Python Module

Modules are used to categorize code in Python into smaller part. A module is simply a file, where classes, functions and variables are defined. Grouping similar code into a single file makes it easy to access.

**Have a look over example:**

If the content of a book is not indexed or categorized into individual chapters, then the book might have turned boring and hectic. Hence, dividing book into chapters made it easy to understand.

In the same sense python modules are the files which have similar code. Thus module is simplify a python code where classes, variables and functions are defined.

**Advantage:**

Python provides the following advantages for using module:

**1) Reusability:** Module can be used in some other python code. Hence it provides the facility of code reusability.

**2) Categorization:** Similar type of attributes can be placed in one module.

Importing a Module:

There are different ways by which you we can import a module. These are as follows:

1) Using import statement:

"import" statement can be used to import a module.

**Syntax:**

**import** <file\_name1, file\_name2,...file\_name(n)="">

</file\_name1,>

Have a look over an example:

eg:

**def** add(a,b):

    c=a+b

**print** c

**return**

Save the file by the name addition.py. To import this file "import" statement is used.

**import** addition

addition.add(10,20)

addition.add(30,40)

Create another python file in which you want to import the former python file. For that, import statement is used as given in the above example. The corresponding method can be used by file\_name.method (). (Here, addition. add (), where addition is the python file and add () is the method defined in the file addition.py)

Output:

1. >>>
2. 30
3. 70
4. >>>

NOTE: You can access any function which is inside a module by module name and function name separated by dot. It is also known as period. Whole notation is known as dot notation.

Example of importing multiple modules:

**Eg:**

**1) msg.py:**

**def** msg\_method():

**print** "Today the weather is rainy"

**return**

**2) display.py:**

**def** display\_method():

**print** "The weather is Sunny"

**return**

**3) multiimport.py:**

**import** msg,display

msg.msg\_method()

display.display\_method()

Output:

1. >>>
2. Today the weather **is** rainy
3. The weather **is** Sunny
4. >>>

2) Using from.. import statement:

from..import statement is used to import particular attribute from a module. In case you do not want whole of the module to be imported then you can use from ?import statement.

Syntax:

**from**  <module\_name> **import** <attribute1,attribute2,attribute3,...attributen>

</attribute1,attribute2,attribute3,...attributen></module\_name>

Have a look over the example:

**1) area.py**

**Eg:**

**def** circle(r):

**print** 3.14\*r\*r

**return**

**def** square(l):

**print** l\*l

**return**

**def** rectangle(l,b):

**print** l\*b

**return**

**def** triangle(b,h):

**print** 0.5\*b\*h

**return**

**2) area1.py**

1. **from** area **import** square,rectangle
2. square(10)
3. rectangle(2,5)

**Output:**

1. >>>
2. 100
3. 10
4. >>>

3) To import whole module:

You can import whole of the module using "from? import \*"

**Syntax:**

**from** <module\_name> **import** \*

</module\_name>

Using the above statement all the attributes defined in the module will be imported and hence you can access each attribute.

**1) area.py**

Same as above example

**2) area1.py**

1. **from** area **import** \*
2. square(10)
3. rectangle(2,5)
4. circle(5)
5. triangle(10,20)

**Output:**

1. >>>
2. 100
3. 10
4. 78.5
5. 100.0
6. >>>

Built in Modules in Python:

There are many built in modules in Python. Some of them are as follows:

math, random , threading , collections , os , mailbox , string , time , tkinter etc..

Each module has a number of built in functions which can be used to perform various functions.

Let?s have a look over each module:

1) math:

Using math module , you can use different built in mathematical functions.

**Functions:**

|  |  |
| --- | --- |
| **Function** | **Description** |
| ceil(n) | Returns the next integer number of the given number |
| sqrt(n) | Returns the Square root of the given number. |
| exp(n) | Returns the natural logarithm e raised to the given number |
| floor(n) | Returns the previous integer number of the given number. |
| log(n,baseto) | Returns the natural logarithm of the number. |
| pow(baseto, exp) | Returns baseto raised to the exp power. |
| sin(n) | Returns sine of the given radian. |
| cos(n) | Returns cosine of the given radian. |
| tan(n) | Returns tangent of the given radian. |

**Useful Example of math module:**

**Eg:**

**import** math

a=4.6

**print** math.ceil(a)

**print** math.floor(a)

b=9

**print** math.sqrt(b)

**print** math.exp(3.0)

**print** math.log(2.0)

**print** math.pow(2.0,3.0)

**print** math.sin(0)

**print** math.cos(0)

**print** math.tan(45)

**Output:**

1. >>>
2. 5.0
3. 4.0
4. 3.0
5. 20.0855369232
6. 0.69314718056
7. 8.0
8. 0.0
9. 1.0
10. 1.61977519054
11. >>>

**Constants:**

The math module provides two constants for mathematical Operations:

|  |  |
| --- | --- |
| **Constants** | **Descriptions** |
| Pi | Returns constant ? = 3.14159... |
| ceil(n) | Returns constant e= 2.71828... |

**Eg:**

1. **import** math
3. **print** math.pi
4. **print** math.e

**Output:**

1. >>>
2. 3.14159265359
3. 2.71828182846
4. >>>

**2) random:**

The random module is used to generate the random numbers. It provides the following two built in functions:

|  |  |
| --- | --- |
| **Function** | **Description** |
| random() | It returns a random number between 0.0 and 1.0 where 1.0 is exclusive. |
| randint(x,y) | It returns a random number between x and y where both the numbers are inclusive. |

**Eg:**

**import** random

**print** random.random()

**print** random.randint(2,8)

**Output:**

1. >>>
2. 0.797473843839
3. 7
4. >>>

Other modules will be covered in their respective topics.

Package

A Package is simply a collection of similar modules, sub-packages etc..

Steps to create and import Package:

1) Create a directory, say Info

2) Place different modules inside the directory. We are placing 3 modules msg1.py, msg2.py and msg3.py respectively and place corresponding codes in respective modules. Let us place msg1() in msg1.py, msg2() in msg2.py and msg3() in msg3.py.

3) Create a file \_\_init\_\_.py which specifies attributes in each module.

4) Import the package and use the attributes using package.

Have a look over the example:

**1) Create the directory:**

**import** os

os.mkdir("Info")

**2) Place different modules in package: (Save different modules inside the Info package)**

**msg1.py**

**def** msg1():

**print** "This is msg1"

**msg2.py**

**def** msg2():

**print** "This is msg2"

**msg3.py**

1. **def** msg3():
2. **print** "This is msg3"

**3) Create \_\_init\_\_.py file:**

**from** msg1 **import** msg1

**from** msg2 **import** msg2

**from** msg3 **import** msg3

**4)Import package and use the attributes:**

**import** Info

Info.msg1()

Info.msg2()

Info.msg3()

**Output:**

1. >>>
2. This **is** msg1
3. This **is** msg2
4. This **is** msg3
5. >>>

What is \_\_init\_\_.py file? \_\_init\_\_.py is simply a file that is used to consider the directories on the disk as the package of the Python. It is basically used to initialize the python packages.