9. Python 3 – Numbers

Number data types store numeric values. They are immutable data types. This means, changing the value of a number data type results in a newly allocated object.

Number objects are created when you assign a value to them. For example-

```
var1 = 1
var2 = 10
```

You can also delete the reference to a number object by using the **del** statement. The syntax of the **del** statement is –

```
del var1[,var2[,var3[....,varN]]]]
```

You can delete a single object or multiple objects by using the **del** statement. For example-

```
del var
del var_a, var_b
```

Python supports different numerical types-

- **int (signed integers)**: They are often called just integers or **ints**. They are positive or negative whole numbers with no decimal point. Integers in Python 3 are of unlimited size. Python 2 has two integer types - int and long. There is no **'long integer'** in Python 3 anymore.

- **float (floating point real values)** : Also called floats, they represent real numbers and are written with a decimal point dividing the integer and the fractional parts. Floats may also be in scientific notation, with E or e indicating the power of 10 (2.5e2 = 2.5 x 10^2 = 250).

- **complex (complex numbers)** : are of the form a + bJ, where a and b are floats and J (or j) represents the square root of -1 (which is an imaginary number). The real part of the number is a, and the imaginary part is b. Complex numbers are not used much in Python programming.

It is possible to represent an integer in hexa-decimal or octal form.

```
>>> number = 0xA0F #Hexa-decimal
>>> number
2575

>>> number=0o37 #Octal
>>> number
```

```
31
```

Examples

Here are some examples of numbers.

int	float	complex
10	0.0	3.14j
100	15.20	45.j
-786	-21.9	9.322e-36j
080	32.3+e18	.876j
-0490	-90.	-.6545+0J
-0x260	-32.54e100	3e+26J
0x69	70.2-E12	4.53e-7j

A complex number consists of an ordered pair of real floating-point numbers denoted by a + bj, where a is the real part and b is the imaginary part of the complex number.

Number Type Conversion

Python converts numbers internally in an expression containing mixed types to a common type for evaluation. Sometimes, you need to coerce a number explicitly from one type to another to satisfy the requirements of an operator or function parameter.

- Type **int(x)** to convert x to a plain integer.
- Type **long(x)** to convert x to a long integer.
- Type **float(x)** to convert x to a floating-point number.
- Type **complex(x)** to convert x to a complex number with real part x and imaginary part zero.
- Type **complex(x, y)** to convert x and y to a complex number with real part x and imaginary part y. x and y are numeric expressions.

Mathematical Functions

Python includes the following functions that perform mathematical calculations.

Function	Returns (Description)
abs(x)	The absolute value of x: the (positive) distance between x and zero.
ceil(x)	The ceiling of x: the smallest integer not less than x.
cmp(x, y)	-1 if x < y, 0 if x == y, or 1 if x > y. **Deprecated** in Python 3; Instead use **return (x>y)-(x<y).**
exp(x)	The exponential of x: e^x
fabs(x)	The absolute value of x.
floor(x)	The floor of x: the largest integer not greater than x.
log(x)	The natural logarithm of x, for x> 0.
log10(x)	The base-10 logarithm of x for x> 0.
max(x1, x2,...)	The largest of its arguments: the value closest to positive infinity.
min(x1, x2,...)	The smallest of its arguments: the value closest to negative infinity.
modf(x)	The fractional and integer parts of x in a two-item tuple. Both parts have the same sign as x. The integer part is returned as a float.
pow(x, y)	The value of x**y.
round(x [,n])	x rounded to n digits from the decimal point. Python rounds away from zero as a tie-breaker: round(0.5) is 1.0 and round(-0.5) is -1.0.
sqrt(x)	The square root of x for x > 0.

Let us learn about these functions in detail.

Number abs() Method

Description

The **abs()** method returns the absolute value of x i.e. the positive distance between x and zero.

Syntax

Following is the syntax for abs() method-

```
abs( x )
```

Parameters

x - This is a numeric expression.

Return Value

This method returns the absolute value of x.

Example

The following example shows the usage of the abs() method.

```
#!/usr/bin/python3
print ("abs(-45) : ", abs(-45))
print ("abs(100.12) : ", abs(100.12))
```

When we run the above program, it produces the following result-

```
abs(-45) :  45
abs(100.12) :  100.12
```

Number ceil() Method

Description

The **ceil()** method returns the ceiling value of x i.e. the smallest integer not less than x.

Syntax

Following is the syntax for the **ceil()** method-

```
import math
math.ceil( x )
```

Note: This function is not accessible directly, so we need to import math module and then we need to call this function using the math static object.

Parameters

x - This is a numeric expression.

Return Value

This method returns the smallest integer not less than x**.**

Example

The following example shows the usage of the ceil() method.

```
#!/usr/bin/python3
import math   # This will import math module
print ("math.ceil(-45.17) : ", math.ceil(-45.17))
print ("math.ceil(100.12) : ", math.ceil(100.12))
print ("math.ceil(100.72) : ", math.ceil(100.72))
print ("math.ceil(math.pi) : ", math.ceil(math.pi))
```

When we run the above program, it produces the following result-

```
math.ceil(-45.17) :  -45
math.ceil(100.12) :  101
math.ceil(100.72) :  101
math.ceil(math.pi) :  4
```

Number exp() Method

Description

The **exp()** method returns exponential of x: e^x**.**

Syntax

Following is the syntax for the exp() method-

```
import math
math.exp( x )
```

Note: This function is not accessible directly. Therefore, we need to import the math module and then we need to call this function using the math static object.

Parameters

X - This is a numeric expression.

Return Value

This method returns exponential of x: ex.

Example

The following example shows the usage of exp() method.

```
#!/usr/bin/python3
import math   # This will import math module
print ("math.exp(-45.17) : ", math.exp(-45.17))
print ("math.exp(100.12) : ", math.exp(100.12))
print ("math.exp(100.72) : ", math.exp(100.72))
print ("math.exp(math.pi) : ", math.exp(math.pi))
```

When we run the above program, it produces the following result-

```
math.exp(-45.17) :  2.4150062132629406e-20
math.exp(100.12) :  3.0308436140742566e+43
math.exp(100.72) :  5.522557130248187e+43
math.exp(math.pi) :  23.140692632779267
```

Number fabs() Method

Description

The **fabs()** method returns the absolute value of x. Although similar to the abs() function, there are differences between the two functions. They are-

- abs() is a built in function whereas fabs() is defined in math module.
- fabs() function works only on float and integer whereas abs() works with complex number also.

Syntax

Following is the syntax for the fabs() method-

```
import math
math.fabs( x )
```

Note: This function is not accessible directly, so we need to import the math module and then we need to call this function using the math static object.

Parameters

x - This is a numeric value.

Return Value

This method returns the absolute value of x.

Example

The following example shows the usage of the fabs() method.

```
#!/usr/bin/python3
import math   # This will import math module
print ("math.fabs(-45.17) : ", math.fabs(-45.17))
print ("math.fabs(100.12) : ", math.fabs(100.12))
print ("math.fabs(100.72) : ", math.fabs(100.72))
print ("math.fabs(math.pi) : ", math.fabs(math.pi))
```

When we run the above program, it produces following result-

```
math.fabs(-45.17) :  45.17
math.fabs(100) :  100.0
math.fabs(100.72) :  100.72
math.fabs(math.pi) :  3.141592653589793
```

Number floor() Method

Description

The **floor()** method returns the floor of **x** i.e. the largest integer not greater than x.

Syntax

Following is the syntax for the **floor()** method-

```
import math
math.floor( x )
```

Note: This function is not accessible directly, so we need to import the math module and then we need to call this function using the math static object.

Parameters

x - This is a numeric expression.

Return Value

This method returns the largest integer not greater than x.

Example

The following example shows the usage of the floor() method.

```
#!/usr/bin/python3
import math   # This will import math module
print ("math.floor(-45.17) : ", math.floor(-45.17))
print ("math.floor(100.12) : ", math.floor(100.12))
print ("math.floor(100.72) : ", math.floor(100.72))
print ("math.floor(math.pi) : ", math.floor(math.pi))
```

When we run the above program, it produces the following result-

```
math.floor(-45.17) :  -46
math.floor(100.12) :  100
math.floor(100.72) :  100
math.floor(math.pi) :  3
```

Number log() Method

Description

The **log()** method returns the natural logarithm of x, for x > 0.

Syntax

Following is the syntax for the **log()** method-

```
import math
math.log( x )
```

Note: This function is not accessible directly, so we need to import the math module and then we need to call this function using the math static object.

Parameters

x - This is a numeric expression.

Return Value

This method returns natural logarithm of x, for x > 0.

Example

The following example shows the usage of the log() method.

```
#!/usr/bin/python3
import math   # This will import math module
print ("math.log(100.12) : ", math.log(100.12))
print ("math.log(100.72) : ", math.log(100.72))
print ("math.log(math.pi) : ", math.log(math.pi))
```

When we run the above program, it produces the following result-

```
math.log(100.12) :  4.6063694665635735
math.log(100.72) :  4.612344389736092
math.log(math.pi) :  1.1447298858494002
```

Number log10() Method

Description

The **log10()** method returns base-10 logarithm of x for x > 0.

Syntax

Following is the syntax for **log10()** method-

```
import math
math.log10( x )
```

Note: This function is not accessible directly, so we need to import the math module and then we need to call this function using the math static object.

Parameters

x - This is a numeric expression.

Return Value

This method returns the base-10 logarithm of x for x > 0.

Example

The following example shows the usage of the **log10()** method.

```
#!/usr/bin/python3
import math   # This will import math module
print ("math.log10(100.12) : ", math.log10(100.12))
print ("math.log10(100.72) : ", math.log10(100.72))
print ("math.log10(119) : ", math.log10(119))
print ("math.log10(math.pi) : ", math.log10(math.pi))
```

When we run the above program, it produces the following result-

```
math.log10(100.12) :  2.0005208409361854
math.log10(100.72) :  2.003115717099806
math.log10(119) :  2.0755469613925306
math.log10(math.pi) :  0.49714987269413385
```

Number max() Method

Description

The **max()** method returns the largest of its arguments i.e. the value closest to positive infinity.

Syntax

Following is the syntax for **max()** method-

```
max( x, y, z, .... )
```

Parameters

- **x** - This is a numeric expression.
- **y** - This is also a numeric expression.
- **z** - This is also a numeric expression.

Return Value

This method returns the largest of its arguments.

Example

The following example shows the usage of the max() method.

```
#!/usr/bin/python3
print ("max(80, 100, 1000) : ", max(80, 100, 1000))
print ("max(-20, 100, 400) : ", max(-20, 100, 400))
print ("max(-80, -20, -10) : ", max(-80, -20, -10))
print ("max(0, 100, -400) : ", max(0, 100, -400))
```

When we run the above program, it produces the following result-

```
max(80, 100, 1000) :  1000
max(-20, 100, 400) :  400
max(-80, -20, -10) :  -10
max(0, 100, -400) :  100
```

Number min() Method

Description

The method **min()** returns the smallest of its arguments i.e. the value closest to negative infinity.

Syntax

Following is the syntax for the **min()** method-

```
min( x, y, z, .... )
```

Parameters

- **x** - This is a numeric expression.
- **y** - This is also a numeric expression.
- **z** - This is also a numeric expression.

Return Value

This method returns the smallest of its arguments.

Example

The following example shows the usage of the **min()** method.

```
#!/usr/bin/python3
print ("min(80, 100, 1000) : ", min(80, 100, 1000))
print ("min(-20, 100, 400) : ", min(-20, 100, 400))
print ("min(-80, -20, -10) : ", min(-80, -20, -10))
print ("min(0, 100, -400) : ", min(0, 100, -400))
```

When we run the above program, it produces the following result-

```
min(80, 100, 1000) :  80
min(-20, 100, 400) :  -20
min(-80, -20, -10) :  -80
min(0, 100, -400) :  -400
```

Number modf() Method

Description

The **modf()** method returns the fractional and integer parts of x in a two-item tuple. Both parts have the same sign as x. The integer part is returned as a float.

Syntax

Following is the syntax for the **modf()** method-

```
import math

math.modf( x )
```

Note: This function is not accessible directly, so we need to import the math module and then we need to call this function using the math static object.

Parameters

x - This is a numeric expression.

Return Value

This method returns the fractional and integer parts of x in a two-item tuple. Both the parts have the same sign as x. The integer part is returned as a float.

Example

The following example shows the usage of the **modf()** method.

```
#!/usr/bin/python3

import math   # This will import math module

print ("math.modf(100.12) : ", math.modf(100.12))

print ("math.modf(100.72) : ", math.modf(100.72))

print ("math.modf(119) : ", math.modf(119))

print ("math.modf(math.pi) : ", math.modf(math.pi))
```

When we run the above program, it produces the following result-

```
math.modf(100.12) :  (0.12000000000000455, 100.0)

math.modf(100.72) :  (0.7199999999999989, 100.0)

math.modf(119) :  (0.0, 119.0)

math.modf(math.pi) :  (0.14159265358979312, 3.0)
```

Number pow() Method

Return Value

This method returns the value of x^y.

Example

The following example shows the usage of the **pow()** method.

```
#!/usr/bin/python3
import math   # This will import math module
print ("math.pow(100, 2) : ", math.pow(100, 2))
print ("math.pow(100, -2) : ", math.pow(100, -2))
print ("math.pow(2, 4) : ", math.pow(2, 4))
print ("math.pow(3, 0) : ", math.pow(3, 0))
```

When we run the above program, it produces the following result-

```
math.pow(100, 2) :  10000.0
math.pow(100, -2) :  0.0001
math.pow(2, 4) :  16.0
math.pow(3, 0) :  1.0
```

Number round() Method

Description

round() is a built-in function in Python. It returns x rounded to n digits from the decimal point.

Syntax

Following is the syntax for the round() method-

```
round( x [, n]  )
```

Parameters

- **x** - This is a numeric expression.
- **n** - Represents number of digits from decimal point up to which x is to be rounded. Default is 0.

Return Value

This method returns x rounded to n digits from the decimal point.

Example

The following example shows the usage of **round()** method.

```
#!/usr/bin/python3
print ("round(70.23456) : ", round(70.23456))
print ("round(56.659,1) : ", round(56.659,1))
print ("round(80.264, 2) : ", round(80.264, 2))
print ("round(100.000056, 3) : ", round(100.000056, 3))
print ("round(-100.000056, 3) : ", round(-100.000056, 3))
```

When we run the above program, it produces the following result-

```
round(70.23456) :  70
round(56.659,1) :  56.7
round(80.264, 2) :  80.26
round(100.000056, 3) :  100.0
round(-100.000056, 3) :  -100.0
```

Number sqrt() Method

Description

The **sqrt()** method returns the square root of x for x > 0.

Syntax

Following is the syntax for **sqrt()** method-

```
import math
math.sqrt( x )
```

Note: This function is not accessible directly, so we need to import the math module and then we need to call this function using the math static object.

Parameters

x - This is a numeric expression.

Return Value

This method returns square root of x for x > 0.

Example

The following example shows the usage of sqrt() method.

```
#!/usr/bin/python3
import math   # This will import math module
print ("math.sqrt(100) : ", math.sqrt(100))
print ("math.sqrt(7) : ", math.sqrt(7))
print ("math.sqrt(math.pi) : ", math.sqrt(math.pi))
```

When we run the above program, it produces the following result-

```
math.sqrt(100) :  10.0
math.sqrt(7) :  2.6457513110645907
math.sqrt(math.pi) :  1.7724538509055159
```

Random Number Functions

Random numbers are used for games, simulations, testing, security, and privacy applications. Python includes the following functions that are commonly used.

Function	Description
choice(seq)	A random item from a list, tuple, or string.
randrange ([start,] stop [,step])	A randomly selected element from range(start, stop, step).
random()	A random float r, such that 0 is less than or equal to r and r is less than 1.
seed([x])	Sets the integer starting value used in generating random numbers. Call this function before calling any other random module function. Returns None.
shuffle(lst)	Randomizes the items of a list in place. Returns None.
uniform(x, y)	A random float r, such that x is less than or equal to r and r is less than y.

Number choice() Method

Description

The **choice()** method returns a random item from a list, tuple, or string.

Syntax

Following is the syntax for **choice()** method-

```
choice( seq )
```

Note: This function is not accessible directly, so we need to import the random module and then we need to call this function using the random static object.

Parameters

seq - This could be a list, tuple, or string...

Return Value

This method returns a random item.

Example

The following example shows the usage of the **choice()** method.

```
#!/usr/bin/python3
import random
print ("returns a random number from range(100) : ",random.choice(range(100)))
print ("returns random element from list [1, 2, 3, 5, 9]) : ", random.choice([1, 2, 3, 5, 9]))
print ("returns random character from string 'Hello World' : ", random.choice('Hello World'))
```

When we run the above program, it produces a result similar to the following-

```
returns a random number from range(100) :  19
returns random element from list [1, 2, 3, 5, 9]) :  9
returns random character from string 'Hello World' :  r
```

Number randrange() Method

Description

The **randrange()** method returns a randomly selected element from range(start, stop, step).

Syntax

Following is the syntax for the **randrange()** method-

```
randrange ([start,] stop [,step])
```

Note: This function is not accessible directly, so we need to import the random module and then we need to call this function using the random static object.

Parameters

- **start** - Start point of the range. This would be included in the range. Default is 0.
- **stop** - Stop point of the range. This would be excluded from the range.
- **step** - Value with which number is incremented. Default is 1.

Return Value

This method returns a random item from the given range.

Example

The following example shows the usage of the randrange() method.

```
#!/usr/bin/python3
import random
# randomly select an odd number between 1-100
print ("randrange(1,100, 2) : ", random.randrange(1, 100, 2))
# randomly select a number between 0-99
print ("randrange(100) : ", random.randrange(100))
```

When we run the above program, it produces the following result-

```
randrange(1,100, 2) :  83
randrange(100) :  93
```

Number random() Method

Description

The **random()** method returns a random floating point number in the range [0.0, 1.0].

Syntax

Following is the syntax for the random() method-

```
random ( )
```

Note: This function is not accessible directly, so we need to import the random module and then we need to call this function using the random static object.

Parameters

NA

Return Value

This method returns a random float r, such that 0.0 <= r <= 1.0

Example

The following example shows the usage of the **random()** method.

```
#!/usr/bin/python3
import random
# First random number
print ("random() : ", random.random())
# Second random number
print ("random() : ", random.random())
```

When we run the above program, it produces the following result-

```
random() :  0.281954791393
random() :  0.309090465205
```

Number seed() Method

Description

The **seed()** method initializes the basic random number generator. Call this function before calling any other random module function.

Syntax

Following is the syntax for the seed() method-

```
seed ([x], [y])
```

Note: This function initializes the basic random number generator.

Parameters

- **x -** This is the seed for the next random number. If omitted, then it takes system time to generate the next random number. If x is an int, it is used directly.

- **Y -** This is version number (default is 2). str, byte or byte array object gets converted in int. Version 1 used hash() of x.

Return Value

This method does not return any value.

Example

The following example shows the usage of the seed() method.

```
#!/usr/bin/python3
import random
random.seed()
print ("random number with default seed", random.random())
random.seed(10)
print ("random number with int seed", random.random())
random.seed("hello",2)
print ("random number with string seed", random.random())
```

When we run above program, it produces following result-

```
random number with default seed 0.2524977842762465
random number with int seed 0.5714025946899135
random number with string seed 0.3537754404730722
```

Number shuffle() Method

Description

The **shuffle()** method randomizes the items of a list in place.

Syntax

Following is the syntax for the **shuffle()** method-

```
shuffle (lst,[random])
```

Note: This function is not accessible directly, so we need to import the shuffle module and then we need to call this function using the random static object.

Parameters

- **lst** - This could be a list or tuple.
- **random** - This is an optional 0 argument function returning float between 0.0 - 1.0. Default is None.

Return Value

This method returns reshuffled list.

Example

The following example shows the usage of the shuffle() method.

```
#!/usr/bin/python3
import random
list = [20, 16, 10, 5];
random.shuffle(list)
print ("Reshuffled list : ",  list)
random.shuffle(list)
print ("Reshuffled list : ",  list)
```

When we run the above program, it produces the following result-

```
Reshuffled list :  [16, 5, 10, 20]
reshuffled list :  [20, 5, 10, 16]
```

Number uniform() Method

Description

The **uniform()** method returns a random float r, such that x is less than or equal to r and r is less than y.

Syntax

Following is the syntax for the **uniform()** method-

```
uniform(x, y)
```

Note: This function is not accessible directly, so we need to import the uniform module and then we need to call this function using the random static object.

Parameters

- **x -** Sets the lower limit of the random float.
- **y -** Sets the upper limit of the random float.

Return Value

This method returns a floating point number r such that x <=r < y.

Example

The following example shows the usage of the uniform() method.

```
#!/usr/bin/python3
import random
```

```
print ("Random Float uniform(5, 10) : ",  random.uniform(5, 10))
print ("Random Float uniform(7, 14) : ",  random.uniform(7, 14))
```

Let us run the above program. This will produce the following result-

```
Random Float uniform(5, 10) :  5.52615217015
Random Float uniform(7, 14) :  12.5326369199
```

Trigonometric Functions

Python includes the following functions that perform trigonometric calculations.

Function	Description
acos(x)	Return the arc cosine of x, in radians.
asin(x)	Return the arc sine of x, in radians.
atan(x)	Return the arc tangent of x, in radians.
atan2(y, x)	Return atan(y / x), in radians.
cos(x)	Return the cosine of x radians.
hypot(x, y)	Return the Euclidean norm, sqrt(x*x + y*y).
sin(x)	Return the sine of x radians.
tan(x)	Return the tangent of x radians.
degrees(x)	Converts angle x from radians to degrees.
radians(x)	Converts angle x from degrees to radians.

Number acos() Method

Description

The **acos()** method returns the arc cosine of x in radians.

Syntax

Following is the syntax for acos() method-

```
acos(x)
```

Note: This function is not accessible directly, so we need to import the math module and then we need to call this function using the math static object.

Parameters

x - This must be a numeric value in the range -1 to 1. If x is greater than 1 then it will generate 'math domain error'.

Return Value

This method returns arc cosine of x, in radians.

Example

The following example shows the usage of the acos() method.

```
#!/usr/bin/python3
import math
print ("acos(0.64) : ",  math.acos(0.64))
print ("acos(0) : ",  math.acos(0))
print ("acos(-1) : ",  math.acos(-1))
print ("acos(1) : ",  math.acos(1))
```

When we run the above program, it produces the following result-

```
acos(0.64) :  0.876298061168
acos(0) :  1.57079632679
acos(-1) :  3.14159265359
acos(1) :  0.0
```

Number asin() Method

Description

The **asin()** method returns the arc sine of x (in radians).

Syntax

Following is the syntax for the asin() method-

```
asin(x)
```

Note: This function is not accessible directly, so we need to import the math module and then we need to call this function usingthe math static object.

Parameters

x - This must be a numeric value in the range -1 to 1. If x is greater than 1 then it will generate 'math domain error'.

Return Value

This method returns arc sine of x, in radians.

Example

The following example shows the usage of the asin() method.

```
#!/usr/bin/python3
import math
print ("asin(0.64) : ",  math.asin(0.64))
print ("asin(0) : ",  math.asin(0))
print ("asin(-1) : ",  math.asin(-1))
print ("asin(1) : ",  math.asin(1))
```

When we run the above program, it produces the following result-

```
asin(0.64) :  0.694498265627
asin(0) :  0.0
asin(-1) :  -1.57079632679
asin(1) :  1.5707963267
```

Number atan() Method

Description

The atan() method returns the arc tangent of x, in radians.

Syntax

Following is the syntax for atan() method-

```
atan(x)
```

Note: This function is not accessible directly, so we need to import the math module and then we need to call this function using the math static object.

Parameters

x - This must be a numeric value.

Return Value

This method returns arc tangent of x, in radians.

Example

The following example shows the usage of the atan() method.

```
#!/usr/bin/python3
import math
print ("atan(0.64) : ",  math.atan(0.64))
print ("atan(0) : ",  math.atan(0))
print ("atan(10) : ",  math.atan(10))
print ("atan(-1) : ",  math.atan(-1))
print ("atan(1) : ",  math.atan(1))
```

When we run the above program, it produces the following result-

```
atan(0.64) :  0.569313191101
atan(0) :  0.0
atan(10) :  1.4711276743
atan(-1) :  -0.785398163397
atan(1) :  0.785398163397
```

Number atan2() Method

Description

The **atan2()** method returns atan(y / x), in radians.

Syntax

Following is the syntax for atan2() method-

```
atan2(y, x)
```

Note: This function is not accessible directly, so we need to import the math module and then we need to call this function using the math static object.

Parameters

- **y** - This must be a numeric value.
- **x** - This must be a numeric value.

Return Value

This method returns atan(y / x), in radians.

Example

The following example shows the usage of atan2() method.

```
#!/usr/bin/python3
import math
print ("atan2(-0.50,-0.50) : ",  math.atan2(-0.50,-0.50))
print ("atan2(0.50,0.50) : ",  math.atan2(0.50,0.50))
print ("atan2(5,5) : ",  math.atan2(5,5))
print ("atan2(-10,10) : ",  math.atan2(-10,10))
print ("atan2(10,20) : ",  math.atan2(10,20))
```

When we run the above program, it produces the following result-

```
atan2(-0.50,-0.50) :  -2.35619449019
atan2(0.50,0.50) :  0.785398163397
atan2(5,5) :  0.785398163397
atan2(-10,10) :  -0.785398163397
atan2(10,20) :  0.463647609001
```

Number cos() Method

Description

The **cos()** method returns the cosine of x radians.

Syntax

Following is the syntax for **cos()** method-

```
cos(x)
```

Note: This function is not accessible directly, so we need to import the math module and then we need to call this function using the math static object.

Parameters

x - This must be a numeric value.

Return Value

This method returns a numeric value between -1 and 1, which represents the cosine of the angle.

Example

The following example shows the usage of cos() method.

```
#!/usr/bin/python3
import math
print ("cos(3) : ",  math.cos(3))
print ("cos(-3) : ",  math.cos(-3))
print ("cos(0) : ",  math.cos(0))
print ("cos(math.pi) : ",  math.cos(math.pi))
print ("cos(2*math.pi) : ",  math.cos(2*math.pi))
```

When we run the above program, it produces the following result-

```
cos(3) :  -0.9899924966
cos(-3) :  -0.9899924966
cos(0) :  1.0
cos(math.pi) :  -1.0
cos(2*math.pi) :  1.0
```

Number hypot() Method

Description

The method hypot() return the Euclidean norm, sqrt(x*x + y*y). This is length of vector from origin to point (x,y)

Syntax

Following is the syntax for hypot() method-

```
hypot(x, y)
```

Note: This function is not accessible directly, so we need to import math module and then we need to call this function using math static object.

Parameters

- **x** - This must be a numeric value.
- **y** - This must be a numeric value.

Return Value

This method returns Euclidean norm, sqrt(x*x + y*y).

Example

The following example shows the usage of hypot() method.

```
#!/usr/bin/python3
import math
print ("hypot(3, 2) : ",  math.hypot(3, 2))
print ("hypot(-3, 3) : ",  math.hypot(-3, 3))
print ("hypot(0, 2) : ",  math.hypot(0, 2))
```

When we run the above program, it produces the following result-

```
hypot(3, 2) :  3.60555127546
hypot(-3, 3) :  4.24264068712
hypot(0, 2) :  2.0
```

Number sin() Method

Description

The **sin()** method returns the sine of x, in radians.

Syntax

Following is the syntax for sin() method-

```
sin(x)
```

Note: This function is not accessible directly, so we need to import the math module and then we need to call this function using the math static object.

Parameters

x - This must be a numeric value.

Return Value

This method returns a numeric value between -1 and 1, which represents the sine of the parameter x.

Example

The following example shows the usage of sin() method.

```
#!/usr/bin/python3
import math
print ("sin(3) : ",  math.sin(3))
print ("sin(-3) : ",  math.sin(-3))
print ("sin(0) : ",  math.sin(0))
print ("sin(math.pi) : ",  math.sin(math.pi))
print ("sin(math.pi/2) : ",  math.sin(math.pi/2))
```

When we run the above program, it produces the following result-

```
sin(3) :  0.14112000806
sin(-3) :  -0.14112000806
sin(0) :  0.0
sin(math.pi) :  1.22460635382e-16
sin(math.pi/2) :  1
```

Number tan() Method

Description

The **tan()** method returns the tangent of x radians.

Syntax

Following is the syntax for tan() method.

```
tan(x)
```

Note: This function is not accessible directly, so we need to import math module and then we need to call this function using math static object.

Parameters

x - This must be a numeric value.

Return Value

This method returns a numeric value between -1 and 1, which represents the tangent of the parameter x.

Example

The following example shows the usage of tan() method.

```
#!/usr/bin/python3
import math
print ("(tan(3) : ",  math.tan(3))
print ("tan(-3) : ",  math.tan(-3))
print ("tan(0) : ",  math.tan(0))
print ("tan(math.pi) : ",  math.tan(math.pi))
print ("tan(math.pi/2) : ",  math.tan(math.pi/2))
print ("tan(math.pi/4) : ",  math.tan(math.pi/4))
```

When we run the above program, it produces the following result-

```
print ("(tan(3) : ",  math.tan(3))
print ("tan(-3) : ",  math.tan(-3))
print ("tan(0) : ",  math.tan(0))
print ("tan(math.pi) : ",  math.tan(math.pi))
print ("tan(math.pi/2) : ",  math.tan(math.pi/2))
print ("tan(math.pi/4) : ",  math.tan(math.pi/4))
```

Number degrees() Method

Description

The **degrees()** method converts angle x from radians to degrees..

Syntax

Following is the syntax for degrees() method-

```
degrees(x)
```

Note: This function is not accessible directly, so we need to import the math module and then we need to call this function using the math static object.

Parameters

x - This must be a numeric value.

Return Value

This method returns the degree value of an angle.

Example

The following example shows the usage of degrees() method.

```
#!/usr/bin/python3
import math
print ("degrees(3) : ",  math.degrees(3))
print ("degrees(-3) : ",  math.degrees(-3))
print ("degrees(0) : ",  math.degrees(0))
print ("degrees(math.pi) : ",  math.degrees(math.pi))
print ("degrees(math.pi/2) : ",  math.degrees(math.pi/2))
print ("degrees(math.pi/4) : ",  math.degrees(math.pi/4))
```

When we run the above program, it produces the following result-

```
degrees(3) :  171.88733853924697
degrees(-3) :  -171.88733853924697
degrees(0) :  0.0
degrees(math.pi) :  180.0
degrees(math.pi/2) :  90.0
degrees(math.pi/4) :  45.0
```

Number radians() Method

Description

The **radians()** method converts angle x from degrees to radians.

Syntax

Following is the syntax for radians() method-

```
radians(x)
```

Note: This function is not accessible directly, so we need to import the math module and then we need to call this function using the math static object.

Parameters

x - This must be a numeric value.

Return Value

This method returns radian value of an angle.

Example

The following example shows the usage of radians() method.

```
#!/usr/bin/python3
import math
print ("radians(3) : ",  math.radians(3))
print ("radians(-3) : ",  math.radians(-3))
print ("radians(0) : ",  math.radians(0))
print ("radians(math.pi) : ",  math.radians(math.pi))
print ("radians(math.pi/2) : ",  math.radians(math.pi/2))
print ("radians(math.pi/4) : ",  math.radians(math.pi/4))
```

When we run the above program, it produces the following result-

```
radians(3) :  0.0523598775598
radians(-3) :  -0.0523598775598
radians(0) :  0.0
radians(math.pi) :  0.0548311355616
radians(math.pi/2) :  0.0274155677808
radians(math.pi/4) :  0.0137077838904
```

Mathematical Constants

The module also defines two mathematical constants-

Constants	Description
pi	The mathematical constant pi.
e	The mathematical constant e.