KV6006 practical session - 2 - Data Source - APIs

These exercises pull data from web sources, then parse it a bit. The main purpose here is to explore handling JSON data in Python, though using a basic GUI library and drawing graphs might also come in handy.

People in Spaaaace!

Open the Chrome browser, and paste or type this into the address bar: http://api.open-notify.org/astros.json

You should get data which looks something like:

```
{"message": "success", "people": [{"name": "Cai Xuzhe", "craft": "Tiangong"}, {"name"
```

That’s a bit messy, so let’s reformat it:

```
{
  "message": "success",
  "people": [
    {
      "name": "Cai Xuzhe",
      "craft": "Tiangong"
    },
    {
      "name": "Chen Dong",
      "craft": "Tiangong"
    },
    {
      "name": "Liu Yang",
      "craft": "Tiangong"
    },
    {
      "name": "Sergey Prokopyev",
      "craft": "ISS"
    },
    [ ... ]
    {
      "name": "Anna Kikina",
      "craft": "ISS"
    }
  ],
  "number": 10
}
```

This is JSON-structured data, containing information about all the humans who are currently in space. Documentation for this API may be found at http://open-notify.org/Open-Notify-API/People-In-Space/, where you'll notice that the data source is… a guy called Nathan who's really obsessed with space missions, who updates this by hand every time there's a launch. Seriously.

Let's do something with this programatically.

Open the Thonny editor – there's a terrible `Th` icon in the top menu bar – and make yourself a new file in the `student_work` directory. In the upper pane enter the following Python:

```
import requests

r = requests.get('http://api.open-notify.org/astros.json')
data = r.json()

print(data['number'])
```

Try running the code (click the green run button, choose 'Run current script' from the Run menu, or hit `F5`), and you should receive a number in the lower `Shell` pane.

Congratulations, you just retrieved and parsed some JSON data.

What does the number represent?

If it doesn't work

If you can't run the code, click the text in the lower-right corner of the window and check it says something like `Local Python 3 — Thonny's Python`.

If Thonny gives you a package error on `requests`, go to Tools -> Manage Packages. Search for `requests`, then install it. You may have to do this with other packages during the workshop.

Prettier output via a basic GUI

Not many users like viewing data in a terminal, so let's build them a GUI window. There are dozens of ways of doing this; we're going to use one of the simplest, a toolkit called GUIzero. Documentation for GUIzero is here: https://lawsie.github.io/guizero/.

Edit the program in Thonny so it looks like this (you can omit the comments if you wish):

```
import requests
from guizero import App, Text

# Give ourselves an application window to put things in
app = App(title="People iiiiin Spaaaaaace!", height=150)
```

```python
# Get the data and parse it as JSON
r = requests.get('http://api.open-notify.org/astros.json')
data = r.json()

# Write text into the GUI window
message0 = Text(app, " ", height=2) # Spacer to push things down.
message1 = Text(app, "Number of people in space: ", size=24)
message2 = Text(app, data['number'], size=48, color='red')

# Now show the window
app.display()
```

Run that, and you should see a nice neat GUI window. Excellent.

More data

Those are real people up there. They have names, families, hopes and dreams. One would hope their dreams included 'going to space,' in which case: good job. And we know nothing about their families. But we can at least display their names.

We can use a python iterator to step through (`data['people']`), and extract their names.

```python
# [...] after message2:
people_string = ""
for person in data['people']:
    people_string += person['name']
    people_string += "\n" # Add a new line
message3 = Text(app, people_string, size=14, color='blue')
```

You'll need to fix at least one bug: the window won't be tall enough to display all the names.

If you get stuck, you'll find code in `examples/peopleinspace`. That goes for this whole worksheet: try to write the code yourself, but draw on the `examples` directory when you need to. You'll also find examples of the JSON data structures for each exercise, to explore.

Energy generation, and drawing graphs

Let's try something different. You'll want a fresh file for this.

The National Grid publishes extensive data around electricity generation, via a well-documented API: https://carbonintensity.org.uk. Let's grab some data!

```python
import requests

r = requests.get('https://api.carbonintensity.org.uk/generation')

```

```
# Parse the JSON response
mix = r.json()

# Now step through the fuels list; see example.json for the structure we're walking t
for fuel in mix['data']['generationmix']:
    fueltype = fuel['fuel']
    percentage = fuel['perc']
    # Need to cast percentage to string to concatenate it for printing:
    print(fueltype + ": " + str(percentage))
```

OK, a badly-formatted table is fine, but… let's draw a graph! Modify the code above to include the following (or open `elecgenapi-2.py` from the examples folder).

```
import matplotlib.pyplt as plt

# [...]
mix = r.json()

# Give ourselves some empty lists
fueltype = []
percentage =[]

for fuel in mix['data']['generationmix']:
    fueltype.append(fuel['fuel'])
    percentage.append(fuel['perc'])

# Set up a chart
fig1, ax1 = plt.subplots()
# Plot a pie chart of the percentage data, using fueltype as labels
ax1.pie(percentage, labels=fueltype, autopct='%1.1f%%', shadow=False, startangle=90)
ax1.axis('equal')

plt.show()
```

Run that, and after a few seconds (possibly *quite a few seconds*) you should have a pie chart. An ugly one, probably, but you can immerse yourself in the matplotlib documentation at a later date.

Other APIs

There are, of course, rather a lot of these sorts of API out there. Some you may wish to explore at a later date:

- OpenWeatherMap. https://openweathermap.org/api. Terrific breadth and depth of data in a service that's free for the first 1000 API calls per day.
- The 'people in space' guy also publishes data about the International Space Station: http://open-notify.org/.
- Here's a decent list of 'awesome' APIs: https://github.com/TonnyL/Awesome_APIs, though it's no longer updated.

- Need George R.R. Martin data in JSON format? https://anapioficeandfire.com has you covered. Because… nope, I’m struggling here.
- Not only does SpaceX have a wonderfully complete public API, there’s even a python wrapper for it: https://pypi.org/project/spacexpypi/.

Recap

You’ve:

- Used a couple of APIs to retrieve data
- Parsed that data as JSON
- Built a (very) basic GUI app to present data
- Drawn a graph constructed from some other data.

A little quality time with the API and library documentation, and you could build a GUI app which periodically refreshed the electricity generation data, and drew a line chart as the mix changed. But let’s move on…