Session 6 Bonus: Making Python Code Reusable and Scriptable

In this bonus sheet we accumulate some miscellaneous but important applications of Python to make the code more easily reusable.

Exercise 1. Python Scripts

Up to this point, you've written and run Python code in Jupyter notebooks. This is great for interactive work, especially when you're experimenting, visualizing, or running code locally a small number of times.

However, when your code needs to be run repeatedly, automated, or executed on a computing cluster (e.g. using a job scheduler), it becomes far more practical to run your Python code as a script from the terminal. Scripts are more portable, reproducible, and easier to integrate into pipelines or larger systems.

1. Copy the `accretion_luminosity` function you created in the practical session 5 worksheet into a text file (e.g. using `vim`) and save as `accretion.py`.
2. Save and close the file, change permissions on the file to make it executable, and run it from the command line

   ```
   python accretion.py
   ```

   You shouldn't see any output yet – remember, we haven't called the function!
3. Now below the function add code that computes the accretion luminosity and prints it as was in your `Jupyter` notebook. Save it and run again from the terminal to check it works.

Exercise 2. Creating Python Modules

We may want to reuse parts of our code in other scripts or notebooks. This could be variables, functions, classes etc. We do this by *importing* our code.

1. Create a new Python script called `import_test.py` in the same folder as `accretion.py`, where you will import your `accretion_luminosity` function and use it to do a calculation. You can import code in a few different ways:

   (a) Import all contents

   ```
   import accretion

   L = accretion.accretion_luminosity(....
   ```

   (b) Import all contents with an alias for convenience, this alias must then be used when calling functions

```
import accretion as ac

L = ac.accretion_luminosity(....
```

(c) Import all contents directly to script (Avoid using this unless you are absolutely sure there will be no name clashes. It can lead to unpredictable behavior in larger projects.)

```
from accretion import *

L = accretion_luminosity(....
```

(d) Import a single function

```
from accretion import accretion_luminosity

L = accretion_luminosity(....
```

Choose your preferred method of importing, and add some lines below using the function and printing the result.

2. Save and exit `import_test.py`, change permissions to make it executable and run with

```
python import_test.py
```

What do you notice? You should see this runs the function twice, one from `accretion.py` when it is imported, and one from `import_test.py`.

3. Fix this by adding a block in `accretion.py` such that the code below the function only runs if `accretion.py` is being ran from the command line, but not if it is being imported

```
if __name__ == "__main__":
        # Code here that will only run if it's being ran from the terminal
```

4. Verify this worked by running both `accretion.py` and `import_test.py` and checking each one only runs `accretion_luminosity` once.

5. Check that you can also import code within a `Jupyter` notebook, try creating a blank notebook in the same directory as `accretion.py`, import `accretion_luminosity` and use it in the notebook.