In [2]:
```
# procedure style -- (1)
# functional style -- (2)
# object  oriented design style -- (3)

# in python - class <--->object
# ---------
# types - int float str bool bytes list tuple dict set //class
#  |
# values   10  10.5  'a' True //object

a=5 # a | 5 | 0x01224
    # |  |__Value - 5 is belongs to int (type)
    # |_variable

b=int(5) # oops style - int class -> int() //constructor
print(a)
print(b)
```

```
5
5
```

In [3]:
```
a=5
a="Hello"
print(a)
```

```
Hello
```

In [ ]:
```
# New to python
# -----------------
#  step 1:  understand the topic definition( what is _______?)
#  step 2:  understand the syntax (Rules)
#  step 3:  Examples
#  step 4:  Exercise (activity)
```

In [4]:
```
# Variable
# -----------
# what is variable ?  - namesapce - it's holding the value
# Sytnax:-
# ---------
# variablename = value
# ------------
#  |__ startswith A-Za-z_ (not starts with digits) ; not allowed space and specia
#
name="root"    # name  | root | 0x2344
Dept = 'DBA'
IP = '10.20.30.40'
count=56       # count | 56 |0x5455
F=1.45
status=True

print(name)
print(Dept)
print(IP)
print(count)
print(F)
print(status)
```

```
root
DBA
10.20.30.40
56
1.45
True
```

In [9]:
```
# type() - determine python type
# type(variable/Value)

print(type(10))
a=56
print(type(a))
print(type(True))
print(type(''))
```

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

In [11]:
```
print(len('a5^ H'),type('a5^ H'))
```

```
5 <class 'str'>
```

In [12]:
```
print("data1\ndata2\ndata3\ndata4")
print("""DATA1
DATA2
DATA3
DATA4
""")
```

```
data1
data2
data3
data4
DATA1
DATA2
DATA3
DATA4
```

In [16]:
```
v1=134
v2=45.56
v3='data'

print("v1 value is:",v1,"\nV2 value:",v2,"\nV3 value:",v3) ##(1)
print("") # empty line
print("v1 value is:%d\nv2 value:%0.3f\nv3 value:%s"%(v1,v2,v3)) ##(2)
print("") # empty line
print("v1 value is:{}\nv2 value:{}\n v3 value:{}".format(v1,v2,v3)) ##(3)
```

```
v1 value is: 134
V2 value: 45.56
V3 value: data

v1 value is:134
v2 value:45.560
v3 value:data

v1 value is:134
v2 value:45.56
 v3 value:data
```

In [ ]:
```
# Task1
# --------
# declare and intialize value about filesystem details
# partition (ex: /dev/sda1)
# filesystem type (ex: xfs)
# mount point  (ex: /D1)
# psize (ex:456GB)

# use print() - display file system details line by line
```

In [22]:
```
partition='/dev/sda1'
fstype='xfs'
fmount='/D1'
fsize=120
print("""Partition name:{}
----------------------------
File System Type:{}
----------------------------
File Mount Point:{}
----------------------------
File Size:{}GB
----------------------------
""".format(partition,fstype,fmount,fsize))
```

```
Partition name:/dev/sda1
----------------------------
File System Type:xfs
----------------------------
File Mount Point:/D1
----------------------------
File Size:120GB
----------------------------
```

In [21]:
```
print(type(''),type(""))
```

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

In [30]:
```
# typecasting
# ------------
# int float str bytes bool NoneType <== Scalar (Single data)
#                         |    |__ None
#                         |__True/False

n=60 # int
print(float(n))
str(n)
bool(n)
```

```
60.0
```

Out[30]:
```
True
```

In [39]:
```
s='abc' # string
v=b'abc' # bytes # python 3.x version

print(type(s),type(v))
type(b'')
```

```
<class 'str'> <class 'bytes'>
```

Out[39]:
```
bytes
```

In [33]:
```
print(bool(0),type(0))
print(bool(0.0),type(0.0))
print(bool(''),type(''))
print(bool(b''),type(b''))
print(bool([]),type([]))
print(bool(()),type(()))
print(bool({}),type({}))
print(bool(None),type(None))
```

```
False <class 'int'>
False <class 'float'>
False <class 'str'>
False <class 'bytes'>
False <class 'list'>
False <class 'tuple'>
False <class 'dict'>
False <class 'NoneType'>
```

In [36]:
```
import re
re.search("bash","root:x:bin:bash:0")
```

Out[36]:
```
<re.Match object; span=(11, 15), match='bash'>
```

In [37]:
```
bool(re.search("bash","root:x:bin:bash:0"))
```

Out[37]:
```
True
```

In [44]:
```
if(re.search("bash","root:x:bin:bash:0")):
    print("True section")
    print("bash is matched")
else:
    print("Not-MAtched")
```

```
True section
bash is matched
```

In [43]:
```
bool(re.search("bash","root:x:bin:ksh:0"))
```

Out[43]:
```
False
```

In [45]:
```
# in not in - membership operators (str,bytes,list,tuple,dict,set)

"bash" in "root:x:bin:bash" Vs  re.search("bash","root:x:bin:bash")
                                |__ Regx ^ $ ^bash$ + {}
```

Out[45]:
```
True
```

In [53]:
```
n=56
print(float(n))
print(type(n),n)
n=float(n)
print(type(n),n)
s=str(n)
print(s,type(s))
```

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

In [56]:
```
fsize=150
print("File Size is:",fsize,"GB")
print("File Size is:"+str(fsize)+"GB")
```

```
File Size is: 150 GB
File Size is:150GB
```

In [62]:
```
cost="4567.56"
float(cost)*0.545
```

Out[62]:
```
2489.3202000000006
```

In [59]:
```
s1="456"
i=int(s1)
f=float(s1)
print(i,f)
```

```
456 456.0
```

In [67]:
```
s2="abc"
#int(s2) # Error
s2.isdigit()
"12345".isdigit()

s1="456F"
if s1.isdigit():
    r=int(s1)+1345
    print(r)
else:
    print("Non-digits")
```

```
Non-digits
```

In [72]:
```
s1="560"
s2="545"
s3="235"
str(int(s1)+int(s2)+int(s3))+"Rs"
   #------- ======== -----
   # 1340
   #  |
   # '1340'+"Rs" ->'str'
```

Out[72]:
```
'1340Rs'
```

In [73]:
```
10+20*5+20
```

Out[73]:
```
130
```

In [74]:
```
10+20*(5+20)
```

Out[74]:
```
510
```

In [ ]:
```
input() ------------ PythonScript-------------------print()
Keyboard                                          Monitor (STDOUT/STDERR)
<STDIN>

Syntax:-
==========
variable=input("PromptMessage") # in bash script  read -p "promptMessage" var
|              |---STDOUT----|
<STDIN>
```

In [75]:
```
fstype=input("Enter a filesystem name:")
print("input file system name is:{}".format(fstype))
```

```
Enter a filesystem name:xfs
input file system name is:xfs
```

In [76]:
```
fstype=input("Enter a filesystem name:")
print("input file system name is:{}".format(fstype))
```

```
Enter a filesystem name:ext4
input file system name is:ext4
```

In [78]:
```
n=input("Enter N value:")
print(type(n),n)
n
```

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

Out[78]:
```
'56'
```

In [79]:
```
n=input("Enter N value:")
n=int(n)
print(n+100)
```

```
Enter N value:56
156
```

In [80]:
```
n=int(input("Enter N value:"))
print(n+100)
```

```
Enter N value:56
156
```

In [ ]:
```
# operators
# ---------------
# + - * / // % ** (int,float)  -> int/float
# + *             (int,str) --> str
# == != < <= > >= (int,float,str) -> bool(True/False)
# and or not      (int,float,str) -> bool(True/False)
# in not in -->(str,list,tuple,dict,set) -> bool(True/False)
```

In [95]:
```
10>50.45
"root" == "root"
0.45 >0.23
```

Out[95]:
```
True
```

In [97]:
```
user="root"
sh="bash"
user == "root" and sh == "bash" or sh == "ksh"
```

Out[97]:
```
True
```

In [88]:
```
'python---programming'*5
```

Out[88]:
```
'python---programmingpython---programmingpython---programmingpython---programmingpython---programming'
```

In [98]:
```
# <SearchPattern> in <inputCollection> ->True/False

":" in "root:x"
```

Out[98]:
```
True
```

In [99]:
```
"p1.log" in ['p1.log','p2.log']
```

Out[99]:
```
True
```

In [100]:
```
":" not in "root:x"
```

Out[100]:
```
False
```

In [101]:
```
"," not in "root:x"
```

Out[101]:
```
True
```

In [102]:
```
# in python any expression (or) method ->retruns bool ->Conditional statements
# if/else - code block will execute only one time.
if(True):
    print("ONE")
    print("TWO")
    if(True):
        print("THREE")
        if(False):
            print("FIVE")
        else:
            print("SIX")
```

```
ONE
TWO
THREE
SIX
```

In [105]:
```
if '':
    print("Yes")
else:
    print("No")
```

```
No
```

In [107]:
```
name=input("Enter your name:")
if(name): # if(len(name)!=0)
    print("Hello..{}".format(name))
else:
    print("Sorry you not entered input")
```

```
Enter your name:
Sorry you not entered input
```

In [ ]:
```
# Multiconditional statements - more than one test
# if..elif

if(condition1):
    Successblock1
elif(condition2):
    Successblock2
elif(condition3):
    Successblock3
..
elif(conditionN):
    conditionN
else:
    False block
```

In [ ]:
```
Q1. Write a python program
'''
read a disk partition name from <STDIN>
read a disk partition Size from <STDIN>
|_ another disk name and disk Size
Calculate Sum of disk Size
Display - each partition name and partition Size
display Total Partition Size
Enter a partition name: /dev/sda1
Enter a /dev/sda1 partition Size:120
Enter a partition name: /dev/sda2
Enter a /dev/sda2 partition Size:200

Partition Name: /dev/sda1     Size:120
Partition Name: /dev/sda2     Size:200
-------------------------------------------------
         Total                    :320 GB
-------------------------------------------------

Q2. Write a python program:
    read a user name from <STDIN>
    test input user name is "root" or not
                               |       |--->exit from code block
                          read a shell name from <STDIN>
            test shell name is bash - initialize profile file name is bashrc
            test shell name is ksh  - initialize profile file name is kshrc
            test shell name is psh  - initialize profile file name is PSprofile
            |
            default shell name is /bin/nologin    profile file name /etc/profile

Display user name,shell name, profile filename
'''
```

In [109]:
```
p1=input("Enter a partition name:")
s1=int(input("Enter {} Size:".format(p1)))
p2=input("Enter a partition name:")
s2=input("Enter {} Size:".format(p2))

total=s1+int(s2)
print('''
Partition name:{}\t  Size:{} GB
-------------------------------------
Partition name:{}\t  Size:{} GB
-------------------------------------
      Total: {} GB
-------------------------------------'''.format(p1,s1,p2,s2,total))
```

```
Enter a partition name:/dev/sda1
Enter /dev/sda1 Size:100
Enter a partition name:/dev/sda2
Enter /dev/sda2 Size:120

Partition name:/dev/sda1          Size:100 GB
-------------------------------------
Partition name:/dev/sda2          Size:120 GB
-------------------------------------
      Total: 220 GB
-------------------------------------
```

In [115]:
```
name=input("Enter a name:")
if name == "root":
    var=input("Enter a shell name:")
    if var == "bash":
        fname="bashrc"
    elif var == "ksh":
        fname="kshrc"
    elif var == "psh":
        fname="PSprofile"
    else:
        print("Sorry your input shell is not matched")
        print("default shell details:-")
        var="/bin/nologin"
        fname="/etc/profile"
    print("Login name:{}\t Shell name:{}\t Profile:{}".format(name,var,fname))
else:
    print("Sorry your not root user")
```

```
Enter a name:root
Enter a shell name:tcsh
Sorry your input shell is not matched
default shell details:-
Login name:root  Shell name:/bin/nologin         Profile:/etc/profile
```

In [ ]:
```
if var == "bash" or var == "sh" or var == "ksh" :
    fname="/etc/profile"
elif var == "ps1" or var == "ps2":
    fname="C:\\winprofile"
else:
    var="defaultshell"
    fname="defaultfile"
```

In [ ]:
```
# Conditional statements(if statement) - code block will execute only onetime
# -----------------------------------
# Looping statements - Code block will execute more than one time
# -----------------------------------
#     |___ Conditional style - based on True/False  - while
#     |___ Collection style  - based on data(str,bytes,list,tuple,dict,set,iterat
#                for

conditional style loop

step 1: initialization
step 2: condition
step 3: increment/decrement

Syntax:-
------------
initialization # step 1
while(condition): # step 2
    codeblock
    arithemtic # step 3
else:
    optional block
```

In [116]:
```
i=0
while(i<5):
    print("Test code:{}".format(i))
    i=i+1
```

```
Test code:0
Test code:1
Test code:2
Test code:3
Test code:4
```

In [118]:
```
while(False):
    print("Hello")
```

In [119]:
```
while(False):
    print("Hello")
else:
    print("thank you!!!")
```

```
thank you!!!
```

In [120]:
```
i=0
while(i<5):
    print("Test code:{}".format(i))
    i=i+1
else:
    print("-"*15)
    print("Thank you")
    print("-"*15)
```

```
Test code:0
Test code:1
Test code:2
Test code:3
Test code:4
---------------
Thank you
---------------
```

In [122]:
```
# for variable in collection:
#         codeblock

for var in "welcome":
    print(var)
else:
    print("Thank you")
```

```
w
e
l
c
o
m
e
Thank you
```

In [ ]:
```
# break - exit from loop
# continue - next ;ignore
```

In [123]:
```
for var in 'abcde':
    if(var == 'c'):
        break # exit from loop
    else:
        print(var)
```

```
a
b
```

In [124]:
```
for var in 'abcde':
    if(var == 'c'):
        continue # ignore
    else:
        print(var)
```

```
a
b
d
e
```

In [125]:
```
# int float str bytes bool None # Single data

# List - Collection of ordered elements - indexbased - mutable - []

# Syntax:-
# -----------
# Listname=[collection of items]
L=[10,2.45,'data',b'ab',True,None]
print(type(L),len(L))
```

```
<class 'list'> 6
```

In [136]:
```
L=[10,2.45,'data',b'ab',True,None]
#   0   1     2     3     4    5  <== index
#  -6  -5    -4    -3    -2    -1 <== index
L[1]
L[3]
L[3:]
L[:3]
L[1]=43.56
L
```

Out[136]:
```
[10, 43.56, 'data', b'ab', True, None]
```

In [137]:
```
print(type(L),type(L[0]))
print(type(L),type(L[-1]))
print(type(L),type(L[3]))
```

```
<class 'list'> <class 'int'>
<class 'list'> <class 'NoneType'>
<class 'list'> <class 'bytes'>
```

In [ ]:
```
Listname.append(Value) (or)Listname.insert(index,Value)

Listname.pop(Index) ->removed item
                |__outofrange/notexisting - IndexError
Listname[Existing_Index]=Updated_Value
           |
        outofrange/notexisting ->IndexError
```

In [138]:
```
for var in 'python':
    print(var)
```

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

In [139]:
```
for var in ['python','java','test',124,45.344]:
    print(var)
```

```
python
java
test
124
45.344
```

In [149]:
```
# List - Collection of ordered elements - indexbased - mutable - []
# tuple - Collection of ordered elements - indexbased - immutable - ()
#                                                      ^^^^^^^^^
L=['sql','mysql','sqlite3','oralce']
T=('sql','mysql','sqlite3','oralce')

L[1] == T[1]
```

Out[149]:
```
True
```

In [142]:
```
L[1]='pl/sql'
L
```

Out[142]:
```
['sql', 'pl/sql', 'sqlite3', 'oralce']
```

In [150]:
```
#T[1]='pl/Sql' ->Error
s='abc'
s[1]
#s[1]='X' -->Error
```

Out[150]:
```
'b'
```

In [151]:
```
# dictionary (dict) {}
# Collection of unordered elements - data - key:value - Like HashTable - mutable
# ------------------------------------------------------------------------
#
# Syntax:-
# ------------
# dictname={"Key1":Value,"Key2":Value2,"Key3":Value3...Kn:Vn}

emp={'ename':'Mr.Kumar','eid':1234,'edept':'DBA','eplace':'pune'}
print(type(emp),len(emp))

'''
Key      |     Value
---------------------
ename    |     Mr.Kumar
------------------------
eid      |     1234
------------------------
edept    |     DBA
------------------------
eplace   |     pune
-------------------------
'''
```

```
<class 'dict'> 4
```

In [152]:
```
emp['edept']
```

Out[152]:
```
'DBA'
```

In [153]:
```
emp['edept']='sales'
emp
```

Out[153]:
```
{'ename': 'Mr.Kumar', 'eid': 1234, 'edept': 'sales', 'eplace': 'pune'}
```

In [154]:
```
emp.pop('eplace')
```

Out[154]:
```
'pune'
```

In [155]:
```
emp
```

Out[155]:
```
{'ename': 'Mr.Kumar', 'eid': 1234, 'edept': 'sales'}
```

In [ ]:
```
dict - key - unique ; value can duplicate
                |
            int,float,str,bytes,bool,None,tuple - immutable //dict_key
```

In [157]:
```
d={1:"V1",2:[],3:{},4.55:"V1",():"V2"}
d
```

Out[157]:
```
{1: 'V1', 2: [], 3: {}, 4.55: 'V1', (): 'V2'}
```

In [158]:
```
d={[]:"V"} #  Error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-158-69265d6e1e21> in <module>
----> 1 d={[]:"V"}

TypeError: unhashable type: 'list'
```

In [159]:
```
# membership - key is existing or not?
#
# "input_key" in dictionary ->True/False
#
d={"K1":"V1","K2":"V2"}
"K1" in d
```

Out[159]:
```
True
```

In [160]:
```
"K3" in d
```

Out[160]:
```
False
```

In [161]:
```
"V1" in d
```

Out[161]:
```
False
```

In [163]:
```
for var in d:
    print(var,d[var])
```

```
K1 V1
K2 V2
```

In [ ]:
```
Login:<input type='text' name='n1' value=''> //htmlform
|    __________
Login:| root  |     =====>    {"n1":"root"}
      ----------              ----------------//dict style

bs4 - parsing
<a href='python.org' ...>
 | ----------------
       |__{'href':'python.org'}
          ------------------------//dict

framework ->GET/POST  --> formobject['key'] ->Value
...
```

In [ ]:
```
10
20
L=[10,20,[10,20],[[10,20]]]
     |  |   ^   | |  ^  |
     |  |   ^   | |  ^  |

List of List,tuple,dict
tuple of List,tuple,dict
dict of List,tuple,dict
|      =================
|           |__unnamed struct
named
struct
```

In [164]:
```
L=[10,11,[12,13,14,'d1','d2'],('d3','d4','d5')]
 # 0   1 --------2nd--------- --------3rd-----  <== index
 # -4  -3          -2                  -1 <== index
type(L) # <class 'list'>
```

Out[164]:
```
list
```

In [165]:
```
print(type(L),type(L[0]))
print(type(L),type(L[1]))
print(type(L),type(L[2]))
print(type(L),type(L[3]))
```

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

In [167]:
```
print(L[2][0])
print(L[2][1])
print(L[2][-1])
```

```
12
13
d2
```

In [171]:
```
# tuple of List/Tuple/dict

t=([10,20,30,40],('d1','d2','d3'))
print(type(t),len(t),type(t[0]))
```

```
<class 'tuple'> 2 <class 'list'>
```

In [173]:
```
t[0].append(50)
t[0].append("Dx")
t[0].append("Dy")
t[0].append("Dz")
print(t,len(t))
```

```
([10, 20, 30, 40, 50, 'Dx', 'Dy', 'Dz', 50, 'Dx', 'Dy', 'Dz'], ('d1', 'd2', 'd
3')) 2
```

In [175]:
```
t[0][1]="https://www.abc.com"
t
```

Out[175]:
```
([10,
  'https://www.abc.com',
  30,
  40,
  50,
  'Dx',
  'Dy',
  'Dz',
  50,
  'Dx',
  'Dy',
  'Dz'],
 ('d1', 'd2', 'd3'))
```

In [ ]:
```
L=[{"K1":"V1","K2":"V2"}] # List of dict
|  ---------- ---------
|
T=({"Key1":"Value","Key2:"Value2"},{"K1":"V1"}) # tuple of dict
|    ----------------------------    ---------
|__ Size of tuple(T) - fixed size
```

In [178]:
```
L=[{"url":'https://www.ab.com',"port":80,"fname":"/etc/passwd"}]
print(len(L))
print(type(L[0]),len(L[0]))
```

```
1
<class 'dict'> 3
```

In [181]:
```
for var in L[0]:
    print(var,"-->",L[0][var])
```

```
url --> https://www.ab.com (https://www.ab.com)
port --> 80
fname --> /etc/passwd
```

In [182]:
```
L=[] # empty list
L.append("ONE") # Single
L.append(["server1","server2","server3","server4","server5"])
L.append((110,200,300))
L.append({"DB1":"DBNAME1","DB2":"DBNAME2"})
L
```

Out[182]:
```
['ONE',
 ['server1', 'server2', 'server3', 'server4', 'server5'],
 (110, 200, 300),
 {'DB1': 'DBNAME1', 'DB2': 'DBNAME2'}]
```

In [183]:
```
print(L[0])
print(L[1][1])
print(L[2][1])
print(L[3]['DB1'])
```

```
ONE
server2
200
DBNAME1
```

In [184]:
```
import pprint
pprint.pprint(L)
```

```
['ONE',
 ['server1', 'server2', 'server3', 'server4', 'server5'],
 (110, 200, 300),
 {'DB1': 'DBNAME1', 'DB2': 'DBNAME2'}]
```

In [191]:
```
emp={'ename':"Mr.Kumar",'eid':1234,'ecost':3343433.55} # 1 to 1

# 1 to many value
#      ----------- [ ]  ( ) { }
#

L=[{"K":"V"}] #  L[0]['K'] ->"V"

Emp={"enames":[],"edept":('sales','admin','QA','DBA')} # dict of List / dict of t

type(Emp['enames'])
```

Out[191]:
```
list
```

In [187]:
```
Emp['enames'].append('Mr.X')
Emp['enames'].append('Mr.Y')
Emp['enames'].append('Mr.Z')
Emp
```

Out[187]:
```
{'enames': ['Mr.X', 'Mr.Y', 'Mr.Z'], 'edept': ('sales', 'admin', 'QA', 'DBA')}
```

In [188]:
```
Emp['enames'][1]
```

Out[188]:
```
'Mr.Y'
```

In [199]:
```
Emp={"enames":[],"edepts":("sales","admin","QA")}

Emp['enames'].append('kumar')
Emp['enames'].append('arun')
Emp['enames'].append('Anu')
Emp
```

Out[199]:
```
{'enames': ['kumar', 'arun', 'Anu'], 'edepts': ('sales', 'admin', 'QA')}
```

In [204]:
```
Emp['enames'][1]
print("Emp name is:{} working dept is:{}".format(Emp['enames'][1],Emp['edepts'][1
```

```
Emp name is:arun working dept is:admin
```

In [209]:
```
d={"K1":{"K1":"V1","K2":"V2","K3":"V3"},"K2":{"K1":101,"K2":202,"K3":345}}

# d['K1']['K1']
# d['K1']['K2']
#    |      |
#  outer   innerkey
#            |_nested key
```

In [212]:
```
d['K1']['K1']='/var/log/repo.log'
d
```

Out[212]:
```
{'K1': {'K1': '/var/log/repo.log', 'K2': 'V2', 'K3': 'V3'},
 'K2': {'K1': 101, 'K2': 202, 'K3': 345}}
```

In [213]:
```
d['K1']['K4']='V4'
d['K2']['K4']=134455
d
```

Out[213]:
```
{'K1': {'K1': '/var/log/repo.log', 'K2': 'V2', 'K3': 'V3', 'K4': 'V4'},
 'K2': {'K1': 101, 'K2': 202, 'K3': 345, 'K4': 134455}}
```

In [ ]:
```
d={"Key":[{},{},{},{}]}
     |   |           |
     |   | 0  1  2  3 <== index
     |   |-4  -3 -2 -1 <== index
     |      index based
     |
    key:Value based
import pprint
pprint.pprint(d)
```

In [215]:
```
d={'instance1':[{'name':'10.20.30.40'}]}

d['instance1'].append({'name':'12.34.5.66'})

d['instance1'].append({'name':'190.160.15.69'})
d
import pprint
pprint.pprint(d)
```

```
{'instance1': [{'name': '10.20.30.40'},
               {'name': '12.34.5.66'},
               {'name': '190.160.15.69'}]}
```

In [217]:
```
d['instance1'][0]['name']
```

Out[217]:
```
'10.20.30.40'
```

In [ ]:
```
'''
Q1. write a program:
step 1: declare a pin number (ex: pin=1234)
step 2: using while loop - 3 times
             read a pin number from <STDIN>
             compare input pin with existing pin
             display pin is matched - at <Count>
step 3: all 3 inputs are failed - display pin is blocked

Note: store all the input details to list
      ..................................

Q2. step 1: create an empty list
    step 2: use while loop - 5times
            read a hostname from <STDIN>
            append input hostname to existing list
    step 3: display list of hostdetails - use forloop

Ex:
1. host01
2. host02
3. host03
..
5. host05
Total no.of host is:5
'''
```

In [229]:
```
import time
pin=1234
pin_history=[]
count=0
while(count<3):
    p=int(input("Enter a pin number:"))
    count=count+1
    if p == pin:
        print("pin is matched at {} attempt.".format(count))
        d=time.ctime()
        pin_history.append("VALID_ENTRY:"+str(p)+" Entered on "+d)
        break # exit from loop
    else:
        pin_history.append("InvalidEntry"+str(p)+" Entered on "+d)
if(pin != p):
    print("Sorry your pin is blocked")

choice=input("Wish to View list of input pins:")
if(choice == "Yes" or choice == "YES" or choice == "yes"):
    for var in pin_history:
        print(var)
```

```
Enter a pin number:1234
pin is matched at 1 attempt.
Wish to View list of input pins:yes
VALID_ENTRY:1234 Entered on Mon Apr 26 14:29:50 2021
```

In [230]:
```
hosts=[]
c=0
while(c<5):
    var=input("Enter a hostname:")
    hosts.append(var)
    c=c+1
count=0
for var in hosts:
    count=count+1
    print("{}. {}".format(count,var))
```

```
Enter a hostname:host01
Enter a hostname:host02
Enter a hostname:host03
Enter a hostname:host04
Enter a hostname:host05
1. host01
2. host02
3. host03
4. host04
5. host05
```

In [233]:
```
item_details={"itemcode1":[10,20,30,40,50],"itemcode2":[100,200,300]}

# itemcode1 total sales count is:150
# itemcode2 total sales count is:600
# -----------------------------------

for var in item_details:
    t=0
    for v in item_details[var]:
        t=t+v
    else:
        print("{} total sales count is:{}".format(var,t))
```

```
itemcode1 total sales count is:150
itemcode2 total sales count is:600
```

In [234]:
```
item_details={"itemcode1":[10,20,30,40,50],"itemcode2":[100,200,300]}

item_details['itemcode3']=[15,25,35,45,65]
```

In [235]:
```
for var in item_details:
    t=0
    for v in item_details[var]:
        t=t+v
    else:
        print("{} total sales count is:{}".format(var,t))
```

```
itemcode1 total sales count is:150
itemcode2 total sales count is:600
itemcode3 total sales count is:185
```

In [237]:
```
# dict - Collection of unordered elements - data - key:value {}
#  |
# set  - Collection of unordered elements - data - keyonly
# |__ collection of keys
#                    |__unique element
#
s={"K1","V1","K2",10,20,"K1","K2",10,20,30,10,20} # Vs d={"K1":"V1","Kx":"Vx"}
print(type(s),len(s))
print(s)
```

```
<class 'set'> 6
{'V1', 10, 'K2', 20, 'K1', 30}
```

In [239]:
```
# set is not key:value, not index - so there is no modification
#
# we can add newelement/remove existing element
s={10,20}
s.add(30)
s.add("data")
s
```

Out[239]:
```
{10, 20, 30, 'data'}
```

In [240]:
```
 s.remove(30)
s
```

Out[240]:
```
{10, 20, 'data'}
```

In [242]:
```
s
```

Out[242]:
```
{10, 20, 'data'}
```

In [ ]:
```
# s.remove(30) # KeyError
```

In [245]:
```
s.discard(30)
s.discard(20)
s
```

Out[245]:
```
{10, 'data'}
```

In [ ]:
```
union intersection  difference symmetric_difference

1. operator ways -> | & - ^
     (or)
2. method ways -> union() intersection()  difference() symmetric_difference()
```

In [246]:
```
s={1,2,3,4}
s.add(5)
s.add('D1')
s.remove(2)
s
```

Out[246]:
```
{1, 3, 4, 5, 'D1'}
```

In [252]:
```
s=frozenset(s)
s.remove(4)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-252-a5f367eed367> in <module>
      1 s=frozenset(s)
----> 2 s.remove(4)

AttributeError: 'frozenset' object has no attribute 'remove'
```

In [256]:
```
L1=["D1","d2","d3","d4","d5","D6","d7"]
L2=["D1","d2","dx","dy","dz"]

set(L1)-set(L2)
```

Out[256]:
```
{'D6', 'd3', 'd4', 'd5', 'd7'}
```

In [257]:
```
L=[10,20,30,40,10,20,20,10,10,20,10,20]
s=set(L)
print(s)
L=list(s)
print(L)
```

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

In [263]:
```
s[0] # set is not indexbased
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-263-9a37f9b68524> in <module>
----> 1 s[0] # set is not indexbased

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

In [261]:
```
for var in s:
    print(var)
```

```
40
10
20
30
```

In [266]:
```
L=['p1.log','p2.log','p3.log','p6.py','p3.log','p3.log']
L.index('p3.log')
```

Out[266]:
```
2
```

In [269]:
```
if "p6.py" in L:
    print("Index number is:{}".format(L.index("p6.py")))
    print(L[L.index("p6.py")])
else:
    print("Not exists")
```

```
Index number is:3
p6.py
```

In [276]:
```
# File Handling
# ---------------
# str,list
#
# Keyboard(STDIN) -----------------Python--------------------Monitor
#                                    |
#                                    |read()/readlines()/write()
#                                  Storage(FILE)
#
# 1. reading data from <FILE> -->-- python ---->--display to monitor
# 2. python ->-- create/Write data to FILE
# 3. read data from <oneFILE>-->--Python --->--- Create/Write data to another FIL
#
#   open() --> open("filename","mode")  # read(r) write(w) append(a)
#

FH=open("C:\\users\\karthikeyan\\emp.csv")
s=FH.read()
FH.close()
```

In [277]:
```
print(s)
```

```
101,ram,sales,pune,10000
203,arun,prod,bglore,2000
304,vijay,QA,chennai,30000
545,xerox,sales,mumbai,34234
456,anu,sales,noida,56780
```

In [279]:
```
FH=open("C:\\users\\karthikeyan\\emp.csv")
L=FH.readlines()
FH.close()
```

In [285]:
```
s1="data\n"
s2=" data\t"
s3=" data:"
s3.strip(":")
```

Out[285]:
```
' data'
```

In [289]:
```
for var in L[-3:]:
    print(var.strip())
```

```
304,vijay,QA,chennai,30000
545,xerox,sales,mumbai,34234
456,anu,sales,noida,56780
```

In [290]:
```
# read data from <FILE1> to script -->create/Write data to FILE2
# ----------------------------------------------------------------
FH=open("D:\\TEMP\\r1.log")
WH=open("D:\\TEMP\\r2.log","w")
s=FH.read()
WH.write(s)
FH.close()
WH.close()
```

In [291]:
```
F=open("D:\\TEMP\\r2.log")
F.read()
```

Out[291]:
```
'Interface=eth0\nIP=10.20.30.40\nPORT=456\neth0,10.20.30.40,456\n'
```

In [ ]:
```
# Block style
# -----------------
# with as - keywords
# --------
#  no need to write object.close()

FH=open("inputfile","mode")  --> with open("inputfile","mode") as FH:
s=FH.read()                                 s=FH.read()
FH.close()

WH=open("resultfile","w") --> with open("resultfile","w") as WH:
WH.write("SingleString\n") ---->          WH.write("data\n")
WH.close()
```

In [292]:
```
with open("result.log","w") as WH:
    with open("D:\\TEMP\\r1.log") as FH:
        s=FH.read()
        WH.write(s)
```

In [293]:
```
with open("result.log") as FH:
    print(FH.read())
```

```
Interface=eth0
IP=10.20.30.40
PORT=456
eth0,10.20.30.40,456
```

In [296]:
```
d
```

Out[296]:
```
'Mon Apr 26 15:54:14 2021'
```

In [299]:
```
import time
pin=1234
WH=open("pin_history.log","a")
count=0
while(count<3):
    p=int(input("Enter a pin number:"))
    count=count+1
    if p == pin:
        print("pin is matched at {} attempt.".format(count))
        WH.write("VALID_ENTRY:"+str(p)+" Entered on "+time.ctime()+"\n")
        break # exit from loop
    else:
        WH.write("InvalidEntry"+str(p)+" Entered on "+time.ctime()+"\n")

WH.close()
if(pin != p):
    print("Sorry your pin is blocked")

choice=input("Wish to View list of input pins:")
if(choice == "Yes" or choice == "YES" or choice == "yes"):
    for var in open("pin_history.log").readlines():
        print(var)
```

```
Enter a pin number:13454
Enter a pin number:4343
Enter a pin number:1234
pin is matched at 3 attempt.
Wish to View list of input pins:yes
InvalidEntry13454 Entered on Mon Apr 26 15:56:08 2021

InvalidEntry4343 Entered on Mon Apr 26 15:56:13 2021

VALID_ENTRY:1234 Entered on Mon Apr 26 15:56:15 2021
```

In [ ]:
```
# Function
# ------------
#  user defined function
#  ----------------------
#  function - codeblock - operation
#  1. function definition - operation block
#  2. function call - to invoke a definition

# Syntax:-
# ----------
# def functionname():
#       .........
#       .........
# functionname()
```

In [301]:
```
def display():
    print("This is display block")

display()
```

```
This is display block
```

In [302]:
```
def f2():
    print("Exit from block")
def f1():
    f2() # nested function call
    print("Exit from f1 block")
f1()
```

```
Exit from block
Exit from f1 block
```

In [306]:
```
L=[]
# L.append("D1","D2") Error
# L.append() Error
L.append(10)
L.append(1.345)
L.append("data")
L.append(343)
L.append(["D1","D2"])
L
```

Out[306]:
```
[10, 1.345, 'data', 343, ['D1', 'D2']]
```

In [307]:
```
L.pop(1)
```

Out[307]:
```
1.345
```

In [308]:
```
L
```

Out[308]:
```
[10, 'data', 343, ['D1', 'D2']]
```

In [309]:
```
L.pop() ##<<<<
```

Out[309]:
```
['D1', 'D2']
```

In [ ]:
```
# def pop(arg=-1):
#
```

In [314]:
```
def f1(a1,a2,a3): #<<< Required arguments
    print("Hello")
#f1()
#f1(10)
#f1(10,20)
#f1(10,20,30,40)

f1(10,0.44,[])
```

```
Hello
```

In [316]:
```
def f2(user="root",dept="DBA",port=1234): # default args
    print(user)
    print(dept)
    print(port)

f2()
f2("admin")
```

```
root
DBA
1234
admin
DBA
1234
```

In [321]:
```
def fx(a1,a2,a3=0,a4=0): # required,default args
    print("Hello")
fx(10,20,30,40)
```

```
Hello
```

In [322]:
```
d={}
d['K1']="V1" # adding data to dict

d.setdefault("K2","V2") # adding ndata to dict
d
```

Out[322]:
```
{'K1': 'V1', 'K2': 'V2'}
```

In [323]:
```
d.setdefault("K3")
d
```

Out[323]:
```
{'K1': 'V1', 'K2': 'V2', 'K3': None}
```

In [ ]:
```
def setdefault(Key,Value=None):
    pass
```

In [328]:
```
del(v3)
```

In [332]:
```
def fx():
    global v1,v2
    v1=100
    v2="/var/log/repo.log"
    v3=0.56
fx()
print(v1,v2)

def fy():
    print(v1,v2)
fy()
```

```
100 /var/log/repo.log
100 /var/log/repo.log
```

In [331]:
```
print(v1,v2)
```

```
100 /var/log/repo.log
```

In [334]:
```
def fx():
    v=1234
    return v
rv=fx()

print(rv)
```

```
1234
```

In [336]:
```
def fx():
    return True

if fx():
    print("Yes")
else:
    print("Not")
```

```
Yes
```

In [343]:
```
def fx():
    return "STDOUT","STDERR",[],(),{}
fx()
```

Out[343]:
```
('STDOUT', 'STDERR', [], (), {})
```

In [339]:
```
def fx():
    x=123+123
fx()
fx() == None
```

Out[339]:
```
True
```

In [ ]:
```
# module ->existing python file
```

In [345]:
```
import os
```

In [ ]:
```
D:\\TEMP> ab.py      p1.py
                       import ab
                       ab.name

D:\\TEMP> ab.py     E:\\>test1.py
                         import ab - ImportError(2.x)/ModuleNotFoundError(3.x)

python -->refer sys.path ->[ ]
```

In [ ]:
```

```