In [ ]:
```
# Python - General purpose programming language
#        - dynamic type programming
#          ..........................
#               variable = value <== initialization
                ........
                 |->variable name starts with a-zA-Z_ (not starts with digit)
                 |->variable name not allows space,specialchars
                 |->Don't use python keywords as a variable
# In compiler code => <dataType> <variable> = Value <= declaration &
#                                                      initialization
#                     ------------------------------

# In python -> class <--> object model

10    - int class
10.0  - float class
'abc' - str class
Python native types are: int float complex str bytes NoneType bool
                         list tuple dict set

number: int float complex
bool : True False
NoneType : None
Collection: str bytes list tuple dict set

str : Collection of chars - index based - immutable  '' <or> ""
bytes: Collection of ASCII - index based - immutable b''

list: Collection of items - index based - mutable (we can add/modify/delete)[]
tuple: Collection of items - index based - immutable () - fixed record set
|
dict: Collection of item - key:value - mutable(we can add/modify/delete){}
|
set: Collection of item - This is NOT index based
                          This is NOT Key:Value
                          Allows unique items
                    set Operation(Union,Intersection,difference,
                                  symmetric_difference)
```

In [5]:
```
#File
#  |->Filename,Fsize,IndexNumber,Open_status .... //
#
Fname = "emp.csv"
fsize = "15KB"  # fsize=15 ->OK  fsize=15.23 ->OK
findex = 3465
fstatus = True
print(Fname)
print("Fname")
print("Fname is Fname")
# print(fname) NameError: name 'fname' is not defined

print("Fname is:",Fname)
```

```
emp.csv
Fname
Fname is Fname
Fname is: emp.csv
```

In [6]:
```
print("File name is:",Fname,"Index Number:",findex,"Size:",fsize,
      "Open status:",fstatus)
```

```
File name is: emp.csv Index Number: 3465 Size: 15KB Open status: True
```

In [7]:
```
va = 56
vb = 1.45
vc = True
vd = 'data'
print('va value is:',va,'vb value is:',vb,'vc value is:',vc,
      'vd value is:',vd)
```

```
va value is: 56 vb value is: 1.45 vc value is: True vd value is: data
```

In [8]:
```
# from C language %d %f %s
print('va value is:%d vb value is:%f vc value is:%s vd value:%s'%(va,vb,vc,vd))
```

```
va value is:56 vb value is:1.450000 vc value is:True vd value:data
```

In [10]:
```
n=56
print('n value is:%d'%(n))
n="Hello"
print('n value is:%d'%(n))
```

```
n value is:56

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[10], line 4
      2 print('n value is:%d'%(n))
      3 n="Hello"
----> 4 print('n value is:%d'%(n))

TypeError: %d format: a real number is required, not str
```

In [11]:
```
n=56
print("n value is:{}".format(n))
n="Hello"
print("n value is:{}".format(n))
```

```
n value is:56
n value is:Hello
```

In [12]:
```
print('va value is:{} vb value is:{} vc value is:{} vd value:{}'
      .format(va,vb,vc,vd))
```

```
va value is:56 vb value is:1.45 vc value is:True vd value:data
```

In [13]:
```
print(f"va value is:{va} vb value is:{vb} vc value is:{vc} vd value is:{vd}")
```

```
va value is:56 vb value is:1.45 vc value is:True vd value is:data
```

In [14]:
```
print(f"va value is:{va},vb value is:{vb},vc value is:{vc},vd value is:{vd}")
```

```
va value is:56,vb value is:1.45,vc value is:True,vd value is:data
```

In [15]:
```
print(f"va value is:{va}\nvb value is:{vb}\tvc value is:{vc}\nvd value is:{vd}")
```

```
va value is:56
vb value is:1.45        vc value is:True
vd value is:data
```

In [ ]:
```
# Task
initialize an application details(appName,appPort,appConfig,appVersion,status)
use single print() - display app details line by line
```

In [16]:
```
app_Name = 'Flask'
app_port = 5000
app_config = "D:\\test.conf"
app_version = 1.2
debug_mode = True

print(f'App name is:{app_Name}\n{app_Name} running port:
      {app_port}\nConfig:{app_config}\nversion:{app_version}\n
      debug mode is:{debug_mode}')
```

```
App name is:Flask
Flask running port:5000
Config:D:\test.conf
version:1.2
debug mode is:True
```

In [17]:
```
# Multiline string
print('''data1
data2
data3
data4
data5''')
```

```
data1
data2
data3
data4
data5
```

In [18]:
```
print(f'''App name is:{app_Name}
-----------------------------------
{app_Name} running port:{app_port}
-----------------------------------
{app_Name} Config file is:{app_config}
-------------------------------------
version:{app_version}
-------------------------------------
debug mode is:{debug_mode}''')
```

```
App name is:Flask
-----------------------------------
Flask running port:5000
-----------------------------------
Flask Config file is:D:\test.conf
-------------------------------------
version:1.2
-------------------------------------
debug mode is:True
```

In [22]:
```
app = ['Flask',5000,1.0,"D:\\test.conf",True] # list
#          0    1    2         3         4   <===  index
#         -5   -4   -3        -2        -1   <== index
print(app[0])
print(app[-5])
print(app[2:]) # slicing from 2nd index to list of all
print(app[-3:]) # last 3 items
```

```
Flask
Flask
[1.0, 'D:\\test.conf', True]
[1.0, 'D:\\test.conf', True]
```

In [23]:
```
data = ['D1','D2','D3','D4','D5','D6','D7','D8']
data[2:7] # from 2nd index to 6th index(7-1)
```

Out[23]:
```
['D3', 'D4', 'D5', 'D6', 'D7']
```

In [24]:
```
app = ('Flask',5000,1.0,"D:\\test.conf",True) # tuple
#          0    1    2         3         4   <===  index
#         -5   -4   -3        -2        -1   <== index
print(app[0])
print(app[-5])
print(app[2:]) # slicing from 2nd index to list of all
print(app[-3:]) # last 3 items
```

```
Flask
Flask
(1.0, 'D:\\test.conf', True)
(1.0, 'D:\\test.conf', True)
```

In [26]:
```
print(type([]))
print(type(()))
```

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

In [27]:
```
L=['D1','D2','D3']
L[1]='Data-2' # modification
L
```

Out[27]:
```
['D1', 'Data-2', 'D3']
```

In [29]:
```
T=('D1','D2','D3')
T[1]='Data-2' # Error
#TypeError: 'tuple' object does not support item assignment
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[29], line 2
      1 T=('D1','D2','D3')
----> 2 T[1]='Data-2'

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

In [30]:
```
d={'Key1':'Value'}
#  -----   =====
#   |->

app={'K1':'Flask','K2':5000,'K3':'D:\\test.conf','K4':1.0,'K5':True}
print(app)
```

```
{'K1': 'Flask', 'K2': 5000, 'K3': 'D:\\test.conf', 'K4': 1.0, 'K5': True}
```

In [32]:
```
print(app['K1'],app['K2'])
```

```
Flask 5000
```

In [ ]:
```
NameError - undefined -> variable name (or) function name (or) classname
IndexError - output of index str,bytes,list,tuple
KeyError - invalid key - dict,set
```

In [ ]:
```
list of list,tuple,dict
tuple of list,tuple,dict
dict of list,tuple,dict
```

In [ ]:
```
L = [['flask','fastAPI','testApp'],(5000,8000,3030),
     {'K1':'test.conf','K2':'project.conf','K3':'repo.conf'}]
#    ----------0------------------|-------1------- |---------------------------2
```

In [33]:
```
L = [['flask','fastAPI','testApp'],(5000,8000,3030),
     {'K1':'test.conf','K2':'project.conf','K3':'repo.conf'}]
print(L)
print(L[0])
print(L[1])
print(L[-1])
print("") # empty line
print(L[0][0],L[1][0],L[-1]['K1'])
```

```
[['flask', 'fastAPI', 'testApp'], (5000, 8000, 3030), {'K1': 'test.conf', 'K2':
'project.conf', 'K3': 'repo.conf'}]
['flask', 'fastAPI', 'testApp']
(5000, 8000, 3030)
{'K1': 'test.conf', 'K2': 'project.conf', 'K3': 'repo.conf'}

flask 5000 test.conf
```

In [ ]:
```
L=[]
L.append("D1") # adding new item at the end.
L[OldIndex] = updatedValue # modification

d={"Kx":"Vx"}

d['OldKey']='Value' = modification
     Vs
d['NewKey']='Value' = Adding new data

d['K1']='ValueA' # adding new data
d['Kx']="ValueB" # modification
```

In [ ]:
```
L = [['flask','fastAPI','testApp'],(5000,8000,3030),
     {'K1':'test.conf','K2':'project.conf','K3':'repo.conf'}]

From the given list
Add new a App (ex: Django)
Add new a config file(ex: app.conf)

modify ->fastAPI => lemma2
modify ->project.conf => main.conf
display updated list_(L)
```

In [34]:
```
L = [['flask','fastAPI','testApp'],(5000,8000,3030),
     {'K1':'test.conf','K2':'project.conf','K3':'repo.conf'}]

L[0].append('Django')
L[-1]['K4']='app.conf'

L[0][1]='lemma2'
L[-1]['K2']='main.conf'
```

In [35]:
```
L
```

Out[35]:
```
[['flask', 'lemma2', 'testApp', 'Django'],
 (5000, 8000, 3030),
 {'K1': 'test.conf', 'K2': 'main.conf', 'K3': 'repo.conf', 'K4': 'app.conf'}]
```

In [ ]:
```
L.append('D1') # OK
L[0]='DATA' # OK
```

In [40]:
```
T=([],[]) # tuple of list
print(len(T))
print(type(T))
print(type(T[0]))
T[0].append('D1')
T[0].append('D2')
T[0].append('D3')
T[0].append('D4')
print(len(T))
T
```

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

Out[40]:
```
(['D1', 'D2', 'D3', 'D4'], [])
```

In [42]:
```
T = (['flask','fastAPI','testApp'],(5000,8000,3030),
     {'K1':'test.conf','K2':'project.conf','K3':'repo.conf'})

# Tuple of list  - 0th index
# Tuple of tuple - 1st index
# Tuple of dict  - 2nd index

T[0].append('Django')
T[0][1]='lemma2'

T[-1]['K4']='app.conf'
T[-1]['K2']='main.conf'
T
```

Out[42]:
```
(['flask', 'lemma2', 'testApp', 'Django'],
 (5000, 8000, 3030),
 {'K1': 'test.conf', 'K2': 'main.conf', 'K3': 'repo.conf', 'K4': 'app.conf'})
```

In [43]:
```
d={"K1":'V1'} # 1D

      # Vs

d={"K1":["V1","V2","V3"]} # dict of list

d={"K2":("V1","V2","V3")} # dict of tuple

d={"K3":{"K1":"V1","K2":"V2"}} # dict of dict
```

In [46]:
```
d={"K1":["V1","V2","V3"],"K3":{"K1":"V1","K2":"V2"}}

print(d['K1'][0],d['K1'][-1]) # dict of list

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

```
V1 V3
V1
```

In [48]:
```
d['K3']['K1'] = 'updated_Value'
d
```

Out[48]:
```
{'K1': ['V1', 'V2', 'V3'], 'K3': {'K1': 'updated_Value', 'K2': 'V2'}}
```

In [49]:
```
products={'pid':[101,102,103,104,105],
          'pnames':['proA','prodB','prodC','prodD','prodE'],
         'pcost':[1000,2000,1200,3400,4000]}

print(products['pid'][0],products['pnames'][0],products['pcost'][0])
```

```
101 proA 1000
```

In [ ]:
```
# Python Operators
# arithmetic input_types are int,float ->int,float
# string     input_types are int,str -> str
# relational input_types are int,float,str ->bool
# logical  input_types are int,float,str ->bool
# membership input_types are str,bytes,list,tuple,dict,set ->bool
#   (in not in)
# identity operators type specific ->bool
#          is is not
```

In [54]:
```
print(10+20.0)
print(10*2.34)
print('A'+'B')
print('ABC'*5)
```

```
30.0
23.4
AB
ABCABCABCABCABC
```

In [55]:
```
s='101,raj,sales,pune,1000'
'sales' in s
```

Out[55]:
```
True
```

In [56]:
```
'prod' in s
```

Out[56]:
```
False
```

In [57]:
```
files = ['p1.log','p2.java','p2.txt','r1.log']
'p2.java' in files
```

Out[57]:
```
True
```

In [58]:
```
'emp.csv' in files
```

Out[58]:
```
False
```

In [59]:
```
'pid' in products
```

Out[59]:
```
True
```

In [60]:
```
'pcost' in products
```

Out[60]:
```
True
```

In [63]:
```
va = 10
print(type(va) is int,type(va) is float)
```

```
True False
```

In [64]:
```
L=[]
type(L) is list
```

Out[64]:
```
True
```

In [ ]:
```
In Python any expression <or> function <or> method ->bool value(True/False)
we can use conditional statement
           ---------------------- Validation based
                |
              The block will execute only one time
if statment
===========
1. if only 2.if..else 3.if..elif..elif..elif...else

if(condition):
    <True Block>

if(condition):
    <True Block>
else:
    <False Block>

if(condition1):
    <True Block>
elif(condition2):
    <True Block>
elif(condition3):
    <True Block>
..
else:
    <False Block>
```

In [ ]:
```
Write a python program:
    1. create an empty list
    2. read a hostname from <STDIN>
    3. use membership operator - test input host name is exists or not
            3.1 not exists ->Append operation(add input host to list)
    4. display list
```

In [67]:
```
# input_variable = input('prompt message')
#################
```

In [70]:
```
hosts = []
hosts.append('localhost')
h = input('Enter a hostname:')
if(h in hosts):
    print(f'Yes given host {h} already exists')
else:
    hosts.append(h)

hosts
```

```
Enter a hostname:host02
```

Out[70]:
```
['localhost', 'host02']
```

In [ ]:
```
Tasks:
    read a port number from <STDIN>
    test input port range 5001-5999 => initialize app name is TestApp
    test input port range 501-599 ==> initialize app name is demoApp
    |
    default app name is main.app
    |
    display app name and port number
```

In [71]:
```
port = input('Enter a port number:')

if(int(port) >5000 and int(port)<6000):
    app = 'TestApp'
elif(int(port)>500 and int(port)<600):
    app = 'demoApp'
else:
    app = 'main.app'

print(f'App name is:{app} running port number is:{port}')
```

```
Enter a port number:5689
App name is:TestApp running port number is:5689
```

In [73]:
```
port = input('Enter a port number:')

if(int(port) >5000 and int(port)<6000):
    app = 'TestApp'
elif(int(port)>500 and int(port)<600):
    app = 'demoApp'
else:
    app = 'main.app'

print(f'App name is:{app} running port number is:{port}')
```

```
Enter a port number:567
App name is:demoApp running port number is:567
```

In [74]:
```
port = input('Enter a port number:')

if(int(port) >5000 and int(port)<6000):
    app = 'TestApp'
elif(int(port)>500 and int(port)<600):
    app = 'demoApp'
else:
    app = 'main.app'

print(f'App name is:{app} running port number is:{port}')
```

```
Enter a port number:123
App name is:main.app running port number is:123
```

In [76]:
```
v1 = 45
v2 = '46'
int(v2) # type cast to int
```

Out[76]:
```
46
```

In [77]:
```
str(v1) # type cast to str
```

Out[77]:
```
'45'
```

In [78]:
```
print(bool(v1),bool(v2))
```

```
True True
```

In [79]:
```
print(bool(0),bool(0.0),bool(''),bool([]),bool(()),bool({}),bool(None))
```

```
False False False False False False False
```

In [80]:
```
name = input('Enter your name:')
if(len(name) == 0):
    print("Sorry your input is empty")
else:
    print(f"Hello...{name}")
```

```
Enter your name:
Sorry your input is empty
```

In [81]:
```
name = input('Enter your name:')
if(len(name) == 0):
    print("Sorry your input is empty")
else:
    print(f"Hello...{name}")
```

```
Enter your name:raj
Hello...raj
```

In [83]:
```
bool(''),bool(' ')
```

Out[83]:
```
(False, True)
```

In [84]:
```
name = input('Enter your name:')
if(name):
    print(f'Hello...{name}')
else:
    print('Sorry your input is missed')
```

```
Enter your name:
Sorry your input is missed
```

In [85]:
```
name = input('Enter your name:')
if(name):
    print(f'Hello...{name}')
else:
    print('Sorry your input is missed')
```

```
Enter your name:Raj
Hello...Raj
```

In [86]:
```
# Operator + Conditional statement + Looping statements
# ------------------------------------------------------

# Looping statements
#  ->Code block ->execute more than one times
#
#  1. Conditional Style - based on the bool value
#       |-> while
#  2. Collection style - based on the items(collection - str,bytes,list,tuple,dic
#       |-> for

# while loop
# 1st step => initialization
# 2nd step => condition
# 3rd step => arithmetic
# python won't support ++ -- operators

i=0
while(i<5):
    print(f'i value is:{i}')
    i=i+1 # i+=1
```

```
i value is:0
i value is:1
i value is:2
i value is:3
i value is:4
```

In [88]:
```
# for variable in <Collection>:
#         <Code Block>
#
# for in - keywords

for var in 'sales':
    print(f'var value is: {var}')
```

```
var value is: s
var value is: a
var value is: l
var value is: e
var value is: s
```

In [90]:
```
for var in ['sales','prod']:
    print(f'var value is:{var}')
```

```
var value is:sales
var value is:prod
```

In [93]:
```
# python 2.x
# range(5) --->[0,1,2,3,4]

# python 3.x
# range(5) --><class 'range'>(0,5)

for var in range(15):
    print(var)

print("\n")
for var in range(3,15):
    print(var)

print("\n")
for var in range(3,15,2):
    print(var)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14


3
4
5
6
7
8
9
10
11
12
13
14


3
5
7
9
11
13
```

In [96]:
```
while(False):
    print("OK")


while(False):
    print("OK")
else:
    print("Thank you")

t=0
for var in '12345':
    t=t+int(var)
else:
    print(f'Sum of {var} is:{t}')
```

```
Thank you
Sum of 5 is:15
```

In [ ]:
```
>>> for var in '12345':
...     print(var,type(var))
...
1 <class 'str'>
2 <class 'str'>
3 <class 'str'>
4 <class 'str'>
5 <class 'str'>
>>>
```

In [97]:
```
def display():
    print("display list of files")

display()
```

```
display list of files
```

In [100]:
```
def display(f1,f2,f3): # required arguments
    print("display list of files")
    print("file1:",f1,"file2:",f2,"file3:",f3)

display("p1.log","p2.log","p3.log")

#display()
#TypeError: display() missing 3 required positional arguments:
#                                          'f1', 'f2', and 'f3'

#display("test1.c")
#TypeError: display() missing 2 required positional arguments: 'f2' and 'f3'

#display("t1.c","t2.c") ->Error
```

```
display list of files
file1: p1.log file2: p2.log file3: p3.log

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[100], line 10
      5 display("p1.log","p2.log","p3.log")
      6 #display()
      7 #TypeError: display() missing 3 required positional arguments:
      8 #                                          'f1', 'f2', and 'f3'
---> 10 display("test1.c")

TypeError: display() missing 2 required positional arguments: 'f2' and 'f3'
```

In [103]:
```
def display(f1="users.csv"): # default args
    print("list of files")
    print("f1 value=",f1)

display()

display("emp.csv")

display("f1","f2")
```

```
list of files
f1 value= users.csv
list of files
f1 value= emp.csv

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[103], line 9
      5 display()
      7 display("emp.csv")
----> 9 display("f1","f2")

TypeError: display() takes from 0 to 1 positional arguments but 2 were given
```

In [104]:
```
def fx(a1,a2,a3,a4=True,a5):
    print("OK")
```

```
  Cell In[104], line 1
    def fx(a1,a2,a3,a4=True,a5):
                               ^
SyntaxError: non-default argument follows default argument
```

In [105]:
```
def fx(a1,a2,a3,a4,a5="OK",a6="True"):
    print("OK")
```

In [106]:
```
def fx(*a): # Variable length args
    print("List of files")
    print(a,type(a))

fx()
```

```
List of files
() <class 'tuple'>
```

In [107]:
```
fx("f1","f2","f3")
```

```
List of files
('f1', 'f2', 'f3') <class 'tuple'>
```

In [109]:
```
fx(['p1.conf','p2.conf'],('p1.yml','p2.yml'),{"K1":"V1"})
```

```
List of files
(['p1.conf', 'p2.conf'], ('p1.yml', 'p2.yml'), {'K1': 'V1'}) <class 'tuple'>
```

In [110]:
```
fx(app='flask',port=5000) # keyword args
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[110], line 1
----> 1 fx(app='flask',port=5000)

TypeError: fx() got an unexpected keyword argument 'app'
```

In [111]:
```
def fx(**a):
    for var in a:
        print(f'{var} - {a[var]}')
```

In [112]:
```
fx()
```

In [113]:
```
fx(config="/etc/passwd",user="root",db="MysqL")
```

```
config - /etc/passwd
user - root
db - MysqL
```

In [125]:
```
name='raj'
print(name.title(),name.upper())
```

```
Raj RAJ
```

In [ ]:
```
Task:
------
Q1. Given List
------------------
Emp = ['101,raj,sales,pune,1000','102,tom,prod,bglore,2000','103,leo,hr,hyd,3000'
|
|->display emp name(TitleCase) and working City(UpperCase)
|->Calculate Sum of Emp's Cost
|->Display Total Cost at the end of the line.
```

In [114]:
```
s='101,raj,sales,pune,1000'
s.split(',')
```

Out[114]:
```
['101', 'raj', 'sales', 'pune', '1000']
```

In [115]:
```
L = s.split(',')
L
```

Out[115]:
```
['101', 'raj', 'sales', 'pune', '1000']
```

In [122]:
```
help(str.split)
```

```
Help on method_descriptor:

split(self, /, sep=None, maxsplit=-1)
    Return a list of the substrings in the string, using sep as the separator st
ring.

      sep
        The separator used to split the string.

        When set to None (the default value), will split on any whitespace
        character (including \\n \\r \\t \\f and spaces) and will discard
        empty strings from the result.
      maxsplit
        Maximum number of splits (starting from the left).
        -1 (the default value) means no limit.

    Note, str.split() is mainly useful for data that has been intentionally
    delimited.  With natural text that includes punctuation, consider using
    the regular expression module.
```

In [121]:
```
eid,ename,edept,ecity,ecost = s.split(',') # multiple initialization
print(eid,ename,edept,ecity,ecost)
```

```
101 raj sales pune 1000
```

In [129]:
```
Emp = ['101,raj,sales,pune,1000','102,tom,prod,bglore,2000',
       '103,leo,hr,hyd,3000']

total = 0
for var in Emp:
    eid,ename,edept,ecity,ecost=var.split(",")
    print(f'Emp name:{ename.title()}\tWorkingCity:{ecity.upper()}')
    total = total + int(ecost)
else:
    print("-"*40)
    print(f"\tSum of Emp's Cost:{total}")
    print("-"*40)
```

```
Emp name:Raj    WorkingCity:PUNE
Emp name:Tom    WorkingCity:BGLORE
Emp name:Leo    WorkingCity:HYD
----------------------------------------
        Sum of Emp's Cost:6000
----------------------------------------
```

In [ ]:
```
# Tasks
Q2. Given dictionary
------------------------
net_info={"Type":"Ethernet","Onboot":"No","bootproto":"dhcp","defroute":"yes"}

print -> given dict details
|-> dictionary key modification
        Onboot ->Yes
        bootproto ->static
        defroute ->no

|-> dictionary add operation
        IP ->10.20.30.40
        prefix ->24

print -> updated dict content
```

In [133]:
```
net_info={"Type":"Ethernet","Onboot":"No","bootproto":"dhcp",
          "defroute":"yes"}

for var in net_info:
    print(f'{var} - {net_info[var]}')

# dict modification
net_info['Onboot']='yes'
net_info['bootproto']='static'
net_info['defroute']='no'

# dict add operation
net_info['IP']='10.20.30.40'
net_info['prefix']=24

print('\nUpdated network Info')
for var in net_info:
    print(f'{var} - {net_info[var]}')
```

```
Type - Ethernet
Onboot - No
bootproto - dhcp
defroute - yes

Updated network Info
Type - Ethernet
Onboot - yes
bootproto - static
defroute - no
IP - 10.20.30.40
prefix - 24
```

In [134]:
```
s='Type=Ethernet'
d={}

s.split("=")
```

Out[134]:
```
['Type', 'Ethernet']
```

In [135]:
```
k,v = s.split("=")
d[k]=v
d
```

Out[135]:
```
{'Type': 'Ethernet'}
```

In [136]:
```
import time
time.ctime()
```

Out[136]:
```
'Mon Nov 25 14:42:48 2024'
```

In [137]:
```
time.ctime().split()
```

Out[137]:
```
['Mon', 'Nov', '25', '14:43:08', '2024']
```

In [138]:
```
time.ctime().split()[-2]
```

Out[138]:
```
'14:44:12'
```

In [139]:
```
time.sleep(3)
```

In [ ]:
```
Write a python program:
    1. initialize a pin Number(ex: pin=1234)
    2. use while loop - limit is 3
            |-> read a inputPin from <STDIN>
            |-> test inputPin with an existing pin
                    ----------------------------
                        ->Both pin is matched -> Pin is matched - <count>

    3. if 3 inputs are failed - pin is blocked.

-> create an empty dictionary
-> use time as key - user input pin details as value(Success/Failed)
   -------------    ================================================

-> display dict details
```

In [143]:
```
pin = 1234
c = 0
while(c < 3):
    p = input('Enter a pin number:')
    c=c+1
    if(pin == int(p)):
        print(f'Success - pin is matched {c}')
        break # exit from loop

if(int(p) != pin):
    print('pin is blocked')
```

```
Enter a pin number:13132
Enter a pin number:123123
Enter a pin number:12321
pin is blocked
```

In [146]:
```
pin = 1234
c = 0
while(c < 3):
    p = input('Enter a pin number:')
    c=c+1
    if(pin == int(p)):
        print(f'Success - pin is matched {c}')
        break # exit from loop
    else:
        print(f'Failed - pis is not-matched')

if(int(p) != pin):
    print('pin is blocked')
```

```
Enter a pin number:12321
Failed - pis is not-matched
Enter a pin number:12321
Failed - pis is not-matched
Enter a pin number:1234
Success - pin is matched 3
```

In [146]:
```
pin = 1234
c = 0
while(c < 3):
    p = input('Enter a pin number:')
    c=c+1
    if(pin == int(p)):
        print(f'Success - pin is matched {c}')
        break # exit from loop
    else:
        print(f'Failed - pis is not-matched')

if(int(p) != pin):
    print('pin is blocked')
```

```
Enter a pin number:12321
Failed - pis is not-matched
Enter a pin number:12321
Failed - pis is not-matched
Enter a pin number:1234
Success - pin is matched 3
```

In [151]:
```
import time

pin_history = {} # empty dict

pin = 1234
c = 0
while(c < 3):
    p = input('Enter a pin number:')
    c=c+1
    if(pin == int(p)):
        pin_history[time.ctime().split()[-2]]=f'Success - pin is matched{c}'
        time.sleep(2)
        #print(f'Success - pin is matched {c}')
        break # exit from loop
    else:
        pin_history[time.ctime().split()[-2]]=f'Failed - user input pin:{p} is no
        time.sleep(1)
        #print(f'Failed - pin is not-matched')

if(int(p) != pin):
    pin_history[time.ctime().split()[-2]] = f'pin is blocked'
    time.sleep(1)
    #print('pin is blocked')
```

```
Enter a pin number:123456
Enter a pin number:1234
```

In [152]:
```
pin_history
```

Out[152]:
```
{'15:02:57': 'Failed - user input pin:123456 is not-matched',
 '15:03:03': 'Success - pin is matched2'}
```

In [153]:
```
import time

pin_history = {} # empty dict

pin = 1234
c = 0
while(c < 3):
    p = input('Enter a pin number:')
    c=c+1
    if(pin == int(p)):
        pin_history[time.ctime().split()[-2]]=f'Success - pin is matched{c}'
        time.sleep(2)
        #print(f'Success - pin is matched {c}')
        break # exit from loop
    else:
        pin_history[time.ctime().split()[-2]]=f'Failed - user input pin:{p} is no
        time.sleep(1)
        #print(f'Failed - pin is not-matched')

if(int(p) != pin):
    pin_history[time.ctime().split()[-2]] = f'pin is blocked'
    time.sleep(1)
    #print('pin is blocked')
```

```
Enter a pin number:52234
Enter a pin number:43242
Enter a pin number:42422
```

In [154]:
```
pin_history
```

Out[154]:
```
{'15:03:45': 'Failed - user input pin:52234 is not-matched',
 '15:03:50': 'Failed - user input pin:43242 is not-matched',
 '15:03:54': 'Failed - user input pin:42422 is not-matched',
 '15:03:55': 'pin is blocked'}
```

In [155]:
```
import time

#pin_history = {} # empty dict

pin = 1234
c = 0
while(c < 3):
    p = input('Enter a pin number:')
    c=c+1
    if(pin == int(p)):
        pin_history[time.ctime().split()[-2]]=f'Success - pin is matched {c}'
        time.sleep(2)
        #print(f'Success - pin is matched {c}')
        break # exit from loop
    else:
        pin_history[time.ctime().split()[-2]]=f'Failed - user input pin:{p} is no
        time.sleep(1)
        #print(f'Failed - pin is not-matched')

if(int(p) != pin):
    pin_history[time.ctime().split()[-2]] = f'pin is blocked'
    time.sleep(1)
    #print('pin is blocked')
```

```
Enter a pin number:12312
Enter a pin number:1234
```

In [156]:
```
pin_history
```

Out[156]:
```
{'15:03:45': 'Failed - user input pin:52234 is not-matched',
 '15:03:50': 'Failed - user input pin:43242 is not-matched',
 '15:03:54': 'Failed - user input pin:42422 is not-matched',
 '15:03:55': 'pin is blocked',
 '15:04:51': 'Failed - user input pin:12312 is not-matched',
 '15:04:55': 'Success - pin is matched2'}
```

In [158]:
```
L = []

def fx(a):
    if(a > 100):
        return a+500

for var in [50,60,70,120,500]:
    r = fx(var)
    L.append(r)

print(L[:3])
print(L)
```

```
[None, None, None]
[None, None, None, 620, 1000]
```

In [ ]:
```
# File Handling

1. Read data from <FILE> ->python ->display to monitor
    fobj = open('inputFile','mode')
    |
    fobj.read() ->'str' (or) fobj.readlines() ->[list]
    |
    fobj.close()


2. python ->create a newFile and write Data to File
   wobj = open('resultFile','w')
   wobj.write('SingleString\n')
   wobj.close()

3. Read data from <FILE> ->python ->create a newFile and write Data to File
     fobj = open('inputFile','mode')
     fobj.read() ->'str' (or) fobj.readlines() ->[list]
     wobj = open('resultFile','w')
     wobj.write('SingleString\n')
     wobj.close()
     fobj.close()
```

In [159]:
```
open('emp.csv','r')
```

Out[159]:
```
<_io.TextIOWrapper name='emp.csv' mode='r' encoding='cp1252'>
```

In [160]:
```
fobj = open('emp.csv','r')
fobj.read()
```

Out[160]:
```
'eid,ename,edept,eplace,ecost\n101,raj,sales,pune,1000\n102,leo,prod,bglore,2000
\n103,paul,HR,chennai,3000\n104,anu,hr,hyderabad,4000\n456,kumar,,bglore,\n105,z
ion,Hr,mumbai,5000\n106,bibu,sales,bglore,1450\n107,theeb,sales,noida,4590\n102,
leo,prod,bglore,2000\n104,anu,hr,hyderabad,4000\n'
```

In [161]:
```
fobj = open('emp.csv','r')
fobj.readlines()
```

Out[161]:
```
['eid,ename,edept,eplace,ecost\n',
 '101,raj,sales,pune,1000\n',
 '102,leo,prod,bglore,2000\n',
 '103,paul,HR,chennai,3000\n',
 '104,anu,hr,hyderabad,4000\n',
 '456,kumar,,bglore,\n',
 '105,zion,Hr,mumbai,5000\n',
 '106,bibu,sales,bglore,1450\n',
 '107,theeb,sales,noida,4590\n',
 '102,leo,prod,bglore,2000\n',
 '104,anu,hr,hyderabad,4000\n']
```

In [162]:
```
fobj = open('emp.csv','r')
s = fobj.read()
fobj.close()
print(s)
```

```
eid,ename,edept,eplace,ecost
101,raj,sales,pune,1000
102,leo,prod,bglore,2000
103,paul,HR,chennai,3000
104,anu,hr,hyderabad,4000
456,kumar,,bglore,
105,zion,Hr,mumbai,5000
106,bibu,sales,bglore,1450
107,theeb,sales,noida,4590
102,leo,prod,bglore,2000
104,anu,hr,hyderabad,4000
```

In [163]:
```
fobj = open('emp.csv','r')
L = fobj.readlines()
fobj.close()

L[-3:] # last 3lines
```

Out[163]:
```
['107,theeb,sales,noida,4590\n',
 '102,leo,prod,bglore,2000\n',
 '104,anu,hr,hyderabad,4000\n']
```

In [ ]:
```
'w' - create a newFile and write
'a' - append mode - won't overwrite an existing data
```

In [166]:
```
wobj = open('r1.log','w')
wobj.write('data1\n')

s1='index.html'
code=3435

wobj.write(f'file name is:{s1}\n')
wobj.write(f'code:{code}\n')
# wobj.write(code) TypeError: write() argument must be str, not int
wobj.write(str(code)) # typecast to str
```

Out[166]: 10

In [169]:
```
#wobj.write('data1','data2')
#TypeError: TextIOWrapper.write() takes exactly one argument (2 given)

wobj.write('data1'+'data2\n')
wobj.write('Type'+'='+'Ethernet\n')
wobj.write('leo'+','+'sales'+','+'pune'+','+str(1232.21)+'\n')
```

Out[169]: 23

In [170]:
```
wobj.close()
```

In [171]:
```
fobj = open('r1.log')
s =fobj.read()
fobj.close()
print(s)
```

```
data1
file name is:index.html
code:3435
data1data2
Type=Ethernet
leo,sales,pune,1232.21
```

In [178]:
```
import time

wobj = open('pin_history.log','a') # append operation

pin = 1234
c = 0
while(c < 3):
    p = input('Enter a pin number:')
    c=c+1
    if(pin == int(p)):
        print(f'Success - pin is matched {c}') # display to monitor
        wobj.write(f'Success - pin is matched {c} ')
        wobj.write(f'pin Entry time is:{time.ctime()}\n')
        break # exit from loop
    else:
        wobj.write(f'Failed - pin {p} is Not-Matched')
        wobj.write(f'pin Entry time is:{time.ctime()}\n')

if(int(p) != pin):
    print('pin is blocked')
    wobj.write(f'pin is blocked entry time is:{time.ctime()}\n\n')

wobj.close()
```

```
Enter a pin number:4532
Enter a pin number:2345
Enter a pin number:1234
Success - pin is matched 3
```

In [ ]:
```
OOPs
 - class  - type
 - object - entity <or> instance
 - method - function
 - inheritance - class - parent<-->child
```

In [ ]:
```
class - blueprint of an object
                          |->real value - instance
How to create a class?

class <className>:
    <attribute>
    <attribute>

<className>.<attribute> <== we can access class attributes
```

In [180]:
```
class Fsinfo:
    fname='index.html'
    fsize='2KB'

print(Fsinfo)
print(Fsinfo.fname,Fsinfo.fsize)
```

```
<class '__main__.Fsinfo'>
index.html 2KB
```

In [182]:
```
fname
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[182], line 1
----> 1 fname

NameError: name 'fname' is not defined
```

In [183]:
```
Fsinfo.findex
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[183], line 1
----> 1 Fsinfo.findex

AttributeError: type object 'Fsinfo' has no attribute 'findex'
```

In [185]:
```
# Python user defined class - mutable (we can add,modify,delete)
# ---------------------------------------------------------------

class Fsinfo:
    fname = 'index.html'

print(Fsinfo.fname)

Fsinfo.fname='display.html' # modification
print(Fsinfo.fname)
```

```
index.html
display.html
```

In [186]:
```
print(Fsinfo.fname)
```

```
display.html
```

In [187]:
```
Fsinfo.index = 4567 # we can add new attribute
```

In [188]:
```
print(Fsinfo.index)
```

```
4567
```

In [ ]:
```
del(Fsinfo.index) # we can delete an existing attribute
```

In [ ]:

```
 +----------------------+
 | [ ]   |  |   [ ] [ ]  |
 |                       | <== blueprint sheet   - class
 |     (white)           |
 +-----------|   |-------+


  |            |
 [House-1]    [House-2] ... [House-n] <== real object/entity
   0x124         0x33         0x23    <== address/memory
   1st main     2nd main       nth main

>>> class Fsinfo:
...     fname='abc.log'
...
>>> Fsinfo
<class '__main__.Fsinfo'>
>>> Fsinfo()
<__main__.Fsinfo object at 0x0000017C936606E0>
>>> Fsinfo()
<__main__.Fsinfo object at 0x0000017C934C25D0>
>>>
>>> obj1 = Fsinfo()
>>> obj2 = Fsinfo()
>>> print(type(obj1),type(obj2))
<class '__main__.Fsinfo'> <class '__main__.Fsinfo'>
>>> print(type(10),type(20))
<class 'int'> <class 'int'>
>>> print(type(int),type(Fsinfo))
<class 'type'> <class 'type'>
>>>
>>> class box:
...     bid=101
...
>>> box.bid
101
>>> obj1 = box()
>>> obj2 = box()
>>> obj1.bid
101
>>> obj2.bid
101
>>> obj1.bid='B505' # object based initialization
>>> obj1.bid
'B505'
>>> obj2.bid
101
>>> box.bid=202 # using class name we can modify an existing attribute
>>> box.bid
202
>>> obj1.bid
'B505'
>>> obj2.bid
202
>>>
>>> obj2.bid='B909' # object based initialization
>>>
```

```
>>> obj1.bid
'B505'
>>> obj2.bid
'B909'
>>>
```

In [189]:
```
def f1():
    print("OK-1")

class box:
    def f2():
        print("OK-2")

obj = box()
print(type(f1),type(obj.f2))
```

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

In [190]:
```
s='abc'
L=[]
print(s.upper()) # upper method
L.append('Data1') # append method
print(L)
```

```
ABC
['Data1']
```

In [193]:
```
class box:
    def f2():
        print("OK-2")
obj = box()
#obj.f2() TypeError: box.f2() takes 0 positional arguments but 1 was given

# obj.f2() -> f2(obj)
#  ======     -------
# obj.f2(10,20,30) ->f2(obj,10,20,30)
# =================    ----------------
```

In [194]:
```
class box:
    var=120
    def f2(self):
        print("obj name:",self)
        print(self.var)
        self.var = 500 # object based initialization
obj = box()
print(obj.var) # 120
obj.f2() # f2(obj)
print(obj)
```

```
120
obj name: <__main__.box object at 0x000001C91A3F4F10>
120
<__main__.box object at 0x000001C91A3F4F10>
```

In [195]:
```
obj.var
```

Out[195]:
```
500
```