Modules and Packages

MODULES

- If we have two python programs in the same folder, and we want one to use methods and attributes from the other, all we need to do is say:

```
import <Filename>
```

Modules are files, Packages are folders

- To use specific class from a program (module) in the same folder, we can say:
  - `from <Filename> import <Classname>`
- If there already is a class in the calling program as the name of the class we want to import, we can import it with an alias:
  - `from <Filename> import <Classname> as <Alias>`
- If we want to import two classes from the same program (module):
  - `from <Filename> import <Classname1>, <Classname2>`
- If we want to import all the classes (which isn't really a good idea), we can say:
  - `from <Filename> import *`

PACKAGES

- If we feel there are too many programs (modules) in the current folder, we can create a sub-folder (package) to put some in that. Then to make it possible to access the programs in that subfolder we need to add a blank textfile into the subfolder called the following:

```
__init__.py
```

- To import a full file from that subfolder:
  - `import <SubFolder>.<Filename>`
- To import a specific class from a file in that subfolder:
  - `from <SubFolder>.<Filename> import <Classname>`
- We can also say:
  - `from <SubFolder> import <Filename>`

Modules and Packages

Tips for Using Modules and Packages

We can use the full stop "." to refer to the current directory.
We can put text in the `__init__.py` file to import code directly from a package as opposed to a module.
The "global" command allows us to create a global variable, and we can make changes to that variable in a local context.
If we want to check if a program is being called from another program, or is being run as a stand-alone program, we can say: `if __name__ == "__main__":` If this is true, it means the program is being called as a stand-alone.

These allow us to get the most out of modules and packages.

Access Control

Many object-oriented programming languages allow the programmer to specific the level of access to the methods and attributes of an object, using *Public*, *Protected*, and *Private*. Python does not have an equivant access control mechnaism, but by convention a method or attribute with no underscores at the start of its name is consdiered to be public (any object can access it), a method or attribute with a single underscore (_) at the start of its name indcates it is protected (the object itself and subclasses can access it), and a method or attribute with a double underscore (__) at the start of its name indcates it is private (only the object can access it), but remember there is nothing in the interpreter to stop external objects from access these methods or attributes.

We can also add a comment in the *docstrings* at the start of the class indicating which methods or attributes are for internal use only.

Third-Party Libraries

Python comes with a very big Standard Library with lots of features, but we may be looking for a feature that it doesn't have, if so we have two options; we can write a new package ourselves, or we can use somebody else's code.

If we want to find packages that might be of use, we can use the Python Package Index (PyPI) website: `http://pypi.python.org`.

Once we've found a package that we want to install, we can use the `pip` tool to install it. Pip doesn't come with the Python download, but we can download it from here: `https://pypi.python.org/pypi/pip`

And once pip has been installed, we can install the software into our library using:

- `pip install <package>`