Python Modules

You can extend the capabilities of your Python program by leveraging, or "importing" other files of code called "modules".

Some selected Python modules of interest include:

- The `csv` Module
- The `datetime` Module
- The `itertools` Module
- The `json` Module
- The `math` Module
- The `os` Module
- The `random` Module
- The `statistics` Module
- The `time` Module
- The `webbrowser` Module

You can also create and import your own modules to help you organize your code into separate logical files.

For more details, follow along with this official tutorial on modules:

- https://docs.python.org/3/tutorial/modules.html#modules
- https://docs.python.org/3/tutorial/modules.html#more-on-modules
- https://docs.python.org/3/tutorial/modules.html#executing-modules-as-scripts

Usage

To load any module, whether a built-in module or a custom module you create, use the `import` statement. Then after importing the module, you can reference code contained within.

To see this concept in action, create a new directory on your computer called "modules-overview" and place inside the following two files...

Script:

```python
# modules-overview/my_script.py

import my_module

print("IMPORTING MY MODULE ...")
my_module.my_message()
```

Module:

```
# modules-overview/my_module.py

# anything in the global scope of this file will be executed immediately when
the module is imported.
# ... so we generally wrap all the code inside separate functions, which can
later be invoked as desired.

def my_message():
    print("HELLO FROM A MODULE")

def other_message():
    print("GREETINGS EARTHLING")

# but if we want something to happen when the module is invoked directly from
the command line (as a script)
# ... we can use this special conditional to detect that use case and perform
instructions as desired.
if __name__ == "__main__":
    print("INVOKING MY MODULE AS A SCRIPT...")
    my_message()
```

Then execute the script to prove it has access to code in the module:

```
python my_script.py
#> IMPORTING MY MODULE ...
#> HELLO FROM A MODULE
```

It is also possible to execute the module directly:

```
python my_module.py
#> INVOKING MY MODULE AS A SCRIPT...
#> HELLO FROM A MODULE
```

Modules in Subdirectories

If your python file is located in a subdirectory, you can reference it using the `[directory name].[file name]`. Like this:

```
# modules-overview/things/robot.py

def robot_message():
    print("HELLO I'M A ROBOT")
```

```python
# modules-overview/robot_script.py

import things.robot as bot

bot.robot_message()
```

```
python robot_script.py
#> HELLO I'M A ROBOT
```