In [ ]:
```
class - object


python 2.x - type - class

|
python 3.x - class <--> object

10
```

In [ ]:
```
numbers
 |->int float complex
 |->bool
 |->NoneType
 |->Collection
          |-> str bytes list tuple dict set

Seq -> index based - str,bytes,list,tuple
mapping ->key:value // dict
                       set
```

In [ ]:
```
File
  |->file_index,file_name,file_util,file_status
      101        'emp.csv'     98.34    True

Fname = 'emp.csv'  <==
Findex = 101
file_util=98.34
file_status = True

list - Collection of items
list_name = [<list of items>]

File_info = [Fname,Findex,file_util,file_status]  # list - mutable

File_info = (Fname,Findex,file_util,file_status)  # tuple - immutable

File_info = {'K1':Fname,'K2':Findex,'K3':file_util,'K4':file_status} # dict - m
```

In [3]:
```
n=50
print(n)
n="Hello"
print(n)
print(N)
```

```
50
Hello

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[3], line 5
      3 n="Hello"
      4 print(n)
----> 5 print(N)

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

In [4]:
```
s={10,20,10,10,10,20,30,'K1','K2','V1','K1','K2'}
print(s)
```

```
{'V1', 20, 30, 'K2', 10, 'K1'}
```

In [5]:
```
d={'K1':10,'K2':20,'K1':30,'K3':20}
print(d)
```

```
{'K1': 30, 'K2': 20, 'K3': 20}
```

In [ ]:
```
1. procedure style - direct
     |->topic definition
     |->Syntax
     |->Existing Examples
     |->Activity

2. oops - style
    |->class <--> object

3. Functional style program - singleline Code
```

In [12]:
```
Fname = "emp.csv"
Findex = 4677
print(Fname)
print("Fname") # same as print('Fname')
print("File name is:Fname")

# How to combine user defined string and named variable ?

print("File name is:",Fname,"File Index:",Findex)
print("File name is:%s File Index:%d"%(Fname,Findex))
print("File name is:{} File Index:{}".format(Fname,Findex)) # python 2.7
print(f"File name is:{Fname} File Index:{Findex}")
```

```
emp.csv
Fname
File name is:Fname
File name is: emp.csv File Index: 4677
File name is:emp.csv File Index:4677
File name is:emp.csv File Index:4677
File name is:emp.csv File Index:4677
```

In [14]:
```
print('data1\ndata2\ndata3\ndata4')
print('') # empty line
print('''data1
data2
data3
data4''')
```

```
data1
data2
data3
data4

data1
data2
data3
data4
```

In [15]:
```
# Using multiline string - display file details
Fname = "emp.csv"
Findex = 3456
Fuitls = 98.52
Fstatus = True
print(f'''File name is:{Fname}
{Fname} index number is:{Findex}
{Fname} Utilization is:{Fuitls}
{Fname} open status is:{Fstatus}
---------------------------------------''')
```

```
File name is:emp.csv
emp.csv index number is:3456
emp.csv Utilization is:98.52
emp.csv open status is:True
---------------------------------------
```

In [16]:
```
Fname = input("Enter a file name:")
print("Input File name is:",Fname)
```

```
Enter a file name:/var/log/test.log
Input File name is: /var/log/test.log
```

In [18]:
```
n = input('Enter n value:')
print(n)
n
```

```
Enter n value:56
56
```

Out[18]:
```
'56'
```

In [19]:
```
'A'+'B'
```

Out[19]:
```
'AB'
```

In [20]:
```
'10'+'20'
```

Out[20]:
```
'1020'
```

In [21]:
```
10+20
```

Out[21]:
```
30
```

In [22]:
```
'A'+10
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[22], line 1
----> 1 'A'+10

TypeError: can only concatenate str (not "int") to str
```

In [23]:
```
n+n
```

Out[23]:
```
'5656'
```

In [24]:
```
# type(Value) (or) type(namedVariable)
print(type(56),type('56'))
```

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

In [26]:
```
n=56
print(type(n),n)
str(n) # type cast to string
```

```
<class 'int'> 56
```

Out[26]:
```
'56'
```

In [27]:
```
print("n value is:"+str(n))
```

```
n value is:56
```

In [28]:
```
s='56'
print(type(s))
int(s) # typecast to int
```

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

Out[28]:
```
56
```

In [29]:
```
cost="4563.23"
float(cost) # typecast to float
```

Out[29]:
```
4563.23
```

In [30]:
```
int(cost)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[30], line 1
----> 1 int(cost)

ValueError: invalid literal for int() with base 10: '4563.23'
```

In [31]:
```
int(float(cost))
```

Out[31]:
```
4563
```

In [33]:
```
s1='56'
s2='abc'
```

In [34]:
```
s1.isdigit()
```

Out[34]:
```
True
```

In [35]:
```
s2.isdigit()
```

Out[35]:
```
False
```

In [32]:
```
#int,float
# => + - * / // **  (input:int,float) ->int,float
print(10+2.5)

# == != < <= > >=  (input:int,float) ->bool True False

# string
#  + *     ->str
#  == !=   ->bool

# logical operators -> bool
# int,float,str
and
or
not
```

```
12.5
```

In [36]:
```
print('Hello'+'python'+str(3.13))
```

```
Hellopython3.13
```

In [38]:
```
print("Hello\n"*5)
```

```
Hello
Hello
Hello
Hello
Hello
```

In [40]:
```
print("Hello\n",2*5)
```

```
Hello
 10
```

In [ ]:
```
# In python any expression ->bool
#           any method/function ->bool
#          validate inputdata ->bool
# ------------------------------------//use conditional statement
```

In [ ]:
```
# python supports infix type of expression
Operand1  <operator>  Operand2  - infix
```

In [ ]:
```
Write a python program:
    read a shell name from <STDIN>

    test - input shell is bash - initialize a profile filename /etc/bashrc
    test - input shell is ksh  - initialize a profile filename /etc/kshrc
    test - input shell is csh  - initialize a profile filename /etc/cshrc
    |->default shell /bin/nologin profile filename is /etc/profile

    display shell name and profile filename
```

In [45]:
```
shell_name = input("Enter a shell name:")
if(shell_name == 'bash'):
    fname = '/etc/bashrc'
elif(shell_name == 'ksh'):
    fname = '/etc/kshrc'
elif(shell_name == 'csh'):
    fname = '/etc/cshrc'
else:
    shell_name = '/bin/nologin'
    fname = '/etc/profile'

print(f'Shell name is:{shell_name} profile filename:{fname}')
```

```
Enter a shell name:bash
Shell name is:bash profile filename:/etc/bashrc
```

In [46]:
```
print(bool(0),bool(0.0),bool(''),bool([]),bool(()),bool({}),bool(None))
```

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

In [47]:
```
len('abc')
```

Out[47]:
```
3
```

In [48]:
```
len('')
```

Out[48]:
```
0
```

In [49]:
```
login_name = input('Enter a login name:')

if(len(login_name) == 0):
    print('Input is missed - empty')
else:
    print('Hello',login_name)
```

```
Enter a login name:
Input is missed - empty
```

In [52]:
```
login_name = input('Enter a login name:')
if(login_name):
    print('Hello',login_name)
else:
    print('Input is empty/missing')
```

```
Enter a login name:
Input is empty/missing
```

In [53]:
```
login_name = input('Enter a login name:')
if(login_name):
    print('Hello',login_name)
else:
    print('Input is empty/missing')
```

```
Enter a login name:Raj
Hello Raj
```

In [54]:
```
print(bool(''),bool('Raj'))
```

```
False True
```

In [55]:
```
# python looping statatments
#   |->Conditional style loop - while - life time is depend on bool True
#   |->Collection style loop - for   - life time is depends on data(collection)
#      ==========
#           |->str,bytes,list,tuple,dict,set

i=0
while(i<5):
    print('i value is:',i)
    i=i+1 # i+=1 OK
```

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

In [56]:
```
while(False):
    print("Hello")
```

In [57]:
```
i=0
while(i<5):
    print('i value is:',i)
    i=i+1 # i+=1 OK
else:
    print('-'*15)
    print('Footer message')
    print('-'*15)
```

```
i value is: 0
i value is: 1
i value is: 2
i value is: 3
i value is: 4
---------------
Footer message
---------------
```

In [58]:
```
s='root:x:bin5 ^g'
len(s)
```

Out[58]:
```
14
```

In [59]:
```
for var in 'python':
    print('var value is:',var)
```

```
var value is: p
var value is: y
var value is: t
var value is: h
var value is: o
var value is: n
```

```
# other programming
#  for loop                  in Python
for(i=0;i<5;i++){}  --->  for i in range(5):

for(i=3;i<15;i++){} --->  for i in range(3,15):

for(i=3;i<15;i+4){} ------>for i in range(3,15,4):
```

In [61]:
```
for var in 'python':
    print(f'var value is:{var}')
else:
    print('-'*50)
    print('\tThank you')
    print('-'*50)
```

```
var value is:p
var value is:y
var value is:t
var value is:h
var value is:o
var value is:n
--------------------------------------------------
        Thank you
--------------------------------------------------
```

In [ ]:
```
str - collection of chars ->  ''  =>type('') -><class 'str'>

bytes - collection of ASCII -> b'' =>type(b'') -><class 'bytes'>
```

In [64]:
```
s='Ab5'
for var in s:
    print(var,end = ' ')
print('') # empty line
s=b'Ab5'
for var in s:
    print(var,end = ' ')
```

```
A b 5
65 98 53
```

In [65]:
```
va = 'ab'
vb = b'ab'
print(len(va),len(vb))
```

```
2 2
```

In [66]:
```
# membership operators
#  in    not in  (inputs: str,bytes,list,tuple,dict,set) ->bool

':' in 'root:x'
```

Out[66]:
```
True
```

In [ ]:
```
'pattern_string' in input_collection ->bool
```

In [67]:
```
'a' in 'leo,sales,1000'
```

Out[67]:
```
True
```

In [68]:
```
'ae' in 'leo,sales,1000'
```

Out[68]:
```
False
```

In [69]:
```
'sales' in 'leo,sales,1000'
```

Out[69]:
```
True
```

In [70]:
```
L=['D1','D2','D3']
'D2' in L
```

Out[70]:
```
True
```

In [71]:
```
if('D2' in L):
    print('Yes exists')
else:
    print('Not exists')
```

```
Yes exists
```

In [ ]:
```
function --> functionCall() print() input() type() del()
Vs
method ----> object.functionCall()  "{}".format()
```

In [72]:
```
"abc".upper()
```

Out[72]:
```
'ABC'
```

In [73]:
```
"ABC".lower()
```

Out[73]:
```
'abc'
```

In [75]:
```
[].upper()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[75], line 1
----> 1 [].upper()

AttributeError: 'list' object has no attribute 'upper'
```

In [ ]:
```
class myclass: ---------------->class str:
    def method1(self):              def upper(self):
        pass                                return ...
    def method2(self):              def lower(self):
        pass                                ...
                                          return
```

In [77]:
```
# help(str)
help(str.upper)
```

```
Help on method_descriptor:

upper(self, /)
    Return a copy of the string converted to uppercase.
```

In [ ]:
```
type(<variable>) ->determine type  => help(type.attribute)

help(json.dumps)
```

In [78]:
```
help(str.isupper)
```

```
Help on method_descriptor:

isupper(self, /)
    Return True if the string is an uppercase string, False otherwise.

    A string is uppercase if all cased characters in the string are uppercase
and
    there is at least one cased character in the string.
```

In [79]:
```
import os
help(os.listdir)
```

```
Help on built-in function listdir in module nt:

listdir(path=None)
    Return a list containing the names of the files in the directory.

    path can be specified as either str, bytes, or a path-like object.  If pa
th is bytes,
      the filenames returned will also be bytes; in all other circumstances
      the filenames returned will be str.
    If path is None, uses the path='.'.
    On some platforms, path may also be specified as an open file descripto
r;\
      the file descriptor must refer to a directory.
      If this functionality is unavailable, using it raises NotImplementedErr
or.

    The list is in arbitrary order.  It does not include the special
    entries '.' and '..' even if they are present in the directory.
```

In [ ]:
```
list,dict - mutable
            - add/modify/delete
```

In [80]:
```
L=[] # empty list

L.append(15)
L.append(3.42)
L.append(True)
L.append("data1")
L
```

Out[80]: [15, 3.42, True, 'data1']

In [81]:
```
L.insert(1,"data2")
L
```

Out[81]: [15, 'data2', 3.42, True, 'data1']

In [82]:
```
# How to modifying an existing index
# ListName[old_index] = updatedValue
L[1]='iris'
L
```

Out[82]: [15, 'iris', 3.42, True, 'data1']

In [83]:
```
# How to delete nth data from given list?
# Listname.pop() ->return last item Vs Listname.pop(index) ->return removedNth
L.pop()
```

Out[83]: 'data1'

In [84]:
```
L
```

Out[84]: [15, 'iris', 3.42, True]

In [85]:
```
L.pop(1)
```

Out[85]: 'iris'

In [86]:
```
L
```

Out[86]: [15, 3.42, True]

In [87]:
```
s='python'
s.upper()
```

Out[87]: 'PYTHON'

In [88]:
```
s
```

Out[88]: 'python'

In [89]:
```
s=s.upper()
s
```

Out[89]:
```
'PYTHON'
```

In [ ]:
```
Write a python program:
    create an empty list
    |
    use while loop - limit is 5
          ->read a dataset filename from <STDIN>
          ->append input dataset to an existing list
    |
    use for loop - display list of all the sets one by one
    |
    display total no.of datasets
```

In [90]:
```
files=[] # empty list

c=0
while(c <5):
    fname = input('Enter a filename:')
    files.append(fname) # adding input data to an existing list
    c+=1

print('') # empty line
for var in files: # iterate list of items one by one
    print(var)
else:
    print(f"Total no.of datasets:{len(files)}")
```

```
Enter a filename:prod.csv
Enter a filename:emp.csv
Enter a filename:dbs.csv
Enter a filename:test.csv
Enter a filename:health.csv

prod.csv
emp.csv
dbs.csv
test.csv
health.csv
Total no.of datasets:5
```

In [ ]:
```
dict - collection - mutable
        |->key:value //data
           ----======
```

In [91]:
```
d={'DB':'mysql','port':3306,'proto':'https'}
print(d)
```

```
{'DB': 'mysql', 'port': 3306, 'proto': 'https'}
```

In [92]:
```
print(d['DB'])
```

```
mysql
```

In [93]:
```
len(d)
```

Out[93]:
```
3
```

In [ ]:
```
# To add new data to an existing dict
#   --------------
#   dictName[NewKey] = Value
```

In [94]:
```
d['confg'] = '/var/log/mysql.log'
d['ip'] = '127.0.0.1'
```

In [95]:
```
print(d)
```

```
{'DB': 'mysql', 'port': 3306, 'proto': 'https', 'confg': '/var/log/mysql.lo
g', 'ip': '127.0.0.1'}
```

In [96]:
```
# To modify  an existing data from dict
#   ---------------------------------
#   dictName[OldKey] = Updated_Value

d['proto'] = 'file://'
print(d)
```

```
{'DB': 'mysql', 'port': 3306, 'proto': 'file://', 'confg': '/var/log/mysql.lo
g', 'ip': '127.0.0.1'}
```

In [ ]:
```
# list ->  listName.pop() ->remove last index ;
#          Listname.pop(index) ->removenth index

# dict -> dictName.pop('OldKey') ->remove the value
```

In [97]:
```
d.pop('port')
```

Out[97]:
```
3306
```

In [98]:
```
d
```

Out[98]:
```
{'DB': 'mysql',
 'proto': 'file://',
 'confg': '/var/log/mysql.log',
 'ip': '127.0.0.1'}
```

In [99]:
```
d={}
d['K1']='V1'
d.setdefault('K2','V2') # adding new data to an existing dict
d
```

Out[99]:
```
{'K1': 'V1', 'K2': 'V2'}
```

In [102]:
```
# How to get/fetch data from dict ?
#
print(d['K1'])
print(d.get('K1'))

print(d.get('Kx'))
print(d['Kx'])
```

```
V1
V1
None

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[102], line 4
      2 print(d.get('K1'))
      3 print(d.get('Kx'))
----> 4 print(d['Kx'])

KeyError: 'Kx'
```

In [103]:
```
print(L)
d
```

```
[15, 3.42, True]
```

Out[103]:
```
{'K1': 'V1', 'K2': 'V2'}
```

In [105]:
```
for var in L:
    print(var)
print("\n\n")
for var in d:
    print(var)
```

```
15
3.42
True



K1
K2
```

In [106]:
```
for var in d:
    print(var,d[var])
```

```
K1 V1
K2 V2
```

In [108]:
```
d.keys()
d.values()
```

Out[108]:
```
dict_values(['V1', 'V2'])
```

In [ ]:
```
Login:<input type='text' name='n1'>  --> Login:|_tom__|  ==>{'n1':'tom'}
```

In [ ]:
```
# List , Tuple ,Dict

L=[10,[20,30,40],(50,60,70),{"K1":80,"K2":"V1"}]
    | ----------  ---------  =================
    |   |_1st        |_2nd     |__3rd index
    0th
```

In [116]:
```
L=[10,[20,30,40],(50,60,70),{"K1":80,"K2":"V1"}]
print(L)
print(L[0])
print(L[1])
print(L[1][0])
print(L[2])
print(L[2][0])
print(L[-1])
print(L[-1]['K1'])
```

```
[10, [20, 30, 40], (50, 60, 70), {'K1': 80, 'K2': 'V1'}]
10
[20, 30, 40]
20
(50, 60, 70)
50
{'K1': 80, 'K2': 'V1'}
80
```

In [117]:
```
T=([],[]) # tuple of list
print(len(T))
print(type(T))
```

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

In [118]:
```
type(T[0])
```

Out[118]:
```
list
```

In [119]:
```
T[0].append("D1")
T[0].append("D2")
T[0].append("D3")
T[0].append("D4")
T[0].append("D5")
```

In [120]:
```
print(len(T))
print(type(T))
```

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

In [121]:
```
T
```

Out[121]:
```
(['D1', 'D2', 'D3', 'D4', 'D5'], [])
```

In [124]:
```
# T[0]="Dx"

T[0][0]="Dx"
T
```

Out[124]:
```
(['Dx', 'D2', 'D3', 'D4', 'D5'], [])
```

In [ ]:
```
emp1 = {'eid':101,'ename':'Raj','edept':'sales'} # 1D
# |
#
emp2 = {'eid':[],'ename':(),'edept':{}} # MD
#        -------   ---------   --------
#      dict of list  dict of tuple  |->dict of dict
```

In [125]:
```
emp = {'eid':[101,102,103],'ename':['Raj','Leo','Tom'],
       'edept':['DBA','QA','admin']}   # dict of list
```

In [128]:
```
print(emp['eid'])
print(emp['ename'])
```

```
[101, 102, 103]
['Raj', 'Leo', 'Tom']
```

In [129]:
```
emp['eid'][1] # L=[{'eid':101}] <-- L[0]['eid']
```

Out[129]:
```
102
```

In [130]:
```
d={'K1':101,'K2':[102,103,{'K1':[10,20,30,{'K1':(1,2,3)}]}],'K3':{'K1':{'K1':['
print(d)
```

```
{'K1': 101, 'K2': [102, 103, {'K1': [10, 20, 30, {'K1': (1, 2, 3)}]}], 'K3':
{'K1': {'K1': ['V1', 'V2']}}}
```

In [131]:
```
import pprint
pprint.pprint(d)
```

```
{'K1': 101,
 'K2': [102, 103, {'K1': [10, 20, 30, {'K1': (1, 2, 3)}]}],
 'K3': {'K1': {'K1': ['V1', 'V2']}}}
```

In [137]:
```
d['K2'][2]['K1'][3]['K1'][1]
```

Out[137]: 2

In [139]:
```
yum_repo = ['https','creterepo','ansible','ruby']
apt_repo = ['apache2','apt-get','ansible','ruby','java']

print(set(yum_repo)) # typecast to set
set(apt_repo) # typecast to set
```

```
{'https', 'ruby', 'creterepo', 'ansible'}
```

Out[139]: {'ansible', 'apache2', 'apt-get', 'java', 'ruby'}

In [140]:
```
set(yum_repo)|set(apt_repo)
```

Out[140]: {'ansible', 'apache2', 'apt-get', 'creterepo', 'https', 'java', 'ruby'}

In [144]:
```
set(yum_repo).union(set(apt_repo))
```

Out[144]: {'ansible', 'apache2', 'apt-get', 'creterepo', 'https', 'java', 'ruby'}

In [143]:
```
set(yum_repo)&set(apt_repo) # intersection
```

Out[143]: {'ansible', 'ruby'}

In [145]:
```
open('E:\\emp.csv')
```

Out[145]: <_io.TextIOWrapper name='E:\\emp.csv' mode='r' encoding='cp1252'>

In [146]:
```
fobj = open("E:\\emp.csv")
fobj.read()
```

Out[146]: '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,sales,bglore,3000\n105,zion,Hr,mumbai,5000\n106,bibu,sales,bglore,1450\n107,theeb,sales,noida,4590\n108,bibu,sales,bglore,5000\n113,kumar,prod,hyderabad,5423DATA'

In [159]:
```
fobj = open("E:\\emp.csv")
#help(fobj.read)
#help(fobj.readlines)
L = fobj.readlines()
for var in L:
    var=var.strip() # remove \n
    print(var)
```

```
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,sales,bglore,3000
105,zion,Hr,mumbai,5000
106,bibu,sales,bglore,1450
107,theeb,sales,noida,4590
108,bibu,sales,bglore,5000
113,kumar,prod,hyderabad,5423DATA
```

In [160]:
```
for var in L:
    var=var.strip()
    if('sales' in var):
        print(var)
```

```
101,raj,sales,pune,1000
456,kumar,sales,bglore,3000
106,bibu,sales,bglore,1450
107,theeb,sales,noida,4590
108,bibu,sales,bglore,5000
```

In [164]:
```
def f1(a1,a2):
    print(a1,a2)

# f1()
# f1(10)
f1(10,20)
#f1(10,20,30)
```

```
10 20
```

In [168]:
```
def f2(a1=100,a2=True):
    print(a1,a2)

f2()
f2("data1")
f2("data1","data2")
f2(10,20,30)
```

```
100 True
data1 True
data1 data2

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[168], line 7
      5 f2("data1")
      6 f2("data1","data2")
----> 7 f2(10,20,30)

TypeError: f2() takes from 0 to 2 positional arguments but 3 were given
```

In [169]:
```
def f1(a1,a2,a3,a4=True,a5):
    print("OK")
```

```
  Cell In[169], line 1
    def f1(a1,a2,a3,a4=True,a5):
                               ^
SyntaxError: non-default argument follows default argument
```