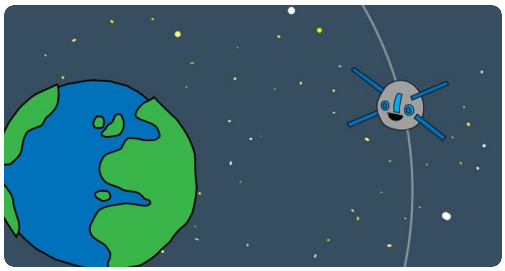
Projects

Where is the Space Station?

Find the exact location of the ISS

Python

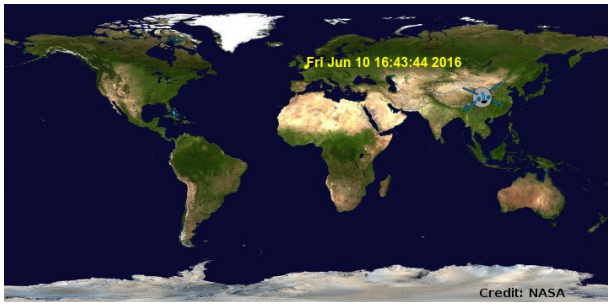


Step 1 Introduction

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.

Instructions

The space station icon shows the current location of the ISS. The yellow text shows when the ISS will next pass over the Space Centre in Houston, US.



What you will learn

This project covers elements from the following strands of the **Raspberry Pi Digital Making Curriculum (http://rpf.io/curriculum)**:

- **Combine programming constructs to solve a problem.** (https://www.raspberrypi.org/curriculum/programming/builder)

Step 2 What you will need

Hardware

- A computer with internet connection

Software

- **Trinket** (https://trinket.io/) online editor

Additional resources

- Starter project - **rpf.io/iss-on** (http://rpf.io/iss-on)
- A completed version of this project - **trinket.io/python/5d3327334d** (https://trinket.io/python/5d3327334d)
- Open Notify webservices - **api.open-notify.org** (http://api.open-notify.org/)

Step 3 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 website does. Instead of returning HTML for a web page, it returns data.

- Open **the web service** (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 probably see a slightly different result. The data format is called `JSON` (pronounced like 'Jason').

What is JSON?

JSON is a format for storing and sharing data. JSON (say Jason) stands for JavaScript Object Notation, but it isn't just used with JavaScript.

JSON is a text format that can be used in code and is fairly easy for people to read.

```
{
  "name": "Ogre",
  "size": 90,
```

```
  "power": 86,
  "intelligence": 12,
  "magic": 0
}
```

A JSON **object** is a list of key-value pairs inside curly brackets `{}`.

A value can also be a **list** inside square brackets `[]`:

```
{
  "name": "Ogre",
  "size": 90,
  "power": 86,
  "intelligence": 12,
  "magic": 0,
  "weapons" : ["club", "rock", "bone"]
}
```

You need to call the web service from a Python script, so you can use the results.

- Open this trinket: **http://rpf.io/iss-on** (http://rpf.io/iss-on).

The `urllib.request` and `json` modules have already been imported for you at the top of the `main.py` script.

- Add the following code to `main.py` to store the URL of the web service you just accessed as a variable:

```
url = 'http://api.open-notify.org/astros.json'
```

- Now call the web service:

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

- 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 three keys: `message`, `number`, and `people`.

Using key:value pairs in Python

Here is a dictionary of band members. The **key** is the first part (e.g. 'john'), and its associated **value** is the second part (e.g. 'rhythm guitar').

```
band = {
  'john' : 'rhythm guitar',
  'paul' : 'bass guitar',
  'george' : 'lead guitar',
  'ringo' : 'bass guitar'
        }
```

Here's how to add a key:value pair to the dictionary:

```
# Add a key:value pair
band['yoko'] = 'vocals'
```

Here's how to remove a key:value pair from the dictionary:

```
# Remove a key:value pair
del band['paul']
```

That `message` has the value `success` tells you that you successfully accessed the web service. Note that you will see different results for `number` and `people` 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 Python `for` loop to do this.

For loop with a list in Python

This **for loop** which will print each item in the `animals` list.

```
animals = ["fox", "wolf", "panda", "squirrel"]

for animal in animals:
  print(animal)
```

The output is:

```
fox
wolf
panda
squirrel
```

Notice that the `print` line of code is slightly further to the right. This is called **indentation** - the line is **indented** to show that it is inside the loop. Any lines of code inside the loop will be repeated.

- 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`. Let's show the names of the people in space:

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

people = result['people']

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

You should see something like this:

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

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

Step 4 Challenge: show the craft

In addition to the name of the astronauts, the web service also provides the craft that they are on, such as the ISS.

- Can you add to your script so that it also prints out the craft for each astronaut?

Example:

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

Change your `for` loop so it looks like this:

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

Step 5 Where is the ISS?

The International Space Station is in orbit around Earth. It completes an orbit of the earth roughly every hour and a half, and 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.

- First open the URL for the web service in a new tab in your web browser: **http://api.open-notify.org/iss-now.json** **(http://api.open-notify.org/iss-now.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.

What are latitude and longitude?

Latitude and Longitude

Latitude and longitude are used to give coordinates to locations on the Earth's surface.

Latitude indicates the position along the north-south axies, and can be any value between 90 and -90. 0 marks the equator.

Longitude indicates the position along the east-west axis, and can be any value between -180 and 180. 0 marks the prime meridian, which runs through Greenwich in London, UK.

Coordinates are given as **(latitude, longitude)**. The coordinates of the Royal Observatory at Greenwich are (51.48, 0). As you can see, the latitude (north-south) position is given first.

You can look up the latitude and longitude of places at **latlong.net (http://www.latlong.net/)**.

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

```
Latitude:  40.0603
Longitude:  138.2383
```

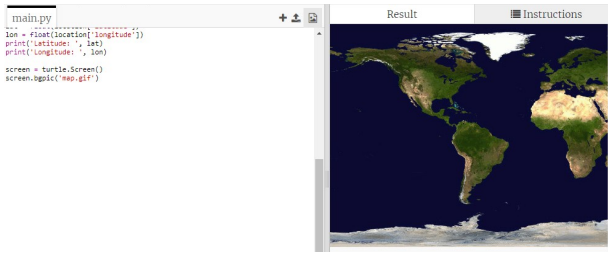
Step 6 Plotting the ISS on a map

It would be useful to show the position on a map. You can do this using Python Turtle graphics!

- First we'll need to import the `turtle` Python library:

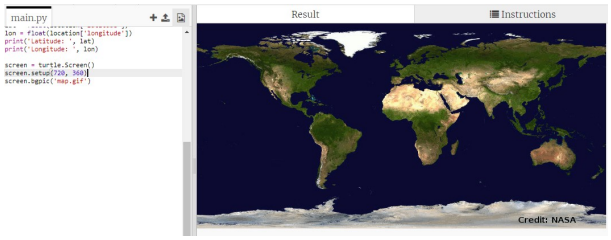
```
import json
import urllib.request
import turtle
```

- Next, load a world map as the background image. There's one already included in your trinket called 'map.gif'! NASA has provided this beautiful map and given permission for reuse.



The map is centered at `(0,0)` latitude and longitude, 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 pixel. Add `screen.setup(720, 360)` :



- You want to be able to send the turtle to a particular latitude and longitude. To make this easy, you can set the screen to match the coordinates you're using:

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

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

- Let's create a turtle icon for the ISS. Your trinket includes 'iss.gif' and 'iss2.gif' — try them both and see which one you prefer.

Changing Python Turtle icons

Changing Python Turtle icons

Instead of always using a turtle, you can tell the Python Turtle icon to use a different image. The image should be small, so that it does not cover up too much of the screen: 50 × 50 pixels will give you a large icon.

- First you need to register the image with the `screen`:

```
screen = turtle.Screen()
screen.register_shape('happy.png')
```

- Then you can set the `shape`:

```
turtle.shape('happy.png')
```

- Turtle icons face right to start with. You can change the heading to get your image to face upwards:

```
turtle.setheading(90) # face upwards
```

See an example here:

Your code should look like this:

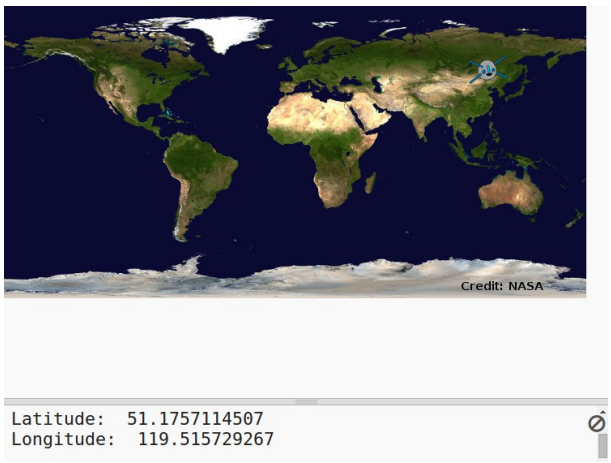
- The ISS starts off in the centre of the map, now let's move it to the correct location:

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

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

Note: 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.



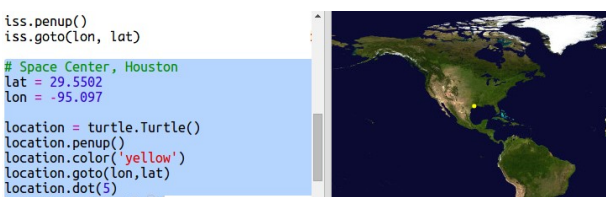
- Wait a few seconds and run your program again to see where the ISS has moved to.

Step 7 When will the ISS be overhead?

There's also a web service that you can use 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, USA, which is at latitude `29.5502` and longitude `95.097`.

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



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

- As before, you can call the web service by entering its URL into the address bar of a web browser: **api.open-notify.org/iss-pass.json** **(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 you have to include them in the URL. Inputs are added after a `?` and separated with `&`.

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

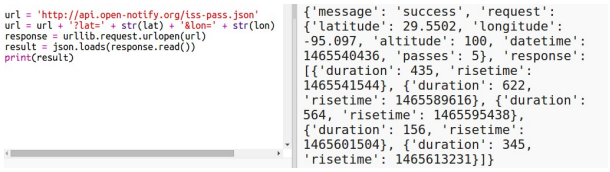

```
{
  "message": "success",
  "request": {
    "altitude": 100,
    "datetime": 1465541028,
    "latitude": 29.55,
    "longitude": 95.1,
    "passes": 5
  },
  "response": [
    {
      "duration": 630,
      "risetime": 1465545197
    },
    {
      "duration": 545,
      "risetime": 1465551037
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

}

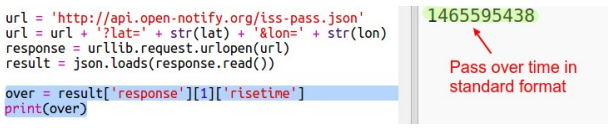
The response includes several pass-over times, and we'll just look at the first one. The time is given as a Unix time stamp (you'll be able to convert it to a readable time in your Python script).

> **Unix timestamps**
>
> Unix timestamps are a convenient way to store a date and time as a single number.
>
> A Unix timestamp is the number of seconds since 1 January 1970 in UTC (an international standard for time). For example, `1498734934` is 29 June 2017 at 11:15am.
>
> You can find the current Unix timestamp at **unixtimestamp.com (http://www.unixtimestamp.com/)**.

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

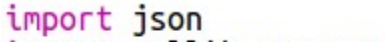


- Now let's get the first pass-over time from the result. Add the following code:



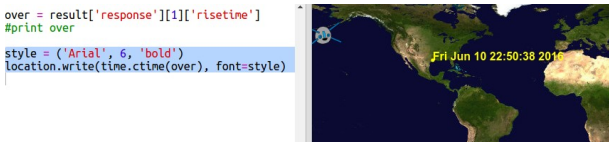
We'll need the Python `time` module so we can print it in a readable form and convert it to local time. Then we'll get the script to write the pass-over time by the dot for Houston.

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



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

- The `time.ctime()` function will convert the time stamp to a readable form that you can write onto your map:



(You can remove the `print` line, or turn it into a comment by adding `#` at the start so your script will ignore it.)

- If you like, you can change the colour and format of the text.

Writing text with Python turtle

You can use a turtle to write text.

```
turtle.write('Hello!')
```

Set the turtle's color to create coloured text:

```
turtle.color('deep pink')
turtle.write('Hello!')
```

You can also change the font and alignment of the text.

```
style = ('Courier', 30, 'italic')
turtle.write('Hello!', font=style, align='center')
```

The font is a tuple containing:

- The font name such as 'Arial', 'Courier', or 'Times New Roman'
- The font size in pixels
- The font type, which can be 'normal', 'bold', or 'italic'

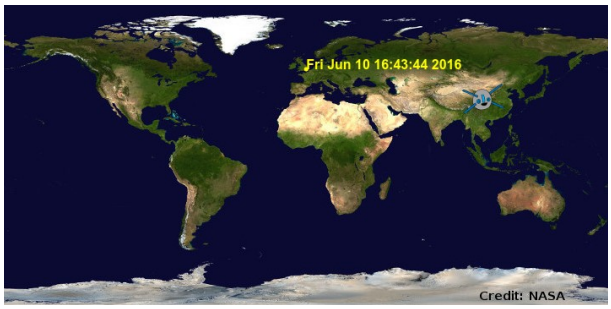
To set the alignment which controls how the text is positioned based on the position of the turtle, use the `align` parameter. `align` can be set to one of these options: 'left', 'center', 'right'

Example:

Step 8 Challenge: find more pass-over times

To look up the latitude and longitude of a location you are interested in, you can use a website such as **www.latlong.net/** **(http://www.latlong.net/)**.

- Can you look up and plot the pass-over times for more locations?



Here's an example using the location of the Baikonur Cosmodrome, a spaceport in southern Kazakhstan. The code goes at the end of your program, after plotting the Houston Space Center pass-over time.

```
# Baikonur Cosmodrome
lat = 45.86
lon = 63.31

location.penup()
location.color('orange')
location.goto(lon,lat)
location.dot(5)
```

```
location.hideturtle()

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

#print(result)
over = result['response'][1]['risetime']
location.write(time.ctime(over))
```

Try adding more locations!
