Python Programming

RGM College of Engineering & Technology (Autonomous)

Department of Computer Science & Engineering

Academic Year : 2020-2021

MODULES - 1

UNIT – VI

Modules: Creating modules, import statement, from Import statement.

Topics to be Covered:

1. Introduction
2. Advantages of Modules
3. Renaming a module at the time of import (module aliasing)
4. from … Import
5. member aliasing
6. Various possibilities of import statement
7. Reloading a Module
8. Finding members of module by using 'dir()' function
9. The Special variable '_ name _‘
10. Working with math module
11. Working with random module
12. Example programs

Guido Van Rossum

Learning Mantra

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

Agenda:

1. Introduction
2. Advantages of Modules
3. Renaming a module at the time of import (module aliasing)
4. from … Import
5. member aliasing

INTRODUCTION

MODULE

- A group of functions, variables and classes saved to a file, which is nothing but module.
- Every Python file (**.py**) acts as a module.

Eg:

```
x = 888
y = 999
def add(a,b):        # Use Edit plus (OR) Atom editor
    print('The Sum : ',a+b)
def product(a,b):
    print('The product :', a*b)
```

Let me save this code as **rgm.py**, and itself is a module.

- **rgm.py** module contains **two variables** and **2 functions**.
- If we want to use members of module in our program then we should import that module.

Syntax of importing a module:

import modulename

- We can access members by using module name.

modulename.variable

modulename.function()

Eg:

```
import rgm

print(rgm.x)

rgm.add(10,20)              # Executed in Editplus & Atom editor

rgm.product(10,20)
```

Output:

```
888

The Sum : 30

The Product : 200
```

Note:

- Whenever we are using a module in our program, for that module compiled file will be generated and stored in the hard disk permanently.
- This is available at _ _ **pycache**_ _ file, which is available at current working directory.

2.Advantages of Modules

1. Code Reusability
2. Readability improved
3. Maintainability improved

3. Renaming a module at the time of import (module aliasing)

- We can create alias name for a module. This can be done as follows:

import rgm as r

- Here, **rgm** is original module name and **r** is alias name. We can access members by using alias name **r.**

Eg:

```
import rgm as r
print(r.x)
r.add(10,20) # Executed in Editplus editor
r.product(10,20)
```

Output

```
888
The Sum : 30
The Product : 200
```

Eg:

```
import rgm as r

print(rgm.x)

rgm.add(10,20)      # Executed in Editplus editor

rgm.product(10,20)
```

Output

```
Traceback (most recent call last):

File "test.py", line 2, in

    print(rgm.x)

NameError: name 'rgm' is not defined
```

Note:

- Once we define alias name for a module, compulsory you should use alias name only. Original names by default will be gone related to that particular file.

4. from … import

- We can import particular members of module by using from … import.
- The main advantage of this is we can access members directly without using module name.

Eg:

```
from rgm import x,add
print(x)
add(10,20) # Executed in Editplus editor
product(10,20)
```

Output

```
888
The Sum : 30
Traceback (most recent call last):
File "test.py", line 4, in
    product(10,20)
NameError: name 'product' is not defined
```

- **We can import all members of a module as follows,**

from rgm import *

Eg :

```
from rgm import *

print(x)

add(10,20)          # Executed in Editplus editor

product(10,20)
```

Output

```
888

The Sum : 30

The Product : 200
```

5. member aliasing

Similar to module aliasing, member aliasing also possible in python. This can be done as follows:

```
from rgm import x as y,add as sum
print(y)
sum(10,20)
```

Note: Once we defined as alias name, we should use alias name only and we should not use original name.

Eg:

```
from rgm import x as y

print(x)
```

Output

```
Traceback (most recent call last):

File "test.py", line 2, in

    print(x)

NameError: name 'x' is not defined
```

6. Various possibilities of ‘import’ statement

- import module_name
- import module1,module2,module3
- import module1 as m
- import module1 as m1,module2 as m2,module3
- from module import member
- from module import member1,member2,memebr3
- from module import memeber1 as x
- from module import *

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