nustem

People in Spaaaace!

Pull data from a web feed, and display it in GUI window

Ingredients

Platform:	Raspberry Pi / Raspbian Python3
Libraries:	GUIzero

Glossary

GUI	"Graphical User Interface" – The thing we were all terribly excited about back in the mid-80s.
JSON	"JavaScript Object Notation" – a standardised way of packaging structured data so the structure can be unpacked by a program.
parsing	We parse a sentence by breaking it into its constituent parts. Similarly, 'parsing' in computing means 'to break down into logical components.'

Fetching data from the web

Several of these examples rely on fetching data from the web, then doing something with it. This is probably the simplest.

All the code for this worksheet is in `Digital-Making/examples/peopleinspace`

Start by opening `peopleinspace.py` in Thonny, have a read through, and run it.

The `requests` module does all the hard work for us here. It retrieves some data from the web which lists all the people who are currently in space. The data is returned in a form which looks like `example.json`:

```
{
    "message": "success",
    "people": [
        { "name": "Alexey Ovchinin", "craft": "ISS" },
        { "name": "Nick Hague", "craft": "ISS" },
        { "name": "Christina Koch", "craft": "ISS" },
        { "name": "Alexander Skvortsov", "craft": "ISS" },
        { "name": "Luca Parmitano", "craft": "ISS" },
        { "name": "Andrew Morgan", "craft": "ISS" }
    ],
    "number": 6
}
```

This is JSON-formatted data. Ignore all the brackets, and it looks reasonably understandable. The brackets, however, help Python (via `requests`) parse the information almost as easily as you can.

In the simplest possible program, the code would go:

```
import requests
r = requests.get('http://api.open-notify.org/astros.json')
data = r.json()
print(data['number'])
```

Here, `r` holds the response from the website; `data` holds that response parsed as json, and `data['number']` returns the value with the key '`number`'. This key:value sort of structure is called a *dictionary* in Python, and you'll see it a lot.

Northumbria University NEWCASTLE

Iterators

`peopleinspace.py` also spits out the names of the astronauts currently on orbit. It does this by stepping through the set of values for the `people` key (lines 18 and 19). This sort of `for thing in list:` construction is called an *iterator*, and again, you'll see loads of those in Python code.

In this case, the variable `person` is given the values from each line in turn, so each pass through the loop spits out another person's name until all of the `people` data is exhausted.

GUIzero

Printing text to the Terminal is conveniently simple, but sometimes you need more. Unfortunately, building a complete Graphical User Interface can be complex. The GUIzero library exists to keep your choices minimal, and in return keep things as simple as possible.

Open `peopleinspace-2.py` in Thonny, and you'll see what this looks like.

Line 13 asks GUIzero's App object for a window, with a title.

Lines 20-23 build the window contents, as individual Text objects, each of which can have their own size and colour.

Line 26 then instructs the App object to display itself. Job done.

Challenge: display names in the GUI window

You already have the code to print out the list of names, so we need to do just a few things:

1. Set up an empty string variable to hold the list of names. Try:
   ```
   peopleString = ""
   ```

2. Iterate over the people, as in `peopleinspace.py`. For each `person`:

3. Append the person's name to `peopleString`:
   ```
   peopleString += person['name']
   ```

4. Also append a new line character:
   ```
   peopleString += "\n"
   ```

5. When peopleString is complete, add it to the GUI window with something like:
   ```
   message3 = Text(app, peopleString, size=14, color='blue')
   ```

See if you can puzzle out where to put these lines so the list of names displays in your GUI window. You might also encounter a bug which needs fixing elsewhere in your code. Our end result looked like this: