# List of Functions in Python Math Module

| Function | Description |
|---|---|
| acos(x) | Returns the arc cosine of x. |
| acosh(x) | Returns the inverse hyperbolic cosine of x. |
| asin(x) | Returns the arc sine of x. |
| asinh(x) | Returns the inverse hyperbolic sine of x. |
| atan(x) | Returns the arc tangent of x. |
| atan2(y, x) | Returns atan(y / x). |
| atanh(x) | Returns the inverse hyperbolic tangent of x. |
| ceil(x) | Returns the smallest integer greater than or equal to x. |
| copysign(x, y) | Returns x with the sign of y. |
| cos(x) | Returns the cosine of x. |
| cosh(x) | Returns the hyperbolic cosine of x. |
| degrees(x) | Converts angle x from radians to degrees. |
| e | Mathematical constant e (2.71828...) |
| erf(x) | Returns the error function at x. |

| Function | Description |
|---|---|
| erfc(x) | Returns the complementary error function at x. |
| exp(x) | Returns e**x. |
| expm1(x) | Returns e**x − 1. |
| fabs(x) | Returns the absolute value of x. |
| factorial(x) | Returns the factorial of x. |
| floor(x) | Returns the largest integer less than or equal to x. |
| fmod(x, y) | Returns the remainder when x is divided by y. |
| frexp(x) | Returns the mantissa and exponent of x as the pair (m, e). |
| fsum(iterable) | Returns an accurate floating-point sum of values in the iterable. |
| gamma(x) | Returns the Gamma function at x. |
| hypot(x, y) | Returns the Euclidean norm, sqrt(x*x + y*y). |
| isfinite(x) | Returns True if x is neither infinity Not a Number (NaN). |
| isinf(x) | Returns True if x is positive or negative infinity. |
| isnan(x) | Returns True if x is a NaN. |
| ldexp(x, i) | Returns x * (2**i). |

| Function | Description |
|---|---|
| lgamma(x) | Returns the natural logarithm of the absolute value of the Gamma function at x. |
| log(x[, base]) | Returns the logarithm of x to the base (defaults to e). |
| log10(x) | Returns the base-10 logarithm of x. |
| log1p(x) | Returns the natural logarithm of 1 + x. |
| log2(x) | Returns the base-2 logarithm of x. |
| modf(x) | Returns the fractional and integer parts of x. |
| pi | Mathematical constant, the ratio of the circumference of a circle to its diameter (3.14159...). |
| pow(x, y) | Returns x raised to the power y. |
| radians(x) | Converts angle x from degrees to radians. |
| sin(x) | Returns the sine of x. |
| sinh(x) | Returns the hyperbolic cosine of x. |
| sqrt(x) | Returns the square root of x. |
| tan(x) | Returns the tangent of x. |
| tanh(x) | Returns the hyperbolic tangent of x. |
| trunc(x) | Returns the truncated integer value of x. |

**List of function in the random module**

| **Function** | **Description** |
|---|---|
| betavariate(alpha, beta) | Beta distribution. |
| choice(seq) | Return a random element from the non-empty sequence. |
| expovariate(lambd) | Exponential distribution. |
| gammavariate(alpha, beta) | Gamma distribution. |
| gauss(mu, sigma) | Gaussian distribution. |
| getrandbits(k) | Returns a Python integer with k random bits. |
| getstate() | Returns an object capturing the current internal state of the generator. |
| lognormvariate(mu, sigma) | Log normal distribution. |
| normalvariate(mu, sigma) | Normal distribution. |
| paretovariate(alpha) | Pareto distribution. |
| randint(a, b) | Returns a random integer between a and b inclusive. |
| random() | Return the next random floating point number in the range [0.0, 1.0). |
| randrange(start, stop[, step]) | Returns a random integer from the range. |
| sample(population, k) | Return a k length list of unique elements chosen from the population sequence. |

| Function | Description |
|---|---|
| seed(a=None, version=2) | Initialize the random number generator. |
| setstate(state) | Restores the internal state of the generator. |
| shuffle(seq) | Shuffle the sequence. |
| triangular(low, high, mode) | Return a random floating point number between low and high, with the specified mode between those bounds. |
| uniform(a, b) | Return a random floating point number between a and b inclusive. |
| vonmisesvariate(mu, kappa) | Vonmises distribution. |
| weibullvariate(alpha, beta) | Weibull distribution. |

## **Creating Python Library/Package**

Python **library** is a collection of Python **packages**. We may consider a Python package as a library and create that package that will be considered as the library. Most of the time, we use the terms **library** and **package** interchangeably. In this section we will discuss how a Python package is created. In order to create a Python package we have to go through the following steps:

1. Create a **folder** with the name of the package.
2. Create an **empty** __init__.py file in the folder.
3. Create **module file**(s) and save the file(s) with .py extension in the folder.
4. Associate the folder with Python Installation

Let us create a package named **mypackage** with three modules **mathop**, **stringop**, and **listop**. The mathop module is used for basic mathematical operations. The stringop module is used for basic string operations and the listop module is used for basic list operations. To create the **mypackage** with the above mentioned modules, first create a **folder** with the name **mypackage**. Then using any text editor or IDLE, create an empty __init__.py file in the mypackage folder. Then create required three Python files mathop.py, stringop.py, and listop.py for modules mathop, stringop, and listop respectively inside the mypackage folder.

```
# mathop.py
def sum(num1,num2):
    return num1+num2
def average(num1,num2):
    return (num1+num2)/2
def power(num1,num2):
    return num1**num2
```

```
# stringop.py
def member(str1,str2):
    if str2 in str1:
        print('Second string is a member of first')
    else:
        print('Second string is not a member of first')
def concatenate(str1,str2):
    return str1+str2

def repeat(str,num):
    return str*num
```

```
# listop.py
def add(l):
    sum = 0
    for i in l:
        sum += i
    return sum
def multiply(l):
    prod = 1
    for i in l:
        prod *= i
    return prod
```

Finally the **mypackage** folder is associated with Python installation. We can associate the folder with Python installation by attaching it to Python's site-packages folder. The **site-packages** folder is the target folder of manually built Python packages. When we install Python packages from other source, the packages are installed in site-packages by default.

## Importing Python Library

A program must **import** a **library module** before using it. We can import a Python library module to load it into a program's memory. We can import modules from a **standard library** or a library created by the user without installing it. But to use other **external libraries** from PyPI, it must be installed. Importing a module from a library without installing it returns an import error. The following example illustrates how an error message is thrown when **matplotlib** library is imported without installing it.

If we receive an ImportError message when we are importing a module from the library then we have to deactivate the Python interpreter and install the library with pip. To install a library run the following commands in the terminal:

For Windows OS-

```
pip install library_name
```

Or

```
python –m pip install library_name
```

For Linux OS-

```
sudo apt-get install library_name
```

Or

```
python version –m pip install library_name
```