

**Modules**

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: durgamath.py

|  |  |
| --- | --- |
| **1)** | **x=888** |
| **2)** | |
| **3)** | **def add(a,b):** |
| **4)** | **print("The Sum:",a+b)** |
| **5)** | |
| **6)** | **def product(a,b):** |
| 7) | **print("The Product:",a\*b)** |

durgamath module contains one variable and 2 functions.

If we want to use members of module in our program then we should import that module. import modulename

We can access members by using module name. modulename.variable

modulename.function() test.py:

|  |  |
| --- | --- |
| **1)** | **import durgamath** |
| **2)** | **print(durgamath.x)** |
| **3)** | **durgamath.add(10,20)** |
| **4)** | **durgamath.product(10,20)** |
| **5)** | |
| **6)** | **Output** |
| **7)** | **888** |
| **8)** | **The Sum: 30** |
| **9)** | **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.

## Renaming a module at the time of import (module aliasing):

Eg:

import durgamath as m

here durgamath is original module name and m is alias name.

We can access members by using alias name m test.py:

|  |  |
| --- | --- |
| **1)** | **import durgamath as m** |
| **2)** | **print(m.x)** |
| **3)** | **m.add(10,20)** |
| **4)** | **m.product(10,20)** |

### 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 durgamath import x,add print(x)

add(10,20)

product(10,20)==> NameError: name 'product' is not defined

We can import all members of a module as follows from durgamath import \*

test.py:

|  |  |
| --- | --- |
| **1)** | **from durgamath import \*** |
| **2)** | **print(x)** |
| **3)** | **add(10,20)** |
| **4)** | **product(10,20)** |



# Various possibilties of import:

import modulename

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 \*

# member aliasing:

from durgamath import x as y,add as sum print(y)

sum(10,20)

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

Eg:

from durgamath import x as y print(x)==>NameError: name 'x' is not defined

# Reloading a Module:

By default module will be loaded only once eventhough we are importing multiple multiple times.

### Demo Program for module reloading:

|  |  |
| --- | --- |
| **1)** | **import time** |
| **2)** | **from imp import reload** |
| **3)** | **import module1** |
| **4)** | **time.sleep(30)** |
| **5)** | **reload(module1)** |
| **6)** | **time.sleep(30)** |
| **7)** | **reload(module1)** |
| **8)** | **print("This is test file")** |

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



module1.py:

print("This is from module1") test.py

|  |  |
| --- | --- |
| **1)** | **import module1** |
| **2)** | **import module1** |
| **3)** | **import module1** |
| **4)** | **import module1** |
| **5)** | **print("This is test module")** |
| **6)** | |
| **7)** | **Output** |
| **8)** | **This is from module1** |
| **9)** | **This is test module** |

In the above program test module will be loaded only once eventhough 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.

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

import imp imp.reload(module1)

test.py:

|  |  |
| --- | --- |
| **1)** | **import module1** |
| **2)** | **import module1** |
| **3)** | **from imp import reload** |
| **4)** | **reload(module1)** |
| **5)** | **reload(module1)** |
| **6)** | **reload(module1)** |
| **7)** | **print("This is test module")** |

In the above program module1 will be loaded 4 times in that 1 time by default and 3 times explicitly. In this case output is

|  |  |
| --- | --- |
| **1)** | **This is from module1** |
| **2)** | **This is from module1** |
| **3)** | **This is from module1** |
| **4)** | **This is from module1** |
| **5)** | **This is test module** |



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

### 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: test.py

|  |  |
| --- | --- |
| **1)** | **x=10** |
| **2)** | **y=20** |
| **3)** | **def f1():** |
| **4)** | **print("Hello")** |
| **5)** | **print(dir()) # To print all members of current module** |
| **6)** | |
| **7)** | **Output** |
| **8)** | **[' annotations ', ' builtins ', ' cached ', ' doc ', ' file ', ' loader ', ' nam**  **e ', ' package ', ' spec ', 'f1', 'x', 'y']** |

Eg 2: To display members of particular module: durgamath.py:

|  |  |
| --- | --- |
| **1)** | **x=888** |
| **2)** | |
| **3)** | **def add(a,b):** |
| **4)** | **print("The Sum:",a+b)** |
| **5)** | |
| **6)** | **def product(a,b):** |
| **7)** | **print("The Product:",a\*b)** |

test.py:

|  |  |
| --- | --- |
| **1)** | **import durgamath** |
| **2)** | **print(dir(durgamath))** |
| **3)** | |
| **4)** | **Output** |
| **5)** | **[' builtins ', ' cached ', ' doc ', ' file ', ' loader ', ' name ',** |
| **6)** | **' package ', ' spec ', 'add', 'product', 'x']** |

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: test.py:

|  |
| --- |
| **1) print( builtins )** |
| **2) print( cached )** |
| **3) print( doc )** |
| **4) print( file )** |
| **5) print( loader )** |
| **6) print( name )** |
| **7) print( package )** |
| **8) print( spec )** |
| **9)** |
| **10) Output** |
| **11) <module 'builtins' (built-in)>** |
| **12) None** |
| **13) None** |

test.py

|  |
| --- |
| **1) <\_frozen\_importlib\_external.SourceFileLoader object at 0x00572170>** |
| **2) main** |
| **3) None** |
| 4) **None** |

**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:

|  |  |
| --- | --- |
| **1)** | **def f1():** |
| **2)** | **if name ==' main ':** |
| **3)** | **print("The code executed as a program")** |
| **4)** | **else:** |
| **5)** | **print("The code executed as a module from some other program")** |
| **6)** | **f1()** |

test.py:

|  |  |
| --- | --- |
| **1)** | **import module1** |
| **2)** | **module1.f1()** |
| **3)** | |
| **4)** | **D:\Python\_classes>py module1.py** |
| **5)** | **The code executed as a program** |
| **6)** | |
| **7)** | **D:\Python\_classes>py test.py** |
| **8)** | **The code executed as a module from some other program** |
| **9)** | **The code executed as a module from some other program** |

## Working with math module:

Python provides inbuilt module math.

This module defines several functions which can be used for mathematical operations. The main important functions are

1. **sqrt(x)**
2. **ceil(x)**
3. **floor(x)**
4. **fabs(x) 5.log(x)**
5. **sin(x)**
6. **tan(x)**

....

Eg:

|  |  |
| --- | --- |
| **1)** | **from math import \*** |
| **2)** | **print(sqrt(4))** |
| **3)** | **print(ceil(10.1))** |
| **4)** | **print(floor(10.1))** |
| **5)** | **print(fabs(-10.6))** |
| **6)** | **print(fabs(10.6))** |



|  |
| --- |
| **7)** |
| **8) Output** |
| **9) 2.0** |
| **10) 11** |
| **11) 10** |
| **12) 10.6** |
| **13) 10.6** |

### Note:

We can find help for any module by using help() function Eg:

import math help(math)

# Working with random module:

This module defines several functions to generate random numbers.

We can use these functions while developing games,in cryptography and to generate random numbers on fly for authentication.

### random() function:

This function always generate some float value between 0 and 1 ( not inclusive)

0<x<1

Eg:

|  |
| --- |
| **1) from random import \*** |
| **2) for i in range(10):** |
| **3) print(random())** |
| **4)** |
| **5) Output** |
| **6) 0.4572685609302056** |
| **7) 0.6584325233197768** |
| **8) 0.15444034016553587** |
| **9) 0.18351427005232201** |
| **10) 0.1330257265904884** |
| **11) 0.9291139798071045** |
| **12) 0.6586741197891783** |
| **13) 0.8901649834019002** |
| **14) 0.25540891083913053** |
| **15) 0.7290504335962871** |



### randint() function:

To generate random integer beween two given numbers(inclusive) Eg:

|  |
| --- |
| **1) from random import \*** |
| **2) for i in range(10):** |
| **3) print(randint(1,100)) # generate random int value between 1 and 100(inclusive)** |
| **4)** |
| **5) Output** |
| **6) 51** |
| **7) 44** |
| **8) 39** |
| **9) 70** |
| **10) 49** |
| **11) 74** |
| **12) 52** |
| **13) 10** |
| **14) 40** |
| **15) 8** |

### uniform():

It returns random float values between 2 given numbers(not inclusive) Eg:

|  |
| --- |
| **1) from random import \*** |
| **2) for i in range(10):** |
| **3) print(uniform(1,10))** |
| **4)** |
| **5) Output** |
| **6) 9.787695398230332** |
| **7) 6.81102218793548** |
| **8) 8.068672144377329** |
| **9) 8.567976357239834** |
| **10) 6.363511674803802** |
| **11) 2.176137584071641** |
| **12) 4.822867939432386** |
| **13) 6.0801725149678445** |
| **14) 7.508457735544763** |
| **15) 1.9982221862917555** |

random() ===>in between 0 and 1 (not inclusive) randint(x,y) ==>in between x and y ( inclusive) uniform(x,y) ==> in between x and y ( not inclusive)



### randrange([start],stop,[step])

returns a random number from range start<= x < stop

start argument is optional and default value is 0 step argument is optional and default value is 1

randrange(10)-->generates a number from 0 to 9 randrange(1,11)-->generates a number from 1 to 10 randrange(1,11,2)-->generates a number from 1,3,5,7,9

Eg 1:

|  |
| --- |
| **1) from random import \*** |
| **2) for i in range(10):** |
| **3) print(randrange(10))** |
| **4)** |
| **5) Output** |
| **6) 9** |
| **7) 4** |
| **8) 0** |
| **9) 2** |
| **10) 9** |
| **11) 4** |
| **12) 8** |
| **13) 9** |
| **14) 5** |
| **15) 9** |

Eg 2:

|  |
| --- |
| **1) from random import \*** |
| **2) for i in range(10):** |
| **3) print(randrange(1,11))** |
| **4)** |
| **5) Output** |
| **6) 2** |
| **7) 2** |
| **8) 8** |
| **9) 10** |
| **10) 3** |
| **11) 5** |
| **12) 9** |
| **13) 1** |
| **14) 6** |
| **15) 3** |



Eg 3:

|  |
| --- |
| **1) from random import \*** |
| **2) for i in range(10):** |
| **3) print(randrange(1,11,2))** |
| **4)** |
| **5) Output** |
| **6) 1** |
| **7) 3** |
| **8) 9** |
| **9) 5** |
| **10) 7** |
| **11) 1** |
| **12) 1** |
| **13) 1** |
| **14) 7** |
| 15) **3** |

### choice() function:

It wont return random number.

It will return a random object from the given list or tuple. Eg:

|  |  |
| --- | --- |
| **1)** | **from random import \*** |
| **2)** | **list=["Sunny","Bunny","Chinny","Vinny","pinny"]** |
| **3)** | **for i in range(10):** |
| **4)** | **print(choice(list))** |

**Output Bunny pinny Bunny Sunny Bunny pinny pinny Vinny Bunny Sunny**