Modules are Files

Modules are actually just plain text files with a `.py` ending. So if you write:

```
import string
```

you actually tell the Python interpreter to fetch the `string.py` file.

Let's have a closer look at that!

All modules that are executed by the Python interpreter get assigned two *magic* variables. `__name__` , the name of the module and `__file__` the absolute path of that file.

You already know these two variables from the debugger in mu-editor.

In [2]:

```
1  import string
2
3
4  print(string.__file__)
5  print(string.__name__)
```

```
/usr/local/anaconda3/lib/python3.7/string.py
string
```

Let's try that ourselves!

- Create a new file `global_warming.py`
- In the file, put these four lines of code:

```
def global_warming_status():
      return 'The globe is heating up!'


status = global_warming_status()
print(status)
```

- What is the code doing?

- Save the file and run it on the terminal via: `python global_warming.py`

If modules are text files, we can import them!

- Create a new file called `united_nations.py` with the following code:

```
import global_warming



status = global_warming.global_warming_status()
print('UN expert panel says: ' + status)
```

- What do you think will happen when you run it?

- Run `united_nations.py` either in Mu or in the terminal.
  - Did you get the expected result?

Python Main Modules vs. Imported Modules

- Python modules that you run with `python <file_name>.py` are considered **main** modules
  - That is, the file we *actually* want to run
- All other files are modules, and should not execute anything until we ask them to do so.

- In your `united_nations.py` file, add the line `print(__file__, __name__)`
  - Run the file again. What happens?

- In your `global_warming.py` file, add the line `print(__file__, __name__)`
  - First run the `global_warming.py` file. What happens?
  - Then run the `united_nations.py` file. What happens?

Ok, so `__name__` is either the string `'__main__'` if it is the *first* file to be run, or the actual name of the file (without extension) otherwise.

We can use that to avoid running code in `global_warming.py`, *unless* we use it as the main file?

In [ ]:

```
def global_warming_status():
    return 'The globe is heating up!'


if __name__ == '__main__':
    status = global_warming_status()
    print(status)
    print(__file__, __name__)
```