`__main__` — Top-level script environment

`'__main__'` is the name of the scope in which top-level code executes. A module's __name__ is set equal to `'__main__'` when read from standard input, a script, or from an interactive prompt.

A module can discover whether or not it is running in the main scope by checking its own `__name__`, which allows a common idiom for conditionally executing code in a module when it is run as a script or with `python -m` but not when it is imported:

```
if __name__ == "__main__":
    # execute only if run as a script
    main()
```

For a package, the same effect can be achieved by including a `__main__.py` module, the contents of which will be executed when the module is run with `-m`.

Source: https://www.freecodecamp.org/news/if-name-main-python-example/

Python files are called modules and they are identified by the `.py` file extension. A module can define functions, classes, and variables.
So when the interpreter runs a module, the `__name__` variable will be set as `__main__` if the module that is being run is the main program.

But if the code is importing the module from another module, then the `__name__` variable will be set to that module's name.

__init__

self

The word 'self' is used to represent the instance of a class. By using the "self" keyword we access the attributes and methods of the class in python.

__init__ method

"__init__" is a reseved method in python classes. It is called as a constructor in object oriented terminology. This method is called when an object is created from a class and it allows the class to initialize the attributes of the class.