In [2]:
```
# procedure style
# object oriented style
# functional style

# numbers - int,float,complex
# str - 'A-Za-z0-9spacespecial'  collection of chars
# bytes - collection of ASCII b'' python 3.x
# bool -True/False
# None
# ---------------------------------------//scalar(Single)

# containers - list,tuple,dict,set
# list,tuple - collection of ordered elements index based
# |      |_immutableabs
# mutable
# dict - collection of unordered elements - data{"Key":"Value"} - mutable
# set  - collection of unordered elements - key only - not allowed duplicate valu

# List,tuple,dict <==MD
# Syntax:-
# --------
# variable = value
# |__ name starts with A-Za-z_ not starts with digits; not allowed space and spec
#
n=56
cost=3.566
COUNT=15
port=4255
# type(namedvariable) (or) type(value) ->determine datatype
print(n,type(n))
print(cost,type(cost))
print(COUNT,type(COUNT))
print(port,type(port))

s='data'
print(s,type(s))
```

```
56 <class 'int'>
3.566 <class 'float'>
15 <class 'int'>
4255 <class 'int'>
data <class 'str'>
```

In [8]:
```
# typecasting
n=56
# convert to float -->float(input_int)
float(n)
# convert to str -->str(input)
str(n)
f=42.4566
print(int(f))
str(f)
s="45678\n"
total=float(s)*0.12
```

```
42
```

In [11]:
```
n=56
print(float(n))
print(n,type(n))
n=float(n)
print(n,type(n))
```

```
56.0
56 <class 'int'>
56.0 <class 'float'>
```

In [14]:
```
# bool - True,False
print(type(True),type(False))
```

```
<class 'bool'> <class 'bool'>
```

In [25]:
```
bool(0) bool(0.0) bool('')bool(b'')bool([])bool(())bool({})
bool(None)
```

Out[25]:
```
False
```

In [26]:
```
# in python any expression (or) any method -> returns bool value True/False
# and input validation/test - we can use conditional statement - if statement
name="root"
if(name == "root"):
    print("Login name is:{}".format(name))
else:
    print("Invalid name")
```

```
Login name is:root
```

In [28]:
```
name=input("Enter a login name:")
if(name == "root"):
    print("Success")
else:
    print("Failed")
```

```
Enter a login name:
Failed
```

In [32]:
```
name=input("Enter your name:")
if(name):
    print("Hello...{}".format(name))
else:
    print("Sorry - empty string!!")
```

```
Enter your name:
Sorry - empty string!!
```

In [36]:
```
N=input("Enter N value:")
print(N,type(N))
```

```
Enter N value:56
56 <class 'str'>
```

In [37]:
```
# Write a python program:
# STEP 1: read a diskpartition name from <STDIN>
# STEP 2: read a diskpartition Size from <STDIN>
# STEP 3: read another disk partition and size from <STDIN>
# STEP 4: calculate sum of disk size
# STPE 5: display each partition name - individual size - total size

p1=input("Enter a disk partition name:")
s1=int(input("Enter {} size:".format(p1)))
p2=input("Enter a disk partition name:")
s2=input("Enter {} size:".format(p2))
total=s1+int(s2)
print("""
Partition name:{}\t   Size:{}
---------------------------------
Partition name:{}\t   Size:{}
---------------------------------
        Total :{}
---------------------------------""".format(p1,s1,p2,s2,total))
```

```
Enter a disk partition name:/dev/sda1
Enter /dev/sda1 size:450
Enter a disk partition name:/dev/sda2
Enter /dev/sda2 size:540

Partition name:/dev/sda1          Size:450
---------------------------------
Partition name:/dev/sda2          Size:540
---------------------------------
        Total :990
---------------------------------
```

In [41]:
```
# Write a python program:
# STEP 1 : read a login name from <STDIN>
# STEP 2: test input login name is 'root' or not
#                                   ------    =====
#                                     |         |__ exit from script
#                        STEP 3:   read a shell name from <STDIN>
#                        STEP 4:   test input shell name is bash or not
#                        STEP 5:  display login name and shell name

name=input("Enter a login name:")
if(name == "root"):
    var=input('Enter a shell name:')
    if(var == 'bash'):
        print("Login name:{}\t Login shell name:{}".format(name,var))
    else:
        print("Sorry input shell name is not bash shell")
else:
    print("Sorry input user name is not root")
```

```
Enter a login name:root
Enter a shell name:bash
Login name:root  Login shell name:bash
```

In [48]:
```
name=input("Enter a login name:")
if(name == "root"):
    var=input('Enter a shell name:')
    if(var == 'bash' or var == 'ksh' or var == 'csh'): # test any shell name is m
        print("Login name:{}\t Login shell name:{}".format(name,var))
    else:
        print("Sorry input shell name is not valid shell")
else:
    print("Sorry input user name is not root")
```

```
Enter a login name:root
Enter a shell name:psh
Sorry input shell name is not valid shell
```

In [51]:
```
# Write a python program
# read a port number from <STDIN>
# test input port number -> 501-599   ==> valid port
port=int(input("enter a port number:"))
if(port >500 and port<600):
    print("Valid port")
else:
    print("Invalid port")
```

```
enter a port number:5666
Invalid port
```

In [ ]:
```
# Multiconditional statements
# ----------------------------
# if(condition1):
#          True block1
# elif(condition2):
#          True block2
# elif(condition3):
#          True block3
#  ..
# elif (conditionN):
#           True block
# else:
#           False block
'''
Write a python program:
    read a login name test input login name is root or not
    if input login name is matched - read a shell name from <STDIN>
    test input shell bash -> initialize profile filename bashrc
    test input shell ksh  -> initialize profile filename kshrc
    test input shell csh  -> initialize profile filename cshrc
    if all 3 shell name is not matched -> default shell name ->/bin/nologin
                                          default profile -> /etc/profile
    display login name , shell name and profilefilename
'''
```

In [56]:
```
name=input("Enter a login name:")
if(name == "root"):
    var=input("Enter a shell name:")
    if(var == 'bash'):
        fname="bashrc"
    elif(var == 'ksh'):
        fname="kshrc"
    elif(var == 'csh'):
        fname="cshrc"
    else:
        print("Sorry your input shell is not matched,so we assigned default shell
        var="/bin/nologin"
        fname="/etc/profile"
    print("Login name:{}\tShell name:{}\tProfile:{}".format(name,var,fname))
else:
    print("Sorry your not root user")
```

```
Enter a login name:root
Enter a shell name:tcsh
Sorry your input shell is not matched,so we assigned default shell and default
profile file
Login name:root Shell name:/bin/nologin Profile:/etc/profile
```

In [57]:
```
# Looping statements
# ----------------------
# in python looping statements we can write two ways:
#     1. Conditional style loop - based on True/False - while
#     2. Collection style loop -  based on data       - for
# while - loop
#  ->initialize
#  ->condition
#  ->arithmetic

# python won't support ++/-- operators

# if    vs   while
# ---        ------
# one time   more than one time execution
c=0
while(c<5):
    print("Test report..{}".format(c))
    c=c+1 # c+=1
```

```
Test report..0
Test report..1
Test report..2
Test report..3
Test report..4
```

In [62]:
```
while(False):
    print("Hello")
```

In [59]:
```
while(True):
    print("Hello")
    break # exit from loop
```

```
Hello
```

In [61]:
```
# for variable in collection:
s="Hello"
print(len(s))
for var in s:
    print("Hello...{}".format(var))
```

```
5
Hello...H
Hello...e
Hello...l
Hello...l
Hello...o
```

In [63]:
```
c=0
while(c<5):
    print("Test report..{}".format(c))
    c=c+1
else:
    print("-"*50)
    print("\t Thank you\t")
    print("-"*50)
```

```
Test report..0
Test report..1
Test report..2
Test report..3
Test report..4
--------------------------------------------------
         Thank you
--------------------------------------------------
```

In [64]:
```
for var in 'abc':
    print("Test code:{}".format(var.upper()))
else:
    print("Thank you")
```

```
Test code:A
Test code:B
Test code:C
Thank you
```

In [ ]:
```
# == != < <= > >=  relational operator ->True/False
# (int,float,str) ->True/False
#
# in not in  membership  ->True/False
# (str,bytes,list,tuple,dict,set) ->True/False

"pattern" in inputcollection ->True
"sales"  in "ram,sales,pune" ->True
 -----
"sales," == "sales" ->False
```

In [70]:
```
if('e' in 'hello'):
    print("Yes")
else:
    print("No")

if('hello' in 'hello '):
    print("Yes")
else:
    print("No")

if('hello' == 'hello '):
    print("Yes")
else:
    print("No")
```

```
Yes
Yes
No
```

In [76]:
```
# List - Collection of ordered elements - index - [] - mutable
# Listname=[list of elements]
a=45
b=2.45
c='data'
d=True
e=b'abc'
f=None
L=[a,b,c,d,e,f,100,'pythonprogramming']
print(type(L))
print(len(L))
print(L)
print(type(L[0]),type(L[1]),type(L[2]))
```

```
<class 'list'>
8
[45, 2.45, 'data', True, b'abc', None, 100, 'pythonprogramming']
<class 'int'> <class 'float'> <class 'str'>
```

In [79]:
```
L=[100,200,'data1','data2','data3']
print(type(L))
print(L)
print(L[1])
L[1]=343.43
print(L)
# name[0] name[1] name[2] ....name[N] --->
#
# L[-1]
# L[-2]
print(L[-1])
print(L[-2])
```

```
<class 'list'>
[100, 200, 'data1', 'data2', 'data3']
200
[100, 343.43, 'data1', 'data2', 'data3']
data3
data2
```

In [84]:
```
s="Linux"
print(s[0])
L=["Linux"]
print(L[0][0])
for var in 'Linux':
            |||||
            01234
for var in ['',123,'']:
```

```
L
L
```

In [91]:
```
L=[]
# adding new element into exsiting list
# Listname.append(value) ->None
# Listname.insert(index,Value) ->None

# Listname.pop(value) ->removed_value (or) del(listname[index]) ->None

# help(list)  help(str) help(dict) help(int)
#help(list.append)
print(len(L))
L.append("D1")
L.append("D2")
L
L.insert(1,"D3")
L.append("D4")
L
L.pop() # Listname.pop() ->default removed last index value

# L.pop(index) -> removed existing index value

# L[index]
#    |__ outofrange ->IndexError <--- L.pop(invalid_index)
```

```
0
```

Out[91]:
```
'D3'
```

In [92]:
```
# Write a python program:
# STEP 1: create a empty list
# STEP 2: display size of given list
# STEP 3: using while loop - 5times
#          read a hostname from <STDIN>
#          append input hostname into existing list
# STEP 4: using for loop - display list of elements
# STEP 5: display size of the list

hosts=[] # empty list
print("Size of given list:{}".format(len(hosts))) # Size of given list
i=0
while(i<5):   # STEP 3
    v=input("Enter a hostname:")
    hosts.append(v) # adding data to existing list
    i+=1 # i=i+1
print("List of host details:-")
for var in hosts:
    print(var)
else:
    print("Size of given list:{}".format(len(hosts))) # Size of given list
```

```
Size of given list:0
Enter a hostname:host01
Enter a hostname:host02
Enter a hostname:host03
Enter a hostname:host04
Enter a hostname:host05
List of host details:-
host01
host02
host03
host04
host05
Size of given list:5
```

In [99]:
```
"e" in "hello" # True
"o" in "hello" # True
"eo" in "hello" # False
# "input" in List ->True/False
"unix" in ["Linux","aix","unix","winx"]
```

Out[99]: True

In [100]:
```
hosts=[] # empty list
print("Size of given list:{}".format(len(hosts))) # Size of given list
i=0
while(i<5):   # STEP 3
    v=input("Enter a hostname:")
    hosts.append(v) # adding data to existing list
    i+=1 # i=i+1
print("List of host details:-")
for var in hosts:
    print(var)
else:
    print("Size of given list:{}".format(len(hosts))) # Size of given list
```

```
Size of given list:0
Enter a hostname:host01
Enter a hostname:host02
Enter a hostname:host01
Enter a hostname:host02
Enter a hostname:host01
List of host details:-
host01
host02
host01
host02
host01
Size of given list:5
```

In [101]:
```
hosts=[] # empty list
print("Size of given list:{}".format(len(hosts))) # Size of given list
i=0
while(i<5):   # STEP 3
    v=input("Enter a hostname:")
    if v in hosts:
        print("Sorry hostname {} is already exists.".format(v))
    else:
        hosts.append(v) # adding data to existing list
    i+=1 # i=i+1
print("List of host details:-")
for var in hosts:
    print(var)
else:
    print("Size of given list:{}".format(len(hosts))) # Size of given list
```

```
Size of given list:0
Enter a hostname:host01
Enter a hostname:host02
Enter a hostname:host01
Sorry hostname host01 is already exists.
Enter a hostname:host03
Enter a hostname:host02
Sorry hostname host02 is already exists.
List of host details:-
host01
host02
host03
Size of given list:3
```

In [107]:
```
s="sample python programming document"
print(len(s))
# slicing name[n:m] # from nth index to m-1index
print(s[3:15]) # from 3rd index to 14th index(15-1)
# name[n] ->nth index(single)
# name[n:] ->from nth index to list of all (multiple)
print(s[3:]) # from 3rd index to list of all
# name[:m] # from 0th index to m-1 index
print(s[:3]) # from 0th index to 2nd (3-1)
print(s[-3:]) # last 3 chars
```

```
34
ple python p
ple python programming document
sam
ent
```

In [108]:
```
L=["D1","D2","D3","D4","D5","D6","D7"]
print(L[1:5])
print(L[:3])
print(L[3:])
print(L[-3:]) # last 3elemnts
```

```
['D2', 'D3', 'D4', 'D5']
['D1', 'D2', 'D3']
['D4', 'D5', 'D6', 'D7']
['D5', 'D6', 'D7']
```

In [118]:
```
# List - collection of ordered elements - index -based - mutable - []
# -----
# tuple - collection of ordered elements - index -based - immutable - ()
# -----
#   |__ record set - fixed

# tuple - supports index,slicing

T=("D1",134,3.45,True,'10.20.30.40')
print(type(T),T)
v=T[1]
for var in T:
    print(var)
else:
    print(len(T))

if(134 in T):
    print("Yes")
else:
    print("No")

"D1" in T[-3:]
```

```
<class 'tuple'> ('D1', 134, 3.45, True, '10.20.30.40')
D1
134
3.45
True
10.20.30.40
5
Yes
```

Out[118]:
```
False
```

In [121]:
```
T1=(100,200,300)
T2=('d1','d2')
T1+T2
print(T1)
print(T2)
```

```
(100, 200, 300)
('d1', 'd2')
```

In [123]:
```
# python support multiple initialization
v1,v2,v3=10,3.45,'data'
a,b,c=100,True,[]
r1,r2=(),[]
```

In [129]:
```
L=['kumar','sales','pune']
name=L[0]
dept=L[1]
city=L[-1]

n,d,c=L # multiple initialization
print("Emp name is:{} working city is:{}".format(n,c))
L.append(1000)
L.append('+91 9923234323')

n,d,c,v1,v2=L # multiple initialization
```

```
Emp name is:kumar working city is:pune
```

In [127]:
```
T=('22','Feb','2021')
d,m,Y=T # multiple initialization
print("Today:{}\tMonth:{}\tYear:{}".format(d,m,Y))
```

```
Today:22	Month:Feb	Year:2021
```

In [152]:
```
L=['d1','D2','D3','D4','D5']
#   0    1     2    3    4
#  -5   -4    -3   -2   -1
L[-3:-1]  # -1-1 ->-2
L[-3:-2]  # -2-1 ->-3

L[-4:-1] ## (1)

r=L[-4:] ##(2)
r[:-1]   ##(2)
```

Out[152]:
```
['D2', 'D3', 'D4']
```

In [157]:
```
# List/tuple -  collection of ordered element - index based - [] -mutable; ()-imm
#
# dict - collection of unordered elements - key:value -> key - {} - mutable
d={} # empty dict

d={"Key1":"Value1","K2":100,"K3":34.4556}
 ##      ^        |        |
# dict - like table ->Row X column
#
#    |   Column1   |    Column2  |
#    |-------------|-------------|
#    |   Key1      |    Value    |   <===>
#    |-----------------------------|-
#    |   Key2      |    Value    |
#    -------------------------------
#        |             |

# dict key - unique element - immutable type
d={1:'V',1.34:'V2','data':'V3',True:'V4',():'V5'}

# dict value  can duplicate
d={"K1":100,"K2":"p1.log","K3":100,"K4":"p1.log"}
print(d)
d={"K1":100,"K2":200,"K1":"p1.log"}
print(d)
```

```
{'K1': 100, 'K2': 'p1.log', 'K3': 100, 'K4': 'p1.log'}
{'K1': 'p1.log', 'K2': 200}
```

In [154]:
```
fileinfo={"F1":"p1.log","Findex":123454,"Fsize":'100KB'}
print(type(fileinfo),len(fileinfo))
```

```
<class 'dict'> 3
```

In [161]:
```
# How to fetch single data from dict?
#   dictname['existing_key'] ->Value
#              |
#              |__invalid key ->KeyError vs IndexError <--Listname[index]
#                                                              invalid index

print("filename:{}\tIndex:{}".format(fileinfo["F1"],fileinfo["Findex"]))

# How to modify existing data from dict?
#  dictname['old_key']=updated_Value

fileinfo['F1']="/var/log/test.log" # modification
print(fileinfo)
```

```
filename:p1.log Index:123454
{'F1': '/var/log/test.log', 'Findex': 123454, 'Fsize': '100KB'}
```

In [164]:
```
d={}  # empty dict -> # L=[] ->L.append('data')
# dictname['newkey']=value
d['K1']=100
d['K2']='p1.log'
d['K3']="/bin/bash"
print(d)
print(fileinfo,len(fileinfo))
fileinfo['perm']="rw-rw-r--" # adding new data to existing dict
fileinfo['count']=1          # adding new data to existing dict
print(fileinfo,len(fileinfo))
fileinfo['count']=2   # modification
```

```
{'K1': 100, 'K2': 'p1.log', 'K3': '/bin/bash'}
{'F1': '/var/log/test.log', 'Findex': 123454, 'Fsize': '100KB'} 3
{'F1': '/var/log/test.log', 'Findex': 123454, 'Fsize': '100KB', 'perm': 'rw-rw-
r--', 'count': 1} 5
```

In [165]:
```
d={"K1":100,"K2":200,"K3":300}
del(d["K1"])
print(d)
```

```
{'K2': 200, 'K3': 300}
```

In [173]:
```
#help(dict)
d={}
d['K1']='V1' ##(1)
# dictname.setdefault("Key","Value") ##(2)
d.setdefault("K2","V2") # adding new element into existing dict
print(d['K1'],d['K2'])
# print(d['K3']) # KeyError
# dictname.get("Key") ->Value/None
print(d.get("K1"))
print(d.get("Kx"))
d.get("Kx") == None

# d.pop("Key") ->removed value vs Listname.pop()->-1index ; Listname.pop(index)
d.pop("K1")
```

```
V1 V2
V1
None
```

Out[173]:
```
'V1'
```

In [177]:
```
for var in fileinfo:
    print(var)
# dictname['key'] ->Value
print("")
for var in fileinfo:
    print(fileinfo[var])
print("")
for var in fileinfo:
    print("{}\t{}".format(var,fileinfo[var]))
```

```
F1
Findex
Fsize
perm
count

/var/log/test.log
123454
100KB
rw-rw-r--
1

F1      /var/log/test.log
Findex  123454
Fsize   100KB
perm    rw-rw-r--
count   1
```

In [181]:
```
# dictname.keys()
for var in fileinfo.keys():
    print(var)
print("")
# dictname.values()
for var in fileinfo.values():
    print(var)
```

```
F1
Findex
Fsize
perm
count

/var/log/test.log
123454
100KB
rw-rw-r--
1
```

In [185]:
```
d={}
d.setdefault("K1","V1")
tmp={"Kx":100,"Ky":222}
# newdictname.update(existict_dict)
tmp.update(d)
print(tmp)
tmp.update(K2=1000,user='root')
tmp
```

```
{'Kx': 100, 'Ky': 222, 'K1': 'V1'}
```

Out[185]:
```
{'Kx': 100, 'Ky': 222, 'K1': 'V1', 'K2': 1000, 'user': 'root'}
```

In [ ]:
```
# List=['d1',12,3,343,343]  -> | d1  | 12 | 3 | 343 | 343 |
#                                0     1    2   3     4   <-- index

# dict={"K1":"V1","K2":"V2","K3":"V3"}

#   Key    |   Value
# ------------------------
#   K1     |   V1
# ---------|--------------------
#   K2     |   V2
# ---------|-----------------
#   K3     |   V3
#--------------------------

# in dict -> pair("Key":"Value")

# "inputkey" in dict -> test key is existing or not
```

In [ ]:
```
Q1. Given List
EMP=['101,kumar,sales,pune,1000','203,arun,prod,bglore,2000,'345,paul,HR,mumbai,3
display emp name and working city name to monitor
calculate sum of emp's cost
Expected result:-
------------------
Emp name is: Kumar  working city:PUNE
Emp name is: Arun   working city:BGLORE
Emp name is: Paul   working city:MUMBAI
----------------------------------------
     Total Emp's cost:6000
----------------------------------------

========================================================================
Q2. LB=[10.34,0.32,0.334]
Calculate sum of LB and average of LB
========================================================================

Q3. shell details: /bin/bash version is bash-4.24  login path:/root
create a dict - display key/value details
```

In [188]:
```
EMP=['101,kumar,sales,pune,1000','203,arun,prod,bglore,2000','345,paul,HR,mumbai,
total=0
for var in EMP:
     eid,ename,edept,ecity,ecost=var.split(",")
     print("Emp name is:{}\t Working city:{}".format(ename.title(),ecity.upper())
     total=total+int(ecost)
else:
     print("-"*50)
     print("\tSum of Emp cost:{}".format(total))
     print("-"*50)
```

```
Emp name is:Kumar        Working city:PUNE
Emp name is:Arun         Working city:BGLORE
Emp name is:Paul         Working city:MUMBAI
--------------------------------------------------
        Sum of Emp cost:6000
--------------------------------------------------
```

In [196]:
```
LB=[10.34,0.32,0.334]
t=0
for var in LB:
    t=t+var
else:
    print("-"*45)
    print("Total LB:{}".format(t))
    print("Avg:{:.3}".format(t/len(LB)))
    print("-"*45)
```

```
---------------------------------------------
Total LB:10.994
Avg:3.66
---------------------------------------------
```

In [195]:
```
# Q3. shell details: /bin/bash version is bash-4.24  login path:/root
# create a dict - display key/value details
shellinfo={} # empty dict

shellinfo['name']='/bin/bash'
shellinfo['version']='bash-4.24'
shellinfo['path']='/root'
for var in shellinfo:
    print("{}\t{}".format(var,shellinfo[var]))
```

```
name    /bin/bash
version bash-4.24
path    /root
```

In [198]:
```
# Write a python program:
# STEP 1: create a empty dict
# STEP 2: display size of given dict
# STEP 3: using while loop - 5times
#          read a alias from <STDIN> (ex: host01)
#          read a IP Address from <STDIN> (ex: 10.20.30.40)
#          add input details into existing dict {'host01':'10.20.30.40'}
#          Validation: key(alias name) must be unique
# STEP 4: using for loop - display dict details
# STEP 5: display size of the list

hosts={} # empty dict
print("Size of dict:{}".format(len(hosts)))
c=0
while(c<5):
    K=input("Enter a alias name:")
    if K in hosts: # test dict key is exists or not
        print("Sorry input alias name:{} is already exists".format(K))
    else:
        V=input("Enter an IP Address:")
        hosts[K]=V # hosts.setdefault(K,V)
    c=c+1

print("Alias and IP-Address details:-")
for var in hosts:
    print("{}\t{}".format(var,hosts[var]))
```

```
Size of dict:0
Enter a alias name:host01
Enter an IP Address:10.20.30.40
Enter a alias name:host02
Enter an IP Address:10.20.34.55
Enter a alias name:host03
Enter an IP Address:10.20.40.34
Enter a alias name:host04
Enter an IP Address:10.43.34.11
Enter a alias name:host01
Sorry input alias name:host01 is already exists
Alias and IP-Address details:-
host01  10.20.30.40
host02  10.20.34.55
host03  10.20.40.34
host04  10.43.34.11
```

In [200]:
```
print(hosts)
hosts['host03']='127.0.0.1'
print(hosts)
```

```
{'host01': '10.20.30.40', 'host02': '10.20.34.55', 'host03': '10.20.40.34', 'ho
st04': '10.43.34.11'}
{'host01': '10.20.30.40', 'host02': '10.20.34.55', 'host03': '127.0.0.1', 'host
04': '10.43.34.11'}
```

In [ ]:
```
# List of List,tuple,dict  --> L=[[],(),{}]
# tuple of List,tuple,dict ---> T=([],(),{})
# dict of List,tuple,dict------------------->d={"K1":[],"K2":(),"K3":{}}
# -----   ========================
#  |              |__ unnamed
# named

# list,dict - mutable ; tuple - immutable
# list,tuple - index based ; supports slicing
# dict - keybased
```