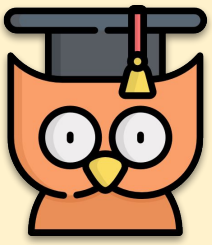
Can you use a loop to make the track lines dashed instead of solid?



Hint: Find the code that draws a straight line.

Try using:`for`, `forward()`, `penup()` and `pendown()`

Lesson 6 - Python Project III - ISS

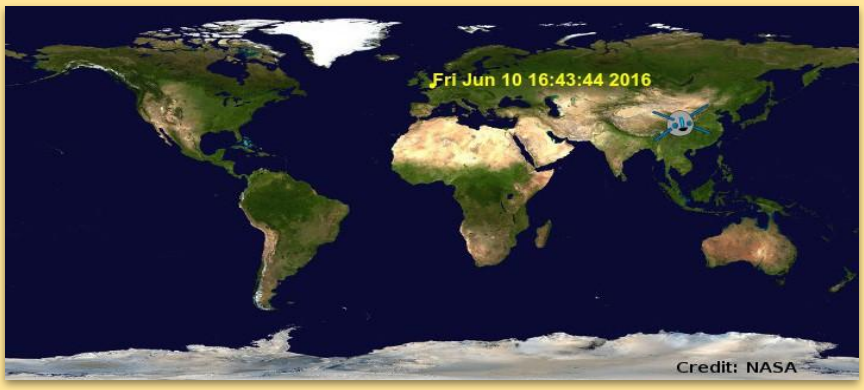


Learning Outcomes:

- Using Web Service to find the location of the International Space Station
- Plotting its location on a map.

Introduction

In this project you will use a **web service** to find out the **current location** of the International Space Station (ISS) and **plot its location** on a map.



Step 1: Who is in Space?

You're going to use a **web service** that provides live information about space. First let's find out who is currently in space. A web service has an address (url) just like a web page does. Instead of returning HTML for a web page it returns data.

1. Open **http://api.open-notify.org/astros.json** in a **web browser** (Chrome, Internet Explorer, Firefox etc.). You should see something like what is here (the data is live so your figures will be different). The data format is called **JSON** (pronounced Jay-Son).


```
{
 "message":
 "success",
 "number": 3,
 "people": [
  {
   "craft": "ISS",
   "name": "Yuri Malenchenko"
  },
  {
   "craft": "ISS",
   "name": "Timothy Kopra"
  },
  {
   "craft": "ISS",
   "name": "Timothy Peake"
  }
 ]
}
```

2 Let's call the web service from Python so we can use the results. Open this trinket: www.taoc.ie/iss. The **urllib.request** and **json** modules have already been imported for you.

3 Add the following code to **main.py** to put the web address you just used into a variable: `url = 'http://api.open-notify.org/astros.json'`

4 Now let's call the web service:

```
url = 'http://api.open-notify.org/astros.json'
response = urllib.request.urlopen(url)
```

5 Next you need to load the JSON response into a Python data structure:

```
url = 'http://api.open-notify.org/astros.json'
response = urllib.request.urlopen(url)
result = json.loads(response.read())
print(result)
```

You should see something like this.

```
{'message': 'success', 'number': 3, 'people': [{'craft': 'ISS', 'name': 'Yuri Malenchenko'}, {'craft': 'ISS', 'name': 'Timothy Kopra'}, {'craft': 'ISS', 'name': 'Timothy Peake'}]}
```

This is a Python dictionary with 3 keys: **message, number and people.**
The 'success' value of message tells you that the request was successful.

6 Now let's print the information in a more readable way. First, let's look up the number of people in space and print it:

```
url = 'http://api.open-notify.org/astros.json'
response = urllib.request.urlopen(url)
result = json.loads(response.read())

print('People in Space: ', result['number'])
```

`result['number']` will print the value associated with the key 'number' in the result dictionary. In the example this is 3. The value associated with the 'people' key is a list of dictionaries! Let's put that value into a variable so you can use it.

```
print('People in Space: ', result['number'])

people = result['people']
print(people)
```

You should see something like this:

```
[{'craft': 'ISS', 'name': 'Yuri Malenchenko'}, {'craft': 'ISS', 'name': 'Timothy Kopra'}, {'craft': 'ISS', 'name': 'Timothy Peake'}]
```

Now you need to **print out a line** for each astronaut.

You can use a **for loop** to do this in Python. Each time through the loop p will be set to a dictionary for a different astronaut.

```
print('People in Space: ', result['number'])

people = result['people']

for p in people:
  print(p)
```

You can then look up the values for ‘name’ and ‘craft’

You should see something like:

```
People in Space:
Yuri Malenchenko
Timothy Kopra
Timothy Peake
```

```
print('People in Space: ', result['number'])

people = result['people']

for p in people:
  print(p['name'])
```

You are using live data so your results will depend on the number of people currently in space.

Challenge: Show the Craft

As well as the name of the astronaut the web service also provides the craft that they are in (such as the ISS.)

Can you add to your script so that it also prints out the craft that the astronaut is in.

Example:

```
People in Space: 3
Yuri Malenchenko in ISS
Timothy Kopra in ISS
Tim Peake in ISS
```

Step 2: Where is the ISS?

The International Space Station is in orbit around Earth. It orbits the earth roughly every hour and a half. The ISS travels at an average speed of 7.66 km per second. Let's use another web service to find out where the International Space Station is.

1 First open the url for the web service in a new tab in your web browser:

http://api.open-notify.org/iss-now.json

You should see something like this:

```
{"timestamp": 1569932551, "iss_position": {"longitude": "78.5006", "latitude": "51.2045"}, "message": "success"}
```

2 Refresh the page a few times to see the position constantly being updated. The result contains the coordinates of the spot on Earth that the ISS is currently over. Longitude is the East-West position and runs from -180 to 180. 0 is the Prime Meridian which runs through Greenwich in London, UK.

3 Latitude is the North-South position and runs from 90 to -90. 0 is the Equator. Now you need to call the same web service from Python. Add the following code to the end of your script to get the current location of the ISS and make variables to store and display its positions.

```
url = 'http://api.open-notify.org/iss-now.json'
response = urllib.request.urlopen(url)
result = json.loads(response.read())

location = result['iss_position']
lat = location['latitude']
lon = location['longitude']
print('Latitude: ', lat)
print('Longitude: ', lon)
```

```
Latitude:  26.4169023793
Longitude:  58.378453289
```

4 It would be more useful to show the position on a map. First we'll need to import the turtle graphics library.

```
import json
import urllib.request
import turtle
```

Let's load a world map as the background image, there's one already included in your trinket.

NASA have provided this map and given permission for reuse. The map is centered at 0, 0 which is just what you need. You need to **set the screen size** to match the size of the image which is **720 by 360.**

```
main.py
lon = location['longitude']
print('Latitude: ', lat)
print('Longitude: ', lon)

screen = turtle.Screen()
screen.bgpic('map.jpg')
```

5 Add "**screen.setup(720, 360)"** and **"screen.setworldcoordinates (-180,-90, 180, 90)"**

```
screen = turtle.Screen()
screen.setup(720, 360)
screen.setworldcoordinates(-180, -90, 180, 90)
screen.bgpic('map.jpg')
```

You want to be able to send the turtle to a particular latitude and longitude. **Insert the two lines of code above**.

6 Let's **create a turtle** for the ISS. Your project includes 'iss.png' and 'iss2.png', try them both and see which one you prefer.

```
screen = turtle.Screen()
screen.setup(720, 360)
screen.setworldcoordinates(-180, -90, 180, 90)
screen.bgpic('map.jpg')

screen.register_shape('iss.png')
iss = turtle.Turtle()
iss.shape('iss.png')
iss.setheading(90)
```

7 The ISS starts off in the centre of the map, now let's move it to the correct location on the map.

```
screen.register_shape('iss.png')
iss = turtle.Turtle()
iss.shape('iss.png')
iss.setheading(90)

iss.penup()
iss.goto(lon, lat)
```

Note that latitude is normally given first, but we need to give longitude first when plotting (x,y) coordinates.

8 **Test your program** by running it. The ISS should move to its current location above Earth. Wait a few seconds and run your program again to see where the ISS has moved to.