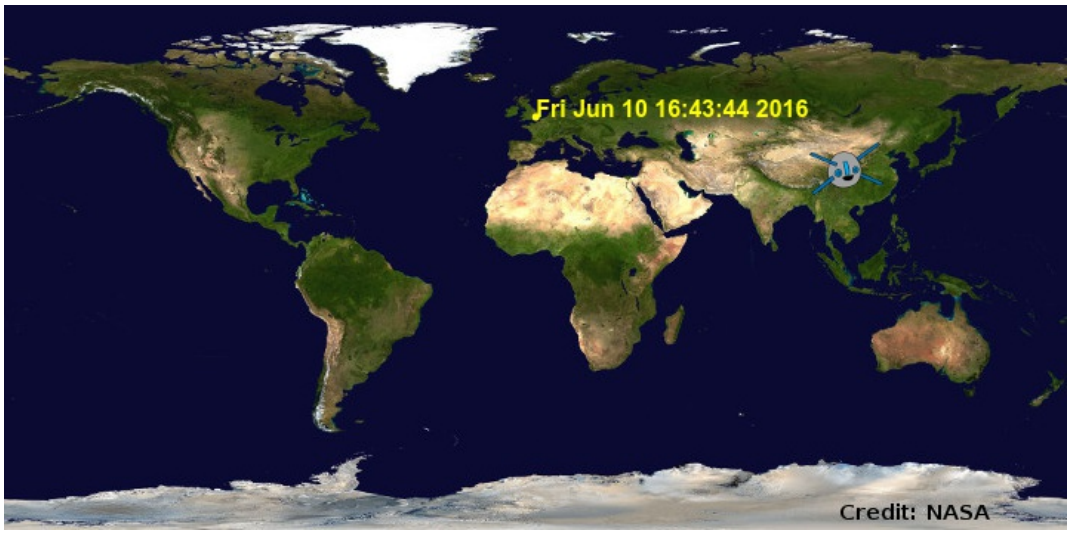
Where is the Space Station?

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.

Activity Checklist

- [ ] A web service has an address (url) just like a web page does. Instead of returning HTML for a web page it returns data.

Open **http://api.open-notify.org/astros.json** in a web browser.

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"
    }
  ]
}
```


The data is live so you will see a different result. The format is called JSON (say Jason).

- [ ] Let's call the web service from Python so we can use the results.

Open this trinket: **jumpto.cc/iss-go**. If you're reading this online, you can also use the embedded version of this trinket below.

- [ ] The `urllib.request` and `json` modules have already been imported for you.

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'
```

- [ ] Now let's call the web service:

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

- [ ] Next you need to load the JSON reponse 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. Good.

Note that you will see different results depending on who is currently in space!

- [ ] 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:

[{'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'

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

people = result['people']

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

You should see something like:

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

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

Save Your Project

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
Timothy Peake in ISS
```

Save Your Project

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. It's fast!

Let's use another web service to find out where the International Space Station is.

Activity Checklist

- [ ] First open the url for the web service in a new tab in your web browser: **http://api.open-notify.org/astros.json**

You should see something like this:

```
{ "iss_position": { "latitude": 8.54938193505081, "longitude": 73.16560793639105 }, "message": "success", "timestamp": 1461931913 }
```

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.

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:

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

print(result)
```

```
{'message': 'success',
'iss_position': {'latitude':
17.0762447364, 'longitude':
66.6454000717}, 'timestamp':
1461931742}
```

- [ ] Let's create variables to store the latitude and longitude, and then print them:

```
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)
```

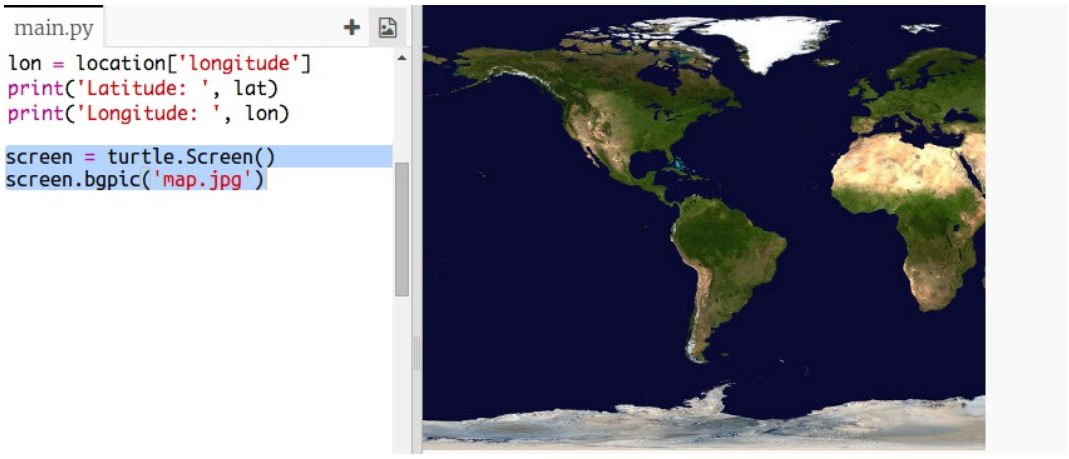
```
Latitude:  26.4169023793
Longitude:  58.378453289
```

- [ ] 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.

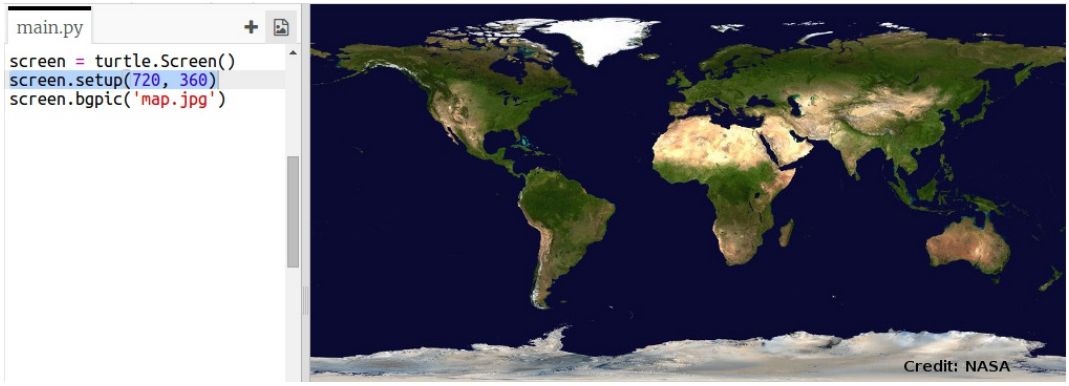


NASA have provided this gorgeous map and given permission for reuse.

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.

Add `screen.setup(720, 360)`:



- [ ] You want to be able to send the turtle to a particular latitude and longitude. To make this easy we can set the screen to match the coordinates we are using:

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

Now the coordinates will match the latitude and longitude coordinates that we get back from the web service.

- [ ] Let's create a turtle for the ISS.

```
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)
```

Your project includes 'iss.png' and 'iss2.png', try them both and see which one you prefer.

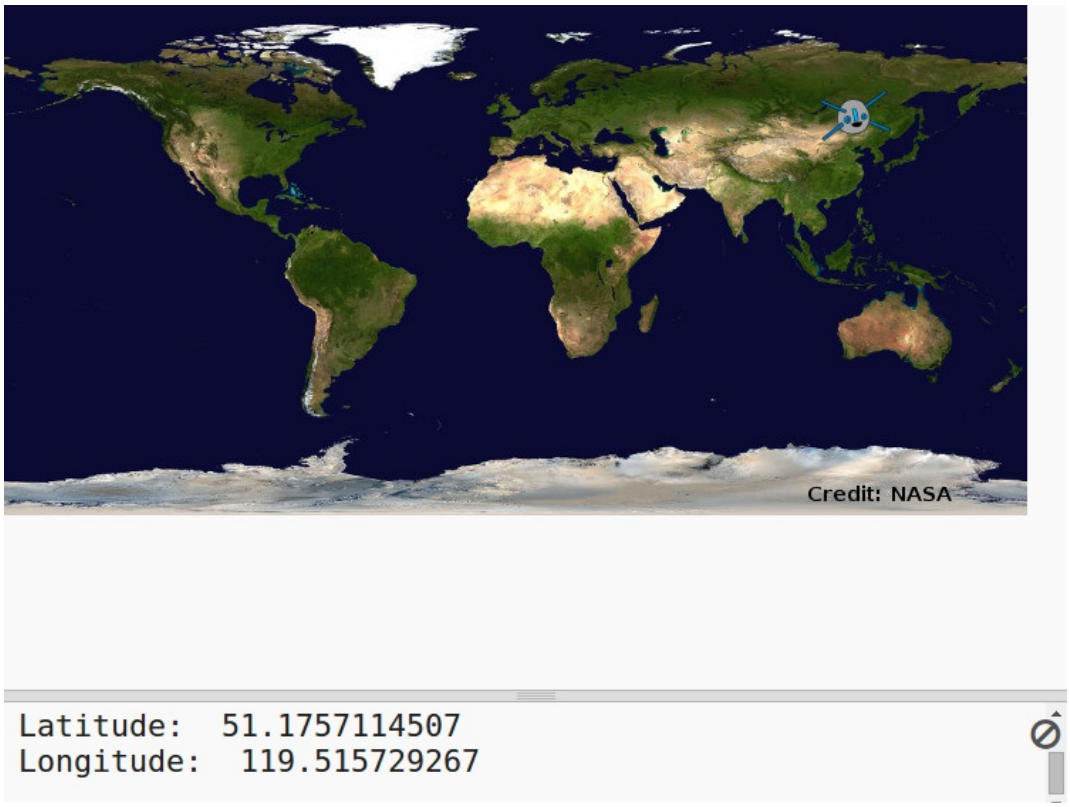
- [ ] The ISS starts off in the centre of the map, now let's move it to the correct location on the map:

```
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.

- [ ] 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.



Step 3: When will the ISS be overhead?

There's also a web service that you can call to find out when the ISS will next be over a particular location. Let's find out when the ISS will next be over the Space Centre in Houston, US which is at latitude 29.5502 and longitude = 95.097.

Activity Checklist

- [ ] First let's plot a dot on the map at these coordinates:

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

# Space Center, Houston
lat = 29.5502
lon = -95.097

location = turtle.Turtle()
location.penup()
location.color('yellow')
location.goto(lon,lat)
location.dot(5)
location.hideturtle()
```

- [ ] Now let's get the date and time that the ISS is next overhead.

As before we can call the web service by entering the url into the address bar of a web browser: **http://api.open-notify.org/iss-pass.json**

You should see an error:

```
{
  "message": "failure",
  "reason": "Latitude must be specified"
}
```

- [ ] This web service takes latitude and longitude as inputs so we have to include them in the url we use.

Inputs are added after a `?` and separated with `&`.

Add the `lat` and `lon` inputs to the url as shown: http://api.open-notify.org/iss-pass.json?lat=29.55&lon=95.1

```
{
  "message": "success",
  "request": {
    "altitude": 100,
    "datetime": 1465541028,
    "latitude": 29.55,
    "longitude": 95.1,
    "passes": 5
  },
  "response": [
    {
      "duration": 630,
      "risetime": 1465545197
    },
    {
      "duration": 545,
      "risetime": 1465551037
    },
    {
      "duration": 382,
      "risetime": 1465568806
    },
    {
      "duration": 625,
      "risetime": 1465574518
    }
  ]
}
```

The response includes several pass over times, we'll just look at the first one. The time is given in a standard time format, you'll be able to convert it to a

readable time in Python.

- [ ] Now let's call the web service from Python. Add the following code to the end of your script:

```
url = 'http://api.open-notify.org/iss-pass.json'
url = url + '?lat=' + str(lat) + '&lon=' + str(lon)
response = urllib.request.urlopen(url)
result = json.loads(response.read())
print(result)
```

```
{'message': 'success', 'request':
{'latitude': 29.5502, 'longitude':
-95.097, 'altitude': 100, 'datetime':
1465540436, 'passes': 5}, 'response':
[{'duration': 435, 'risetime':
1465541544}, {'duration': 622,
'risetime': 1465589616}, {'duration':
564, 'risetime': 1465595438},
{'duration': 156, 'risetime':
1465601504}, {'duration': 345,
'risetime': 1465613231}]}
```

- [ ] Now let's get the first pass over time from the result.

Add the following code:

```
url = 'http://api.open-notify.org/iss-pass.json'
url = url + '?lat=' + str(lat) + '&lon=' + str(lon)
response = urllib.request.urlopen(url)
result = json.loads(response.read())

over = result['response'][1]['risetime']
print(over)
```

```
1465595438
```

Pass over time in standard format

- [ ] The time is given as a timestamp so we'll need the Python time module so we can print it in a readable form and convert it to local time. Let's get the turtle to write the passover time by the dot.

- [ ] Add an `import time` line at the top of your script:

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

- [ ] The `time.cime()` function will convert the time to a readable form that you can write with the turtle:

```
over = result['response'][1]['risetime']
#print over

style = ('Arial', 6, 'bold')
location.write(time.ctime(over), font=style)
```

(You can remove or comment out the `print` line.)

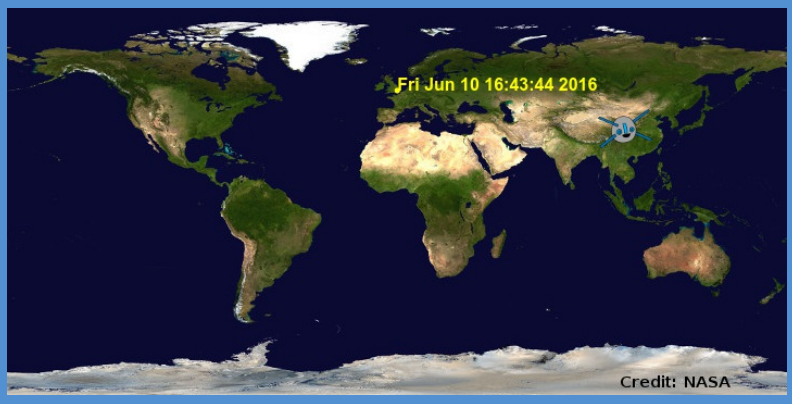
Save Your Project

Challenge: Find more passover times

You can use a website such as **http://www.latlong.net/** to look up the latitude and longitude of locations you are interested in.

Can you look up and plot the passover times for more locations?

- [ ] You'll need to change the latitude and longitude inputs to the web service.
- [ ] You'll need to plot the location and result on the map.



Save Your Project