In [ ]:

```
a=100
```

Revisiting Numbers

In [ ]:

```
#finding absolute and power
print(pow(a,2)) #a**2
print(abs(a))
```

In [ ]:

```
x=-23.345
abs(x)
```

In [ ]:

```
y=3.14159835956
round(y,4)
```

In [ ]:

```
max(23,45,21,78,90,35)
```

In [ ]:

```
min(-23,56,-12,-90,67)
```

In [ ]:

```
#other operations using math class
#1. math.ceil
import math
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))
```

In [ ]:

```
#2. math.floor
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))
```

In [ ]:

```
#Python number method cmp() returns the sign of the difference of two numbers :
#-1 if x < y, 0 if x == y, or 1 if x > y.
#deprecated in python 3
```

In [ ]:

```
x = 80
y = 100 #cmp(x,y)=-1
print(x>y)
print(x<y)
print("Compare in python3 :",(x>y)-(x<y))
```

In [ ]:

```
import math
```

```
#Python number method exp() returns exponential of x: e power x.
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))
```

In [ ]:

```
math.exp(2)
```

In [ ]:

```
# The math.exp() method returns E raised to the power of x (Ex).

# 'E' is the base of the natural system of logarithms (approximately 2.718282)
# and x is the number passed to it.
math.exp(1)
```

In [ ]:

```
(2.718281828459045)**2
```

In [ ]:

```
#Python number method fabs() returns the absolute value of x.
print("math.fabs(-45.17) : ", math.fabs(-45.17))
print(abs(-20))
print(math.fabs(-20))
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))
```

In [ ]:

```
#Python number method log() returns natural logarithm of x, for x > 0.
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))
```

In [ ]:

```
#Python number method log10() returns base-10 logarithm of x for x > 0.
print("math.log10(100.12) : ", math.log10(100.12))
print("math.log10(100.72) : ", math.log10(100.72))
print("math.log10(math.pi) : ", math.log10(math.pi))
```

In [ ]:

```
#Python number method modf() 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.

print("math.modf(100.12) : ", math.modf(-100.12))
print("math.modf(100.72) : ", math.modf(100.72))
print("math.modf(math.pi) : ", math.modf(math.pi))
```

In [ ]:

```
#Python number method pow() returns x to the power of y. If the third argument (z) is giv
en,
#it returns x to the power of y modulus z, i.e. pow(x, y) % z.
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))
```

In [ ]:

```
#Python number method sqrt() returns the square root of x for x > 0.
print("math.sqrt(100) : ", math.sqrt(100))
```

```
print("math.sqrt(7) : ", math.sqrt(7))
print("math.sqrt(math.pi) : ", math.sqrt(math.pi))
```

In [3]:

```
#using random to generate the random number
#in order to use any function related to random number generation,
#it is mandatory to import the random module
import random
listr1=[1,2,3,4]
#choice randomly takes any value from the given list
print(random.choice(listr1))
#choice can be used with any sequential data
string1="Kavita"
print(random.choice(string1))

dictr1={'k1':'One','k2':'two','k3':'three'}
#print(random.choice(dictr1))
keys=list(dictr1.keys())
keys
print(random.choice(keys))
```

```
1
v
k2
```

In [10]:

```
#generate a number between 0 and 1
print(random.random())
```

```
0.5293970187902274
```

In [11]:

```
print(random.random())
random.seed(1)
print(random.random())

#Again printing any arbitrary random number
#print(random.random())
random.seed(2)
print(random.random())

print(random.random())
random.seed(1)
print(random.random())
```

```
0.0610756302208012
0.13436424411240122
0.9560342718892494
0.9478274870593494
0.13436424411240122
```

In [ ]:

```
#randrange function- The random module offers a function that can generate random numbers
from a specified range and also allowing rooms for steps to be included, called randrange
().
```

In [ ]:

```
print(random.random())
```

In [ ]:

```
# using random() to generate a random number between 0 and 1
print ("A random number between 0 and 1 is : ", end="")
print (random.random())

# using seed() to seed a random number
```

```
random.seed(5)

# printing mapped random number
print ("The mapped random number with 5 is : ", end="")
print (random.random())

# using seed() to seed different random number
random.seed(7)

# printing mapped random number
print ("The mapped random number with 7 is : ", end="")
print (random.random())

# using seed() to seed to 5 again
random.seed(5)

# printing mapped random number
print ("The mapped random number with 5 is : ",end="")
print (random.random())

# using seed() to seed to 7 again
random.seed(7)
print(random.random())
```

In [17]:

```
#generate the random number between 0 and 100
num=int((random.random())*100)
num
```

Out[17]:

```
65
```

In [ ]:

```
random.seed(1)
print(random.random())
```

In [18]:

```
sample_list=list(range(100))
print(sample_list)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47
, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69,
70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92
, 93, 94, 95, 96, 97, 98, 99]
```

In [21]:

```
#shuffling the data from the list
random.shuffle(sample_list)
sample_list
```

Out[21]:

```
[86,
 32,
 57,
 78,
 68,
 23,
 85,
 27,
 70,
 37,
 16,
 0,
 46,
 92,
 91
```

91,
90,
60,
31,
58,
94,
67,
71,
11,
42,
20,
9,
62,
97,
73,
49,
88,
99,
41,
10,
30,
81,
82,
95,
87,
74,
40,
5,
36,
38,
33,
64,
83,
75,
65,
21,
45,
12,
18,
44,
8,
3,
6,
13,
2,
79,
29,
61,
25,
50,
1,
59,
89,
77,
28,
48,
63,
7,
56,
39,
35,
22,
52,
80,
53,
14,
51,
96,
69,
72,
98,
76,
24

```
 24,
 19,
 15,
 47,
 55,
 93,
 66,
 4,
 34,
 84,
 43,
 17,
 54,
 26]
```

In [36]:

```
#randrange function
#The random module offers a function that can generate random numbers
# from a specified range and also allowing rooms for steps to be included,
# called randrange().

print(random.randrange(20,60,2))

#print(random.randrange(4.5,25.5,2.5))
```

```
24
```

In [43]:

```
#using random(a,b) generates a floating point random number between a(inclusive)
#and b(non-inclusive)

print(random.uniform(5,15))
```

```
14.26165442005095
```

In [49]:

```
#Generating integer random number

print(random.randint(0,100))
```

```
10
```

In [58]:

```
#selects more than one random values from the given sequence
new_list=[1,2,3,1,2,3,1,2,3]
print(random.sample(new_list,3))
```

```
[1, 3, 3]
```

Strings revisited

In [59]:

```
string1="sample String"
print(string1.capitalize()) #capitalize the first letter
```

```
Sample string
```

In [63]:

```
print(string1.casefold()) #all the characters in lower case
print(string1)
print(string1.lower())
```

```
sample string
sample String
sample string
```

In [64]:

```
len(string1)
```

Out[64]:

```
13
```

In [65]:

```
print(string1.center(20,'a')) #change the string length to 20 with a padded
```

```
aaasample Stringaaaa
```

In [66]:

```
print(string1.count('a'))
```

```
1
```

In [67]:

```
type((string1.encode(encoding="ascii")))
```

Out[67]:

```
bytes
```

In [68]:

```
print(string1.endswith('ple'))
print(string1.endswith('ing'))
```

```
False
True
```

In [70]:

```
print(string1)
suffix='ple'

print(string1.endswith(suffix,0,6))
```

```
sample String
True
```

In [71]:

```
print(string1.find('ample'))
print(string1.find('ing'))
print(string1.find('ample',3))
sample_string="The demo string for string function"
print(sample_string.find('string'))
print(sample_string.rfind('string')) #gives the last occurence
```

```
1
10
-1
9
20
```

In [76]:

```
string2="Company12 string!!"
print(string2.isalnum())
```

```
False
```

In [78]:

```
string3="Hello "
```

```
print(string3.isalpha()) #should strictly contains alphabet, not even space
print(string3)
```

```
False
Hello
```

In [84]:

```
string4='def'
print(string4.isdigit()) #only digits are permissible
```

```
False
```

In [85]:

```
print(string4.islower())
```

```
True
```

In [86]:

```
string5=' '
print(string5.isspace())
```

```
True
```

In [89]:

```
string6="Ademostring!!"
print(max(string6))
print(min(string6))
print(len(string6))
```

```
t
!
13
```

In [90]:

```
string6.lower() #converts into lower case
```

Out[90]:

```
'ademostring!!'
```

In [91]:

```
string6.upper() #converts into upper case
```

Out[91]:

```
'ADEMOSTRING!!'
```

In [92]:

```
string6.swapcase()
```

Out[92]:

```
'aDEMOSTRING!!'
```

In [93]:

```
string6.index('demo')
#print(string6)
```

Out[93]:

```
1
```

In [94]:

```
string7="This is a demo string to demonstrate the function"
```

```
print(string7.index('demo'))
print(string7.index('demo',15)) #starting from index 15
print(string7.rindex('demo')) #gives the last index
```

```
10
25
25
```

In [95]:

```
print(string7.replace("is","was"))
string7
```

```
Thwas was a demo string to demonstrate the function
```

Out[95]:

```
'This is a demo string to demonstrate the function'
```

In [96]:

```
#using split method
print(string7.split(' '))
print(string7.split(' ',2)) #limites the separator
```

```
['This', 'is', 'a', 'demo', 'string', 'to', 'demonstrate', 'the', 'function']
['This', 'is', 'a demo string to demonstrate the function']
```

In [97]:

```
another_demo="Hello \n this is a \n another demo \n string"
another_demo
```

Out[97]:

```
'Hello \n this is a \n another demo \n string'
```

In [98]:

```
another_demo.splitlines() #only if new line is found
```

Out[98]:

```
['Hello ', ' this is a ', ' another demo ', ' string']
```

In [99]:

```
just="  Hello   this  is a string"
print(just.ljust(50,"*")) #create the length of string as 50 and left justifies after fil
ling *
```

```
  Hello   this  is a string**********************
```

In [100]:

```
print(just.rjust(50,"*")) #create the length of string as 500 and righ justifies after fi
lling *
```

```
**********************  Hello   this  is a string
```

In [101]:

```
print(just.strip()) #removes spaces from the start and end of the string
```

```
Hello   this  is a string
```

In [105]:

```
just1='****Hello this is a string*****'
print(just1)
print(just1.strip('*'))  #removes occurences of * from beggining and end

print(just1.rstrip('*')) #removes right spaces
```

```
print(just1.lstrip('*')) #removes left spaces
```

```
****Hello this is a string*****
Hello this is a string
****Hello this is a string
Hello this is a string*****
```

In [103]:

```
just.zfill(40) #zero fill at the start of the string to make length as 40
```

Out[103]:

```
'0000000000000  Hello   this  is a string'
```

In [104]:

```
just.swapcase() #swap the cases of original string
```

Out[104]:

```
'  hELLO   THIS  IS A STRING'
```