Get unlimited access Open in app

Published in CodeX

Ishaan Gupta Follow
May 12 · 3 min read · Listen

Save

What is __name__ == “__main__” in Python?

When and how the main method is executed in Python and what does it means?

Photo by asier_relampagoestudio on freepik

You can now subscribe to get stories delivered directly to your inbox.
Got it

If you are new to Python, you might have noticed that it is possible to run a Python script with or without a main method. But now you be wondering why do we need it then.

In this story, I am going to explain what is the use of the main method and what happens when you define it.

What does the if __name__ == “__main__”: do?

Before executing the code, the Python interpreter reads the source file and defines a few special variables/global variables. If the python interpreter is running that module (the source file) as the main program, it sets the special `__name__` variable to have a value `“__main__”`. If this file is being imported from another module, `__name__` will be set to the **module’s name.** The module’s name is available as value to `__name__` global variable.

A module is a file containing Python definitions and statements. The file name is the module name with the suffix .py appended.

When we execute a file as the command to the python interpreter,

```
python follow.py

print ("Executed")

if __name__ == "__main__":
 print ("Executed when invoked directly")
else:
 print ("Executed when imported")
```

- All of the code that is at indentation level 0 [Block 1] gets executed. Functions and classes that are defined are, well, defined, but none of their code runs.
- Here, as we executed script.py directly `__name__` variable will be `__main__`. So, code in this if block[Block 2] will only run if that module is the entry

Get unlimited access Open in app

- If the script is getting imported by some other module at that time **__name__** will be module name.

Why Do we need it?

For example, we are developing a script that is designed to be used as a module:

```
# Python program to execute function directly

def my_function():
 print ("I am inside function")

# We can test function by calling it.
my_function()
```

Now if we want to use that module by importing we have to comment out our call. Rather than that approach best approach is to use the following code:

```
# Python program to use main for function call.

if __name__ == "__main__":
 my_function()

import myscript

myscript.my_function()
```

Advantages :

1. Every Python module has its `__name__` defined and if this is `'__main__'`, it implies that the module is being run standalone by the user and we can do corresponding appropriate actions.
2. If you import this script as a module in another script, the `__name__` is set to the name of the script/module.
3. Python files can act as either reusable modules or as standalone programs.
4. `if __name__ == "main":` is used to execute some code **only** if the file was run directly, and not imported.

Final Thoughts

Well, hopefully now you know the use of `__name__ == "__main__"` in Python. I hope you find this article helpful and have learned some new things. Share this article with your Pythoneer Friends.

Happy Coding!

Sign up for CrunchX
By CodeX
A weekly newsletter on what's going on around the tech and programming space Take a look.
Get this newsletter
Emails will be sent to cesarnr21@gmail.com.
Not you?