Modules and Imports

What we will cover...

1. Modules and files
2. Packages
3. PIP

Files

You've practised writing to a single file, `exercises.py`.

Often, however, we like to split our files into multiple files.

In python, each file is called a **module**.

Creating a module

Creating a module in python is trivial: just create a file!

Imagine you are working on a file called `hello.py` but would like to create a module called `foo` to hold some useful functions.

Simply create another file called `foo.py`.

```
.
├── hello.py
└── foo.py
```

Using modules

Let's put a simple function in our new module `foo`:

```
# foo.py

def greet(name):
    return 'Hey ' + name
```

Using modules

Now we can **import** this function into our `hello.py` file and use it:

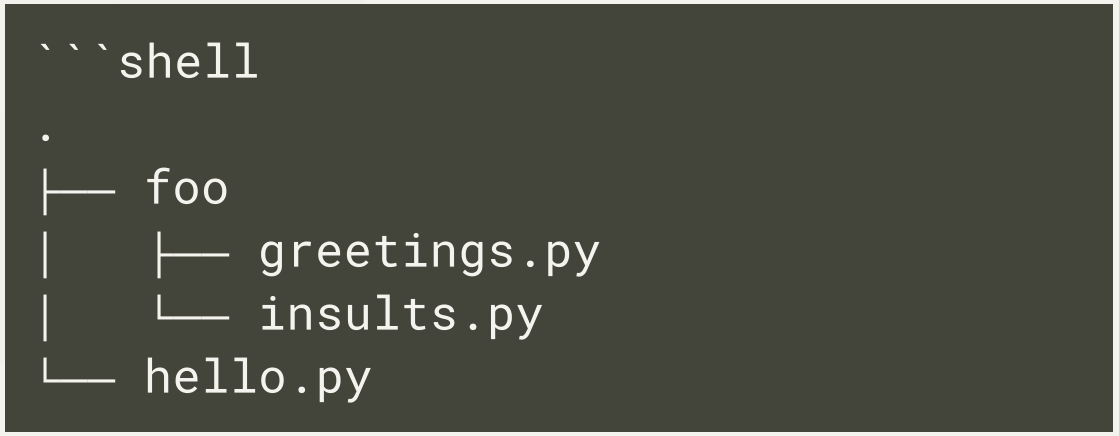
1. We can import only the variable we want with the `from` syntax.
2. We can import the entire module with `import foo`. The variables in the module are then accessible via dot `.` notation.

```
# hello.py

from foo import greet
greet('nandan')

import foo
foo.greet('nandan')
```

Packages

Packages are collections of modules. Creating a local package in python is also easy:

1. Create a folder and put some `.py` files in the folder.
2. Done!



Packages

Technically, you should put a an empty file called `__init__.py` that makes it explicit to everyone that this is meant to be used as a package.

However, this is usually not necessary for simple projects.

```shell
.
├── foo
│   ├── greetings.py
│   ├── __init__.py
│   └── insults.py
└── hello.py
```

Packages

Now we can **import** a function from a module inside a package:

```
# hello.py

from foo.greetings import greet
greet('nandan')

from foo import greetings
greetings.greet('nandan')

import foo.greetings
foo.greetings.greet('nandan')
```

Standard modules

You've used some builtin functions in python, variables that are availabe in the **global scope** at all times (like `len`).

But python comes with many variables and functions that are not always available. You need to import them explicitly from their modules.

These modules, however, are always available.

Standard modules

You can import standard modules the same way you import your own modules.

You should always be careful not to name your own modules the same name as standard modules!

```
from math import factorial

factorial(5)
```

PIP

Sometimes, we want to share packages across projects and with other people.

Python has a "global" package database that allows everyone to share their packages with each other.

That global package database is accessed via a **package manager** called `pip`.

PIP

```
python3 -m pip install requests
```

You've already used this package manager to install packages on your computer.

But maybe you didn't notice that you can import any package installed on your computer into a python file!

Using packages

Using a package installed via `pip` is just like using a local package in a folder. When you try to `import` a package or module, python searches:

1. The folder in which the file is found.
2. The folder given by the `PYTHONPATH` environment variable.
3. A particular folder created by the python installer on your machine.

```
import requests

requests.get('https://google.com')
```

Review

1. Modules and files
2. Packages
3. PIP