Python Module: Python Math Module

In [1]:
```
# importing the required module
import math
```

In [22]:
```
print (math.sin(0), math.cos(0), math.tan(0))
print (math.pow(10, 3), 10 ** 3, 10.0 ** 3, 10 ** 3.0)
print ("pi:", math.pi)
print ("e:", math.e)
print ("tau:", math.tau)
print (math.log(1000), math.log(1000, math.e), math.log(1000, 10))
print (math.log2(1024), math.log(1024, 2))
print (math.log10(1000), math.log(1000, 10))
print (math.ceil(10.1), math.ceil(10.9), math.floor(10.1), math.floor(10.9))
print (math.factorial(5), math.factorial(7))
print ("GCD:", math.gcd(1000, 450), "LCM:", (1000 * 450) / math.gcd(1000, 450))
print (abs(-1000), abs(1000))  # abs() function is inbuilt
```

```
0.0 1.0 0.0
1000.0 1000 1000.0 1000.0
pi: 3.141592653589793
e: 2.718281828459045
tau: 6.283185307179586
6.907755278982137 6.907755278982137 2.9999999999999996
10.0 10.0
3.0 2.9999999999999996
11 11 10 10
120 5040
GCD: 50 LCM: 9000.0
1000 1000
```

In [23]:
```
help (math.pow)
```

```
Help on built-in function pow in module math:

pow(x, y, /)
    Return x**y (x to the power of y).
```

In [ ]:
```
help (math)
```

Dealing with String

> String or text consists of alpha-numeric characters and special characters

string indexing and slicing

index from left to right: 0 1 2 3 4 5 6 7 8 9 mystr: u n i v e r s i t y index from right to left: -10 -9 -8 -7 -6 -5 -4 -3 -2 -1

In [86]:
```
mystr = "university"
print (mystr, len(mystr), type(mystr), id(mystr))
print (mystr[3], mystr[-7], mystr[8], mystr[-2])  # indexing
mystr[3] = "x"    # will cause ERROR as string is immutable
print (mystr[0:6], mystr[-10:-4], mystr[:6], mystr[:-4])  # slicing means sub-string retrieval
print (mystr[3:9], mystr[-7:-1], mystr[3:-1], mystr[-7:9])
print (mystr[::2], mystr[1::2], mystr[::-1], mystr[8::-2], mystr[9::-2])
print (mystr[3:6], mystr[-7:-4], mystr[3:-4], mystr[-7:6])
```

```
university 10 <class 'str'> 2199054384048
v v t t
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-86-a9bf2aa5da4b> in <module>
      2 print (mystr, len(mystr), type(mystr), id(mystr))
      3 print (mystr[3], mystr[-7], mystr[8], mystr[-2])  # indexing
----> 4 mystr[3] = "x"
      5 print (mystr[0:6], mystr[-10:-4], mystr[:6], mystr[:-4])  # slicing means sub-string retrieval
      6 print (mystr[3:9], mystr[-7:-1], mystr[3:-1], mystr[-7:9])

TypeError: 'str' object does not support item assignment
```

In [43]:
```
mystr = "caLCuTTa uniVeRSitY"
print (mystr, len(mystr), type(mystr), id(mystr))
print (mystr.upper())
print (mystr.lower())
print (mystr.capitalize())
```

```
print (mystr.title())
print (mystr.center(40))
print (mystr.swapcase())
```

```
caLCuTTa uniVeRSitY 19 <class 'str'> 2199053140016
CALCUTTA UNIVERSITY
calcutta university
Calcutta university
Calcutta University
          caLCuTTa uniVeRSitY
CAlcUttA UNIvErsITy
```

In [87]:

```
mystr = "caLCuTTa uniVeRSitY"
print (mystr, len(mystr), type(mystr), id(mystr))
capstr = mystr.upper()
print (capstr, len(capstr), type(capstr), id(capstr))
```

```
caLCuTTa uniVeRSitY 19 <class 'str'> 2199058219296
CALCUTTA UNIVERSITY 19 <class 'str'> 2199058219616
```

In [44]:

```
mystr = "abcd"
print (mystr, mystr.isalpha(), mystr.isalnum(), mystr.isdigit())
mystr = "1234"
print (mystr, mystr.isalpha(), mystr.isalnum(), mystr.isdigit())
mystr = "abcd1234"
print (mystr, mystr.isalpha(), mystr.isalnum(), mystr.isdigit())
mystr = "abcd@1234"
print (mystr, mystr.isalpha(), mystr.isalnum(), mystr.isdigit())
```

```
abcd True True False
1234 False True True
abcd1234 False True False
abcd@1234 False False False
```

In [ ]:

```
help(str)
```

In [48]:

```
mystr = "Good morning"
print (mystr)
mystr = mystr.replace("morning", "night")
print (mystr)
mystr = mystr.replace("evening", "afternoon")
print (mystr)
```

```
Good morning
Good night
Good night
```

In [52]:

```
mystr = "charity begins at home"
print (mystr)
print (mystr.find("at"))
print (mystr.find("at", 10))
print (mystr.find("at", 15))
print (mystr.find("at", 16))
```

```
charity begins at home
15
15
15
-1
```

In [56]:

```
mystr = "charity begins at home"
print (mystr)
print (mystr.find("i"))
print (mystr.find("i", 10))
print (mystr.find("i", 11))
print (mystr.find("i", 16))
```

```
charity begins at home
4
11
11
-1
```

In [58]:

```
try:
    mystr = "charity begins at home"
    print (mystr)
    print (mystr.index("i"))
    print (mystr.index("i", 10))
    print (mystr.index("i", 11))
    print (mystr.index("i", 16))
```

```
except ValueError as ve:
    print (f"Exception type: {type(ve)} and error message: {ve}...")
```

```
charity begins at home
4
11
11
Exception type: <class 'ValueError'> and error message: substring not found...
```

In [59]:

```
mystr = "university"
print (mystr)
print (mystr.count("u"), mystr.count("i"), mystr.count("x"))
```

```
university
1 2 0
```

In [65]:

```
mystr = "charity begins at home"
print (mystr)
print (mystr.endswith("home"), mystr.endswith("Home"))
print (mystr.endswith("at"))
print (mystr.endswith("at", 0, 17))
```

```
charity begins at home
True False
False
True
```

In [71]:

```
mystr = "charity begins at home"
print (mystr)
print (mystr.startswith("char"))
print (mystr.startswith("gin"))
print (mystr.startswith("gin", 10))
print (mystr.startswith("gin", 10, 15))
print (mystr.startswith("gin", 10, 12))
```

```
charity begins at home
True
False
True
True
False
```

In [75]:

```
mystr = "    calcutta    university     "
print (mystr, len(mystr))
print (mystr.strip(), len(mystr.strip()))
print (mystr.lstrip(), len(mystr.lstrip()))
print (mystr.rstrip(), len(mystr.rstrip()))
```

```
    calcutta    university      31
calcutta    university 22
calcutta    university      27
    calcutta    university 26
```

In [77]:

```
mystr = "#@@#calcutta #@ university@@##@"
print (mystr, len(mystr))
print (mystr.strip("#@"), len(mystr.strip("#@")))
print (mystr.lstrip("#@"), len(mystr.lstrip("#@")))
print (mystr.rstrip("#@"), len(mystr.rstrip("#@")))
```

```
#@@#calcutta #@ university@@##@ 31
calcutta #@ university 22
calcutta #@ university@@##@ 27
#@@#calcutta #@ university 26
```

In [81]:

```
mystr = "university"
for ch in mystr:
    print (ch, end = ", ")
print ()
for i in range(len(mystr)):
    print (mystr[i], end = ", ")
```

```
u, n, i, v, e, r, s, i, t, y,
u, n, i, v, e, r, s, i, t, y,
```

In [83]:

```
mystr = "charity begins at home"
print (mystr)
list1 = mystr.split(" ")
print (list1, type(list1))
list1 = mystr.split("i")
print (list1, type(list1))
```

```
charity begins at home
```

```
['charity', 'begins', 'at', 'home'] <class 'list'>
['char', 'ty beg', 'ns at home'] <class 'list'>
```

In [85]:
```
list1 = ['charity', 'begins', 'at', 'home']
mystr = " ".join(list1)
print (mystr)
list1 = ['char', 'ty beg', 'ns at home']
mystr = "i".join(list1)
print (mystr)
```

```
charity begins at home
charity begins at home
```

Dealing with List

> A list consists of items of same or different data types separated by commas and enclosed within []. List is mutable[]. That means we can perform insert, delete and update operations.

In [94]:
```
list1 = [100, 400, False, 200.5, 300, 500, True]
print (list1, len(list1), type(list1), id(list1))
print (max(list1), min(list1))
print (sum(list1), sum(list1)/len(list1))
```

```
[100, 400, False, 200.5, 300, 500, True] 7 <class 'list'> 2199063381760
500 False
1501.5 214.5
```

In [97]:
```
list1 = ["Monday", "Friday", "Tuesday", "thursday"]
print (list1, len(list1), type(list1), id(list1))
print (max(list1), min(list1))
# print (sum(list1), sum(list1)/len(list1))
```

```
['Monday', 'Friday', 'Tuesday', 'thursday'] 4 <class 'list'> 2199057656000
thursday Friday
```

In [101…
```
list1 = ["Monday", "Friday", 100, "Tuesday", 34.56, True,"Thursday"]
print (list1, len(list1), type(list1), id(list1))
# print (max(list1), min(list1))
# print (sum(list1), sum(list1)/len(list1))
```

```
['Monday', 'Friday', 100, 'Tuesday', 34.56, True, 'Thursday'] 7 <class 'list'> 2199057211136
```

In [112…
```
# indexing and slicing
#  L2R =>   0         1         2          3
daylist = ["Monday", "Friday", "Tuesday", "Thursday"]
#  R2L =>  -4        -3        -2         -1
print (daylist, len(daylist))
print (daylist[1], daylist[-3])
print (daylist[1:], daylist[-3:])
print (daylist[2][2:5], daylist[-2][-5:-2])
```

```
['Monday', 'Friday', 'Tuesday', 'Thursday'] 4
Friday Friday
['Friday', 'Tuesday', 'Thursday'] ['Friday', 'Tuesday', 'Thursday']
esd esd
```

In [127…
```
colorlist = ["Red", "Black", "Brown", "Yellow"]
print (colorlist, len(colorlist))
daylist = ["Monday", "Friday", "Tuesday", "Thursday"]
print (daylist, len(daylist))
list1 = colorlist + daylist
print (list1, len(list1))
list1 = daylist + colorlist
print (list1, len(list1))
daylist.extend(colorlist)
print (daylist, len(daylist))
```

```
['Red', 'Black', 'Brown', 'Yellow'] 4
['Monday', 'Friday', 'Tuesday', 'Thursday'] 4
['Red', 'Black', 'Brown', 'Yellow', 'Monday', 'Friday', 'Tuesday', 'Thursday'] 8
['Monday', 'Friday', 'Tuesday', 'Thursday', 'Red', 'Black', 'Brown', 'Yellow'] 8
['Monday', 'Friday', 'Tuesday', 'Thursday', 'Red', 'Black', 'Brown', 'Yellow'] 8
```

In [114…
```
colorlist = ["Red", "Black", "Brown", "Yellow"]
print (colorlist, len(colorlist))
daylist = ["Monday", "Friday", "Tuesday", "Thursday"]
print (daylist, len(daylist))
list1 = [colorlist, daylist]   # list of lists
print (list1, len(list1))
```

```
list2 = daylist + colorlist    # list concatenation
print (list2, len(list2))
```

```
['Red', 'Black', 'Brown', 'Yellow'] 4
['Monday', 'Friday', 'Tuesday', 'Thursday'] 4
[['Red', 'Black', 'Brown', 'Yellow'], ['Monday', 'Friday', 'Tuesday', 'Thursday']] 2
['Monday', 'Friday', 'Tuesday', 'Thursday', 'Red', 'Black', 'Brown', 'Yellow'] 8
```

In [119…

```
list1 = [['Red', 'Black', 'Brown', 'Yellow'], ['Monday', 'Friday', 'Tuesday', 'Thursday']]
print (list1[1][2], list1[-1][-2])  # indexing
print (list1[0][2][1:4], list1[-2][-2][-4:-1])  # slicing
```

```
Tuesday Tuesday
row row
```

In [125…

```
list1 = [10] * 5
print (list1, len(list1))
list1 = [[10] * 3] * 4
print (list1, len(list1))
list1 = [[10 for i in range(5)] for j in range(3)]
print (list1, len(list1))
```

```
[10, 10, 10, 10, 10] 5
[[10, 10, 10], [10, 10, 10], [10, 10, 10], [10, 10, 10]] 4
[[10, 10, 10, 10, 10], [10, 10, 10, 10, 10], [10, 10, 10, 10, 10]] 3
```

In [133…

```
list1 = ['Red', 'Black', 'Brown', 'Yellow']
print (list1, len(list1), type(list1), id(list1))
list1.append("Silver")
list1.append("Magenta")
list1.append("Golden")
print (list1, len(list1), type(list1), id(list1))
list1.insert(3, "Cyan")
print (list1, len(list1), type(list1), id(list1))
list1.insert(500, "Blue")
print (list1, len(list1), type(list1), id(list1))
```

```
['Red', 'Black', 'Brown', 'Yellow'] 4 <class 'list'> 2199061612480
['Red', 'Black', 'Brown', 'Yellow', 'Silver', 'Magenta', 'Golden'] 7 <class 'list'> 2199061612480
['Red', 'Black', 'Brown', 'Cyan', 'Yellow', 'Silver', 'Magenta', 'Golden'] 8 <class 'list'> 2199061612480
['Red', 'Black', 'Brown', 'Cyan', 'Yellow', 'Silver', 'Magenta', 'Golden', 'Blue'] 9 <class 'list'> 2199061612480
```

In [135…

```
list1 = ['Red', 'Black', 'Brown', 'Cyan', 'Yellow', 'Silver', 'Magenta', 'Golden']
print (list1, len(list1), type(list1), id(list1))
del list1
print (list1, len(list1), type(list1), id(list1))
```

```
['Red', 'Black', 'Brown', 'Cyan', 'Yellow', 'Silver', 'Magenta', 'Golden'] 8 <class 'list'> 2199054465536
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-135-647681fcefbf> in <module>
      2 print (list1, len(list1), type(list1), id(list1))
      3 del list1
----> 4 print (list1, len(list1), type(list1), id(list1))

NameError: name 'list1' is not defined
```

In [136…

```
list1 = ['Red', 'Black', 'Brown', 'Cyan', 'Yellow', 'Silver', 'Magenta', 'Golden']
print (list1, len(list1), type(list1), id(list1))
list1.clear()
print (list1, len(list1), type(list1), id(list1))
```

```
['Red', 'Black', 'Brown', 'Cyan', 'Yellow', 'Silver', 'Magenta', 'Golden'] 8 <class 'list'> 2199053815488
[] 0 <class 'list'> 2199053815488
```

In [138…

```
list1 = ['Red', 'Black', 'Brown', 'Cyan', 'Yellow', 'Silver', 'Magenta', 'Golden']
print (list1, len(list1), type(list1), id(list1))
list1.remove("Brown")
print (list1, len(list1), type(list1), id(list1))
list1.remove("Cyan")
print (list1, len(list1), type(list1), id(list1))
list1.remove("Green")
print (list1, len(list1), type(list1), id(list1))
```

```
['Red', 'Black', 'Brown', 'Cyan', 'Yellow', 'Silver', 'Magenta', 'Golden'] 8 <class 'list'> 2199063558400
['Red', 'Black', 'Cyan', 'Yellow', 'Silver', 'Magenta', 'Golden'] 7 <class 'list'> 2199063558400
['Red', 'Black', 'Yellow', 'Silver', 'Magenta', 'Golden'] 6 <class 'list'> 2199063558400
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-138-ae262204b5b3> in <module>
      5 list1.remove("Cyan")
      6 print (list1, len(list1), type(list1), id(list1))
```

```
----> 7 list1.remove("Green")
      8 print (list1, len(list1), type(list1), id(list1))

ValueError: list.remove(x): x not in list
```

In [139…]
```
list1 = ['Red', 'Black', 'Brown', 'Cyan', 'Yellow', 'Silver', 'Magenta', 'Golden']
print (list1, len(list1), type(list1), id(list1))
print (list1.pop())
print (list1.pop())
print (list1.pop())
print (list1.pop())
print (list1, len(list1), type(list1), id(list1))
```

```
['Red', 'Black', 'Brown', 'Cyan', 'Yellow', 'Silver', 'Magenta', 'Golden'] 8 <class 'list'> 2199068070016
Golden
Magenta
Silver
Yellow
['Red', 'Black', 'Brown', 'Cyan'] 4 <class 'list'> 2199068070016
```

In [140…]
```
list1 = ['Red', 'Black', 'Brown', 'Cyan', 'Yellow', 'Silver', 'Magenta', 'Golden']
print (list1, len(list1), type(list1), id(list1))
print (list1.pop(3))
print (list1, len(list1), type(list1), id(list1))
print (list1.pop(4))
print (list1, len(list1), type(list1), id(list1))
print (list1.pop(2))
print (list1, len(list1), type(list1), id(list1))
print (list1.pop(1))
print (list1, len(list1), type(list1), id(list1))
```

```
['Red', 'Black', 'Brown', 'Cyan', 'Yellow', 'Silver', 'Magenta', 'Golden'] 8 <class 'list'> 2199053425472
Cyan
['Red', 'Black', 'Brown', 'Yellow', 'Silver', 'Magenta', 'Golden'] 7 <class 'list'> 2199053425472
Silver
['Red', 'Black', 'Brown', 'Yellow', 'Magenta', 'Golden'] 6 <class 'list'> 2199053425472
Brown
['Red', 'Black', 'Yellow', 'Magenta', 'Golden'] 5 <class 'list'> 2199053425472
Black
['Red', 'Yellow', 'Magenta', 'Golden'] 4 <class 'list'> 2199053425472
```

In [141…]
```
list1 = ['Red', 'Black', 'Brown', 'Cyan', 'Yellow']
print (list1, len(list1), type(list1), id(list1))
list1[3] = "Silver"
print (list1, len(list1), type(list1), id(list1))
list1[4] = "Golden"
print (list1, len(list1), type(list1), id(list1))
```

```
['Red', 'Black', 'Brown', 'Cyan', 'Yellow'] 5 <class 'list'> 2199061767232
['Red', 'Black', 'Brown', 'Silver', 'Yellow'] 5 <class 'list'> 2199061767232
['Red', 'Black', 'Brown', 'Silver', 'Golden'] 5 <class 'list'> 2199061767232
```

In [143…]
```
list1 = ['Red', 'Black', 'Brown', 'Cyan', 'Yellow']
print (list1, len(list1), type(list1), id(list1))
list1.sort()
print (list1, len(list1), type(list1), id(list1))
list1.sort(reverse=True)
print (list1, len(list1), type(list1), id(list1))
```

```
['Red', 'Black', 'Brown', 'Cyan', 'Yellow'] 5 <class 'list'> 2199057196032
['Black', 'Brown', 'Cyan', 'Red', 'Yellow'] 5 <class 'list'> 2199057196032
['Yellow', 'Red', 'Cyan', 'Brown', 'Black'] 5 <class 'list'> 2199057196032
```

In [144…]
```
list1 = ['Red', 'Black', 'Brown', 'Cyan', 'Yellow']
print (list1, len(list1), type(list1), id(list1))
list1 = sorted(list1)
print (list1, len(list1), type(list1), id(list1))
list1 = sorted(list1, reverse=True)
print (list1, len(list1), type(list1), id(list1))
```

```
['Red', 'Black', 'Brown', 'Cyan', 'Yellow'] 5 <class 'list'> 2199061222400
['Black', 'Brown', 'Cyan', 'Red', 'Yellow'] 5 <class 'list'> 2199061225088
['Yellow', 'Red', 'Cyan', 'Brown', 'Black'] 5 <class 'list'> 2199057642816
```

In [148…]
```
list1 = ['Red', 'Black', 'Brown', 'Cyan', 'Yellow']
print (list1, len(list1), type(list1), id(list1))
print (all(list1), any(list1))
list1 = ['Red', 'Black', 'Brown', False, 'Cyan', 'Yellow']
print (list1, len(list1), type(list1), id(list1))
print (all(list1), any(list1))
list1 = ['Red', 'Black', '', 'Cyan', 'Yellow']
print (list1, len(list1), type(list1), id(list1))
print (all(list1), any(list1))
```

```
list1 = ['', '', '', '']
print (list1, len(list1), type(list1), id(list1))
print (all(list1), any(list1))
```

```
['Red', 'Black', 'Brown', 'Cyan', 'Yellow'] 5 <class 'list'> 2199060958336
True True
['Red', 'Black', 'Brown', False, 'Cyan', 'Yellow'] 6 <class 'list'> 2199060901056
False True
['Red', 'Black', '', 'Cyan', 'Yellow'] 5 <class 'list'> 2199061299328
False True
['', '', '', ''] 4 <class 'list'> 2199060463168
False False
```

In [150…

```
list1 = ['Red', 'Black', 'Brown', 'Cyan', 'Yellow']
print (list1, len(list1), type(list1), id(list1))
print (list1.index("Black"), list1.index("Cyan"))
print (list1.index("Silver"))
```

```
['Red', 'Black', 'Brown', 'Cyan', 'Yellow'] 5 <class 'list'> 2199058782656
1 3
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-150-d9a86219478c> in <module>
      2 print (list1, len(list1), type(list1), id(list1))
      3 print (list1.index("Black"), list1.index("Cyan"))
----> 4 print (list1.index("Silver"))

ValueError: 'Silver' is not in list
```

In [153…

```
list1 = ['Red', 'Black', 'Brown', 'Cyan', 'Yellow', 'Brown', 'Cyan', 'Yellow']
print (list1, len(list1), type(list1), id(list1))
print (list1.count('Yellow'), list1.count('Cyan'), list1.count('Red'))
```

```
['Red', 'Black', 'Brown', 'Cyan', 'Yellow', 'Brown', 'Cyan', 'Yellow'] 8 <class 'list'> 2199055253696
2 2 1
```

In [ ]:

```
help(list)
```

Class Assignment-2

- Take n number of numbers from the user and put them in a list and print it.
- Then split the list into even_list and odd_list depending on whether the number is even or odd.
- And print those two newly created lists.

In [155…

```
num = int(input("Please enter the number of terms: "))
num_list = []
for i in range(num):
    data = int(input("Please enter item number " + str(i) + ": "))
    num_list.append(data)
print ("So the data list is", num_list)
print ("Now segregating numbers into even list and odd list...")
even_list = []
odd_list = []
for data in num_list:
    if (data % 2 == 0):
        even_list.append(data)
    else:
        odd_list.append(data)
print ("So the list containing EVEN numbers:", even_list)
print ("So the list containing ODD numbers:", odd_list)
print ("End of the program...")
```

```
So the data list is [22, 11, 44, 33, 66]
Now segregating numbers into even list and odd list...
So the list containing EVEN numbers: [22, 44, 66]
So the list containing ODD numbers: [11, 33]
End of the program...
```

Dealing with Tuple

> A tuple consists of items of same or different data types separated by commas and enclosed within (). Tuple is immutable[]. That means we can not perform insert, delete and update operations.

In [156…

```
tuple1 = (100, 400, False, 200.5, 300, 500, True)
print (tuple1, len(tuple1), type(tuple1), id(tuple1))
print (max(tuple1), min(tuple1))
print (sum(tuple1), sum(tuple1)/len(tuple1))
```

```
(100, 400, False, 200.5, 300, 500, True) 7 <class 'tuple'> 2199037333120
```

```
500 False
1501.5 214.5
```

In [158…]
```
tuple1 = ("Monday", "Friday", "Tuesday", "Thursday")
print (tuple1, len(tuple1), type(tuple1), id(tuple1))
print (max(tuple1), min(tuple1))
# print (sum(tuple1), sum(tuple1)/len(tuple1))
```

```
('Monday', 'Friday', 'Tuesday', 'Thursday') 4 <class 'tuple'> 2199065385856
Tuesday Friday
```

In [161…]
```
tuple1 = ("Monday", "Friday", 100, "Tuesday", 34.56, True,"Thursday")
print (tuple1, len(tuple1), type(tuple1), id(tuple1))
# print (max(tuple1), min(tuple1))
# print (sum(tuple1), sum(tuple1)/len(tuple1))
```

```
('Monday', 'Friday', 100, 'Tuesday', 34.56, True, 'Thursday') 7 <class 'tuple'> 2199037333120
```

In [162…]
```
# indexing and slicing
#  L2R =>   0         1         2          3
daytuple = ("Monday", "Friday", "Tuesday", "Thursday")
#  R2L =>  -4        -3        -2         -1
print (daytuple, len(daytuple))
print (daytuple[1], daytuple[-3])
print (daytuple[1:], daytuple[-3:])
print (daytuple[2][2:5], daytuple[-2][-5:-2])
```

```
('Monday', 'Friday', 'Tuesday', 'Thursday') 4
Friday Friday
('Friday', 'Tuesday', 'Thursday') ('Friday', 'Tuesday', 'Thursday')
esd esd
```

In [3]:
```
colortuple = ("Red", "Black", "Brown", "Yellow")
print (colortuple, len(colortuple))
daytuple = ("Monday", "Friday", "Tuesday", "Thursday")
print (daytuple, len(daytuple))
tuple1 = colortuple + daytuple
print (tuple1, len(tuple1))
tuple1 = daytuple + colortuple
print (tuple1, len(tuple1))
```

```
('Red', 'Black', 'Brown', 'Yellow') 4
('Monday', 'Friday', 'Tuesday', 'Thursday') 4
('Red', 'Black', 'Brown', 'Yellow', 'Monday', 'Friday', 'Tuesday', 'Thursday') 8
('Monday', 'Friday', 'Tuesday', 'Thursday', 'Red', 'Black', 'Brown', 'Yellow') 8
```

In [5]:
```
colortuple = ("Red", "Black", "Brown", "Yellow")
print (colortuple, len(colortuple))
daytuple = ("Monday", "Friday", "Tuesday", "Thursday")
print (daytuple, len(daytuple))
tuple1 = (colortuple, daytuple)   # tuple of tuples
print (tuple1, len(tuple1))
tuple2 = daytuple + colortuple    # tuple concatenation
print (tuple2, len(tuple2))
```

```
('Red', 'Black', 'Brown', 'Yellow') 4
('Monday', 'Friday', 'Tuesday', 'Thursday') 4
((('Red', 'Black', 'Brown', 'Yellow'), ('Monday', 'Friday', 'Tuesday', 'Thursday')) 2
('Monday', 'Friday', 'Tuesday', 'Thursday', 'Red', 'Black', 'Brown', 'Yellow') 8
```

In [6]:
```
tuple1 = (('Red', 'Black', 'Brown', 'Yellow'), ('Monday', 'Friday', 'Tuesday', 'Thursday'))
print (tuple1[1][2], tuple1[-1][-2])  # indexing
print (tuple1[0][2][1:4], tuple1[-2][-2][-4:-1])  # slicing
```

```
Tuesday Tuesday
row row
```

In [11]:
```
colortuple = ("Red", "Black", "Brown", "Yellow")
print (colortuple, len(colortuple))
print (colortuple.index("Red"))
print (colortuple.index("Brown"))
print (colortuple.index("yellow"))
```

```
('Red', 'Black', 'Brown', 'Yellow') 4
0
2
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-11-038296ee2237> in <module>
      3 print (colortuple.index("Red"))
      4 print (colortuple.index("Brown"))
----> 5 print (colortuple.index("yellow"))
```

```
ValueError: tuple.index(x): x not in tuple
```

In [10]:
```
colortuple = ("Red", "Black", "Brown", "Yellow", "Black", "Brown", "Yellow", "Black", "Brown", "Yellow")
print (colortuple, len(colortuple))
print (colortuple.count("Red"))
print (colortuple.count("Brown"))
print (colortuple.count("Green"))
```

```
('Red', 'Black', 'Brown', 'Yellow', 'Black', 'Brown', 'Yellow', 'Black', 'Brown', 'Yellow') 10
1
3
0
```

In [15]:
```
colortuple = ("Red", "Black", "Brown", "Yellow")
print (colortuple, len(colortuple), id(colortuple))
# colortuple[2] = "Green"
colorlist = list(colortuple)
print(colorlist, len(colorlist), id(colorlist))
colorlist[2] = "Green"
print(colorlist, len(colorlist), id(colorlist))
colortuple = tuple(colorlist)
print (colortuple, len(colortuple), id(colortuple))
```

```
('Red', 'Black', 'Brown', 'Yellow') 4 2658962566176
['Red', 'Black', 'Brown', 'Yellow'] 4 2658968293312
['Red', 'Black', 'Green', 'Yellow'] 4 2658968293312
('Red', 'Black', 'Green', 'Yellow') 4 2658962565776
```

In [37]:
```
colortuple = ("Red", "Black", "Brown", "Yellow")
print (colortuple, len(colortuple), id(colortuple))
del colortuple
print (colortuple, len(colortuple), id(colortuple))
```

```
('Red', 'Black', 'Brown', 'Yellow') 4 2658949625872
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-37-a10c05918267> in <module>
      2 print (colortuple, len(colortuple), id(colortuple))
      3 del colortuple
----> 4 print (colortuple, len(colortuple), id(colortuple))

NameError: name 'colortuple' is not defined
```

Dealing with Dictionary

> Dictionary is a collection of Key-Value pairs enclosed within {}. All keys must be unique and values may repeat. All keys must be of immutable datatype.

In [23]:
```
dict1 = {"mango":100, "banana":200, "pineapple":500, "apple":400}
print (dict1, len(dict1), type(dict1), id(dict1))
print (dict1.items())  # returns list of tuples
print (dict1.keys())    # returns a list
print (dict1.values())  # returns a list
```

```
{'mango': 100, 'banana': 200, 'pineapple': 500, 'apple': 400} 4 <class 'dict'> 2658970197440
dict_items([('mango', 100), ('banana', 200), ('pineapple', 500), ('apple', 400)])
dict_keys(['mango', 'banana', 'pineapple', 'apple'])
dict_values([100, 200, 500, 400])
```

In [28]:
```
dict1 = {"mango":100, "banana":200, "pineapple":500, "apple":400}
print (dict1, len(dict1), type(dict1), id(dict1))
print (dict1.get("banana"))
print (dict1.get("pineapple"))
print (dict1.get("goava"))
print (dict1.get("goava", "Not in the dictionary..."))
print (dict1.get("apple", "Not in the dictionary..."))
```

```
{'mango': 100, 'banana': 200, 'pineapple': 500, 'apple': 400} 4 <class 'dict'> 2658971810112
200
500
None
Not in the dictionary...
400
```

In [32]:
```
dict1 = {"mango":100, "banana":200, "pineapple":500, "apple":400}
print (dict1, len(dict1), type(dict1), id(dict1))
print (dict1["banana"])
print (dict1["pineapple"])
print (dict1["goava"])
```

```
{'mango': 100, 'banana': 200, 'pineapple': 500, 'apple': 400} 4 <class 'dict'> 2658971727744
200
500
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-32-aae1a4ba32ee> in <module>
      3 print (dict1["banana"])
      4 print (dict1["pineapple"])
----> 5 print (dict1["goava"])

KeyError: 'goava'
```

In [34]:

```
dict1 = {"mango":100, "banana":200, "pineapple":500, "apple":400}
print (dict1, len(dict1), type(dict1), id(dict1))
dict1["orange"] = 750   # insert operation
print (dict1, len(dict1), type(dict1), id(dict1))
dict1["banana"] = 1050   # update operation
print (dict1, len(dict1), type(dict1), id(dict1))
```

```
{'mango': 100, 'banana': 200, 'pineapple': 500, 'apple': 400} 4 <class 'dict'> 2658971590208
{'mango': 100, 'banana': 200, 'pineapple': 500, 'apple': 400, 'orange': 750} 5 <class 'dict'> 2658971590208
{'mango': 100, 'banana': 1050, 'pineapple': 500, 'apple': 400, 'orange': 750} 5 <class 'dict'> 2658971590208
```

In [35]:

```
dict1 = {'mango': 100, 'banana': 1050, 'pineapple': 500, 'apple': 400, 'orange': 750}
print (dict1, len(dict1), type(dict1), id(dict1))
dict1.clear()
print (dict1, len(dict1), type(dict1), id(dict1))
```

```
{'mango': 100, 'banana': 1050, 'pineapple': 500, 'apple': 400, 'orange': 750} 5 <class 'dict'> 2658970140416
{} 0 <class 'dict'> 2658970140416
```

In [36]:

```
dict1 = {'mango': 100, 'banana': 1050, 'pineapple': 500, 'apple': 400, 'orange': 750}
print (dict1, len(dict1), type(dict1), id(dict1))
del dict1
print (dict1, len(dict1), type(dict1), id(dict1))
```

```
{'mango': 100, 'banana': 1050, 'pineapple': 500, 'apple': 400, 'orange': 750} 5 <class 'dict'> 2658971444224
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-36-b739bbde912a> in <module>
      2 print (dict1, len(dict1), type(dict1), id(dict1))
      3 del dict1
----> 4 print (dict1, len(dict1), type(dict1), id(dict1))

NameError: name 'dict1' is not defined
```

In [38]:

```
dict1 = {'mango': 100, 'banana': 1050, 'pineapple': 500, 'apple': 400, 'orange': 750}
print (dict1, len(dict1), type(dict1), id(dict1))
del dict1["pineapple"]
print (dict1, len(dict1), type(dict1), id(dict1))
del dict1["apple"]
print (dict1, len(dict1), type(dict1), id(dict1))
```

```
{'mango': 100, 'banana': 1050, 'pineapple': 500, 'apple': 400, 'orange': 750} 5 <class 'dict'> 2658971005952
{'mango': 100, 'banana': 1050, 'apple': 400, 'orange': 750} 4 <class 'dict'> 2658971005952
{'mango': 100, 'banana': 1050, 'orange': 750} 3 <class 'dict'> 2658971005952
```

In [39]:

```
dict1 = {'mango': 100, 'banana': 1050, 'pineapple': 500, 'apple': 400, 'orange': 750}
print (dict1, len(dict1), type(dict1), id(dict1))
print (dict1.popitem())
print (dict1, len(dict1), type(dict1), id(dict1))
print (dict1.popitem())
print (dict1, len(dict1), type(dict1), id(dict1))
print (dict1.popitem())
print (dict1, len(dict1), type(dict1), id(dict1))
```

```
{'mango': 100, 'banana': 1050, 'pineapple': 500, 'apple': 400, 'orange': 750} 5 <class 'dict'> 2658970842240
('orange', 750)
{'mango': 100, 'banana': 1050, 'pineapple': 500, 'apple': 400} 4 <class 'dict'> 2658970842240
('apple', 400)
{'mango': 100, 'banana': 1050, 'pineapple': 500} 3 <class 'dict'> 2658970842240
('pineapple', 500)
{'mango': 100, 'banana': 1050} 2 <class 'dict'> 2658970842240
```

In [41]:

```
dict1 = {'mango': 100, 'banana': 1050, 'pineapple': 500, 'apple': 400, 'orange': 750}
print (dict1, len(dict1), type(dict1), id(dict1))
print (dict1.pop("banana"))
print (dict1, len(dict1), type(dict1), id(dict1))
print (dict1.pop("pineapple"))
print (dict1, len(dict1), type(dict1), id(dict1))
print (dict1.pop("apple"))
print (dict1, len(dict1), type(dict1), id(dict1))
```

```
{'mango': 100, 'banana': 1050, 'pineapple': 500, 'apple': 400, 'orange': 750} 5 <class 'dict'> 2658968926336
1050
{'mango': 100, 'pineapple': 500, 'apple': 400, 'orange': 750} 4 <class 'dict'> 2658968926336
500
{'mango': 100, 'apple': 400, 'orange': 750} 3 <class 'dict'> 2658968926336
400
{'mango': 100, 'orange': 750} 2 <class 'dict'> 2658968926336
```

In [43]:

```
MyKeyList = ["Red", "Green", "Blue", "Purple", "Magenta"]
MyValue = 100
dict1 = dict.fromkeys(MyKeyList, MyValue)
print (dict1, len(dict1), type(dict1), id(dict1))
```

```
{'Red': 100, 'Green': 100, 'Blue': 100, 'Purple': 100, 'Magenta': 100} 5 <class 'dict'> 2658970434176
```

In [45]:

```
dict1 = {'Red': 100, 'Green': 100, 'Blue': 100, 'Purple': 100}
print (dict1, len(dict1), type(dict1), id(dict1))
dict1.setdefault("Blue", 555)
print (dict1, len(dict1), type(dict1), id(dict1))
dict1.setdefault("Magenta", 999)
print (dict1, len(dict1), type(dict1), id(dict1))
```

```
{'Red': 100, 'Green': 100, 'Blue': 100, 'Purple': 100} 4 <class 'dict'> 2658972395648
{'Red': 100, 'Green': 100, 'Blue': 100, 'Purple': 100} 4 <class 'dict'> 2658972395648
{'Red': 100, 'Green': 100, 'Blue': 100, 'Purple': 100, 'Magenta': 999} 5 <class 'dict'> 2658972395648
```

In [48]:

```
dict1 = {'Red': 800, 'Green': 700, 'Blue': 100}
print (dict1, len(dict1), type(dict1), id(dict1))
dict2 = {'Purple': 200, 'Magenta':500}
print (dict2, len(dict2), type(dict2), id(dict2))
dict1.update(dict2)
print (dict1, len(dict1), type(dict1), id(dict1))
print (dict2, len(dict2), type(dict2), id(dict2))
```

```
{'Red': 800, 'Green': 700, 'Blue': 100} 3 <class 'dict'> 2658972081152
{'Purple': 200, 'Magenta': 500} 2 <class 'dict'> 2658972080256
{'Red': 800, 'Green': 700, 'Blue': 100, 'Purple': 200, 'Magenta': 500} 5 <class 'dict'> 2658972081152
{'Purple': 200, 'Magenta': 500} 2 <class 'dict'> 2658972080256
```

In [ ]:

```
help(dict)
```

In [84]:

```
var = []      # empty list
print (var, len(var), type(var), id(var))
var = ()      # empty tuple
print (var, len(var), type(var), id(var))
var = {}      # empty dictionary
print (var, len(var), type(var), id(var))
var = set()   # empty set
print (var, len(var), type(var), id(var))
var = (3,)    # singleton notation of a tuple
print (var, len(var), type(var), id(var))
```

```
[] 0 <class 'list'> 2658976906624
() 0 <class 'tuple'> 2658874490944
{} 0 <class 'dict'> 2658972754816
set() 0 <class 'set'> 2658969906304
(3,) 1 <class 'tuple'> 2658979220544
```

In [62]:

```
colors = ["Red", "Yellow", "Green", "Orange", "Purple", "Magenta"]
fruits = ["Apple", "Banana", "Goava", "Orange"]
print (colors, len(colors), type(colors), id(colors))
print (fruits, len(fruits), type(fruits), id(fruits))
var = zip(fruits, colors)
print (var, type(var))
var = list(zip(fruits, colors))
print (var, type(var))
var = dict(list(zip(fruits, colors)))
print (var, type(var))
var = tuple(zip(fruits, colors))
print (var, type(var))
var = dict(tuple(zip(fruits, colors)))
print (var, type(var))
```

```
['Red', 'Yellow', 'Green', 'Orange', 'Purple', 'Magenta'] 6 <class 'list'> 2658975430848
['Apple', 'Banana', 'Goava', 'Orange'] 4 <class 'list'> 2658975428928
<zip object at 0x0000026B174A48C0> <class 'zip'>
[('Apple', 'Red'), ('Banana', 'Yellow'), ('Goava', 'Green'), ('Orange', 'Orange')] <class 'list'>
{'Apple': 'Red', 'Banana': 'Yellow', 'Goava': 'Green', 'Orange': 'Orange'} <class 'dict'>
(('Apple', 'Red'), ('Banana', 'Yellow'), ('Goava', 'Green'), ('Orange', 'Orange')) <class 'tuple'>
{'Apple': 'Red', 'Banana': 'Yellow', 'Goava': 'Green', 'Orange': 'Orange'} <class 'dict'>
```

Class Assignment - 3

> Take a word from the user and find and print frequency of occurrences of each distinct character in the given word.
>
> - As example: Input is "mississippi" then output will be
> - m -> 1, s -> 4, i -> 4, p -> 2

In [68]:
```
myword = input("Please enter your word:").lower()
mychar = {}
for ch in myword:
    if ch in mychar:
        mychar[ch] += 1
    else:
        mychar[ch] = 1
print (mychar)
```

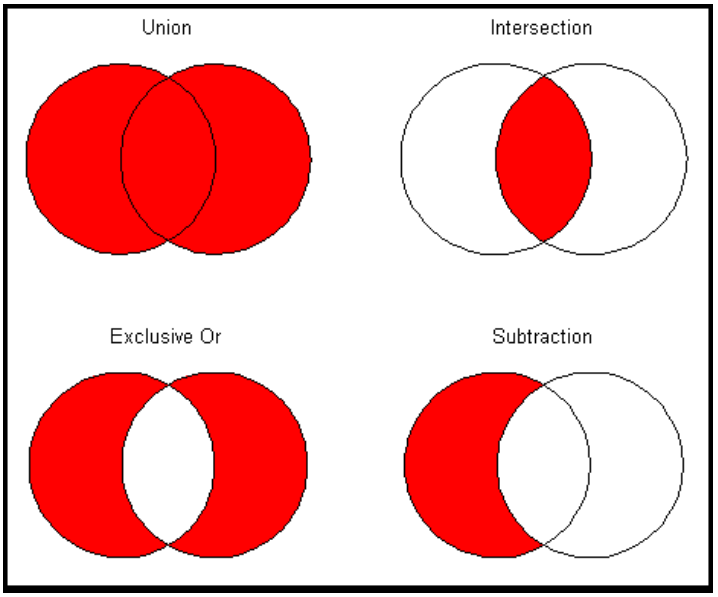
```
{'m': 1, 'i': 4, 's': 4, 'p': 2}
```

Dealing with Set

> Set is an unordered collection of unique values enclosed within {}. And set is mutable.

In [65]:
```
set1 = {"Python", "Java", "C", "Python", "C++", "C", "Java"}
print (set1, len(set1), type(set1), id(set1))
set1 = set(["Python", "Java", "C", "Python", "C++", "C", "Java"])
print (set1, len(set1), type(set1), id(set1))
set1 = set(("Python", "Java", "C", "Python", "C++", "C", "Java"))
print (set1, len(set1), type(set1), id(set1))
```

```
{'C', 'Java', 'C++', 'Python'} 4 <class 'set'> 2658967633472
{'C', 'Java', 'C++', 'Python'} 4 <class 'set'> 2658967633248
{'C', 'Java', 'C++', 'Python'} 4 <class 'set'> 2658967633472
```



In [71]:
```
# union operation
languages = {"python", "java", "c"}
print (languages, len(languages), type(languages), id(languages))
snakes = {"cobra", "viper", "python"}
print (snakes, len(snakes), type(snakes), id(snakes))
result = languages.union(snakes)
print (result, len(result), type(result), id(result))
result = languages | snakes
print (result, len(result), type(result), id(result))
```

```
{'java', 'python', 'c'} 3 <class 'set'> 2658971789792
{'cobra', 'python', 'viper'} 3 <class 'set'> 2658971789568
{'java', 'c', 'cobra', 'python', 'viper'} 5 <class 'set'> 2658971790240
{'java', 'c', 'cobra', 'python', 'viper'} 5 <class 'set'> 2658975571520
```

In [74]:
```
# intersection operation
languages = {"python", "java", "c"}
print (languages, len(languages), type(languages), id(languages))
snakes = {"cobra", "viper", "python"}
print (snakes, len(snakes), type(snakes), id(snakes))
result = languages.intersection(snakes)
print (result, len(result), type(result), id(result))
result = languages & snakes
```

```
print (result, len(result), type(result), id(result))
languages.intersection_update(snakes)
print (languages, len(languages), type(languages), id(languages))
print (snakes, len(snakes), type(snakes), id(snakes))
```

```
{'java', 'python', 'c'} 3 <class 'set'> 2658980694944
{'cobra', 'python', 'viper'} 3 <class 'set'> 2658980695168
{'python'} 1 <class 'set'> 2658980693152
{'python'} 1 <class 'set'> 2658980695392
{'python'} 1 <class 'set'> 2658980694944
{'cobra', 'python', 'viper'} 3 <class 'set'> 2658980695168
```

In [76]:

```
# excusive-or / xor / symmetric difference operation
languages = {"python", "java", "c"}
print (languages, len(languages), type(languages), id(languages))
snakes = {"cobra", "viper", "python"}
print (snakes, len(snakes), type(snakes), id(snakes))
result = languages.symmetric_difference(snakes)
print (result, len(result), type(result), id(result))
result = languages ^ snakes
print (result, len(result), type(result), id(result))
languages.symmetric_difference_update(snakes)
print (languages, len(languages), type(languages), id(languages))
print (snakes, len(snakes), type(snakes), id(snakes))
```

```
{'java', 'python', 'c'} 3 <class 'set'> 2658980817376
{'cobra', 'python', 'viper'} 3 <class 'set'> 2658980829440
{'java', 'c', 'cobra', 'viper'} 4 <class 'set'> 2658980828320
{'java', 'c', 'cobra', 'viper'} 4 <class 'set'> 2658980828992
{'java', 'c', 'cobra', 'viper'} 4 <class 'set'> 2658980817376
{'cobra', 'python', 'viper'} 3 <class 'set'> 2658980829440
```

In [78]:

```
# subtraction / set difference operation
languages = {"python", "java", "c"}
print (languages, len(languages), type(languages), id(languages))
snakes = {"cobra", "viper", "python"}
print (snakes, len(snakes), type(snakes), id(snakes))
result = languages.difference(snakes)
print (result, len(result), type(result), id(result))
result = languages - snakes
print (result, len(result), type(result), id(result))
languages.difference_update(snakes)
print (languages, len(languages), type(languages), id(languages))
print (snakes, len(snakes), type(snakes), id(snakes))
```

```
{'java', 'python', 'c'} 3 <class 'set'> 2658977622496
{'cobra', 'python', 'viper'} 3 <class 'set'> 2658976843584
{'java', 'c'} 2 <class 'set'> 2658976842240
{'java', 'c'} 2 <class 'set'> 2658976844928
{'java', 'c'} 2 <class 'set'> 2658977622496
{'cobra', 'python', 'viper'} 3 <class 'set'> 2658976843584
```

In [88]:

```
set1 = {11, 22, 33}
set2 = {11, 22, 33, 44, 55, 66}
set3 = {77, 88, 99}
print (set1, set2, set3)
print (set1.issubset(set2), set1.issubset(set3), set2.issubset(set1))
print (set1.issuperset(set2), set1.issuperset(set3), set2.issuperset(set1))
print (set1.isdisjoint(set2), set1.isdisjoint(set3), set2.isdisjoint(set1))
```

```
{33, 11, 22} {33, 66, 11, 44, 22, 55} {88, 99, 77}
True False False
False False True
False True False
```

In [90]:

```
languages = {"python", "java", "c"}
print (languages, len(languages), type(languages), id(languages))
languages.add("cobol")
print (languages, len(languages), type(languages), id(languages))
languages.add("perl")
print (languages, len(languages), type(languages), id(languages))
languages.add("r")
print (languages, len(languages), type(languages), id(languages))
```

```
{'java', 'python', 'c'} 3 <class 'set'> 2658979007840
{'java', 'python', 'cobol', 'c'} 4 <class 'set'> 2658979007840
{'perl', 'python', 'cobol', 'java', 'c'} 5 <class 'set'> 2658979007840
{'perl', 'python', 'cobol', 'java', 'r', 'c'} 6 <class 'set'> 2658979007840
```

In [91]:

```
languages = {"python", "java", "c"}
print (languages, len(languages), type(languages), id(languages))
languages.clear()
print (languages, len(languages), type(languages), id(languages))
```

```
{'java', 'python', 'c'} 3 <class 'set'> 2658967292832
set() 0 <class 'set'> 2658967292832
```

In [92]:

```
languages = {"python", "java", "c"}
print (languages, len(languages), type(languages), id(languages))
del languages
print (languages, len(languages), type(languages), id(languages))
```

```
{'java', 'python', 'c'} 3 <class 'set'> 2658967291040
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-92-bc850a3b30be> in <module>
      2 print (languages, len(languages), type(languages), id(languages))
      3 del languages
----> 4 print (languages, len(languages), type(languages), id(languages))

NameError: name 'languages' is not defined
```

In [93]:

```
languages = {'perl', 'python', 'cobol', 'java', 'r', 'c'}
print (languages, len(languages), type(languages), id(languages))
print (languages.pop())
print (languages, len(languages), type(languages), id(languages))
print (languages.pop())
print (languages, len(languages), type(languages), id(languages))
print (languages.pop())
print (languages, len(languages), type(languages), id(languages))
print (languages.pop())
print (languages, len(languages), type(languages), id(languages))
```

```
{'perl', 'python', 'cobol', 'r', 'java', 'c'} 6 <class 'set'> 2658967292832
perl
{'python', 'cobol', 'r', 'java', 'c'} 5 <class 'set'> 2658967292832
python
{'cobol', 'r', 'java', 'c'} 4 <class 'set'> 2658967292832
cobol
{'r', 'java', 'c'} 3 <class 'set'> 2658967292832
r
{'java', 'c'} 2 <class 'set'> 2658967292832
```

In [94]:

```
languages = {'perl', 'python', 'cobol', 'java', 'r', 'c'}
print (languages, len(languages), type(languages), id(languages))
languages.discard("python")
print (languages, len(languages), type(languages), id(languages))
languages.discard("cobol")
print (languages, len(languages), type(languages), id(languages))
languages.discard("r")
print (languages, len(languages), type(languages), id(languages))
languages.discard("kotlin")
print (languages, len(languages), type(languages), id(languages))
```

```
{'perl', 'python', 'cobol', 'r', 'java', 'c'} 6 <class 'set'> 2658972759200
{'perl', 'cobol', 'r', 'java', 'c'} 5 <class 'set'> 2658972759200
{'perl', 'r', 'java', 'c'} 4 <class 'set'> 2658972759200
{'perl', 'java', 'c'} 3 <class 'set'> 2658972759200
{'perl', 'java', 'c'} 3 <class 'set'> 2658972759200
```

In [95]:

```
languages = {'perl', 'python', 'cobol', 'java', 'r', 'c'}
print (languages, len(languages), type(languages), id(languages))
languages.remove("python")
print (languages, len(languages), type(languages), id(languages))
languages.remove("cobol")
print (languages, len(languages), type(languages), id(languages))
languages.remove("r")
print (languages, len(languages), type(languages), id(languages))
languages.remove("kotlin")
print (languages, len(languages), type(languages), id(languages))
```

```
{'perl', 'python', 'cobol', 'r', 'java', 'c'} 6 <class 'set'> 2658971846912
{'perl', 'cobol', 'r', 'java', 'c'} 5 <class 'set'> 2658971846912
{'perl', 'r', 'java', 'c'} 4 <class 'set'> 2658971846912
{'perl', 'java', 'c'} 3 <class 'set'> 2658971846912
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-95-377f0cc8e8d2> in <module>
      7 languages.remove("r")
      8 print (languages, len(languages), type(languages), id(languages))
----> 9 languages.remove("kotlin")
     10 print (languages, len(languages), type(languages), id(languages))

KeyError: 'kotlin'
```

In [99]:

```
languages1 = {'perl', 'python', 'cobol'}
languages2 = {'r', 'java', 'c'}
```

```
print (languages1, len(languages1), type(languages1), id(languages1))
print (languages2, len(languages2), type(languages2), id(languages2))
languages1.update(languages2)
print (languages1, len(languages1), type(languages1), id(languages1))
print (languages2, len(languages2), type(languages2), id(languages2))
```

```
{'perl', 'python', 'cobol'} 3 <class 'set'> 2658968396288
{'r', 'java', 'c'} 3 <class 'set'> 2658968397184
{'r', 'java', 'python', 'c', 'perl', 'cobol'} 6 <class 'set'> 2658968396288
{'r', 'java', 'c'} 3 <class 'set'> 2658968397184
```

In [104…

```
mystr = "mississippi"
print (mystr, len(mystr), type(mystr))
print (list(mystr), len(list(mystr)), type(list(mystr)))
print (tuple(mystr), len(tuple(mystr)), type(tuple(mystr)))
print (set(mystr), len(set(mystr)), type(set(mystr)))
```

```
mississippi 11 <class 'str'>
['m', 'i', 's', 's', 'i', 's', 's', 'i', 'p', 'p', 'i'] 11 <class 'list'>
('m', 'i', 's', 's', 'i', 's', 's', 'i', 'p', 'p', 'i') 11 <class 'tuple'>
{'i', 'm', 'p', 's'} 4 <class 'set'>
```

Dealing with Frozen-Set

> Frozen-Set is an unordered collection of unique values enclosed within {}. And Frozen-Set is immutable.

In [66]:

```
frozen_set1 = frozenset(["Python", "Java", "C", "Python", "C++", "C", "Java"])
print (frozen_set1, len(frozen_set1), type(frozen_set1), id(frozen_set1))
frozen_set1 = frozenset(("Python", "Java", "C", "Python", "C++", "C", "Java"))
print (frozen_set1, len(frozen_set1), type(frozen_set1), id(frozen_set1))
```

```
frozenset({'C', 'Java', 'C++', 'Python'}) 4 <class 'frozenset'> 2658967633248
frozenset({'C', 'Java', 'C++', 'Python'}) 4 <class 'frozenset'> 2658966793568
```

In [72]:

```
# union operation
languages = frozenset(["python", "java", "c"])
print (languages, len(languages), type(languages), id(languages))
snakes = frozenset(["cobra", "viper", "python"])
print (snakes, len(snakes), type(snakes), id(snakes))
result = languages.union(snakes)
print (result, len(result), type(result), id(result))
result = languages | snakes
print (result, len(result), type(result), id(result))
```

```
frozenset({'java', 'python', 'c'}) 3 <class 'frozenset'> 2658967030464
frozenset({'cobra', 'python', 'viper'}) 3 <class 'frozenset'> 2658967031584
frozenset({'java', 'c', 'cobra', 'python', 'viper'}) 5 <class 'frozenset'> 2658967028224
frozenset({'java', 'c', 'cobra', 'python', 'viper'}) 5 <class 'frozenset'> 2658967030688
```

In [75]:

```
# intersection operation
languages = frozenset(["python", "java", "c"])
print (languages, len(languages), type(languages), id(languages))
snakes = frozenset(["cobra", "viper", "python"])
print (snakes, len(snakes), type(snakes), id(snakes))
result = languages.intersection(snakes)
print (result, len(result), type(result), id(result))
result = languages & snakes
print (result, len(result), type(result), id(result))
```

```
frozenset({'java', 'python', 'c'}) 3 <class 'frozenset'> 2658975432992
frozenset({'cobra', 'python', 'viper'}) 3 <class 'frozenset'> 2658975434784
frozenset({'python'}) 1 <class 'frozenset'> 2658975434112
frozenset({'python'}) 1 <class 'frozenset'> 2658975436128
```

In [77]:

```
# excusive-or / xor / symmetric difference operation
languages = frozenset(["python", "java", "c"])
print (languages, len(languages), type(languages), id(languages))
snakes = frozenset(["cobra", "viper", "python"])
print (snakes, len(snakes), type(snakes), id(snakes))
result = languages.symmetric_difference(snakes)
print (result, len(result), type(result), id(result))
result = languages ^ snakes
print (result, len(result), type(result), id(result))
```

```
frozenset({'java', 'python', 'c'}) 3 <class 'frozenset'> 2658980818272
frozenset({'cobra', 'python', 'viper'}) 3 <class 'frozenset'> 2658967413248
frozenset({'java', 'c', 'cobra', 'viper'}) 4 <class 'frozenset'> 2658967414592
frozenset({'java', 'c', 'cobra', 'viper'}) 4 <class 'frozenset'> 2658967412800
```

In [82]:

```
# subtraction / set difference operation
languages = frozenset(["python", "java", "c"])
```

```
print (languages, len(languages), type(languages), id(languages))
snakes = frozenset(["cobra", "viper", "python"])
print (snakes, len(snakes), type(snakes), id(snakes))
result = languages.difference(snakes)
print (result, len(result), type(result), id(result))
result = languages - snakes
print (result, len(result), type(result), id(result))
```

```
frozenset({'java', 'python', 'c'}) 3 <class 'frozenset'> 2658969904512
frozenset({'cobra', 'python', 'viper'}) 3 <class 'frozenset'> 2658969906976
frozenset({'java', 'c'}) 2 <class 'frozenset'> 2658969906304
frozenset({'java', 'c'}) 2 <class 'frozenset'> 2658969903616
```

In [89]:

```
set1 = frozenset([11, 22, 33])
set2 = frozenset([11, 22, 33, 44, 55, 66])
set3 = frozenset([77, 88, 99])
print (set1, set2, set3)
print (set1.issubset(set2), set1.issubset(set3), set2.issubset(set1))
print (set1.issuperset(set2), set1.issuperset(set3), set2.issuperset(set1))
print (set1.isdisjoint(set2), set1.isdisjoint(set3), set2.isdisjoint(set1))
```

```
frozenset({33, 11, 22}) frozenset({33, 66, 11, 44, 22, 55}) frozenset({88, 99, 77})
True False False
False False True
False True False
```

Dealing with Data File

Data file path is "C:\Users\Arnab\USA Batch\Vodafone\emp_data.csv"

In [113…

```
# importing the required module
import csv
with open('emp_data.csv') as data_file:   # relative path
    csv_reader = csv.reader(data_file)
    print (csv_reader)
    # print (len(list(csv_reader)))
    # print (list(csv_reader))
    print (tuple(csv_reader))
```

```
<_csv.reader object at 0x0000026B1710C940>
(['1001', 'Dhiman', 'Kolkata', '39000'], ['1002', 'Anupam', 'Kolkata', '25000'], ['1003', 'Subham', 'Mumbai', '3600
0'], ['1004', 'Dinesh', 'Chennai', '28000'], ['1005', 'Kakali', 'Mumbai', '25000'], ['1006', 'Bimal', 'Hyderabad',
'30000'], ['1007', 'Tarun', 'Chennai', '17000'], ['1008', 'Rittik', 'Durgapur', '45000'], ['1009', 'Barun', 'Hyderab
ad', '39000'], ['1010', 'Utpal', 'Lucknow', '20000'])
```

In [126…

```
# importing the required module
import csv
# with open('C:\\Users\\Arnab\\USA Batch\\Vodafone\\emp_data.csv') as data_file:
# with open('C://Users//Arnab//USA Batch//Vodafone//emp_data.csv') as data_file:
with open('C:/Users/Arnab/USA Batch/Vodafone/emp_data.csv') as data_file:  # fully qualified / absolute path
    csv_reader = csv.reader(data_file)
    print (csv_reader)
    # print (len(list(csv_reader)))
    # print (list(csv_reader))
    print (tuple(csv_reader))
print (type(csv_reader), type(data_file))
csv_reader = csv.reader(data_file)
```

```
<_csv.reader object at 0x0000026B16B4F160>
(['1001', 'Dhiman', 'Kolkata', '39000'], ['1002', 'Anupam', 'Kolkata', '25000'], ['1003', 'Subham', 'Mumbai', '3600
0'], ['1004', 'Dinesh', 'Chennai', '28000'], ['1005', 'Kakali', 'Mumbai', '25000'], ['1006', 'Bimal', 'Hyderabad',
'30000'], ['1007', 'Tarun', 'Chennai', '17000'], ['1008', 'Rittik', 'Durgapur', '45000'], ['1009', 'Barun', 'Hyderab
ad', '39000'], ['1010', 'Utpal', 'Lucknow', '20000'])
<class '_csv.reader'> <class '_io.TextIOWrapper'>
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-126-024e54a52ccb> in <module>
     10     print (tuple(csv_reader))
     11 print (type(csv_reader), type(data_file))
---> 12 csv_reader = csv.reader(data_file)

ValueError: I/O operation on closed file.
```

In [122…

```
with open('emp_data.csv') as data_file:   # relative path
    csv_reader = csv.reader(data_file)
    for row in csv_reader:
        print (row)
```

```
['1001', 'Dhiman', 'Kolkata', '39000']
['1002', 'Anupam', 'Kolkata', '25000']
['1003', 'Subham', 'Mumbai', '36000']
['1004', 'Dinesh', 'Chennai', '28000']
['1005', 'Kakali', 'Mumbai', '25000']
```

```
['1006', 'Bimal', 'Hyderabad', '30000']
['1007', 'Tarun', 'Chennai', '17000']
['1008', 'Rittik', 'Durgapur', '45000']
['1009', 'Barun', 'Hyderabad', '39000']
['1010', 'Utpal', 'Lucknow', '20000']
```

In [123…

```python
with open('emp_data.csv') as data_file:   # relative path
    csv_reader = csv.reader(data_file)
    for row in csv_reader:
        print (f"Emp-ID: {row[0]}, Emp-Name: {row[1]}, Emp-Location: {row[2]}, Emp-Salary: {row[3]}...")
```

```
Emp-ID: 1001, Emp-Name: Dhiman, Emp-Location: Kolkata, Emp-Salary: 39000...
Emp-ID: 1002, Emp-Name: Anupam, Emp-Location: Kolkata, Emp-Salary: 25000...
Emp-ID: 1003, Emp-Name: Subham, Emp-Location: Mumbai, Emp-Salary: 36000...
Emp-ID: 1004, Emp-Name: Dinesh, Emp-Location: Chennai, Emp-Salary: 28000...
Emp-ID: 1005, Emp-Name: Kakali, Emp-Location: Mumbai, Emp-Salary: 25000...
Emp-ID: 1006, Emp-Name: Bimal, Emp-Location: Hyderabad, Emp-Salary: 30000...
Emp-ID: 1007, Emp-Name: Tarun, Emp-Location: Chennai, Emp-Salary: 17000...
Emp-ID: 1008, Emp-Name: Rittik, Emp-Location: Durgapur, Emp-Salary: 45000...
Emp-ID: 1009, Emp-Name: Barun, Emp-Location: Hyderabad, Emp-Salary: 39000...
Emp-ID: 1010, Emp-Name: Utpal, Emp-Location: Lucknow, Emp-Salary: 20000...
```