When your script is run by passing it as a command to the Python interpreter,

python myscript.py

all of the code that is at indentation level 0 gets executed. Functions and classes that are defined are, well, defined, but none of their code gets run. Unlike other languages, there's no main()function that gets run automatically - the main() function is implicitly all the code at the top level.

In this case, the top-level code is an if block. \_\_name\_\_ is a built-in variable which evaluates to the name of the current module. However, if a module is being run directly (as in myscript.py above), then \_\_name\_\_ instead is set to the string "\_\_main\_\_". Thus, you can test whether your script is being run directly or being imported by something else by testing

if \_\_name\_\_ == "\_\_main\_\_":

...

If your script is being imported into another module, its various function and class definitions will be imported and its top-level code will be executed, but the code in the then-body of the if clause above won't get run as the condition is not met. As a basic example, consider the following two scripts:

# file one.py

def func():

print("func() in one.py")

print("top-level in one.py")

if \_\_name\_\_ == "\_\_main\_\_":

print("one.py is being run directly")

else:

print("one.py is being imported into another module")

# file two.py

import one

print("top-level in two.py")

one.func()

if \_\_name\_\_ == "\_\_main\_\_":

print("two.py is being run directly")

else:

print("two.py is being imported into another module")

Now, if you invoke the interpreter as

python one.py

Now, if you invoke the interpreter as

python one.py

The output will be

top-level in one.py

one.py is being run directly

If you run two.py instead:

python two.py

You get

top-level in one.py

one.py is being imported into another module

top-level in two.py

func() in one.py

two.py is being run directly

Thus, when module one gets loaded, its \_\_name\_\_ equals "one" instead of "\_\_main\_\_".