Good Morning To All

Sets

Functions

In [1]:
```
# Set is used o store multiple datatypes in given  set
# Set is declares the {}
# Set is won't allow the duplicates and Unordered list
set1={1,1,2,3,4,5,5,6}
set1
```

Out[1]: {1, 2, 3, 4, 5, 6}

In [2]:
```
set1={1,2,1,3,4,7,6,8,5}
```

In [3]:
```
set1
```

Out[3]: {1, 2, 3, 4, 5, 6, 7, 8}

In [4]:
```
1 dir(set)
```

Out[4]:
```
['__and__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__iand__',
 '__init__',
 '__init_subclass__',
 '__ior__',
 '__isub__',
 '__iter__',
 '__ixor__',
 '__le__',
 '__len__',
 '__lt__',
 '__ne__',
 '__new__',
 '__or__',
 '__rand__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__ror__',
 '__rsub__',
 '__rxor__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__sub__',
 '__subclasshook__',
 '__xor__',
 'add',
 'clear',
```

```
 'copy',
 'difference',
 'difference_update',
 'discard',
 'intersection',
 'intersection_update',
 'isdisjoint',
 'issubset',
 'issuperset',
 'pop',
 'remove',
 'symmetric_difference',
 'symmetric_difference_update',
 'union',
 'update']
```

In [5]:
```
1 set1
```

Out[5]: {1, 2, 3, 4, 5, 6, 7, 8}

In [6]:
```
1 set1.add(9.8)
```

In [7]:
```
1 set1
```

Out[7]: {1, 2, 3, 4, 5, 6, 7, 8, 9.8}

In [9]:
```
1 set1.add('[1,2,3]')
```

In [10]:
```
1 set1
```

Out[10]: {1, 2, 3, 4, 5, 6, 7, 8, 9.8, '[1,2,3]'}

In [11]:
```
1 set1.clear()
```

In [12]:
```
1 set1
```

Out[12]: set()

In [13]:
```
set2={1,1,2,3,4,5}
set3={3,4,5,6,7,8}
set2.difference(set3)
```

Out[13]:
```
{1, 2}
```

In [14]:
```
print(set2,"before update")
set2.difference_update(set3)
print(set2,"after update")
```

```
{1, 2, 3, 4, 5} before update
{1, 2} after update
```

In [16]:
```
set2={1,1,2,3,4,5}
print(set3,"before update")
set3.difference_update(set2)
print(set3,"after update")
```

```
{3, 4, 5, 6, 7, 8} before update
{6, 7, 8} after update
```

In [17]:
```
set2
```

Out[17]:
```
{1, 2, 3, 4, 5}
```

In [18]:
```
set2.discard(4)
```

In [19]:
```
set2
```

Out[19]:
```
{1, 2, 3, 5}
```

In [20]:
```
#Intersection
t={"ravi","anil","satyanarayana"}
t2={"ravi","ayyappa","anil"}
t.intersection(t2)
```

Out[20]:
```
{'anil', 'ravi'}
```

In [21]:
```
# Intersection_Update
t={"ravi","anil","satyanarayana"}
t2={"ravi","ayyappa","anil"}
t.intersection_update(t2)
```

In [22]:
```
t
```

Out[22]: {'anil', 'ravi'}

In [24]:
```
#isdisjoint
t={1,5,10,20}
t2={5,3,4,6}
t.isdisjoint(t2)
```

Out[24]: False

In [25]:
```
#isdisjoint
t={1,5,10,20}
t2={4,3,4,6}
t.isdisjoint(t2)
```

Out[25]: True

In [28]:
```
#issubset
t={1,2,4}
t2={1,2,4}
t.issubset(t2)
```

Out[28]: True

In [34]:
```
#issuperset
t2={1,2,3,4,5,6,7,8}
t={1,2,3,4,5}
t2.issuperset(t)
```

Out[34]: True

In [35]:
```
# Union
t={1,2,3,7}
t2={1,3,6,7}
t.union(t2)
```

Out[35]: {1, 2, 3, 6, 7}

Functions

Function is a group of statements to perform specific and required task

1.Builtin Functions

it's uses the already developed & availablefuntions

2.Userdefined Funtions

it's uses the and creates the own functions

In [37]:
```
for i in range(10):
    print(i,end=" ")
```

0 1 2 3 4 5 6 7 8 9

In [38]:
```
#range(starting,ending,stepvalue)
range(10)

```

Out[38]: range(0, 10)

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

```
|      Return a version of the string where each word is titlecased.
|
|      More specifically, words start with uppercased characters and all remaining
|      cased characters have lower case.
|
|  translate(self, table, /)
|      Replace each character in the string using the given translation table.
|
|        table
|          Translation table, which must be a mapping of Unicode ordinals to
|          Unicode ordinals, strings, or None.
|
|      The table must implement lookup/indexing via __getitem__, for instance a
|      dictionary or list.  If this operation raises LookupError, the character is
|      left untouched.  Characters mapped to None are deleted.
|
|  upper(self, /)
|      Return a copy of the string converted to uppercase.
|
```

In [ ]:
```
1 dir(list)
```

In [1]:
```
1 #abs()
2 abs(-10)
```

Out[1]: 10

In [2]:
```
1 abs(10)
```

Out[2]: 10

In [7]:
```
1 sorted("raviC",reverse=True)
```

Out[7]: ['v', 'r', 'i', 'a', 'C']

In [8]:
```
1 min("Ravi")
```

Out[8]: 'R'

In [9]:
```
1 min('ravi')
```

Out[9]: 'a'

In [10]:
```
1 max('RaVi')
```

Out[10]: 'i'

In [11]:
```
1 max('RaVI')
```

Out[11]: 'a'

In [12]:
```
1 len("ravi sastry")
```

Out[12]: 11

In [13]:
```
1 bin(4)
```

Out[13]: '0b100'

In [14]:
```
1 bin(16)
```

Out[14]: '0b10000'

In [15]:
```
1 bin(65)
```

Out[15]: '0b1000001'

In [16]:
```
1 sum([2,3,4,5])
```

Out[16]: 14

In [17]:
```
round(3.56)
```

Out[17]: 4

In [18]:
```
round(3.5)
```

Out[18]: 4

In [19]:
```
round(3.49)
```

Out[19]: 3

In [20]:
```
ord('A')
```

Out[20]: 65

In [21]:
```
ord('Z')
```

Out[21]: 90

In [22]:
```
ord('a')
```

Out[22]: 97

In [23]:
```
ord('z')
```

Out[23]: 122

In [24]:
```
for i in range(ord('A'),ord('Z')+1):
    print(i,end="  ")
```

```
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
```

In [25]:
```
for i in range(ord('A'),ord('Z')+1):
    if i%2==0:
        print(i,end=' ')
```

```
66 68 70 72 74 76 78 80 82 84 86 88 90
```

In [26]:
```
chr(67)
```

Out[26]: 'C'

In [27]:
```
chr(70)
```

Out[27]: 'F'

In [28]:
```
for i in range(ord('A'),ord('Z')+1):
    if i%2==0:
        print(i,"--->",chr(i))
```

```
66 ---> B
68 ---> D
70 ---> F
72 ---> H
74 ---> J
76 ---> L
78 ---> N
80 ---> P
82 ---> R
84 ---> T
86 ---> V
88 ---> X
90 ---> Z
```

In [29]:
```
# User Defined Functions
# syntax
#def function_name(arg1,...):
##### statements
##### statements
#function_name(arg1,...)
```

In [30]:
```
def add(a,b):
    print(a+b)
add(10,20)
```

```
30
```

In [31]:
```
def add(a,b):
    print(a+b)
a=10
b=40
add(a,b)
```

```
50
```

In [32]:
```
def add(c,b):
    print(c+b)
a=10
b=40
add(a,b)
```

```
50
```

In [35]:
```
def add(a,b):
    c=10
    b=30
    print(c+b)
a=10
b=40
add(a,b)
print(a+b)
```

```
40
50
```

In [36]:
```
def add(a,b):
    return a+b

```

In [37]:
```
1 add(20,30)
```

Out[37]: 50

In [38]:
```
1 add(10,30)
```

Out[38]: 40

In [39]:
```
1 def add(a,b):
2     return a+b
3 def sub(a,b):
4     return a-b
5 def mul(a,b):
6     return a*b
```

In [40]:
```
1 mul(2,4)
```

Out[40]: 8

In [41]:
```
1 mul(2,3)
```

Out[41]: 6

In [42]:
```
1 sub(20,10)
```

Out[42]: 10

In [43]:
```
1 def iseven(num):
2     if num%2==0:
3         return True
4     else:
5         return False
6 iseven(13)
```

Out[43]: False

In [45]:
```
def iseven(num1,num2):
    for num in range(num1,num2+1):
        if num%2==0:
            print(num)
iseven(100,150)
```

```
100
102
104
106
108
110
112
114
116
118
120
122
124
126
128
130
132
134
136
138
140
142
144
146
148
150
```

In [51]:
```
def leapyear(num1,num2):
    for year in range(num1,num2+1):
        if year%400==0 or year%4==0 and year%100!=0:
            print(year)
leapyear(2000,2020)
```

```
2000
2004
2008
2012
2016
2020
```

In [52]:
```
# Four types of function Arguments
#1.Required Arguments
#2.Default Arguments
#3.Keyword Arguments
#4.Variable Arguments
```

In [53]:
```
#1.Required Arguments
def add(a,b):
    print(a+b)
add(10,20)
```

```
30
```

In [54]:
```
# 2.Default Arguments
def add(a,b):
    print(a+b)
a=10
b=20
add(a,b)
```

```
30
```

In [55]:
```
# 3.Keyword Arguments
def add(a,b):
    print(a+b)
add(a=10,b=30)
```

```
40
```

In [61]:
```
# 4.Variable length Arguments
def add(a,b,*var):
    print(a)
    print(b)
    print(var,type(var))
add(10,20,30,40,50)
```

```
10
20
(30, 40, 50) <class 'tuple'>
```

In [64]:
```
set1={1,2,3,(2,3,4)}
set1
```

Out[64]:
```
{(2, 3, 4), 1, 2, 3}
```

In [66]:
```
set1.add('[2,3,4]')
```

In [67]:
```
set1
```

Out[67]:
```
{(2, 3, 4), 1, 2, 3, '[2,3,4]'}
```

In [ ]:
```

```