Python - Importing Modules

- What is Module?
- Importing Modules
- Absolute Imports
- Relative Imports

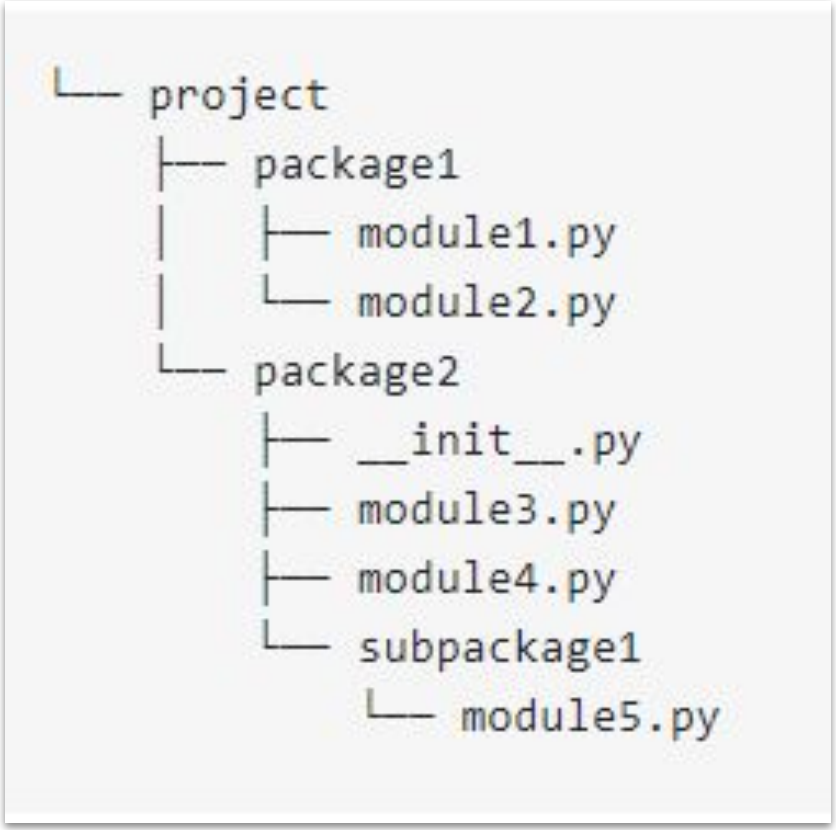
What is Module?

- Module:
  - Is a file (**.py**) that contains a set of definitions (variables and functions) that you can use when they are imported
  - Is considered as objects, just as everything else is in Python
    - Many methods can operate on modules
  - Use **dir()** to list all the objects imported and available to work with in your active Python session

Importing Modules

- There are 2 ways to import modules:
  - **import** ...
    - Example: **import** math
  - **from** … **import** ...
    - Example:
      - **from** math **import** *
      - **from** math **import** pi
- Use **reload(**<module_name>**)** to reload module
  - If you import a module that has already been imported, Python will not re-read the file (even if it has changed!). To reload the module must use **relead()**

Absolute Imports

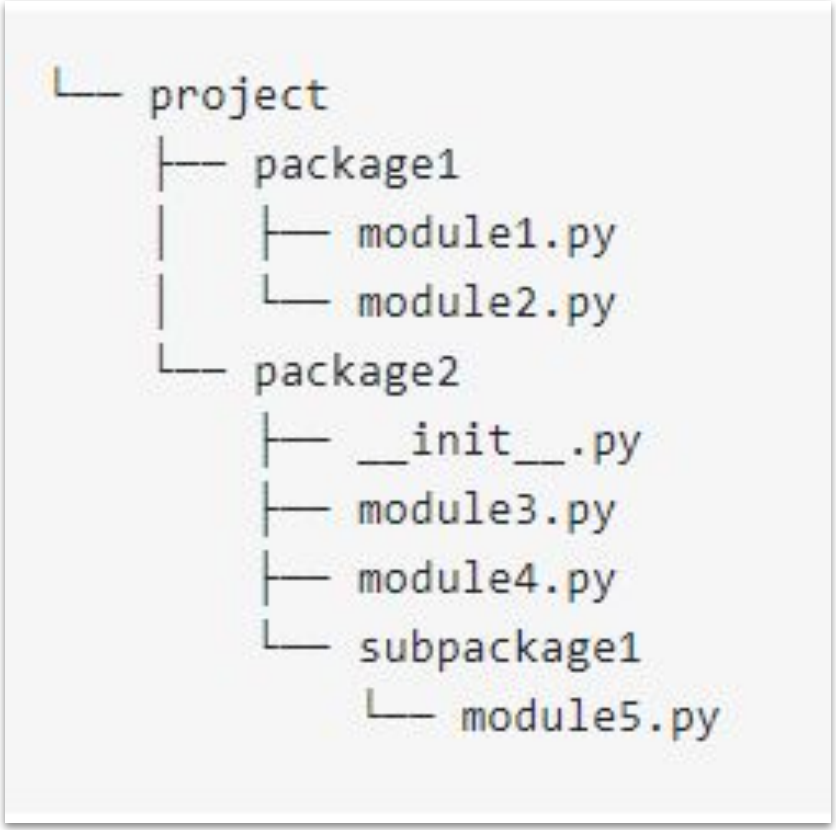


- Let's assume the following:
  - **package1/module2.py**
    - Contains a function, **function1**
  - **package2/__init__.py**
    - Contains a class, **class1**
  - **package2/subpackage1/module5.py**
    - Contains a function, **function2**
- Here is the absolute imports:
  - **from** package1 **import** **module1**
  - **from** package1.module2 **import** **function1**
  - **from** package2 **import** **class1**
  - **from** package2.subpackage1.module5 **import** **function2**

Absolute Imports (2)

- Pros:
  - Absolute Imports are **quite clear** and **straightforward**
  - Absolute imports **remain valid** even if the **current location** of the import statement **changes**
- Cons:
  - Absolute imports can get **quite verbose**, depending on the **complexity of the directory structure**

Relative Imports



- Let's assume the following:
  - **package1/module2.py**
    - Contains a function, **function1**
  - **package2/__init__.py**
    - Contains a class, **class1**
  - **package2/subpackage1/module5.py**
    - Contains a function, **function2**
- Here is the relative imports:
  - In **package1/module1.py**
    - **from .module2 import function1**
  - In **package2/module3.py**
    - **from . import class1**
    - **from .subpackage1.module5 import function2**

Relative Imports (2)

- Pros:
  - Relative Imports are **quite succinct**
- Cons:
  - Relative imports can be **messy**, particularly for **shared projects** where **directory structure** is likely to **change**
  - Relative imports are also **not as readable as absolute ones**

Q & A

Thank You!