```
# Recap
---------
int float str bool

Validation - Testing -> use conditional statement        - Execution - only one time execution
(ex: Expression returns bool - use conditional statement)
                        |
                        |-> Yes / No - True /False

# set of code/statement - execute - more than one time //looping statements
```

```
print("Welcome")
print("Welcome")
print("Welcome")
print("Welcome")
print("Welcome")
#################### display Welcome message - 5times --> 50times
|
Write a code one time - execute - N no.of times
                                 ---//condition - 5times 50times 500times

i=0 .... 49  - Count - 50
1. initialization
2. condition - testing -->True - Code block will execute
                 |-->False - code block will not execute
i = 0
while(i < 50):
  print("Welcome")
  i = i + 1

################# Vs

for var in range(50): # starts from 0 to 49
  print("Welcome")
```

```
s="python"  # s | p | y | t | h | o | n |
            #   | 0 | 1 | 2 | 3 | 4 | 5 | <== index

print(s[0])
print(s[1])
print(s[2])
print(s[3])
print(s[4])
print(s[5])
```

```
p
y
t
h
o
n
```

```
for var in s:
  print(var)
```

```
p
y
t
h
o
n
```

```
len(s)
```

```
6
```

```
i = 0
while(i < len(s)):
  print(s[i]) # print(s[index])
  i = i + 1
```

```
p
y
t
h
o
n
```

```
s="GeneralCode"
i = 0
while(i < len(s)):
  print(s[i]) # print(s[index])
  i = i + 1
```

```
G
e
n
e
r
a
l
C
o
d
e
```

```
s="ab"
i = 0
while(i < len(s)):
  print(s[i]) # print(s[index])
  i = i + 1
```

```
a
b
```

```
# List - Collection of different types of value
#        Each value - own index
#        mutable - we can add/modify/delete
#        index / slicing
#         []
#       ListName = [List of items/values] - each value separated by ,
#
#       len(ListName) -> Total_Count // int
#
pid = 101
pname = 'pA'
pcost = 1000.34
pstatus = True

### Vs
product_infoA = [101,'pA',1000.34,True]

# (or)
product_infoB = [pid,pname,pcost,pstatus]
```

```
product_infoA = [101,'pA',1000.34,True]
print(type(product_infoA))
print(len(product_infoA))
```

```
<class 'list'>
4
```

```
for var in product_infoA:
  print(f"Hello...{var}")
```

```
Hello...101
Hello...pA
Hello...1000.34
Hello...True
```

```
print(type(True))
print(type('True'))
```

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

```
print(type([]))
```

```
<class 'list'>
```

```
# Recap string - str
s='Welcome'
print(s[0])
print(s[1])
print(s[2:]) # from 2nd index to list of ALL
print(s[-2:]) # last 2 items
```

```
W
e
```

```
lcome
me
```

```
L=[10,20,30,40,50.45,'A','B','data','Cap']
print(len(L))

print(L[0])
print(L[1])
print(L[2])
print(L[2:]) # from 2nd index to list of ALL
print(L[-2:]) # display last 2 items
print(L[-4:]) # last 4 items
```

```
9
10
20
30
[30, 40, 50.45, 'A', 'B', 'data', 'Cap']
['data', 'Cap']
['A', 'B', 'data', 'Cap']
```

```
## List is mutable - we can add new item to an existing list ; we can modify an existing item from list ; we can delete nth
#
# To add new data
#  Listname.append(Value) --> None

L=[10,20,30]
print(L,len(L)) # [10,20,30], 3

L.append(40)
L.append('Data1')
print(L,len(L))    # [10,20,30,40,'Data1'], 5
```

```
[10, 20, 30] 3
[10, 20, 30, 40, 'Data1'] 5
```

```
print(L)
```

```
[10, 20, 30, 40, 'Data1']
```

```
L=[10,20]
L.append(10)
L.append(10)
L.append(20)
## List can allow duplicate item
print(L)
```

```
[10, 20, 10, 10, 20]
```

```
# we can modify an existing item from list
#
# ListName[oldIndex] = updatedValue
#          --------

L=[10,20,30,40]   #  --> |  10  | 20  | 30 | 40 |
                  #         0   | 1   | 2  |  3 <== index
print(L[1])
```

```
20
```

```
L[1] = "Data-1" # modification
print(L)
```

```
[10, 'Data-1', 30, 40]
```

```
#  we can delete nth item
#  ListName.pop(index) -> removed_Value Vs  ListName.pop() --> removed_Last_index_Value

L=[10,20,30,40,50]
removed_item = L.pop(1) # remove 1st index value
print(removed_item) # 20
print("Remaining List:",L)
```

```
20
Remaining List: [10, 30, 40, 50]
```

```
L=[10,20,30,40,50]
L.pop() #  ListName.pop() --> removed_Last_index_Value - if not defined any index number - default removal is LastIndex Va
```

```
50
```

```
print(L)
```

```
[10, 20, 30, 40]
```

```
L.pop(6)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/tmp/ipython-input-849041889.py in <cell line: 0>()
----> 1 L.pop(6)

IndexError: pop index out of range
```

Next steps: Explain error

```
L=[10,20]
L.append(30)
L.append(40)
L
```

```
[10, 20, 30, 40]
```

```
# ListName.insert(index,Value) ->None   Vs  ListName.append(Value) ->None
L=['D1','D2','D3','D4','D5']
L.insert(1,'Data-3')
print(L)
```

```
['D1', 'Data-3', 'D2', 'D3', 'D4', 'D5']
```

```
# Write a python program

# Create an empty list  (ex: L = [] )
# use while loop - max limit is 5
#      - read a hostname from <STDIN>
#      - append to an existing list - add input hostname to an existing list
# use for loop
#   - display hostname line by line
# Display - Total no.of hostname
#

# for + range(5) //

L = []
print(f"Total no.of items:{len(L)}")

c=0
while(c < 5):
  host_name = input('Enter a hostname:')
  L.append(host_name)
  c = c + 1
print("\nDisplay - Each hostname ")
for var in L:
  print(f"Host name is:{var}")

print(f"\nTotal no.of items:{len(L)}")
```

```
Total no.of items:0
Enter a hostname:host01
Enter a hostname:host02
Enter a hostname:host03
Enter a hostname:host04
Enter a hostname:host05

Display - Each hostname
Host name is:host01
Host name is:host02
Host name is:host03
Host name is:host04
Host name is:host05

Total no.of items:5
```

```
L = []
print(f"Total no.of items:{len(L)}")

for var in range(5):
  host_name = input('Enter a hostname:')
  L.append(host_name)

print("\nDisplay - Each hostname ")
for var in L:
  print(f"Host name is:{var}")

print(f"\nTotal no.of items:{len(L)}")
```

```
Total no.of items:0
Enter a hostname:host01
Enter a hostname:host02
Enter a hostname:host03
Enter a hostname:host04
Enter a hostname:host05

Display - Each hostname
Host name is:host01
Host name is:host02
Host name is:host03
Host name is:host04
Host name is:host05

Total no.of items:5
```

```
# membership operator
# in
# not in

# inputs are collection - str,bytes,list,tuple,dict,set  ---> output: bool

# 'input_pattern_string'  in inputCollection -->bool

"a" in "raj,sales"
```

```
True
```

```
"sales" in "raj,sales,pune"
```

```
True
```

```
"sales" in ["prod","QA","sales","accounts"]
```

```
True
```

```
"devops" in ["prod","QA","sales","Accounts"]
```

```
False
```

```
"devops" not in ["prod","QA","sales","Accounts"]
```

```
True
```

```
"sales" not in ["prod","QA","sales","Accounts"]
```

```
False
```

```
product_info=[100, "pa", 111.43, True]
str(100) in product_info
```

```
False
```

```
100 in product_info
```

```
True
```

```
100 in 100
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-1729131660.py in <cell line: 0>()
----> 1 100 in 100

TypeError: argument of type 'int' is not iterable
```

Next steps: Explain error

```
100 in str(100)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-3120157947.py in <cell line: 0>()
----> 1 100 in str(100)

TypeError: 'in <string>' requires string as left operand, not int
```

Next steps: Explain error

```
str(100) in str(100)
```

```
True
```

```
for var in 100:
  print(var)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-2869196120.py in <cell line: 0>()
----> 1 for var in 100:
      2   print(var)

TypeError: 'int' object is not iterable
```

Next steps: Explain error

```
for var in str(100):
  print(var)

for var in [100]:
  print(var)
```

```
1
0
0
100
```

```
# List -  Collection of different types of items - [] - mutable - index based
#
# Tuple -  Collection of different types of items - () - immutable - index based
#                                                        ----------//We can't modify an existing structure - there is no ap
#
L = [10,1.34,'data',True]

T = (10,1.34,'data',True)
print(type(T))
print(len(T))
print(T[0])
print(T[-1])
print(T[2:]) # 2nd index to list of all
print(T[-3:]) # Last 3 items
```

```
<class 'tuple'>
4
10
True
('data', True)
(1.34, 'data', True)
```

```
10 in T
```

```
True
```

```
for var in T:
  print(var)
```

```
10
1.34
data
True
```

```
T=(10,20,30,40,50)
T[1]
```

```
20
```

```
T[1] = "Data"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-2169322643.py in <cell line: 0>()
----> 1 T[1] = "Data"

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

Next steps: Explain error

```
# tuple --- single row / record in table
# -----
#   |->functionCall args
#   |->return multiple values
#   |-> In Data Base ->fetch data from DB -> () - 1row
```

```
T=(10,20,30,40,50,60,70,80)
T[:2] # from 0th index to 2  (or) 1st 2 items
```

```
(10, 20)
```

```
T[:-2] # except last two items
```

```
(10, 20, 30, 40, 50, 60)
```

```
T[-2:] # last two items
```

```
(70, 80)
```

```
Conf_files = ("/etc/f1","/etc/f2","/etc/f3")

# remove /etc/f1
# add C:\\myconfig
# modify 1st index -> /etc/f2 -> D:\\test.cfg
# ----------------------------------------------

# typecast to list  ==> list(input_tuple) ->output_list  => initialize to new variable
#

list(Conf_files)
```

```
['/etc/f1', '/etc/f2', '/etc/f3']
```

```
tv = list(Conf_files)
tv
```

```
['/etc/f1', '/etc/f2', '/etc/f3']
```

```
tv.pop(0)
```

```
'/etc/f1'
```

```
tv
```

```
['/etc/f2', '/etc/f3']
```

```
tv[0] = "D:\\test.cfg"
tv.append("C:\\myconfig")
```

```
tv
```

```
['D:\\test.cfg', '/etc/f3', 'C:\\myconfig']
```

```
tuple(tv)
```

```
('D:\\test.cfg', '/etc/f3', 'C:\\myconfig')
```

```
Config_file = tuple(tv) # convert to tuple
```

```
Config_file
```

```
('D:\\test.cfg', '/etc/f3', 'C:\\myconfig')
```

name() <-- functionCall Vs name=() <--- tuple

fetching_single data from collection ==> name[index]

Vs

Listname=[] <== Empty list

```
## Python Support Multiple initialization

##  variable = value

# variable1,variable2,variable3 = value1,value2,value3
#  -------  ========= __________  ------ ====== ________

va,vb,vc = 10,1.34,'data'

print("va=",va)
print("vb=",vb)
print("vc=",vc)
```

```
va= 10
vb= 1.34
vc= data
```

```
va,vb,vc = 10,1.34,'data',20
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipython-input-1126998947.py in <cell line: 0>()
----> 1 va,vb,vc = 10,1.34,'data',20

ValueError: too many values to unpack (expected 3)
```

Next steps: Explain error

```
v1,v2,v3 = 10,20
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipython-input-1026082288.py in <cell line: 0>()
----> 1 v1,v2,v3 = 10,20

ValueError: not enough values to unpack (expected 3, got 2)
```

Next steps: Explain error

```
L=['Data1','Data2']

va = L[0]   # 0th index
vb = L[1]   # 1st index

## Vs
vA,vB = L # multiple initialization

print('vA=',vA,'vB=',vB)
```

```
vA= Data1 vB= Data2
```

```
T=('Data1','Data2')

va = T[0]   # 0th index
vb = T[1]   # 1st index

## Vs
vA,vB = T # multiple initialization

print('vA=',vA,'vB=',vB)
```

```
vA= Data1 vB= Data2
```

```
v1 = 10
v2 = 1.34
v3 = 'data'
v4 = True
print(type(v1),type(v2),type(v3),type(v4))
```

```
<class 'int'> <class 'float'> <class 'str'> <class 'bool'>
```

```
L = [v1,v2,v3,v4]
print(type(L))
```

```
<class 'list'>
```

```
print(type(L[0]),type(L[1]),type(L[2]),type(L[-1]))
```

```
<class 'int'> <class 'float'> <class 'str'> <class 'bool'>
```

```
Emp = ['Raj','E123','Sales','Pune','1000.34Rs']

ename = Emp[0]
eid = Emp[1]
edept = Emp[2]
ecity = Emp[3]
ecost = Emp[4]

#### Vs

en,ei,ed,ec,ecost = Emp   #### Single line - we can initialize from multiple values to individual variable
```

```
print(f'''Emp name is:{en}
      Emp ID:{ei}
      Emp working dept:{ed}
      Emp Working City:{ec}
      Emp Cost is:{ecost}''')
```

```
Emp name is:Raj
      Emp ID:E123
      Emp working dept:Sales
      Emp Working City:Pune
      Emp Cost is:1000.34Rs
```

```
s="101,raj,sales,pune,1000"
print(type(s),len(s))

# Convert input_string into output_list  - based on specialchars
#               ======             =====
#
# input_string.split(sep) -->output_list
#
```

```
<class 'str'> 23
```

```
s.split(",")
```

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

```
Log_data = "10th Feb:data1:LB value-1.34:devServer"
Log_data.split(":")
```

```
['10th Feb', 'data1', 'LB value-1.34', 'devServer']
```

```
s="101,raj,sales,pune,1000"
s.split() # default separator is space
```

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

```
Log_data = "10th Feb:data1:LB value-1.34:devServer"
Log_data.split() # default separator is space
```

```
['10th', 'Feb:data1:LB', 'value-1.34:devServer']
```

```
s="101,raj,sales,pune,1000"
L = s.split(",")
print(type(L),len(L))
print(L)
```

```
<class 'list'> 5
['101', 'raj', 'sales', 'pune', '1000']
```

```
## from list ->str_output

# "".join(input_List) -->output_str

"".join(L)
```

```
'101rajsalespune1000'
```

```
",".join(L)
```

```
'101,raj,sales,pune,1000'
```

```
print(",".join(L))
```

```
101,raj,sales,pune,1000
```

```
###
# int float  bool
# str list tuple  ---- index based
```

string (str) -- Collection of chars - index based [index] - ''(or)"" immutable

list - Collection of different types of items - index based [index] - mutable - []

tuple - Collection of different types of items - index based [index] - immutable - ()

dict - Collection of different types of items - key:value // data NOT index based Key based - operation

```
dict also mutable - {}
```

```
# dictName = {}  <== Empty dictionary

# dictName = {'Key1':Value1,'Key2':Value2,'Key3':Value3,....'KeyN':'ValueN'}
#             --------------------------------------------------------------
#
#       Key   |    Value
#      -------|------------
#      Key1   | Value1
#     --------|------------
#      Key2   | Value2
#   ----------|-------------
#      Key3   |  Value3
#  ------------------------------

# dict key -> unique ; dict value can duplicate
# ----------
```

```
product_records = {'pid':101,'pname':'pA','pcost':1000}
print(type(product_records))
print(len(product_records))
```

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

```
## How to fetch/get single item from dictionary
#
#  dictName['inputKey'] -->Value / KeyError

print(product_records['pid'])
```

```
101
```

```
print(product_records['pname'])
```

```
pA
```

```
print(product_records['Pname'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/tmp/ipython-input-3313335146.py in <cell line: 0>()
----> 1 print(product_records['Pname'])

KeyError: 'Pname'
```

Next steps: Explain error

```
## How to fetch/get single value from given dictionary ?
##  dictionary_Name[inputKey]

## How to modify an existing dict value ?         How to modify an existing list
##  dictionary_Name[oldKey] = updatedValue     Vs  ListName[oldIndex] = Value

product_records['pname'] = 'Box1'
print(product_records)
```

```
{'pid': 101, 'pname': 'Box1', 'pcost': 1000}
```

```
## To add new data to an existing dict
#
#  dictName['newKey'] = value
#            --------    -----

d={} # empty dict
d['K1'] = 'V1'     # adding new data
d['K2'] = 100      # adding new data
d['K3'] = 13.45    # adding new data
print(d)
##
d['K2'] = 'Data-2' # modifing an existing data
print(d)
```

```
{'K1': 'V1', 'K2': 100, 'K3': 13.45}
{'K1': 'V1', 'K2': 'Data-2', 'K3': 13.45}
```

```
# To delete an exiting item from given dict
#  dictName.pop(inputKey) ->removedValue
#                  |->Not Exists ->KeyError

d.pop('K3')
```

```
13.45
```

```
print(d)
```

```
{'K1': 'V1', 'K2': 'Data-2'}
```

```
d.pop('K3')
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/tmp/ipython-input-1010031541.py in <cell line: 0>()
----> 1 d.pop('K3')

KeyError: 'K3'
```

Next steps: Explain error

```
L=[10,20,30,40,50]
for var in L:
  print(var)
```

```
10
20
30
40
50
```

```
d={'K1':'V1','K2':'V2','K3':'V3'}
for var in d:
  print(var) # will get list of keys
```

```
K1
K2
K3
```

```
# d['K1'] ->V1
# d['K2'] ->V2
# d['K3'] ->V3
# d[var] ->value
for var in d:
  print(d[var])
```

```
V1
V2
V3
```

```
# To get list of key - value
for var in d:
  print(f'{var} - {d[var]}')
```

```
K1 - V1
K2 - V2
K3 - V3
```

```
## membership operators
#
# To test - key is existing or not
# ============
# input_key in input_dict ->bool

'K1' in d
```

```
True
```

```
'Kx' in d
```

```
False
```

```
if('K1' in d):
  print(d['K1'])
else:
  print('Sorry given Key is not exists')
```

```
V1
```

```
'''
Write a python program:
Create an empty dictionary (ex: hosts={})
|->use for + range - limit is 5
   -> read hostname from <STDIN>
   -> read host IP Address from <STDIN>
   -> Add input hostname and IP to an existing dict
   -> hostname - key ; IP - Value

|->use for loop - display Key - Value
|->use len() - display total no.of host details
'''
```

```
hosts={} # empty dict
for var in range(5):
  K = input('Enter a hostname:')
  V = input(f'Enter {K} IP Address:')
  hosts[K] = V # dict - adding new data to an existing dict

######
print('\n Hostname and IP Details:-')
for var in hosts:
  print(f'{var} - {hosts[var]}')

print(f'\n---\nTotal No.of hosts:{len(hosts)}')
```

```
Enter a hostname:host01
Enter host01 IP Address:10.20.30.40
Enter a hostname:host02
Enter host02 IP Address:11.23.34.33
Enter a hostname:host03
Enter host03 IP Address:10.33.21.33
Enter a hostname:host04
Enter host04 IP Address:24.43.34.12
Enter a hostname:host05
Enter host05 IP Address:10.33.33.12

 Hostname and IP Details:-
host01 - 10.20.30.40
host02 - 11.23.34.33
host03 - 10.33.21.33
host04 - 24.43.34.12
host05 - 10.33.33.12

---
Total No.of hosts:5
```

```
'''
S1="Interface=eth0"
S2="Type=Ethernet"
S3="Onboot=Yes"
S4="IP=10.20.30.40"
|
split each string into multiple values (list) - based on =
|
Create a dict
 - add above details to dict
 - Ex: {"Interface":"eth0"}

|->display dict

{"Interface":"eth0","Type":"Ethernet","Onboot":"Yes",
  "IP":"10.20.30.40"} //Expected result
  |
  |-->dict operation
      - display Key - value
      - update IP -> 127.0.0.1
      - add new data(ex: Prefix=24)
      - display updated dict
      '''
```

```
S1="Interface=eth0"
S1.split("=")
```

```
['Interface', 'eth0']
```

```
L = S1.split("=")
```

```
d={}
d[L[0]] = L[1]
d
```

```
{'Interface': 'eth0'}
```

```
S1="Interface=eth0"
S2="Type=Ethernet"
S3="Onboot=Yes"
S4="IP=10.20.30.40"

d = {}
L = S1.split("=")
d[L[0]] = L[1]

L = S2.split("=")
d[L[0]] = L[1]

L = S3.split("=")
d[L[0]] = L[1]

L = S4.split("=")
d[L[0]] = L[1]

d
```

```
{'Interface': 'eth0', 'Type': 'Ethernet', 'Onboot': 'Yes', 'IP': '10.20.30.40'}
```

```
d={}
for var in ["Interface=eth0","Type=Ethernet","Onboot=Yes","IP=10.20.30.40"]:
  K,V = var.split("=")
  d[K] = V
d
```

```
{'Interface': 'eth0', 'Type': 'Ethernet', 'Onboot': 'Yes', 'IP': '10.20.30.40'}
```

```
Key = input('Enter a Key:')
if Key in d:
  d[Key] = '127.0.0.1'
else:
  print(f'Sorry Key:{Key} is not found')

print("\nUpdated dict:-")
d
```

```
Enter a Key:IP
```

```
Updated dict:-
{'Interface': 'eth0', 'Type': 'Ethernet', 'Onboot': 'Yes', 'IP': '127.0.0.1'}
```

```python
Key = input('Enter a Key:')
if Key in d:
  d[Key] = '127.0.0.1'
else:
  print(f'Sorry Key:{Key} is not found')

print("\nUpdated dict:-")
d
```

```
Enter a Key:DNS
Sorry Key:DNS is not found

Updated dict:-
{'Interface': 'eth0', 'Type': 'Ethernet', 'Onboot': 'Yes', 'IP': '127.0.0.1'}
```

```python
'''
Given Emp List

EMP = ['101,raj,sales,pune,1000',
'102,leo,prod,bglore,2000',
'103,anu,hr,mumbai,3000',
'104,ram,sales,'bglore,4000']

# Iterate a EMP list
#   -> split each line into multiple values based ,
#   -> display Emp name - Title Case  Emp working dept in upperCase
#   -> Calculate Sum of Emp's cost
#   -> At the end - display Total Emp's Cost

# Expected Result:-
# ------------

Raj    SALES
Leo    PROD
Anu     HR
RAM    SALES
----------------------------
Emp's Total Cost: 10000
-----------------------------
```

```python
EMP = ['101,raj,sales,pune,1000',
'102,leo,prod,bglore,2000',
'103,anu,hr,mumbai,3000',
'104,ram,sales,bglore,4000',
'105,shiv,sales,pune,5000']

total = 0
for var in EMP:
  eid,ename,edept,eplace,ecost = var.split(",")
  print(f'{ename.title()} \t {edept.upper()}')
  total = total + int(ecost)

print(f"\nSum of Emp's Cost is:{total}")
```

```
Raj      SALES
Leo      PROD
Anu      HR
Ram      SALES
Shiv     SALES

Sum of Emp's Cost is:15000
```

```python
EMP = ['101,raj,sales,pune,1000',
'102,leo,prod,bglore,2000',
'103,anu,hr,mumbai,3000',
'104,ram,sales,bglore,4000',
'105,shiv,sales,pune,5000']

total = 0
for var in EMP:
  eid,ename,edept,eplace,ecost = var.split(",")
  if 'sales' in edept:
    print(f'{ename.title()} \t {edept.upper()}')
    total = total + int(ecost)

print(f"\nSum of Emp's Cost is:{total}")
```

```
Raj      SALES
Ram      SALES
Shiv     SALES

Sum of Emp's Cost is:10000
```

```
Write a python program:
create an three empty list (sales,prod,devops,others = [],[],[],[])
- use for loop + range - 5
- read an emp details from <STDIN> (ex: ram,sales,bglore,1000)
- test
   - emp is sales - append to sales list
   - emp is prod  - append to prod list
   - emp is devops - append to devops list
   |
   - not above 3 depts - append to others list

- display each list one by one
```

```
sales,prod,devops,others = [],[],[],[] # multiple empty list - multiple initialization

for var in range(5):
  emp_details = input('Enter emp details:')
  if 'sales' in emp_details:
    sales.append(emp_details)
  elif 'prod' in emp_details:
    prod.append(emp_details)
  elif 'devops' in emp_details:
    devops.append(emp_details)
  else:
    others.append(emp_details)
```

```
Enter emp details:raj,sales,pune,1000
Enter emp details:leo,prod,bglore,2000
Enter emp details:anu,sales,hyderabad,3000
Enter emp details:shiv,admin,bglore,3411
Enter emp details:paul,devops,chennai,3441
```

```
print(sales)
print(prod)
print(devops)
print(others)
```

```
['raj,sales,pune,1000', 'anu,sales,hyderabad,3000']
['leo,prod,bglore,2000']
['paul,devops,chennai,3441']
['shiv,admin,bglore,3411']
```

```
## dict methods
d={'K1':'V1','K2':'V2','K3':'V3'}

# to fetch/get single data from given input dict
# dictName[InputKey] ->Value /KeyError  (or) dictName.get(InputKey) -->Value /None

print(d['K1'])
# print(d['Kx']) -> KeyError: 'Kx'

print(d.get('K1'))
print(d.get('Kx'))
```

```
V1
V1
None
```

```
d={'K1':'V1','K2':'V2','K3':'V3'}
```

```
##
d={'K1':'V1','K2':'V2','K3':'V3'}

# To add new data to an existing dict
# dictName['newKey'] = Value  (or) dictName.setdefault('NewKey',Value)
d['K4'] = 'V4'
```

```
#
# (or)
d.setdefault("K5",'V5')
print(d)
```

```
{'K1': 'V1', 'K2': 'V2', 'K3': 'V3', 'K4': 'V4', 'K5': 'V5'}
```

```
d={'K1':'V1','K2':'V2','K3':'V3'}
# To get list of keys

for var in d:
  print(var)

print("\n")

# another way:  dictName.keys()

for var in d.keys():
  print(var)
```

```
K1
K2
K3


K1
K2
K3
```

```
d={'K1':'V1','K2':'V2','K3':'V3'}
print(d.keys())
print(d.values())
print(d.items())
```

```
dict_keys(['K1', 'K2', 'K3'])
dict_values(['V1', 'V2', 'V3'])
dict_items([('K1', 'V1'), ('K2', 'V2'), ('K3', 'V3')])
```

string (str) -- Collection of chars - index based [index] - ''(or)"" immutable

list - Collection of different types of items - index based [index] - mutable - []

tuple - Collection of different types of items - index based [index] - immutable - ()

dict - Collection of different types of items - key:value // data NOT index based Key based - operation

set - Collection of different types of items

|--> NOT Index based |--> NOT Key:Value based |--> Hold - unique data items - not allows duplicate item |--> we can add new data to set we can remove an existing data from set -- ->There is No modification

```
s = {10,20,30,40,10,10,10,'D1','D1','D1','D2','D1'}
print(type(s),len(s))
```

```
<class 'set'> 6
```

```
print(s)
```

```
{'D2', 'D1', 20, 40, 10, 30}
```

```
s[0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-243613605.py in <cell line: 0>()
----> 1 s[0]

TypeError: 'set' object is not subscriptable
```

Next steps: Explain error

```
s={10,20,30}
s
```

```
{10, 20, 30}
```

```
s.add(40) # adding new data
s.add(10)
s.add(20)
s.add(30)
s
```

```
{10, 20, 30, 40}
```

```
s.remove(10)
```

```
s
```

```
{20, 30, 40}
```

```
L=['D1','D2','D3','D4','D3','D4','D2']
len(L)
```

```
7
```

```
set(L) # typecast to set - omit duplicate values
```

```
{'D1', 'D2', 'D3', 'D4'}
```

```
s = set(L)
list(s) # typecast to list
```

```
['D2', 'D3', 'D1', 'D4']
```

```
## set mathematical operation
##
## union,intersection,difference,symmetric_difference
##  |-> operators (or) |->set methods

s1={10,20,30,40}
s2={10,20,50}
s1.union(s2)
```

```
{10, 20, 30, 40, 50}
```

```
F1_data = ['sales','prod','sales','QA','prod'] # file contents
F2_data = ['QA','Devops','HR'] # file contents

set(F1_data) | set(F2_data) # union
```

```
{'Devops', 'HR', 'QA', 'prod', 'sales'}
```

```
set(F1_data) & set(F2_data) # intersection
```

```
{'QA'}
```

```
set(F1_data) - set(F2_data) # A-B # display F1_data unique contents
```

```
{'prod', 'sales'}
```

```
set(F2_data)- set(F1_data) # B-A - display F2_data unique contents
```

```
{'Devops', 'HR'}
```

```
set(F1_data) ^ set(F2_data)
```

```
{'Devops', 'HR', 'prod', 'sales'}
```

```
list tuple dict
--------------------// We can create multidimensional structure
L = [10,20,30] # 1D
T = (10,20,30) # 1D

d ={'K1':10,'K2':20} # 1D  1Key ->1Value

List of list  -> [[],[],[]]

List of tuple -> [(),(),()]

List of dict  --> [{},{},{}]
---------------------------------------------
Tuple of list --> ([],[],[])
Tuple of tuple ---> ((),(),())

Tuple of dict ---> ({},{},{})
--------------------------------------------
dict of list --->{'K1':[],'K2':[]}

dict of tuple -->{'K1':(),'K2':()}

dict of dict --->{'K1':{},'K2':{}}
-----------------------------------------------
```

```
DBs = ['oracle','sqlite3','MySQL']

Servers=['Unix','Linux','minix']

L = ['D1',DBs,Servers]
print(type(L),len(L))
```

```
<class 'list'> 3
```

```
print(L)
```

```
['D1', ['oracle', 'sqlite3', 'MySQL'], ['Unix', 'Linux', 'minix']]
```

```
L= [['D1','D2','D3'],('D4','D5'),{'K1':'V1','K2':'V2'}]
print(type(L),len(L))
print(L[0])
print(L[1])
print(L[2])
```

```
<class 'list'> 3
['D1', 'D2', 'D3']
('D4', 'D5')
{'K1': 'V1', 'K2': 'V2'}
```

```
print(L)
print(L[0],type(L[0]))
print(L[0][0])
print(L[0][1])
```

```
[['D1', 'D2', 'D3'], ('D4', 'D5'), {'K1': 'V1', 'K2': 'V2'}]
['D1', 'D2', 'D3'] <class 'list'>
D1
D2
```

```
print(L[1])
print(L[1][1])
```

```
('D4', 'D5')
D5
```

```
L[1]='Data' # OK - We can modify - L is a list
# L[1][0] = "Data" # ->Error  #  L ->list but  L[1] is tuple
```

```
L
```

```
[['D1', 'D2', 'D3'], 'Data', {'K1': 'V1', 'K2': 'V2'}]
```

```
L[-1]
```

```
{'K1': 'V1', 'K2': 'V2'}
```

```
L[-1]['K1']
```

```
'V1'
```

```
# Tuple of list / tuple /dict

T=([],[]) # tuple of list
print(len(T))
```

```
2
```

```
print(type(T),type(T[0]))
```

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

```
T[0].append('D1')
T[0].append('D2')
T[0].append('D3')
T[0].append('D1')
T[0].append('D2')
T[0].append('D3')
```

```
print(len(T))
T
```

```
2
(['D1', 'D2', 'D3', 'D1', 'D2', 'D3'], [])
```

```
# dict of list

emp={'eid':[],'ename':[],'edept':[]} # 1Key ->MultipleValues

# dictName={'Key':Value}

print(type(emp))
print(type(emp['eid']))
emp['eid'].append(101)
emp['eid'].append(102)
emp
```

```
<class 'dict'>
<class 'list'>
{'eid': [101, 102], 'ename': [], 'edept': []}
```

```
emp['ename'].append('Arun')
emp['ename'].append('Vijay')
emp['ename'].append('Leo')
```

```
emp
```

```
{'eid': [101, 102], 'ename': ['Arun', 'Vijay', 'Leo'], 'edept': []}
```

```
d={'K1':[10,20,30,40],
   'K2':(100,200,300,400),
   'K3':{'K1':'F1','K2':'F2','K3':'F3'}
   }
```

```
# d ->dict
# d['K1'] ->list - we can do list operation
# d['K2'] ->tuple - we can do tuple operation - immutable
# d['K3'] ->dict  - we can do dict operation

print(d['K3'])
```

```
{'K1': 'F1', 'K2': 'F2', 'K3': 'F3'}
```

```
print(d['K3']['K1']) ## dict of dict
```

```
F1
```

```
print(d['K3']['K2'])
```

```
F2
```

```
print(d['K2'][0])
```

```
100
```

```
import pprint
pprint.pprint(d) #
```

```
{'K1': [10, 20, 30, 40],
 'K2': (100, 200, 300, 400),
 'K3': {'K1': 'F1', 'K2': 'F2', 'K3': 'F3'}}
```

```
net_info = {}
net_info['Types'] = ['eth0','eth1','eth2']
net_info['devices']=[{'D1':'device1','UUID':1234},{'D1':'device2','UUID':34455},{'D1':'device3','UUID':34144}]
net_info['IPs']=['10.20.30.40','10.23.34.44','10.22.43.44']
print(net_info)
```

```
{'Types': ['eth0', 'eth1', 'eth2'], 'devices': [{'D1': 'device1', 'UUID': 1234}, {'D1': 'device2', 'UUID': 34455}, {'D1': 'c
```

```
pprint.pprint(net_info)
```

```
{'IPs': ['10.20.30.40', '10.23.34.44', '10.22.43.44'],
 'Types': ['eth0', 'eth1', 'eth2'],
 'devices': [{'D1': 'device1', 'UUID': 1234},
             {'D1': 'device2', 'UUID': 34455},
             {'D1': 'device3', 'UUID': 34144}]}
```

```
pprint.pprint(net_info['devices'])
```

```
[{'D1': 'device1', 'UUID': 1234},
 {'D1': 'device2', 'UUID': 34455},
 {'D1': 'device3', 'UUID': 34144}]
```

```
net_info['devices'][1]
```

```
{'D1': 'device2', 'UUID': 34455}
```

```
net_info['devices'][1]['UUID'] ## dict of list -> list of dict
```

```
34455
```

```
depts = {}
sales,prod,devops,others = [],[],[],[] # multiple empty list - multiple initialization

for var in range(5):
  emp_details = input('Enter emp details:')
  if 'sales' in emp_details:
    sales.append(emp_details)
  elif 'prod' in emp_details:
    prod.append(emp_details)
  elif 'devops' in emp_details:
    devops.append(emp_details)
  else:
    others.append(emp_details)

depts['S1'] = sales
depts['P1'] = prod
depts['D1'] = devops
depts['O1'] = others
pprint.pprint(depts)
```

```
Enter emp details:ram,sales,bglore
Enter emp details:raj,prod,pune
Enter emp details:paul,admin,chennai
Enter emp details:arun,devops,pune
Enter emp details:zion,sales,mumbai
{'D1': ['arun,devops,pune'],
 'O1': ['paul,admin,chennai'],
 'P1': ['raj,prod,pune'],
 'S1': ['ram,sales,bglore', 'zion,sales,mumbai']}
```

```
depts['S1']
```

```
['ram,sales,bglore', 'zion,sales,mumbai']
```

```
depts['O1']
```

```
['paul,admin,chennai']
```

```
depts['S1']
```

```
['ram,sales,bglore', 'zion,sales,mumbai']
```

```
depts['S1'][0] # dict of list
```

```
'ram,sales,bglore'
```

```
[(row1),(row2),(row3)...(rowN)]  # list of tuple
# 0   |   1   |  2   ..   N <== index


# list and dict
# ---------------//combination list and dict - use more JSON
```

```
##################  End of Day2 session #################
```