# NUMBERS - B

***Utility modules***
**random, math, fractions.**

## RANDOM NUMBERS

To make an interesting computer game, it's good to introduce some randomness into it. Python comes with a module, called random, that allows us to use random numbers in our programs. At this point, there is only one function, called **randint**, that we will need from the **random module**. To load this function, we use the following statement:

```
from random import randint
```

### 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,vstop, 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. |

# NUMBERS - B

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

```
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
```

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

```
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
```

# NUMBERS - B

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

```
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
```

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

```
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
```

# NUMBERS - B

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

```
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]
```

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

```
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
```

Using randint is simple**: randint(a,b)** will return a random integer between a and b including both a and b. (Note that randint includes the right endpoint b unlike the range function). Here is a short example:

```
>>> from random import randint
>>> x = randint(1,10)
>>> print('A random number between 1 and 10: ', x)
A random number between 1 and 10: 7
```

The random number will be different every time we run the program.

## PYTHON'S MATH LIBRARY

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

```
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
```

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

```
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
```

# NUMBERS - B

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

```
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
```

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

```
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
```

# NUMBERS - B

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

```
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
```

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

```
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
```

# NUMBERS - B

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

```
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)
```

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

```
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
```

# NUMBERS - B

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

```
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
```

## TRIGONOMETRY

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

```
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
```

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

```
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
```

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

```
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
```

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

```
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
```

# NUMBERS - B

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

```
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
```

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

```
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
```

# NUMBERS - B

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

```
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
```

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

```
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

```
tan(3) : -0.1425465430742778
tan(-3) : 0.1425465430742778
tan(0) : 0.0
tan(math.pi) : -1.2246467991473532e-16)
tan(math.pi/2) : 1.633123935319537e+16
tan(math.pi/4) : 0.9999999999999999
```

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

```
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
```

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

```
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-

pi The mathematical constant pi.
e The mathematical constant e.

# NUMBERS - B

## FRACTIONS

```
>>> import fractions
>>> x = fractions.Fraction(1, 3)
>>> x
Fraction(1, 3)
>>> x * 2
Fraction(2, 3)
>>> fractions.Fraction(6, 4)
Fraction(3, 2)
```

1. To start using fractions, import the fractions module.
2. To define a fraction, create a Fraction object and pass in the numerator and denominator.
3. You can perform all the usual mathematical operations with fractions. Operations return a new Fraction
object. 2 * (1/3) = (2/3)
4. The Fraction object will automatically reduce fractions. (6/4) = (3/2)
5. Python has the good sense not to create a fraction with a zero denominator.

```
>>> from fractions import Fraction
>>> print(Fraction(11, 35))
11/35
>>> print(Fraction(10, 18))
5/9
>>> print(Fraction())
0
>>> print(Fraction(1.13))
1272266894732165/1125899906842624
>>> print(Fraction('1.13'))
113/100
>>> print(Fraction(18, 5) * Fraction(16, 19))
288/95
>>>
>>>import math
>>> print(math.sqrt(Fraction(28,3)))
3.0550504633038935
>>> print(Fraction(math.sin(math.pi/3)))
3900231685776981/4503599627370496
>>>
```