Module 3: Deep Dive - Functions, Sorting, Errors and Exception Handling

Hands-on Guide

Module 3: Deep Dive - Functions, Sorting, Errors and Exception Handling

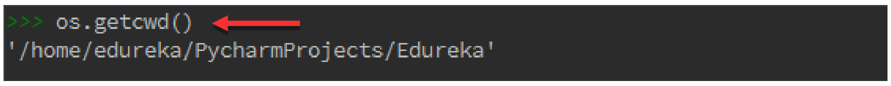
Hands-on Guide

Table of Contents

Os Module

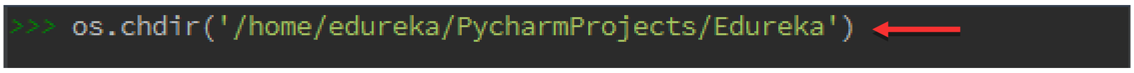
Commonly used function in os module

- os.getcwd()



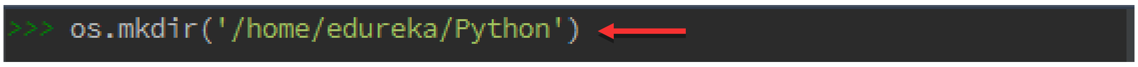
```
>>> os.getcwd()
'/home/edureka/PycharmProjects/Edureka'
```

- os.chdir()



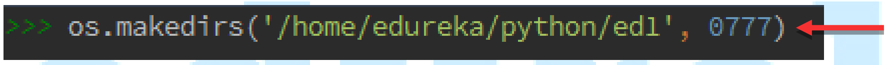
```
>>> os.chdir('/home/edureka/PycharmProjects/Edureka')
```

- os.mkdir()



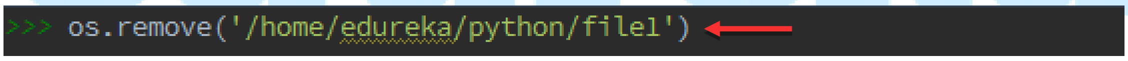
```
>>> os.mkdir('/home/edureka/Python')
```

- os.makedirs()



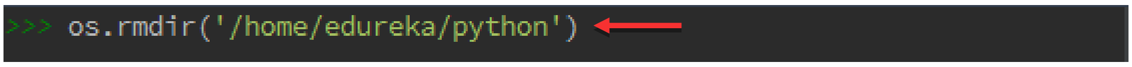
```
>>> os.makedirs('/home/edureka/python/ed1', 0777)
```

- os.remove()



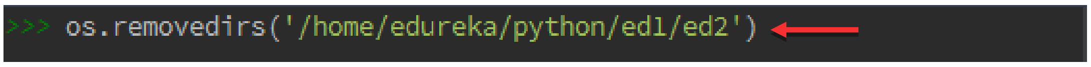
```
>>> os.remove('/home/edureka/python/file1')
```

- os.rmdir()



```
>>> os.rmdir('/home/edureka/python')
```

- os.removedirs()



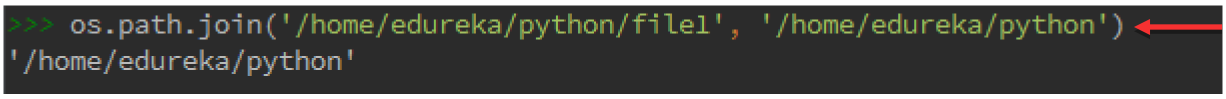
```
>>> os.removedirs('/home/edureka/python/ed1/ed2')
```

os.path

The os.path module is a submodule of os. It provides a range of useful methods to manipulate files and directories

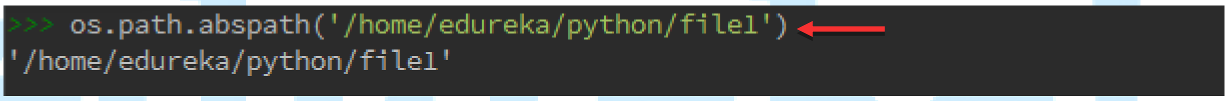
Most of the useful methods are listed here:

- os.path.join() is one of the most important os.path tools. It takes one or more paths and joins them by using the current operating system's path separator. If any component is an absolute path, all previous components are thrown away



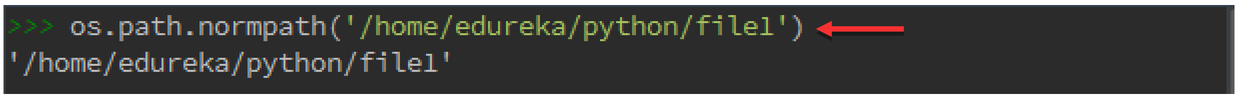
```
>>> os.path.join('/home/edureka/python/file1', '/home/edureka/python')
'/home/edureka/python'
```

- os.path.abspath() takes a relative pathname and returns the corresponding absolute pathname



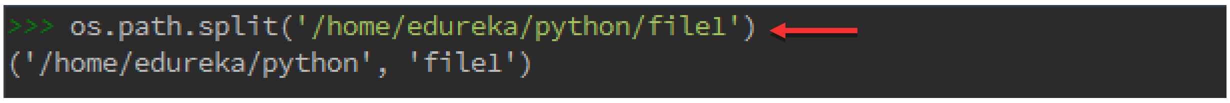
```
>>> os.path.abspath('/home/edureka/python/file1')
'/home/edureka/python/file1'
```

- os.path.normpath() converts path names in nonstandard formats to standard format



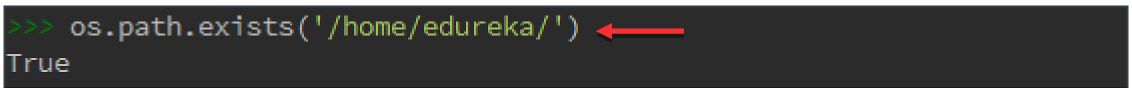
```
>>> os.path.normpath('/home/edureka/python/file1')
'/home/edureka/python/file1'
```

- os.path.split() takes a pathname and returns it in two parts: the directory part and the filename



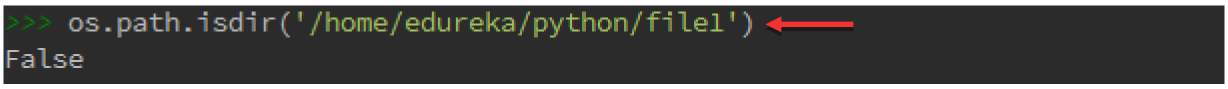
```
>>> os.path.split('/home/edureka/python/file1')
('/home/edureka/python', 'file1')
```

- os.path.exists() takes a pathname and returns true if it exists



```
>>> os.path.exists('/home/edureka/')
True
```

- os.path.isdir() takes a pathname and returns true if it points to a directory

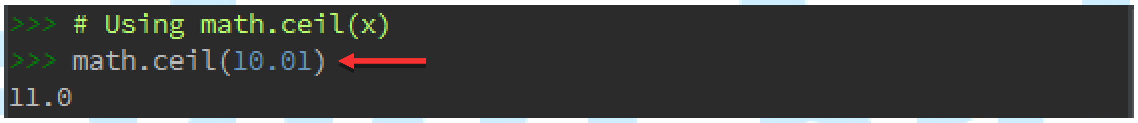


```
>>> os.path.isdir('/home/edureka/python/file1')
False
```

Math Module

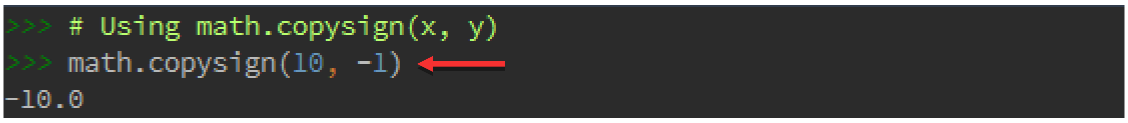
Theoretic Functions

- math.ceil(x)
    - Return the ceiling of x as a float, the smallest integer value greater than or equal to x



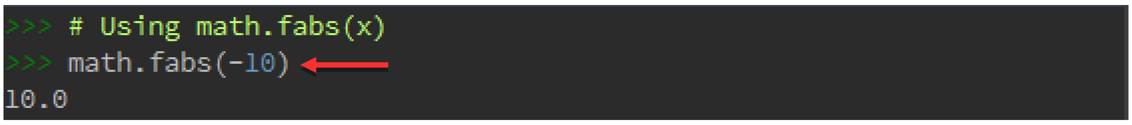
```
>>> # Using math.ceil(x)
>>> math.ceil(10.01)
11.0
```

- math.copysign(x, y)
    - Return x with the sign of y. On a platform that supports signed zeros, copysign(1.0, -0.0) returns -1.0



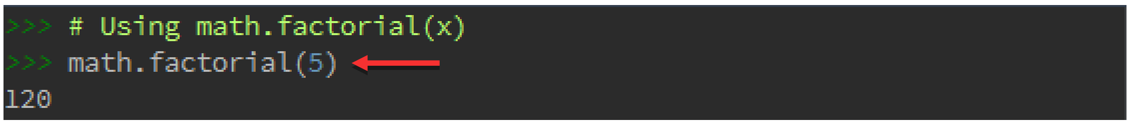
```
>>> # Using math.copysign(x, y)
>>> math.copysign(10, -1)
-10.0
```

- math.fabs(x)
    - Return the absolute value of x



```
>>> # Using math.fabs(x)
>>> math.fabs(-10)
10.0
```

- math.factorial(x)
    - Return x factorial. Raises ValueError if x is not integral or is negative

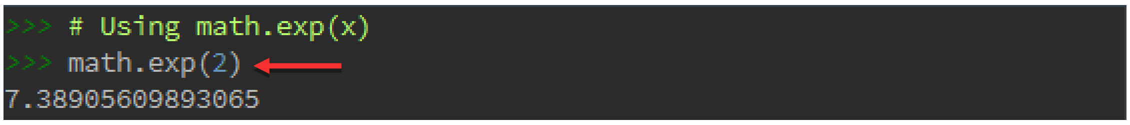


```
>>> # Using math.factorial(x)
>>> math.factorial(5)
120
```

- math.floor(x)
    - Return the floor of x as a float, the largest integer value less than or equal to x

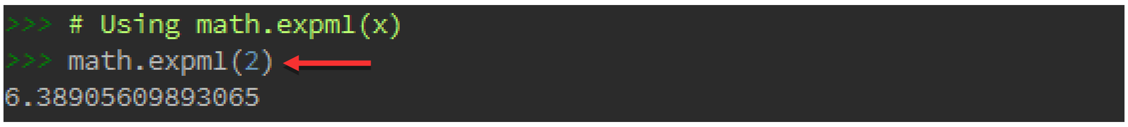
Power and Logarithmic Functions

- math.exp(x)
    - Return e**x



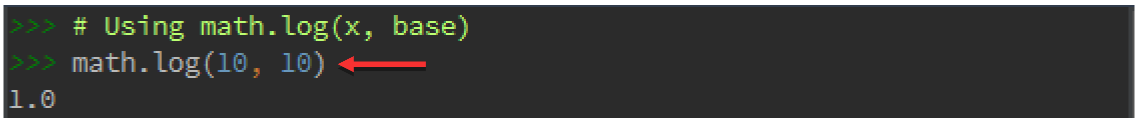
```
>>> # Using math.exp(x)
>>> math.exp(2)
7.38905609893065
```

- math.expm1(x)
    - Return e**x - 1. For small floats x, the subtraction in exp(x) - 1 can result in a significant loss of precision; the expm1() function provides a way to compute this quantity to full precision



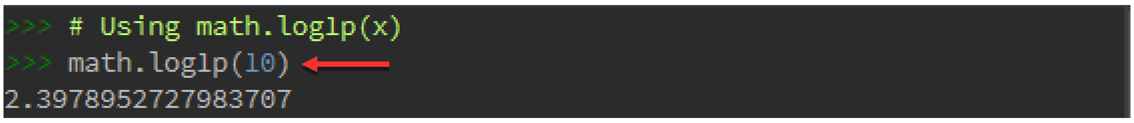
```
>>> # Using math.expm1(x)
>>> math.expm1(2)
6.38905609893065
```

- math.log(x[, base])
    - With one argument, return the natural logarithm of x (to base e)
    - With two arguments, return the logarithm of x to the given base, calculated as log(x)/log(base)



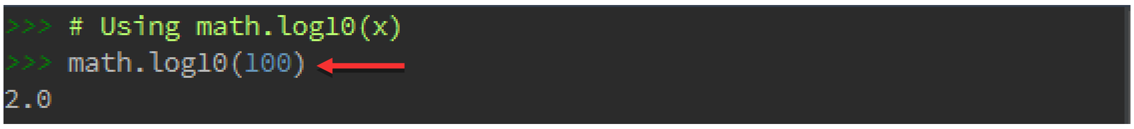
```
>>> # Using math.log(x, base)
>>> math.log(10, 10)
1.0
```

- math.log1p(x)
    - Return the natural logarithm of 1+x (base e). The result is calculated in a way which is accurate for x near zero



```
>>> # Using math.log1p(x)
>>> math.log1p(10)
2.3978952727983707
```

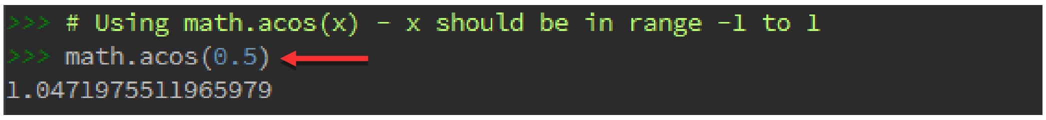
- math.log10(x)
    - Return the base-10 logarithm of x. This is usually more accurate than log(x, 10)



```
>>> # Using math.log10(x)
>>> math.log10(100)
2.0
```

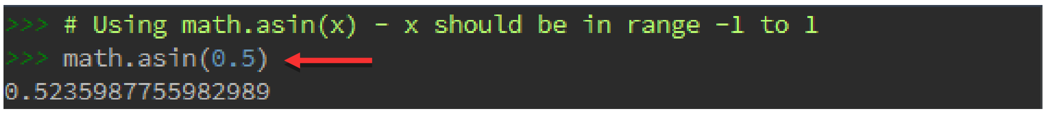
Trigonometric Functions

- math.acos(x)
    - Return the arc cosine of x, in radians



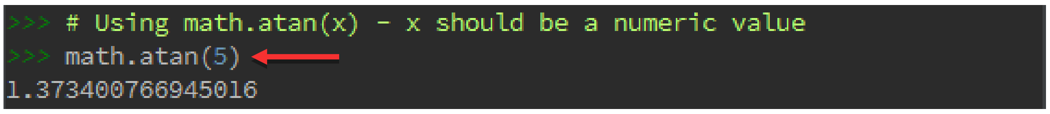
```
>>> # Using math.acos(x) - x should be in range -1 to 1
>>> math.acos(0.5)
1.0471975511965979
```

- math.asin(x)
    - Return the arc sine of x, in radians



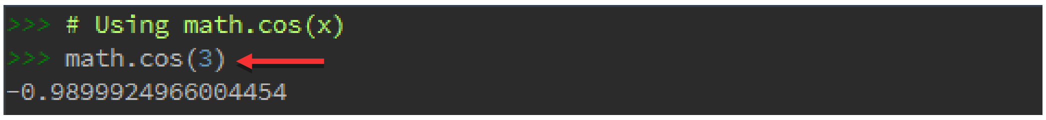
```
>>> # Using math.asin(x) - x should be in range -1 to 1
>>> math.asin(0.5)
0.5235987755982989
```

- math.atan(x)
    - Return the arc tangent of x, in radians



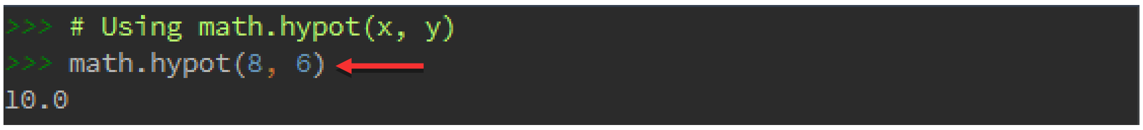
```
>>> # Using math.atan(x) - x should be a numeric value
>>> math.atan(5)
1.373400766945016
```

- math.cos(x)
    - Return the cosine of x radians



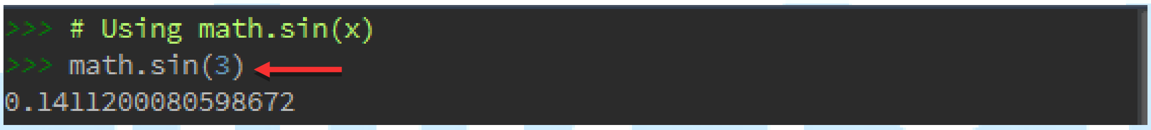
```
>>> # Using math.cos(x)
>>> math.cos(3)
-0.9899924966004454
```

- math.hypot(x, y)
    - Return the Euclidean norm, sqrt(x*x + y*y). This is the length of the vector from the origin to point (x, y)



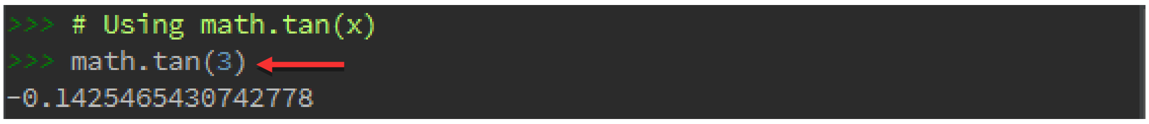
```
>>> # Using math.hypot(x, y)
>>> math.hypot(8, 6)
10.0
```

- math.sin(x)
    - Return the sine of x radians



```
>>> # Using math.sin(x)
>>> math.sin(3)
0.1411200080598672
```

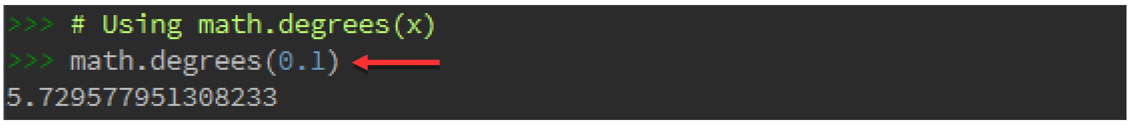
- math.tan(x)
    - Return the tangent of x radians



```
>>> # Using math.tan(x)
>>> math.tan(3)
-0.1425465430742778
```

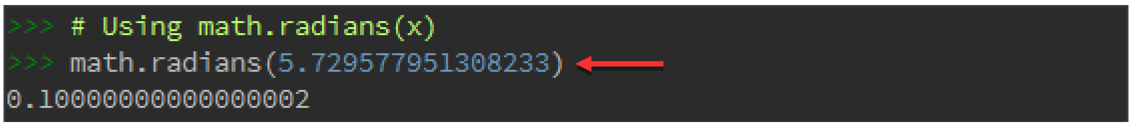
Angular, Hyperbolic and Constant

- math.degrees(x)
    - Converts angle x from radians to degrees



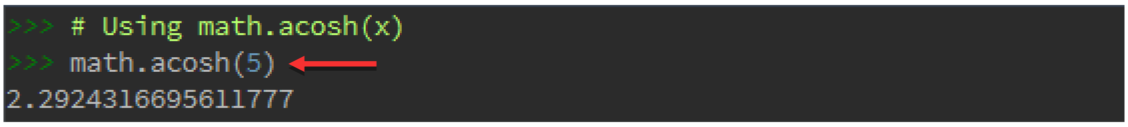
```
>>> # Using math.degrees(x)
>>> math.degrees(0.1)
5.729577951308233
```

- math.radians(x)
    - Converts angle x from degrees to radians



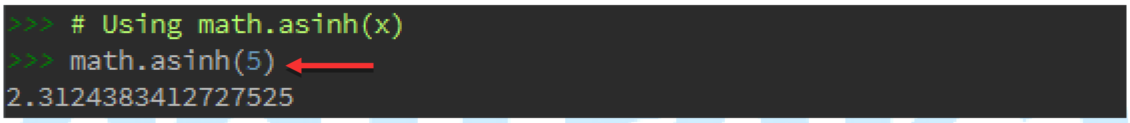
```
>>> # Using math.radians(x)
>>> math.radians(5.729577951308233)
0.10000000000000002
```

- math.acosh(x)
    - Return the inverse hyperbolic cosine of x



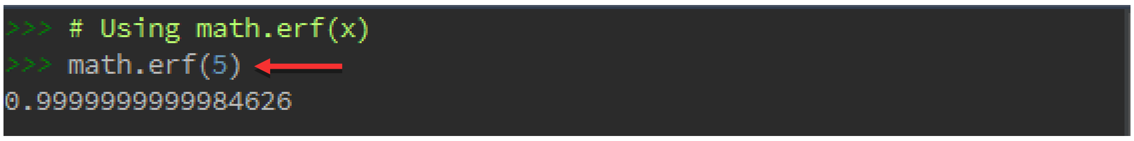
```
>>> # Using math.acosh(x)
>>> math.acosh(5)
2.2924316695611777
```

- math.asinh(x)
    - Return the inverse hyperbolic sine of x



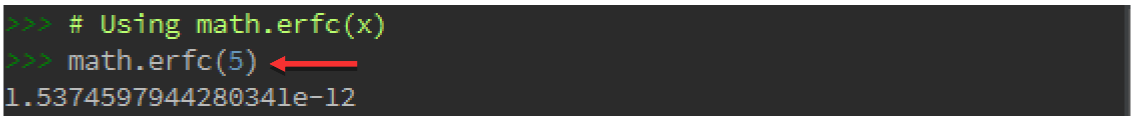
```
>>> # Using math.asinh(x)
>>> math.asinh(5)
2.3124383412727525
```

- math.erf(x)
    - Return the error function at x



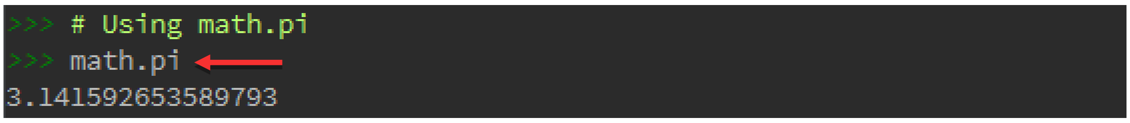
```
>>> # Using math.erf(x)
>>> math.erf(5)
0.9999999999984626
```

- math.erfc(x)
    - Return the complementary error function at x



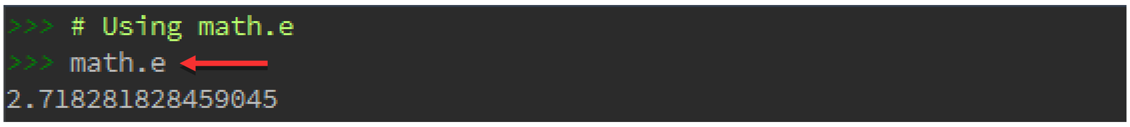
```
>>> # Using math.erfc(x)
>>> math.erfc(5)
1.5374597944280341e-12
```

- math.pi
    - Returns the value of constant π



```
>>> # Using math.pi
>>> math.pi
3.141592653589793
```

- math.e
    - Returns the value of constant e



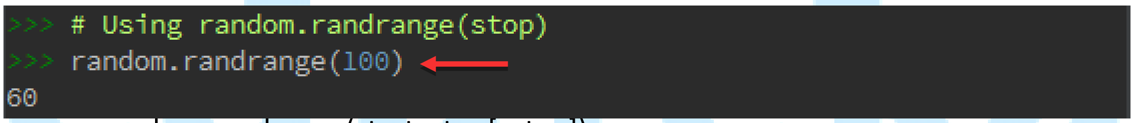
```
>>> # Using math.e
>>> math.e
2.718281828459045
```

Random

This module implements pseudo-random number generators for various distributions

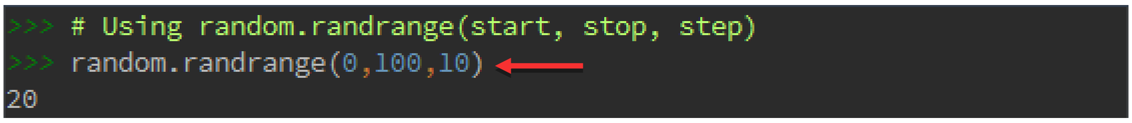
Functions for integers:

- random.randrange(stop)
    - Generates a random integer within the given range



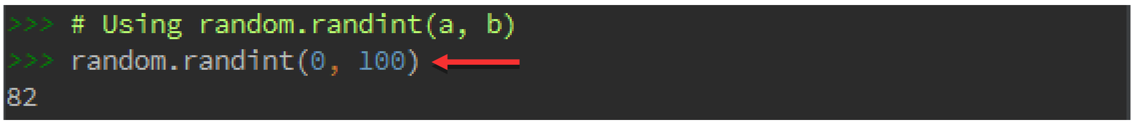
```
>>> # Using random.randrange(stop)
>>> random.randrange(100)
60
```

- random.randrange(start, stop[, step])
    - Return a randomly selected element from range(start, stop, step). This is equivalent to choice(range(start, stop, step)), but doesn't actually build a range object



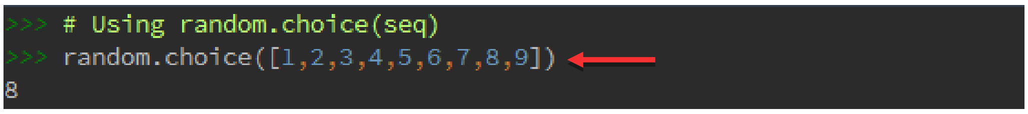
```
>>> # Using random.randrange(start, stop, step)
>>> random.randrange(0,100,10)
20
```

- random.randint(a, b)
    - Return a random integer N such that a <= N <= b



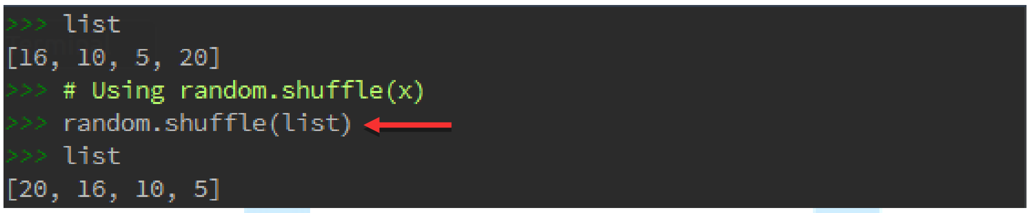
```
>>> # Using random.randint(a, b)
>>> random.randint(0, 100)
82
```

- random.choice(seq)
    - Return a random element from the non-empty sequence seq. If seq is empty, raises IndexError



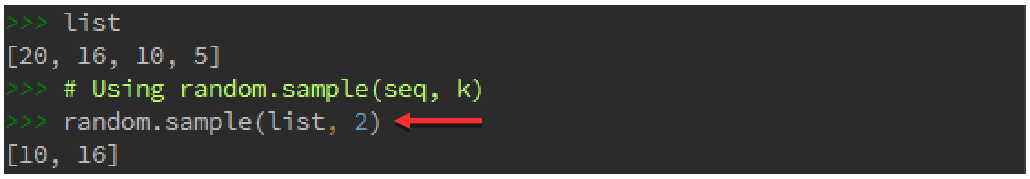
```
>>> # Using random.choice(seq)
>>> random.choice([1,2,3,4,5,6,7,8,9])
8
```

- random.shuffle(x[, random])
    - Shuffle the sequence x in place. The optional argument random is a 0-argument function returning a random float in [0.0, 1.0); by default, this is the function random()



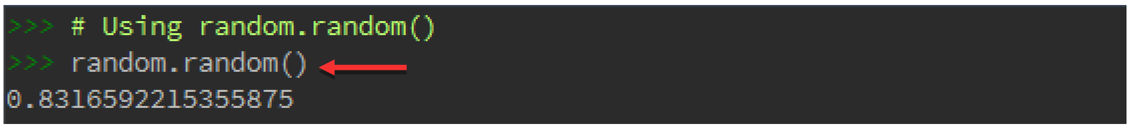
```
>>> list
[16, 10, 5, 20]
>>> # Using random.shuffle(x)
>>> random.shuffle(list)
>>> list
[20, 16, 10, 5]
```

- random.sample(population, k)
    - Return a k length list of unique elements chosen from the population sequence. Used for random sampling without replacement



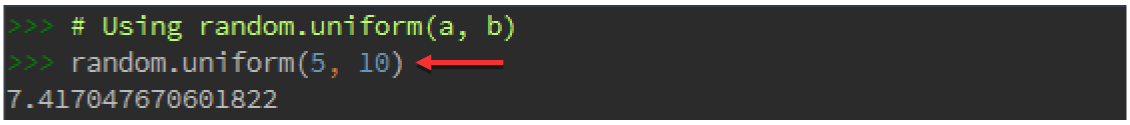
```
>>> list
[20, 16, 10, 5]
>>> # Using random.sample(seq, k)
>>> random.sample(list, 2)
[10, 16]
```

- random.random()
    - Return the next random floating point number in the range [0.0, 1.0)



```
>>> # Using random.random()
>>> random.random()
0.8316592215355875
```

- random.uniform(a, b)
    - Return a random floating point number N such that a <= N <= b for a <= b and b <= N <= a for b < a



```
>>> # Using random.uniform(a, b)
>>> random.uniform(5, 10)
7.417047670601822
```