Python Modules and Packages

Python Module

In programming, a module is a piece of software that has a specific functionality. For example, when building a ping pong game, one module would be responsible for the game logic, and another module would be responsible for drawing the game on the screen. Each module is a different file, which can be edited separately.

Python modules are simply Python files with a `.py` extension. The name of the module will be the name of the Python file. `test.py` is a module, `telephony.py` is a module, `common.py` is a module, etc.

Python Package

Python packages are folders that **can** contain sub-packages and modules. Every Python package is essentially a folder which **must** contain a special file called `__init__.py` which can be left empty, and it indicates that the folder which it resides is a Python package.

If you want to create a Python package, you simply have to create a new folder (e.g. `package1`) and a file named `__init__.py` in the folder. The `__init__.py` file does not need to contain any code; it simply exists to tell Python that the folder which it resides is going to be a Python package.

After creating a new Python package named `package1` and the `__init__.py` file, you can also add modules to the package – if you create a Python file called `module1.py` in the `package1` folder, you are essentially creating a module for the package.

Examples of Modules and Packages

The Enterprise's scripts folder `U50A+ATT_Enterprise` has a folder called `lib`. Within the `lib` folder, there is a Python file called `__init__.py`. The `__init__.py` file designates that the `lib` folder is a Python package. Also, the other Python files within the `lib` folder such as `cfg.py` and `util.py` are all modules.

In order for us to use the modules within the `lib` folder, we must import the modules in the following way:

```python
from lib import cfg
from lib import util
```

The way that the import statements are written depends on the location of the Python file from which you are importing them. Let us imagine that we have a folder that contains the `lib` folder and some test case files:

- `U50A+ATT_Enterprise`
  - `lib`
    - `__init__.py`
    - `cfg.py`
    - `util.py`
  - `01_Messaging.py`

If you wish to use the `cfg` and the `util` modules in `01_Messaging.py`, you need to write the following code in the `01_Messaging.py` file:

```
from lib import cfg
from lib import util
```

If you wish to use the `util` module from the `cfg` module (both modules are inside the `lib` folder), then you need to import the `util` module in the following way within the `cfg` module:

```
import util
```

Notice that the way the import statements are written depends on the location of the module within which you are writing the import statements. In other words, the relative location of the modules will dictate how you write your import statements.

Main Takeaways

> A Python module is nothing more than just a Python file (e.g. `foo.py`) with the `.py` extension.

> A Python package is an ordinary folder that has a file called `__init__.py` which does not need to have any code.

> The name of the Python module is the file name of the `.py` file. For instance, `foo.py` module's name is `foo`.

> The name of the Python module is the name of the folder that has the `__init__.py` file.