Creating and Working with Modules

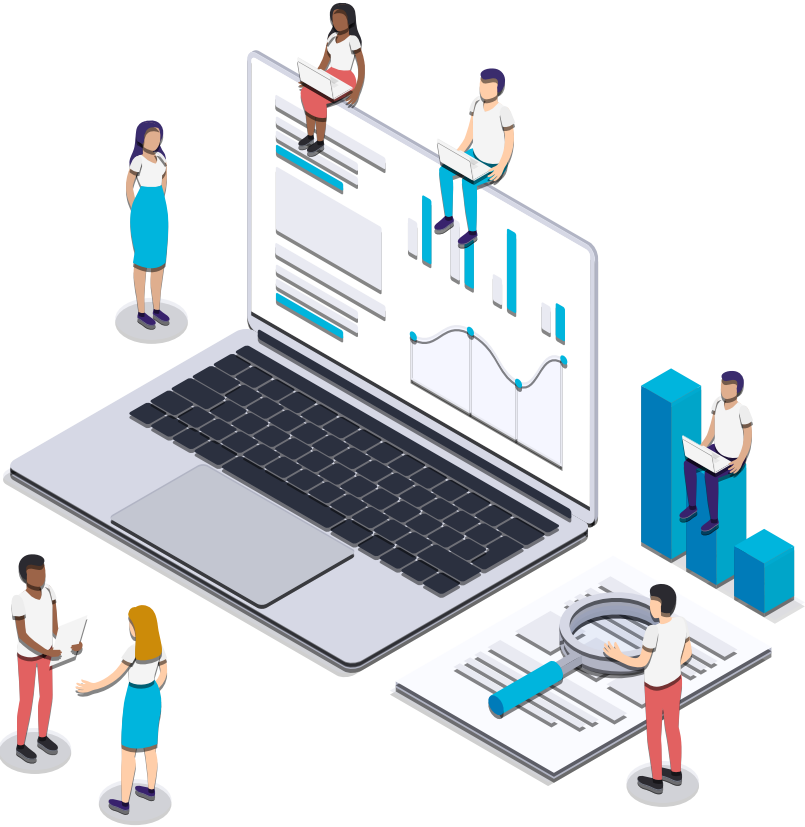

Table of Contents

1

Scripts & Modules Initialization

Can you recap the difference between **scripts** and **modules**?

Students, write your response!

Scripts & Modules Initialization

Recap

- Scripts and modules have essentially identical structures *in terms of creation* and are the files with a **.py** extension, containing some Python codes, statements, operations, and functions.

Scripts & Modules Initialization

- In fact, if you're using an advanced IDE/IDLE, such as Jupyter Notebook/Lab (which we are) or Python IDLE, all these issues about the scripts and the modules don't make much sense. So, these applications have a user-friendly menu on such issues.

💡**Tips:**

- When using Jupyter Lab / Notebook, you will almost always work with files with a **.ipynb** extension.

Scripts & Modules Initialization

- **Task :**
  - Create a **file** named `my_first` with **.py** extension containing of two simple user-defined *functions* and some *statements*.
  - Use it as a script and as a module.
  - Call some functions&variables and use it from your module.
  - Display the docstring of your module.

Scripts & Modules Initialization

- You can see the current path of your Jupyter using **`pwd`** command.

```
In [4]:  1  pwd

Out[4]:  'C:\\Users\\YD'
```

2

Working with the Modules (*Optional*)

Acting of a Module as a Script

optional

hello.py - C:\Users\Dell\AppData\Local\Programs\Python\Python38-32\hello.py (3.8.0)

File Edit Format Run Options Window Help

```
""" this is my first module & script """

print('hello')

def my_func1(x):
    return print(x**2)

my_func1(3)

def my_func2(y):
    return print(*y)

my_func2("clarusway")
```

Python 3.8.0 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [
MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more
information.
>>> import hello
hello
9
c l a r u s w a y
>>> hello.my_func1(2)
4
>>> hello.my_func2('confidence')
c o n f i d e n c e
>>> hello.__doc__   # we can display docstring of the module
' this is my first module & script '
>>>
```

Ln: 13 Col: 4

output of all statements inside the module are displayed

How can we solve this issue?

Students, write your response!

Acting of a Module as a Script

optional

- As you see, when you want to import this file as a module, it acts as a script for the first importing, which is undesirable. It is not normal for a module to generate output when imported. Then why it happens?
- Well. As a Pythonic rule, when the file you created with **.py** extension is imported as a module, Python sets the specific variable `__name__` to the name of the module. But, if the file is run as a *script*, variable `__name__` is set to the string value of "`__main__`". So, using this Pythonic rule, we can fix this issue.

__name__, "__main__" Method

optional

- If we collect the output-generating statements which are in our module under if __name__ == "__main__" : statement we will solve the problem. Let's do it and see what will happen :

hello.py :

```python
""" this is my first module & script """

def my_func1(x):
    return print(x**2)

def my_func2(y):
    return print(*y)

if __name__ == '__main__':  # output-generating statements are here
    print('hello')
    my_func1(3)
    my_func2("clarusway")
```

Working with the Modules

optional

- Let's run it on the Command Prompt (console) as a ***script*** :

Command Prompt

```
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Dell>cd C:\Program Files\Python37

C:\Program Files\Python37>python hello.py
hello
9
c l a r u s w a y

C:\Program Files\Python37>
```

Working with the Modules

optional

- Let's run it on the Command Prompt (console) as a ***module*** :

Command Prompt - python

```
C:\Users\Dell>cd C:\Program Files\Python37

C:\Program Files\Python37>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.19
14 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more informa
tion.
>>> import hello
>>> hello.my_func1(4)
16
>>> hello.my_func2('acting as a module')
a c t i n g   a s   a   m o d u l e
>>>
```

THANKS!

Any questions?

You can find me at:

- @joseph
- joseph@clarusway.com