1 The `import` function

- Python, by default, only has access to a small number of built-in types and functions. The vast majority of functions are located in modules, and before a function can be accessed, the module which contains the function must be imported.
- For example, when using `ipython --pylab` (or any variants), a large number of modules are automatically imported, including ***NumPy*** and ***matplotlib***.
- This is style of importing useful for learning and interactive use, but care is needed to make sure that the correct module is imported when designing more complex programs.
- `import` can be used in a variety of ways. The simplest is to use from module `import *` which imports all functions in module.
- You can give each module an "alias" too, using the `as` specifier.

```
import pandas as pd
import numpy as np
import seaborn as sb
```

Caution

- This method of using import can dangerous since if you use it more than once, it is possible for functions to be hidden by later imports.
- A better method is to just import the required functions.
- This still places functions at the top level of the namespace, but can be used to avoid conflicts.

```
    from pylab import log2 # Will import log2 only
    from scipy import log10 # Will not import the log2 from SciPy
```

- The only difference between these two is that `import scipy` is implicitly calling `import scipy as scipy`.
- When this formof import is used, functions are used with the “as” name. For example, the load provided by NumPy is accessed using `sp.log2`, while the pylab load is `pl.log2` – and both can be used where appropriate.
- While this method is the most general, it does require slightly more typing.