MODULES IN PYTHON

Introduction to Modules:

- A Python file which is used in another file is called a Python module.
- A module can contain Python code, definitions of functions, set of functions or classes.
- The extension of the Python module file is also .py
- Common modules used in Python are:
    - math module
    - random module
    - statistics module

Importing module

There are different ways by which we can import a module

1. import statement

'import' statement can be used to import a module. It provides access to all attributes (like variables, constants etc.) and methods or functions of the module.
import <module_name>
For example: **import math**
To access multiple modules, we can use following syntax
import module1, module2, ……., moduleN
For example: **import math, random**
To access particular methods from any module, use following syntax
import <module_name>.<function_name>
For example: **import random.randint(3,6)**

2. from statement

'from' statement is also used to import specific attributes or objects from a module.
from <module_name> import <function_name>
or
from <module_name> import function1, function2, ……..
For example:
from math import sqrt
or
from math import sqrt, floor

3. import *

import * statement used to import all objects from a module.
from <module_name> import *
For example:

```
from math import *
```

Module aliasing:

You can create an alias (nickname) of a module using 'as' keyword.
import <module_name> as <alias_name>
For example:

```
import random as r
```

Attributes, functions and methods of math module:

1. pi

It is a mathematical constant. It returns the value of pi

```
>>> import math
>>> math.pi
3.141592653589793
```

2. e

It is a mathematical constant that return the value of e, where e = 2.718281, It is also called Euler's number

```
>>> import math
>>> math.e
2.718281828459045
```

3. sqrt()

It returns the square root of a number.
Syntax: **math.sqrt(number)**

```
>>> import math
>>> math.sqrt(25)
5.0
>>>math.sqrt(50)
7.0710678118654755
```

4. ceil()

Syntax: **math.ceil(x)**
It returns the smallest integer that is greater than or equal to x.

```
>>> import math
>>> math.ceil(4.5)
5
```

```
>>> math.ceil(6.0)
6
>>> math.ceil(7.1)
8
```

5. floor()

Syntax: **math.floor(x)**

It returns the largest integer that is less than or equal to x.

```
>>> import math
>>> math.floor(4.5)
4
>>> math.floor(6.0)
6
>>> math.floor(7.1)
7
```

6. pow()

Syntax: **math.pow(x,y)**

It returns the value of xy

```
>>> import math
>>> math.pow(3,2)
9.0
>>> math.pow(5,0)
1.0
>>> math.pow(10,-1)
0.1
```

7. fabs()

Syntax: **math.fabs(x)**

It returns the absolute value of x i.e. the value without sign. Its return type is float.

```
>>> import math
>>> math.fabs(-10)
10.0
>>> math.fabs(15)
15.0
>>> math.fabs(0)
0.0
```

8. sin()

Syntax: **math.sin(x)**

It returns the sine of x in radians

Angle in radian = Angle in degree x (π/180)

Where π = 22/7 or 3.14

Convert 30^0 in radians

30^0 in radians = Angle in degree x (π/180)

= 30 * 3.14 /180

=0.524 (approx)

Convert 60^0 in radians

60^0 in radians = Angle in degree x (π/180)

= 60 * 3.14 /180

=1.047 (approx)

```
>>> import math
>>> math.sin(0)
0.0
>>> math.sin(0.524)
0.5003474302699141
>>> math.sin(1.047)
0.8659266112878228
```

9. cos()

Syntax: **math.cos(x)**

It returns the cosine of x in radians

```
>>> import math
>>> math.cos(0)
1.0
>>> math.cos(0.524)
0.8658247218821448
>>> math.cos(1.047)
0.5001710745970701
```

10. tan()

Syntax: **math.tan(x)**

It returns the tangent of x in radians

```
>>> import math
>>> math.tan(0)
0.0
>>> math.tan(0.524)
0.5778853590392409
>>> math.tan(0.524)
0.5778853590392409
```

Attributes, functions and methods of random module:

random module is used to generate random numbers, which are used to generate captcha code, in computer games like throwing of a dice, picking a number or flipping a coin, shuffling cards, creating lottery scratch cards. Random numbers are also used in online quizzes, tests to shuffle questions etc.To use random module we need to import the random module.

random()

Syntax: **random.random()**

random() method generates a random number from 0 to 1. It does not takes any parameter and returns floating point values between 0 and 1 (including 0, but excluding 1)

```
>>> import random as r
>>> r.random()
0.13939737870230484
>>> r.random()
0.5121394221754262
>>> r.random()
0.5622319595776127
```

randint()

Syntax: **random.randint(a, b)**

randint() method accepts two parameters and returns a random integer number between a and b (both a and b are inclusive). It can generate duplicate numbers.

```
>>> import random as r
>>> r.randint(4,9)
4
>>> r.randint(4,9)
5
>>> r.randint(4,9)
7
```

randrange()

Syntax: **random.randrange(start, stop, step)**

randrange() method generates a random integer number between start and stop, where start is inclusive and stop is exclusive. Here start and step are optional parameters. The default value of start is 0 and step is 1.

```
>>> import random as r
#Generate any random number between 0 to 9 (0 include, but 9 exclude)
>>>r.randrange(9)
```

```
2
#Generate any random number between 4 to 9 (4 include, but 9 exclude)
>>> r.randrange(4,9)
5
#Generate any even random number between 3 to 9 (3 include, but 9 exclude)
>>>r.randrange(3,9,2)
4
```

Attributes, functions and methods of statistics module

The statistics module implements many statistical methods like mean(), median() and mode etc. To use statistical methods we need to import the statistics module.

import statistics as s

mean()

Syntax: **statistics.mean(data)**

The mean() method returns the arithmetic mean (average) of the given data set.

median()

Syntax: **statistics.median(data)**

The mean() method calculates the median (middle value) of the given data set. This method also sorts the data in ascending order before calculating the median.

Median = {(n + 1) / 2}th value, where n is the number of values in a set of data. In order to calculate the median, the data must first be sorted in ascending order. The median is the number in the middle.

Note: If the number of data values is odd, it returns the exact middle value. If the number of data values is even, it returns the average of the two middle values.

mode()

Syntax: **statistics.mode(data)**

The mean() method calculates the mode (central tendency) of the given data set.

```
import statistics as s
L = [1, 4, 5, 7, 4, 7, 4, 10, 12]
print("Mean of the data Set: ", s.mean(L))
print("Median of the data Set: ", s.median(L))
print("Mode of the data Set: ", s.mode(L))
```

Output:

Mean of the data Set: 6
Median of the data Set: 5
Mode of the data Set: 4