Python Programming

RGM College of Engineering & Technology (Autonomous)

Department of Computer Science & Engineering

Academic Year : 2020-2021

MODULES - 2

Guido Van Rossum

Learning Mantra

If you really strong in the basics, then remaining things will become so easy.

Agenda:

1. Reloading a Module
2. Finding members of module by using 'dir()' function
3. The Special variable '_ name _'

7. Reloading a Module

By default, module will be loaded only once even though we are importing multiple times.

Eg: Demo Program for module reloading

Assume that we created a module named as **module1.py**.

```
print('This is from module 1')
```

Now, we want to use **module1.py** in another module **test.py**.

```
import module1

import module1

import module1

import module1

import module1

import module1

print('This is Test Module')        # Executed in Editplus editor
```

Output

```
This is from module 1

This is Test Module
```

Eg:

```
import module1
'''import module1
import module1
import module1
import module1
import module1'''
print('This is Test Module')          # Executed in Editplus editor
```

Output:

```
This is from module 1
This is Test Module
```

In the above program test module will be loaded only once even though we are importing multiple times.

- The problem in this approach is, after loading a module, if it is updated outside then updated version of module1 is not available to our program.

Eg:

```
import time
import module1          # importing original version of module1
print("Program entering into sleeping state ")
time.sleep(30)          # during this time we want to modify something to module1
import module 1 # After 30 seconds we are importing module1, is it going to import updated module1
print("This is last line of program")          # Executed in Editplus editor
```

Output

```
This is from module 1

Program entering into sleeping mode

                                                # After 30 seconds

This is last line of program # Updated version is not available.
```

- We can solve this problem by reloading module explicitly based on our requirement. We can reload by using **reload()** function of **importlib** module.

```
import importlib
importlib.reload(module1)
```

Eg:

```
import time
from importlib import reload
import module1
print("program entering into sleeping state")
time.sleep(30)                                  # It is not mandatory
reload(module1)
print("This is last line of program")
```

Output:

```
This is from module 1                           # original version of module1
Program entering into sleeping mode

                                                # After 30 seconds
This is from updated module 1                   # during this time module1 is updated
This is last line of program                    # Updated version of module1 is now available
```

Note: In the above program, every time updated version of module1 will be available to our program.

- The main advantage of explicit module reloading is we can ensure that updated version is always available to our program.

8. Finding members of module by using 'dir()' function

- Python provides inbuilt function **dir()** to list out all members of current module or a specified module.
- **dir()** ➔ To list out all members of current module.
- **dir(moduleName)** ➔ To list out all members of specified module.

Eg 1: To display members of current module.

```
x=10
y=20
def f1():
    print("Hello")
print(dir())        # To print all members of current module
```

Output:

```
['In', 'Out', '_', '__', '___', '__builtin__', '__builtins__', '__doc__', '_
_loader__', '__name__', '__package__', '__spec__', '_dh', '_i', '_i1', '_i
h', '_ii', '_iii', '_oh', 'exit', 'f1', 'get_ipython', 'quit', 'x', 'y']
```

Eg 2: To display members of particular module.

Consider **rgm.py** module,

```
x = 888
y = 999
def add(a,b):
    print('The Sum :',a+b)
def product(a,b):
    print('The Product :',a*b)
```

import rgm module in another module, called as test.py,

```
import rgm
print(dir(rgm))
```

Output

```
['builtins', 'cached', 'doc', 'file', 'loader', 'name', 'package', 'spec', 'add', 'product', 'x', 'y']
```

Note:

- For every module at the time of execution Python interpreter will add some special properties automatically for internal use.

Eg: _ builtins ,_ _cached ,' doc ,_ _file , _ _loader , _ _name ,_ _package , _ _spec _

Based on our requirement we can access these properties also in our program.

Eg:

```
print(__builtins__ )
print(__cached__ )
print(__doc__)
print(__file__)
print(__loader__)
print(__name__)
print(__package__)
print(__spec__)
```

Output

```
<module 'builtins' (built-in)>
None
None
test.py
<frozenimportlib_external.SourceFileLoader object at 0x000001C8488D2640>
main
None
None
```

9. The Special variable '_ name _'

- For every Python program , a special variable _ **name**_ will be added internally. This variable stores information regarding whether the program is executed as an individual program or as a module.
- If the program executed as an individual program then the value of this variable is **_ main_.**
- If the program executed as a module from some other program then the value of this variable is the name of module where it is defined.
- Hence by using this _ **name**_ variable we can identify whether the program executed directly or as a module.

Demo program:

module1.py

```
def f1():
    if __name__=='__main__':
        print("The code executed directly as a program")
        print("The value of __name__:",__name__)
    else:
        print("The code executed indirectly as a module from some other program")
        print("The value of __name__:",__name__)
f1()
```

Output:

```
The code executed directly as a program
The value of __name__: __main__
```

test.py

```
import module1
print("From test we are executing module f1()")
module1.f1()                    # Executed in Edit plus editor
```

Output:

```
The code executed indirectly as a module from some other program
The value of name: module1
From test we are executing module f1()
The code executed indirectly as a module from some other program
The value of name: module1
```

Any question?

If you try to practice programs yourself, then you will learn many things automatically

Spend few minutes and then enjoy the study

Thank You