Today's Objective

- List
- tuple
- sets
- Dictionaries
- Reglar Expressions

List

- List contains collection of hetrogenoues data set
  - Ex:int,flot,space,special char,..etc
- List Reprasent by []
- LIST
- list is a mutable
- list contains Duplicate elements

In [1]:
```
li=[1,2,3,4]
print(li)
```

```
[1, 2, 3, 4]
```

In [2]:
```
print(dir(list))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir_
_', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem
__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass
__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setat
tr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'c
lear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'revers
e', 'sort']
```

In [3]:
```
len(li)#getting length of the list
```

Out[3]: 4

In [4]:
```
li.index(3)
```

Out[4]: 2

In [5]:
```
nl=['ap','ts','tn','up','ap']
nl[2]
```

Out[5]: 'tn'

In [6]:
```
1 nl[1:]#List slicing
```
Out[6]: ['ts', 'tn', 'up', 'ap']

In [7]:
```
1 nl.insert(2,'apssdc')#for insert the new ele in the list
```

In [8]:
```
1 nl
```
Out[8]: ['ap', 'ts', 'apssdc', 'tn', 'up', 'ap']

In [9]:
```
1 nl.pop()#remove the last ele from list
```
Out[9]: 'ap'

In [10]:
```
1 nl
```
Out[10]: ['ap', 'ts', 'apssdc', 'tn', 'up']

In [11]:
```
1 nl.remove('tn')
```

In [12]:
```
1 nl
```
Out[12]: ['ap', 'ts', 'apssdc', 'up']

In [13]:
```
1 li
```
Out[13]: [1, 2, 3, 4]

In [14]:
```
1 n=[4,7,2,8,120,45]
```

In [15]:
```
1 n.sort()
```

In [16]:
```
1 n.pop(2)
```
Out[16]: 7

In [17]:
```
1 n
```
Out[17]: [2, 4, 8, 45, 120]

In [18]:
```
1 nl
```
Out[18]: ['ap', 'ts', 'apssdc', 'up']

In [19]:
```
1 nl.sort()
```

In [20]:
```
1 nl
```
Out[20]: ['ap', 'apssdc', 'ts', 'up']

In [21]:
```
sorted(nl)
```

Out[21]: ['ap', 'apssdc', 'ts', 'up']

In [22]:
```
names=['sai','venky','muni','rama','Hanuman','veera venkata sai ','ap']
```

In [23]:
```
sorted(names)
```

Out[23]: ['Hanuman', 'ap', 'muni', 'rama', 'sai', 'veera venkata sai ', 'venky']

In [24]:
```
print(sorted(names,key=len))
```

```
['ap', 'sai', 'muni', 'rama', 'venky', 'Hanuman', 'veera venkata sai ']
```

In [25]:
```
ord('H')
```

Out[25]: 72

In [26]:
```
ord('a')
```

Out[26]: 97

In [27]:
```
n
```

Out[27]: [2, 4, 8, 45, 120]

In [28]:
```
n.append(100)#It is adding the ele in list at last position
```

In [29]:
```
n
```

Out[29]: [2, 4, 8, 45, 120, 100]

In [30]:
```
n.clear()#we can clear the total list
```

In [31]:
```
n
```

Out[31]: []

In [32]:
```
d=[1,2,3,1,2,3,4,5,12,37,7,7]
```

In [33]:
```
d.count(7)
```

Out[33]: 2

In [34]:
```
un=[]
for element in d:
    if element not in un:
        un.append(element)
sorted(un)

```

Out[34]: [1, 2, 3, 4, 5, 7, 12, 37]

In [35]:
```
un
```

Out[35]: [1, 2, 3, 4, 5, 12, 37, 7]

In [36]:
```
print("Even Numbers ")
c=0
for i in un:
    if i%2==0:
        print(i,end=" ")
       # c=c+1
        c+=1
print(" \neven no. Count is ",c)
```

```
Even Numbers
2 4 12
even no. Count is  3
```

In [37]:
```
# String
    #Split
    #Join
```

In [38]:
```
s="hello-world-kits"
```

In [39]:
```
s.split('-')
```

Out[39]: ['hello', 'world', 'kits']

In [40]:
```
s1="welcome to python programming"
```

In [41]:
```
s1.split()
```

Out[41]: ['welcome', 'to', 'python', 'programming']

In [42]:
```
s2="hellokits"
```

In [43]:
```
s2.split()
```

Out[43]: ['hellokits']

In [44]:
```
s2
```

Out[44]: 'hellokits'

In [45]:
```
s3='-'.join(s2)
s3
```

Out[45]: 'h-e-l-l-o-k-i-t-s'

In [46]:
```
s3.split('-')
```

Out[46]: ['h', 'e', 'l', 'l', 'o', 'k', 'i', 't', 's']

In [47]:
```
n
```

Out[47]: []

In [48]:
```
li
```

Out[48]: [1, 2, 3, 4]

In [49]:
```
min(li)#min
```

Out[49]: 1

In [50]:
```
max(li)#max value
```

Out[50]: 4

In [52]:
```
s=sum(li)#sum
```

In [53]:
```
s//len(li)#avg
```

Out[53]: 2

In [56]:
```
#generate multplication table for given number
    #4x1=4
    #4x2=8
    #..
    #..
    #4x10=40
n=int(input())
for i in range(1,11):
    print(n,"x",i,"=",n*i)
```

```
498
498 x 1 = 498
498 x 2 = 996
498 x 3 = 1494
498 x 4 = 1992
498 x 5 = 2490
498 x 6 = 2988
498 x 7 = 3486
498 x 8 = 3984
498 x 9 = 4482
498 x 10 = 4980
```

In [67]:
```
=#Check the given number is prime or not
#3,5,7,11,13..
def isprime(n):
    if n<2:
        return False
    for i in range(2,(n//2)+1):
        if n%i==0:
            return False
    return True
isprime(8)
```

Out[67]:
```
False
```

In [68]:
```
# generate the all prime numbers from 1 to 100
lb,ub=1,101
for i in range(lb,ub):
    if isprime(i):
        print(i,end=" ")
```

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

In [69]:
```
#generate the all yeap years from 2000 to 2020
```

In [72]:
```
for n in range(2000,2021):
    if (n%400==0 or (n%100!=0 and n%4==0)):
        print(n,end=" ")
```

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

In [79]:
```
s = 'banana 50 apple 250 greaps 330'
m = s.split()
g = 0
for i in m:
    if i.isdigit():
        g += int(i)
print(g)

```

```
630
```

In [75]:
```
1 dir(str)
```

Out[75]:
```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mod__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmod__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'capitalize',
 'casefold',
 'center',
 'count',
 'encode',
 'endswith',
 'expandtabs',
 'find',
 'format',
 'format_map',
 'index',
 'isalnum',
 'isalpha',
 'isdecimal',
 'isdigit',
 'isidentifier',
 'islower',
 'isnumeric',
 'isprintable',
 'isspace',
 'istitle',
 'isupper',
```

```
 'join',
 'ljust',
 'lower',
 'lstrip',
 'maketrans',
 'partition',
 'replace',
 'rfind',
 'rindex',
 'rjust',
 'rpartition',
 'rsplit',
 'rstrip',
 'split',
 'splitlines',
 'startswith',
 'strip',
 'swapcase',
 'title',
 'translate',
 'upper',
 'zfill']
```

Take multiple inputs from user

In [86]:
```
# 3 8 1
n=list(map(int,input().split()))
print("given input is ",n)
```

```
5 4 2 6 7 7
given input is  [5, 4, 2, 6, 7, 7]
```

In [87]:
```
n=int(input("enter the number "))
```

```
enter the number 67
```

tuple

- It is contain set of Hertogenoues data set
  - Same like as list
- It it is Immutable (do not change)
- short from tuple
- Reprasentation by ()

In [88]:
```
t=(10,20,30,15,10)
```

In [89]:
```
#print the available predefiend funtions
print(dir(tuple))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewa
rgs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__l
e__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__red
uce_ex__', '__repr__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__s
ubclasshook__', 'count', 'index']
```

In [90]:
```
t.count(10)
```

Out[90]:
```
2
```

sets

- represented by curly brace like { }
- It having unique values
- it is aslo mutable
- set

In [93]:
```
s={1,2,4,1,5,44,2}
```

In [94]:
```
s
```

Out[94]:
```
{1, 2, 4, 5, 44}
```

In [95]:
```
print(dir(set))
```

```
['__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__', '__x
or__', 'add', 'clear', 'copy', 'difference', 'difference_update', 'discard', 'i
ntersection', 'intersection_update', 'isdisjoint', 'issubset', 'issuperset', 'p
op', 'remove', 'symmetric_difference', 'symmetric_difference_update', 'union',
'update']
```

In [96]:
```
s.pop()
```

Out[96]:
```
1
```

In [97]:
```
s
```

Out[97]:
```
{2, 4, 5, 44}
```

In [98]:
```
s.discard(5)
```

In [99]:
```
1 s
```

Out[99]:
```
{2, 4, 44}
```

In [100]:
```
1 s.remove(4)
```

In [101]:
```
1 s
```

Out[101]:
```
{2, 44}
```

Dictionary

- the short form is dict
- data is in the form of Key and value pairs
  - like {'key':'value'}
- it is also mutable

In [102]:
```
1 d={"kits":2008,"python":1991,"apssdc":2014}
```

In [103]:
```
1 print(dir(dict))
```

```
['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc_
_', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt_
_', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len_
_', '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'cle
ar', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefaul
t', 'update', 'values']
```

In [104]:
```
1 d.keys()
2
```

Out[104]:
```
dict_keys(['kits', 'python', 'apssdc'])
```

In [130]:
```
1 d.popitem()
```

Out[130]:
```
('apssdc', 2014)
```

In [105]:
```
1 d.values()
```

Out[105]:
```
dict_values([2008, 1991, 2014])
```

In [106]:
```
1 d
```

Out[106]:
```
{'kits': 2008, 'python': 1991, 'apssdc': 2014}
```

In [107]:
```
# kits-->2008
# python-->1991
# apssdc-->2014
for k,v in d.items():
    print(k,"-->",v)
```

```
kits --> 2008
python --> 1991
apssdc --> 2014
```

In [108]:
```
d['python']=1989
```

In [109]:
```
d
```

Out[109]:
```
{'kits': 2008, 'python': 1989, 'apssdc': 2014}
```

In [112]:
```
d.pop('kits')
```

Out[112]:
```
2008
```

In [113]:
```
d
```

Out[113]:
```
{'python': 1989, 'apssdc': 2014}
```

In [115]:
```
nd={'key1':123,'key2':456,'key3':"apssdc"}
```

In [116]:
```
nd
```

Out[116]:
```
{'key1': 123, 'key2': 456, 'key3': 'apssdc'}
```

In [117]:
```
contacts={}
```

In [142]:
```
#name,number
def addContact(name,number):
    if name in contacts:
        print("already exist the name ",name)
    else:
        contacts[name]=number
        print("Contact added sucsessfully..")
name=input("Enter the name  :")
number=input("Enter the number :")
addContact(name,number)
```

```
Enter the name  :murali
Enter the number :5555
Contact added sucsessfully..
```

In [124]:
```
contacts
```

Out[124]:
```
{'muni': '678', 'raja': '0876'}
```

In [127]:
```
#update the mobile number
def updateContact(name):
    if name in contacts:
        m=int(input("enter the number for update..."))
        contacts[name]=m
        print("Contact updated sucsessfully..")
    else:
        print("oops..! not available the given name")
name=input("Enter the name  :")
updateContact(name)
```

```
Enter the name  :muni
enter the number for update...11111111
Contact updated sucsessfully..
```

In [128]:
```
contacts
```

Out[128]:
```
{'muni': 11111111, 'raja': '0876'}
```

In [134]:
```
#delete the contact
def deleteContact(name):
    if name in contacts:
        contacts.pop(name)
        print("Contact deleted sucsessfully..")
    else:
        print("oops..! not available the given name")
name=input("Enter the name  :")
deleteContact(name)
```

```
Enter the name  :h
oops..! not available the given name
```

In [143]:
```
contacts
```

Out[143]:
```
{'muni': 11111111,
 'rishi': '67655',
 'vara': '777777',
 'anusha': '33333',
 'mushi': '565454',
 'shasi': '1222',
 'siri': '420',
 'murali': '5555'}
```

In [144]:
```
#display all the contacts
def displayallContacts(d):
    if len(contacts)>0:
        for k,v in contacts.items():
            print(k,"==>",v)
    else:
        print("empty list")
displayallContacts(name)
```

```
muni ==> 11111111
rishi ==> 67655
vara ==> 777777
anusha ==> 33333
mushi ==> 565454
shasi ==> 1222
siri ==> 420
murali ==> 5555
```

Regular Expressions

- It is lanunge for matching the patterns by using symbols
- short form is re

In [145]:
```
import re
```

In [146]:
```
print(dir(re))
```

```
['A', 'ASCII', 'DEBUG', 'DOTALL', 'I', 'IGNORECASE', 'L', 'LOCALE', 'M', 'MULTI
LINE', 'RegexFlag', 'S', 'Scanner', 'T', 'TEMPLATE', 'U', 'UNICODE', 'VERBOSE',
'X', '_MAXCACHE', '__all__', '__builtins__', '__cached__', '__doc__', '__file_
_', '__loader__', '__name__', '__package__', '__spec__', '__version__', '_alpha
num_bytes', '_alphanum_str', '_cache', '_compile', '_compile_repl', '_expand',
'_locale', '_pattern_type', '_pickle', '_subx', 'compile', 'copyreg', 'enum',
'error', 'escape', 'findall', 'finditer', 'fullmatch', 'functools', 'match', 'p
urge', 'search', 'split', 'sre_compile', 'sre_parse', 'sub', 'subn', 'templat
e']
```

In [149]:
```
#match
re.match('a','helloapssdc')#it will match only first letter
```

In [152]:
```
#search
re.search('a','kitsapssdc')
```

Out[152]:
```
<_sre.SRE_Match object; span=(4, 5), match='a'>
```

In [153]:
```
re.search('a','kitsapssdcap')#it match the only first success occure
```

Out[153]:
```
<_sre.SRE_Match object; span=(4, 5), match='a'>
```

In [154]:
```
#findfAll
re.findall("a",'apssdc')

```

Out[154]: ['a']

In [155]:
```
re.findall("a",'apssdcap')

```

Out[155]: ['a', 'a']

In [ ]:
```
https://bit.ly/2DWIKWV
```