

Creating and Importing Modules in Python:

A module is simply a Python file with the .py extension. It is a self-contained file with Python statements and definitions. A Python module can have a set of functions, classes or variables defined and implemented.

Modules help us break down large programs into small files that are more manageable. With modules, code reusability becomes a reality. Suppose we have a function that is frequently used in different programs. We can define this function in a module then import it into the various programs without having to copy its code each time.

When a program gets bigger then it becomes very difficult to organize and maintain it. So we may want to split it into a different manageable and organized files. This breaking down of python code into different files is made possible by python modules.

Splitting of a big single python program into separate python modules helps us in many ways like;

- > It becomes easy to read the code and understand it.
- Maintenance of neatly grouped code is easy.
- Python module makes easy re-usability of frequently used code. We can group frequently used code into a module. So next time instead of copying the actual code definition into a program, we can just import the module.
- Also there are many python built-in modules available which can be used into our code, like math, time, numpy etc.

Ways to import a module: COMDIETE JaVa Classes

There are various ways to import a module or its particular entity/entities i.e functions, classes or variables in our program as follows:

Syntax:

- > import *ModuleName*
- > import *ModuleName* as *mn*
- from ModuleName import entity1, entity2....
- from ModuleName import *

These various syntaxes have various purpose of them. The *import* statement imports all the entities (functions, classes or variables) within the module, while the *from* statements are used to import particular entity/entities from a module. By using "*" we can also import all the entities from a module using *from* statement.

Imp Note:

According to the syntax used for importing an entity, the syntax for printing a variable or calling a function changes as well. We will understand it through the following table & examples:



Assume that we are having a module **file1** as in the figure below, and we are going to import its entities in **main.py**.

file1.py main.py

var1 = 10

def f1():
 print("This is f1() from file2)



As we read the important note provided above, we know that "According to the syntax used for importing an entity, the syntax for printing a variable or calling a function changes as well". So we'll understand it through a table below how to print a var1 and call f1() in main.py.

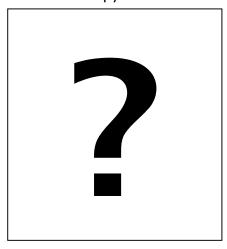
Way to import	To print a variable	To call a function	
import file1	print(file1.var1)	file1.f1()	
import file1 as f	print(f.var1)	f.f1()	
from file1 import var1	print(var1)	can't call f1 as it isn't imported	
from file1 import f1	can't print var1 as it isn't imported	f1()	
from file1 import var1,f1	print(var1)	f1()	
from file1 import *	print(var1)	f1()	

Example 2:

Now in this example we are going to understand, how many entities get imported as we use the various syntaxes.

calculator.py

test.py





Way to import	add()	sub()	mul()	div()
import calculator	calculator.add(10,20)	calculator.sub(10,20)	calculator.mul(10,20)	calculator.div(10,20)
import calculator as c	c.add(10,20)	c.sub(10,20)	c.mul(10,20)	c.div(10,20)
from calculator import add	add(10,20)	0	0	0
from <mark>cal</mark> culator import sub	0	sub(10,20)	0	0
from ca <mark>lc</mark> ulator import mul	0	0	mul(10,20)	0
from calculator import div	0	0	\otimes	div(10,20)
from calculator import add,sub	add(10,20)	sub(10,20)	0	0
from calculator import add,sub,mul	add(10,20)	sub(10,20)	mul(10,20)	0
from calculator import *	add(10,20)	sub(10,20)	mul(10,20)	div(10,20)