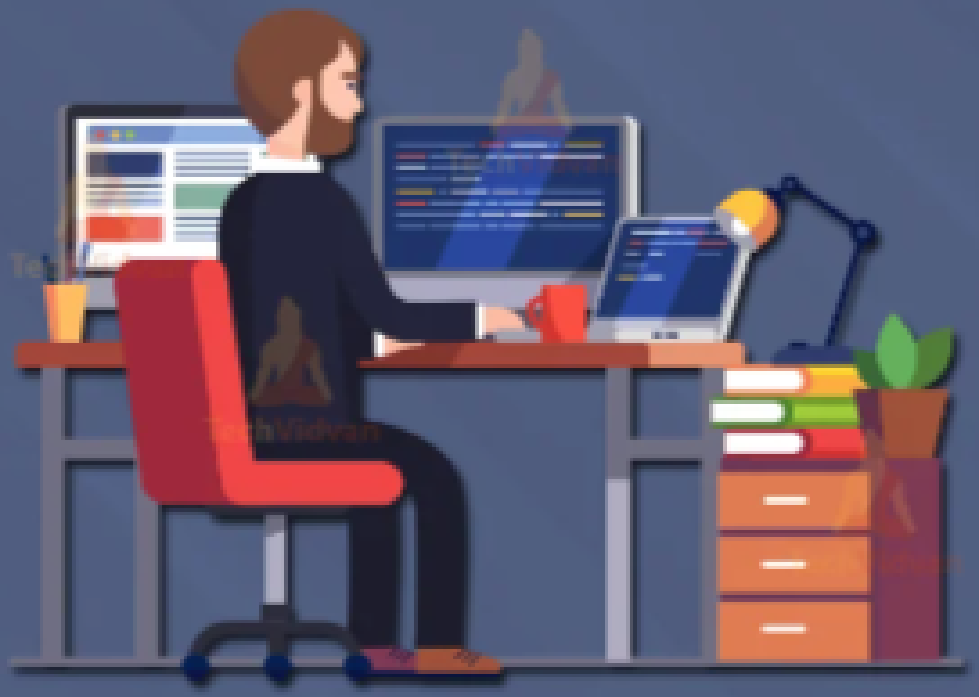
Python

Modules in Python



01 Built-in Modules

User-defined Modules 02

What is Python Module

A Python module is a file containing Python definitions and statements. A module can define functions, classes, and variables. A module can also include runnable code. Grouping related code into a module makes the code easier to understand and use. It also makes the code logically organized.

Create a simple Python module

Let's create a simple calc.py in which we define two functions, one **add** and another **subtract**.

Python3

```
# A simple module, calc.py
def add(x, y):
    return (x+y)

def subtract(x, y):
    return (x-y)
```

Import Module in Python

We can import the functions, and classes defined in a module to another module using the import statement in some other Python source file.

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches for importing a module. For example, to import the module calc.py, we need to put the following command at the top of the script.

Syntax of Python Import

```
import module
```

Note: This does not import the functions or classes directly instead imports the module only. To access the functions inside the module the dot(.) operator is used.

Importing modules in Python

Now, we are importing the **calc** that we created earlier to perform add operation.

```
# importing  module calc.py
import calc

print(calc.add(10, 2))
```

Output:

```
12
```

The from-import Statement in Python

Python's *from* statement lets you import specific attributes from a module without importing the module as a whole.

Importing specific attributes from the module

Here, we are importing specific sqrt and factorial attributes from the math module.

```
# importing sqrt() and factorial from
# module math

from math import sqrt, factorial

# if we simply do "import math", then
# math.sqrt(16) and math.factorial()
# are required.

print(sqrt(16))

print(factorial(6))
```

Output:

```
4.0

720
```

Import all Names

The * symbol used with the from import statement is used to import all the names from a module to a current namespace.

Syntax:

```
from module_name import *
```

From import * Statement

The use of * has its advantages and disadvantages. If you know exactly what you will be needing from the module, it is not recommended to use *, else do so.

```
# importing sqrt() and factorial from
# module math

from math import *

# if we simply do "import math", then
# math.sqrt(16) and math.factorial()
# are required.

print(sqrt(16))

print(factorial(6))
```

Output

```
4.0

720
```

Renaming the Python module

We can rename the module while importing it using the keyword.

> **Syntax:** Import **Module_name** as **Alias_name**

```
# importing sqrt() and factorial from
# module math
import math as mt

# if we simply do "import math", then
# math.sqrt(16) and math.factorial()
# are required.
print(mt.sqrt(16))

print(mt.factorial(6))
```

Output

```
4.0

720
```

Python built-in modules

There are several built-in modules in Python, which you can import whenever you like.

Python3

```
# importing built-in module math
import math
# using square root(sqrt) function con
# in math module
print(math.sqrt(25))
# using pi function contained in math
print(math.pi)
# 2 radians = 114.59 degrees
print(math.degrees(2))
# 60 degrees = 1.04 radians
print(math.radians(60))
# Sine of 2 radians
print(math.sin(2))
# Cosine of 0.5 radians
print(math.cos(0.5))
# Tangent of 0.23 radians
print(math.tan(0.23))
# 1 * 2 * 3 * 4 = 24
print(math.factorial(4))
```

Output:

```
5.0

3.14159265359

114.591559026

1.0471975512

0.909297426826

0.87758256189

0.234143362351

24
```