In [2]:
```
L=[[100,200,300,400],(500,600,'D1'),{"K1":"V1","K2":"V3"}]
print(type(L),len(L))
print(type(L[0]))
print(type(L[1]))
print(type(L[-1]))
```

```
<class 'list'> 3
<class 'list'>
<class 'tuple'>
<class 'dict'>
```

In [5]:
```
L=[[100,200,300,400],(500,600,'D1'),{"K1":"V1","K2":"V3"}]
#    0    1   2  3     0   1   2      --------|---------
#   -4  -3   -2 -1    -3   -2  -1
#  ---------0th---- | ----1st------|---------2nd---------
#        -3               -2                  -1

print(L[0][0])
print(L[0][-1])
print(L[1][0])
print(L[-1]['K1'])
```

```
100
400
500
V1
```

In [8]:
```
# listname.append(Value)
#
# listname.append([]) List of List
# listname.append(()) List of tuple
# listname.append({}) List of dict
L=[]
L.append(['d1','d2','d3','d4'])
L.append(('T1','T2','T3','t4'))
L.append({"url":'python.org','port':80})
L.append({'url':'pypi.org','port':1200})
L[2]['url']
```

Out[8]: 'python.org'

In [11]:
```
import pprint
pprint.pprint(L)
L[0][1]='Data2'
L[-1]['port']=6550
print("")
pprint.pprint(L)
L[-1]['url']='https://www.anaconda.com'
pprint.pprint(L)
```

```
[['d1', 'Data2', 'd3', 'd4'],
 ('T1', 'T2', 'T3', 't4'),
 {'port': 80, 'url': 'python.org'},
 {'port': 6550, 'url': 'pypi.org'}]

[['d1', 'Data2', 'd3', 'd4'],
 ('T1', 'T2', 'T3', 't4'),
 {'port': 80, 'url': 'python.org'},
 {'port': 6550, 'url': 'pypi.org'}]
[['d1', 'Data2', 'd3', 'd4'],
 ('T1', 'T2', 'T3', 't4'),
 {'port': 80, 'url': 'python.org'},
 {'port': 6550, 'url': 'https://www.anaconda.com'}]
```

In [13]:
```
L[0].append("Dx")
L[0].append(["D123","D3445"])
pprint.pprint(L)
```

```
[['d1', 'Data2', 'd3', 'd4', 'Dx', ['D123', 'D3445']],
 ('T1', 'T2', 'T3', 't4'),
 {'port': 80, 'url': 'python.org'},
 {'port': 6550, 'url': 'https://www.anaconda.com'}]
```

In [17]:
```
emp={"name":"arun","dept":'sales','place':'pune'}
emp.keys()
emp.values()
for var in emp.items(): # [("name","arun"),("dept","sales"),("place","pune")]
    print(var)
```

```
('name', 'arun')
('dept', 'sales')
('place', 'pune')
```

In [27]:
```
# function/method call return type -->tuple(),tuple of list ,tuple of tuple,tuple

T=([],[]) # tuple of list
print(len(T))
# type(T[0]) ->list

T[0].append("F1")
T[0].append("F2")
T[0].append("F3")
T[0].append("F4")
T[1].append("F5")
T[1].append("F6")
print(len(T))
pprint.pprint(T)
T[0][0]='DATA'
pprint.pprint(T)
# T[0]="DATA" # Error
```

```
2
2
(['F1', 'F2', 'F3', 'F4'], ['F5', 'F6'])
(['DATA', 'F2', 'F3', 'F4'], ['F5', 'F6'])
```

In [29]:
```
t=([],('d1','d2'),{'K1':"V1","K2":"V2"})
type(t)
t[-1]['K1']
```

Out[29]:
```
'V1'
```

In [32]:
```
d={"K1":"V1"} # 1 to 1 value
d={"K1":[],"K2":(),"K3":{}} # 1 to many
# dict oflist; dict of tuple; dict of dict

emp={'names':[],'depts':('sales','prod','DBA','HR')}
print(type(emp))
print(type(emp['names']))
print(type(emp['depts']))

# object['key'][index]   vs  object[index]['key']
# --------------------       ----------------------
# namedict                                unnamed
```

```
<class 'dict'>
<class 'list'>
<class 'tuple'>
```

In [35]:
```
emp['names'].append('arun')
emp['names'].append('vijay')
emp['names'].append('paul')
pprint.pprint(emp)
```

```
{'depts': ('sales', 'prod', 'DBA', 'HR'), 'names': ['arun', 'vijay', 'paul']}
```

In [37]:
```
# for var in range() same as while() loop
d={}
d['network']=['eth0','IP-Add','netmask']
d['file']=('p1.log','p2.log')
pprint.pprint(d)

d={}
for var in range(3):
    d['K'+str(var+1)]=[]

pprint.pprint(d)
```

```
{'file': ('p1.log', 'p2.log'), 'network': ['eth0', 'IP-Add', 'netmask']}
{'K1': [], 'K2': [], 'K3': []}
```

In [38]:
```
d['K1'].append("V1")
d['K1'].append(1232+3232)
d['K1'].append(['D1','D2'])

d['K2'].append('File1')
pprint.pprint(d)
```

```
{'K1': ['V1', 4464, ['D1', 'D2']], 'K2': ['File1'], 'K3': []}
```

In [39]:
```
d={}
for var in range(3):
    d['K'+str(var+1)]=var+100
print(d)
```

```
{'K1': 100, 'K2': 101, 'K3': 102}
```

In [41]:
```
# d={"K1":"V1"}

d={"K1":{"K1":100,"K2":200,"K3":300,"K4":400},"K2":{"K1":"F1","K2":"F2"}}
#   ---   ---      ---                          ---   ---       ---
pprint.pprint(d)
type(d['K1'])
```

```
{'K1': {'K1': 100, 'K2': 200, 'K3': 300, 'K4': 400},
 'K2': {'K1': 'F1', 'K2': 'F2'}}
```

Out[41]:
```
dict
```

In [43]:
```
for var in d['K1']:
    print(var)

d['K1'].keys()
```

```
K1
K2
K3
K4
```

Out[43]:
```
dict_keys(['K1', 'K2', 'K3', 'K4'])
```

In [46]:
```
print(d['K1']['K1'])
print(d['K1']['K2'])
print(d['K2']['K1'])
d['K1']['K2']='DATA'
pprint.pprint(d)
```

```
100
200
F1
{'K1': {'K1': 100, 'K2': 'DATA', 'K3': 300, 'K4': 400},
 'K2': {'K1': 'F1', 'K2': 'F2'}}
```

In [ ]:
```
# dict of list of dict
#   (or)
# list of dict of list of dict
```

In [48]:
```
s1="101,arun,sales,pune,1000"
s2="102:vijay,prod,bglore,2000"
s3="103-anu,HR,hyderabad,3000"

d={}
print(s1.split(",")) # []
d['K1']=s1.split(",") #  dict of list
d
```

```
['101', 'arun', 'sales', 'pune', '1000']
```

Out[48]:
```
{'K1': ['101', 'arun', 'sales', 'pune', '1000']}
```

In [51]:
```
d['K2']=s2.split(":")
d
```

Out[51]:
```
{'K1': ['101', 'arun', 'sales', 'pune', '1000'],
 'K2': ['102', 'vijay,prod,bglore,2000']}
```

In [54]:
```
d['K3']=s3.split("-")
d['K3'][1].split(",")
```

Out[54]:
```
['anu', 'HR', 'hyderabad', '3000']
```

In [55]:
```
s="root:x:bin:bash"
print(s.split(":"))
s="root:x-bin~bash"
print(s.split(":"))
```

```
['root', 'x', 'bin', 'bash']
['root', 'x-bin~bash']
```

In [60]:
```
import re
s="root:x-bin~bash"
print(re.split("[^\w\s]",s))
s1="root:x:bin:bash"
print(re.split(":",s1))
```

```
['root', 'x', 'bin', 'bash']
['root', 'x', 'bin', 'bash']
```

In [61]:
```
s='bash-4.24'
L=['p1.log','p2.txt','p3.cpp']
t=('t1','t2','t3')
d1={"K1":"V1","K2":"V2"}

d={}
d['K1']=s # 1D
d['K2']=L # 1to many dict of list
d['K3']=t # 1to many dict of list
d['K4']=d1 # 1 to many dict of dict
pprint.pprint(d)
```

```
{'K1': 'bash-4.24',
 'K2': ['p1.log', 'p2.txt', 'p3.cpp'],
 'K3': ('t1', 't2', 't3'),
 'K4': {'K1': 'V1', 'K2': 'V2'}}
```

In [65]:
```
d1={"K1":"V1"}
d2={"K1":100,"K2":200}
#d1.update(d2)
#d1
d1['NEWKEY']=d2
d1
```

Out[65]:
```
{'K1': 'V1', 'NEWKEY': {'K1': 100, 'K2': 200}}
```

In [68]:
```
# dict - collection of unordered elements  -  {"Key":"Value"}
#  |
# set  - Collection of unordered  elements - {Key1}
#  |_not supporting index,key:value  - there is no modification

s={"K1","V1",100,200,"K1","V1","K1",100,100,200,100}
print(type(s))
print(s)
print(len(s))
for var in s:
    print(var)
"K1" in s
```

```
<class 'set'>
{200, 'K1', 100, 'V1'}
4
200
K1
100
V1
```

Out[68]:
```
True
```

In [71]:
```
s={10,20}
# s.add(single)    vs   s.update(list,tuple,dict)
s.add('data')
s.add(True)
s.update(['D1','D2',10,20,30,40])
s
```

Out[71]:
```
{10, 20, 30, 40, 'D1', 'D2', True, 'data'}
```

In [74]:
```
#s.remove(30)
s.discard('D1')
s
```

Out[74]:
```
{10, 20, 40, 'D2', True, 'data'}
```

In [75]:
```
L1=['p1.c','p2.java','index.html','test.py']

L2=['p1.txt','ab.txt','emp.csv','index.html','p2.java','test.py']

set(L1)|set(L2)
set(L1).union(set(L2))
```

Out[75]:
```
{'ab.txt', 'emp.csv', 'index.html', 'p1.c', 'p1.txt', 'p2.java', 'test.py'}
```

In [76]:
```
set(L1).intersection(set(L2)) # set(L1)&(set(L2))
```

Out[76]:
```
{'index.html', 'p2.java', 'test.py'}
```

In [77]:
```
set(L1).difference(set(L2))
```

Out[77]:
```
{'p1.c'}
```

In [78]:
```
set(L2).difference(set(L1))
```

Out[78]:
```
{'ab.txt', 'emp.csv', 'p1.txt'}
```

In [81]:
```
set(L1)^set(L2)
set(L1).symmetric_difference(set(L2))
```

Out[81]:
```
{'ab.txt', 'emp.csv', 'p1.c', 'p1.txt'}
```

In [87]:
```
s1={'d1','d2'}
s2=frozenset(s1)
s1.add('d3')
s1
#s2.add('d4') =>AttributeError
```

Out[87]:
```
{'d1', 'd2', 'd3'}
```

In [93]:
```
# function
# function definition - code block
# ---------------------
# def functionname():
#         .....
#         .....
# function call - to invoke a definition
# ----------------
#   |__functionname()
#

L=[]
L.append("D1")
L.append(['D2','D3'])
# L.append("Dx","Dy") ->Error

def f1(a1,a2):
    print("Hello")

# f1() ->Error
# f1(10)->Error
# f1(10,20,30) ->Error
f1(10,20)
```

```
Hello
```

In [98]:
```
def f2(user='root',port=4566):
    print(user,port)
f2()
f2("userA")
f2("student",5904)
#f2("A",1334,44566) ->Error
```

```
root 4566
userA 4566
student 5904
```

In [109]:
```
def fx(a1,a2,f1,f2,f3,f4,f5,a3=0,a4=None):
    print(a1,a2) # required arguments
    print(a3,a4) # default values
```

In [121]:
```
d={}
d.setdefault("K1","V1")
d.setdefault("K2")
d
# d.setdefault() Error
#d.setdefault("K1","V1","V2")

# def setdefault(a1,a2=None):

L=[10,20,30,40,50,60]
L.pop() # Valid
#L.pop(index) # valid

# def pop(arg=-1):

def fx(*a1):
    print(type(a1))

fx()
fx(10,230,30,40,50,60)
# fx(user="root") # Error
```

```
<class 'tuple'>
<class 'tuple'>
```

In [122]:
```
def fx(**a1):
    print(type(a1))
    print(a1)

fx(user='root',port=5001,count=[100,200,300,400])
```

```
<class 'dict'>
{'user': 'root', 'port': 5001, 'count': [100, 200, 300, 400]}
```

In [126]:
```
def fx(*a1,**a2):
    print(a1,a2)

fx()
fx(123,32,233,232,2,3,23,2)
fx(user='root')
fx(213,123,12,12,12,123213,user='root',sh='/bin/bash')
```

```
() {}
(123, 32, 233, 232, 2, 3, 23, 2) {}
() {'user': 'root'}
(213, 123, 12, 12, 12, 123213) {'user': 'root', 'sh': '/bin/bash'}
```

In [ ]:
```
threading.Thread(target=thread_name,args=(...))
Listname.sort(reverse=True)
subprocess.check_output(shell=True)
```

In [130]:
```
def f1():
    global v1,v2,v3
    v1=100
    v2=200
    v3=300
    v4=400
    v5=500
    print(v1,v2,v3,v4,v5)
f1()
print(v1,v2,v3)

def f2():
    print(v1,v2,v3)
    print(v1+1000)
f2()
```

```
100 200 300 400 500
100 200 300
100 200 300
1100
```

In [132]:
```
def fx():
    var=100

fx() == None
```

Out[132]:
```
True
```

In [133]:
```
def f1():
    print("Test1")
    return 10 # Exitfrom definition
    print("Hello")
    print("Welcome")
f1()
```

```
Test1
```

Out[133]:
```
10
```

In [ ]:
```
# break (exit from loop)    return(exit from function)   exit() - exit from scrip
# we can use inside loop    we can use inside the        we can use exit()
# only                      function                     anywhere
```

In [138]:
```
def f1(a1,a2=0,*a3,**a4):
    print("Hello")

print(type(f1)) # procedure style

class Box:
    def f2(a1,a2=0,*a3,**a4):
        print("Hello")

obj=Box()
print(type(obj.f2)) # object oriented style
```

```
<class 'function'>
<class 'method'>
```

In [ ]:
```
len()
print()
input()
del()
type()

L.append()
s.strip()
s.split()
d.setdefault()
d.pop()
```