Python Modules

we will explain how to construct and import custom Python modules. Additionally, we may import or integrate Python's built-in modules via various methods.

What is Modular Programming?

Modular programming is the practice of segmenting a single, complicated coding task into multiple, simpler, easier-to-manage sub-tasks. We call these subtasks modules.

What are Modules in Python?

A document with definitions of functions and various statements written in Python is called a Python module.

In Python, we can define a module in one of 3 ways:

- Python itself allows for the creation of modules.
- Similar to the re (regular expression) module, a module can be primarily written in C programming language and then dynamically inserted at run-time.
- A built-in module, such as the itertools module, is inherently included in the interpreter.

A module is a file containing Python code, definitions of functions, statements, or classes. An example_module.py file is a module we will create and whose name is example_module.

We employ modules to divide complicated programs into smaller, more understandable pieces. Modules also allow for the reuse of code.

Example:

```
# Here, we are creating a simple Python program to show how to create a module.
# defining a function in the module to reuse it
def square( number ):
      result = number ** 2
   return result
```

How to Import Modules in Python?

In Python, we may import functions from one module into our program, or as we say into, another module.

For this, we make use of the import Python keyword. In the Python window, we add the next to import keyword, the name of the module we need to import. We will import the module we defined earlier example_module.

Syntax:

```
import example_module
```

Example:

```
result = example_module.square( 4 )
print("By using the module square of number is: ", result )
```

Python import Statement

Using the import Python keyword and the dot operator, we may import a standard module and can access the defined functions within it. Here's an illustration.

Code

```
# Here, we are creating a simple Python program to show how to import a standard module
# Here, we are import the math module which is a standard module
import math
print( "The value of euler's number is", math.e )
```

Importing and also Renaming

While importing a module, we can change its name too. Here is an example to show.

Code

```
# Here, we are creating a simple Python program to show how to import a module and rename it
```

```
# Here, we are import the math module and give a different name to it
import math as mt
print( "The value of euler's number is", mt.e )
```

Python from...import Statement

We can import specific names from a module without importing the module as a whole. Here is an example.

Code

```
# Here, we are creating a simple Python program to show how to import specific
# objects from a module
from math import e
print( "The value of euler's number is", e )
```

Code

```
# Here, we are creating a simple Python program to show how to import multiple
# objects from a module
from math import e, tau
print( "The value of tau constant is: ", tau )
print( "The value of the euler's number is: ", e )
```

Import all Names - From import * Statement

To import all the objects from a module within the present namespace, use the * symbol and the from and import keyword.

Syntax:

1. `from name_of_module import *`

There are benefits and drawbacks to using the symbol *. It is not advised to use * unless we are certain of our particular requirements from the module; otherwise, do so.

Here is an example of the same.

Code

```
# Here, we are importing the complete math module using *
from math import *
# Here, we are accessing functions of math module without using the dot operator
print( "Calculating square root: ", sqrt(25) )
print( "Calculating tangent of an angle: ", tan(pi/6) )
```

Locating Path of Modules

The module is initially looked for in the current working directory. Python then explores every directory in the shell parameter PYTHONPATH if the module cannot be located in the current directory. A list of folders makes up the environment variable known as PYTHONPATH. Python examines the installation-dependent set of folders set up when Python is downloaded if that also fails.

Here is an example to print the path.

Code

```
# Here, we are importing the sys module
import sys
print("Path of the sys module in the system is:", sys.path)
```

The dir() Built-in Function

We may use the dir() method to identify names declared within a module.

For instance, we have the following names in the standard module str. To print the names, we will use the dir() method in the following way:

Code

```
# Here, we are creating a simple Python program to print the directory of a module
print( "List of functions:\n ", dir( str ), end=", " )
```