In [ ]:
```
# package
# functional style program
# oops
file:ab.py   ------------------>   file: p1.py
-----------                        ===============
var=10                               import ab
def fx():                            print(ab.var)
    print("Hello")                   ab.fx()
--------------------               ================

import sys
sys.path.append("external filepath")
import ab
ab.fx()

env -> PYTHONPATH="external filepath"
-------------------------------------------------------------------------
```

In [ ]:
```
package
=========
project/m1.py m2.py m3.py ... m50.py

import m1,m2,m3,...m50
(or)
import m1
import m2
import m3
..
import m50
```

In [ ]:
```
# Commandline steps
# --------------------
# STEP 1: create a newdirectory (mkdir <Dir>)
# STEP 2: collect list of .py files into directory (cp..)
# STEP 3: create a special file __init__.py (vi __init__.py  / notepad )
# STEP 4: copy external symbols to __init__.py
#
#         from module import members (or) from module import *
#
# STEP 5: test your directory -> import directoryname
```

In [ ]:
```
file: ab.py                                  ------>   file: p1.py
------------                                          -----------------
var=10                                                 import ab
def fx():                                              port=8080
    print("Hello")                                     print(ab.var) Vs print(var)
----------------------                                 print(port)      ----------
symbol/dict table                                      ab.fx()              |__Error
-------------------                                    -------------------
__main__.var|  10                                 __main__.port | 8080
 ----------------------                        -------------------------
__main__.fx | 0x13345                                  ab.var   | 10
                                              -------------------------
                                                       ab.fx    | 0x12345
```

In [ ]:
```
root@host~]# pwd
/root
root@host~]# ls
a.txt b.txt
root@host~]# cat a.txt
Success                                                port=8080
root@host~]# cat b.txt                                 print(port)
Success                                                import ab
root@host~]# cat /etc/passwd ------------------------  print(ab.var)
Success
root@host~]# cat passwd -------------------------------print(var) ->Error
Error No Such File

root@host~]# cp /etc/passwd .   ================> from module import <datamember>
root@host~]# cat passwd                            from ab import var
Success                                            print(var) -->10
```

In [ ]:
```
project/m1.py m2.py
       /L1/m3.py
       /L1/L2/m4.py
                 |___ def f1
                 |___ def f2
from .project import m1,m2
from .project.L1 import m3
from .project.L1.L2.m4 import f2 <===
from .project.L1.L2.L3.L4 import Workbookoperation as opt

from .project.L1 import *

import project
project.f2()
 ------------------------------------------------------------------------
import subprocess
subprocess.Popen

import subprocess as sp
sp.Popen()        --
```

In [ ]:
```
import filename
        |__ sys.path -> [  ] --> where filename is located?
        |__ filename.py ==> filename.pyc
        |__ Folder/__init.__py ==> __init.__pyc  sales.pyc prod.pyc
```

In [1]:
```
# Functional style programming
# ----------------------------
L=[]
for var in range(5): # [0,1,2,3,4]
    r=var+100
    L.append(r)

L
```

Out[1]:
```
[100, 101, 102, 103, 104]
```

In [2]:
```
'''
[final_value for iterable]
                ---->--
  --<----         (1)
    (2)
'''

[var+100 for var in range(5)]
```

Out[2]:
```
[100, 101, 102, 103, 104]
```

In [3]:
```
L=[]
for var in range(10):
    if(var >5):
        r=var+100
        L.append(r)
    else:
        r=var+500
        L.append(r)
L
```

Out[3]:
```
[500, 501, 502, 503, 504, 505, 106, 107, 108, 109]
```

In [4]:
```
[var+100 if var >5 else var+500 for var in range(10)]
```

Out[4]:
```
[500, 501, 502, 503, 504, 505, 106, 107, 108, 109]
```

In [6]:
```
[var for var in open("C:\\users\\karthikeyan\\emp.csv").readlines()]
```

Out[6]:
```
['101,ram,sales,pune,10000\n',
 '203,arun,prod,bglore,2000\n',
 '304,vijay,QA,chennai,30000\n',
 '545,xerox,sales,mumbai,34234\n',
 '456,anu,sales,noida,56780']
```

In [7]:
```
d={}
d['CSV']=[var for var in open("C:\\users\\karthikeyan\\emp.csv").readlines()]
d
```

Out[7]:
```
{'CSV': ['101,ram,sales,pune,10000\n',
  '203,arun,prod,bglore,2000\n',
  '304,vijay,QA,chennai,30000\n',
  '545,xerox,sales,mumbai,34234\n',
  '456,anu,sales,noida,56780']}
```

In [ ]:
```
lambda
------
 |-->unnamed function
 |-->inline action/expression/simplecall
 |-->function call with arguments ->return value //style - lambda
 |
 |-> lambda list of args:expression/call
```

In [8]:
```
def f1(a1,a2):
    return a1+a2

f1(10,20) #<<<< named function
```

Out[8]: 30

In [10]:
```
f2=lambda a1,a2:a1+a2
f2(10,20)  # <<<< unnamed function
```

Out[10]: 30

In [14]:
```
f3=lambda a1:a1+100
f3(10)
```

Out[14]: 110

In [16]:
```
f4=lambda a1,a2:a1>a2
f4(1000,200)
```

Out[16]: True

In [18]:
```
def fx(a):
    return a+100
```

In [19]:
```
f5=lambda a1:fx(a1)
f5(10)
```

Out[19]: 110

In [20]:
```
f6=lambda a:a.upper()
f6("abc")
```

Out[20]:
```
'ABC'
```

In [ ]:
```
map
filter
reduce (3.x) ->functools ->import functools -> functools.reduce()
```

In [ ]:
```
map(function,collection) ->[ ]

filter(function,collection) -> [ ]

reduce(function,collection) ->Single
```

In [21]:
```
L=[]
def f1(a):
    return a+100

for var in range(5):
    rv=f1(var)
    L.append(rv)

L
```

Out[21]:
```
[100, 101, 102, 103, 104]
```

In [26]:
```
L=list(map(f1,[var for var in range(5)]))
L         # |__ named function
```

Out[26]:
```
[100, 101, 102, 103, 104]
```

In [24]:
```
L=list(map(lambda a:a+100,[var for var in range(5)]))
L
```

Out[24]:
```
[100, 101, 102, 103, 104]
```

In [27]:
```
L=[]
def fx(a):
    if(a>5):
        return True
    else:
        return False

for var in range(10):
    r=fx(var)
    L.append(r)

print(L)
```

```
[False, False, False, False, False, False, True, True, True, True]
```

In [28]:
```
list(map(lambda a:a>5,range(10)))
```

Out[28]:
```
[False, False, False, False, False, False, True, True, True, True]
```

In [29]:
```
list(filter(lambda a:a>5,range(10)))
```

Out[29]:
```
[6, 7, 8, 9]
```

In [31]:
```
L=[10,20,30,40,50]
s=0
for var in L:
    s=s+var
s
```

Out[31]:
```
150
```

In [32]:
```
from functools import reduce
reduce(lambda a,b:a+b,L)
```

Out[32]:
```
150
```

In [33]:
```
if reduce(lambda a,b:a+b,L) >100:
    print("Yes")
else:
    print("No")
```

```
Yes
```

In [45]:
```
reduce(lambda a,b:int(a)+int(b),[var.split(",")[-1] for var in open("C:\\users\\k
```

Out[45]:
```
133014
```

In [49]:
```
s=0
FH=open("C:\\users\\karthikeyan\\emp.csv")
for var in FH.readlines():
    L=var.split(",")
    s=s+int(L[-1])

print(s)
```

```
133014
```

In [ ]:
```
# object oriented programming
# ---------------------------
# class  object method   inheritance

# class - type - codeblock - set of attrs

class classname:
    attribute1
    attribute2
    ..
    attributeN

classname.attribute1
classname.attributeN=Newvalue
```

In [52]:
```
class box:
    fname='/var/log/test.log'
    count=5

print(box.count)
print(box.fname)
box.fname="D:\\repo.log"
print(box.fname)
```

```
5
/var/log/test.log
D:\repo.log
```

In [53]:
```
box.var=123 # we can create newattribute
print(box.var)
```

```
123
```

In [54]:
```
class box:
    v=10
    f=1.35
    s='data'
    L=['F1','F2']
    T=('D1','D2')
    d={"K1":"V1","K2":"V2"}

print(box.v)
print(box.f)
print(box.s)
print(box.s.upper())
print(box.L)
box.L.append("d3")
print(box.L)
```

```
10
1.35
data
DATA
['F1', 'F2']
['F1', 'F2', 'd3']
```

In [58]:
```
for var in box.d:
    print(var,box.d[var])
```

```
K1 V1
K2 V2
```

In [ ]:
```
class - type - blue print of an object
object - value - real entity

 +-------------------------+
 |        [ ]  [  ]        |   blueprint sheet - class
 |        [       ]        |
 +-------------------------+
  |       |     ..     |
  |       |            |
  B1      B2           Bn <== real entity Building - object
  |       |            |
 1st      2nd          nth     <== different address(memory)
 Block    Block        Block
```

In [ ]:
```
obj1=classname()
(building1)

obj2=classname()
(building2)
```

In [60]:
```
class Fax:
    faxno=1234

obj1=Fax()
obj2=Fax()
print(obj1.faxno)
print(obj2.faxno)
```

```
1234
1234
```

In [61]:
```
Fax.faxno=5678
print(obj1.faxno)
print(obj2.faxno)
```

```
5678
5678
```

In [62]:
```
obj1.faxno='F134' # object based initialization
print(obj1.faxno)
print(Fax.faxno)
print(obj2.faxno)
```

```
F134
5678
5678
```

In [63]:
```
Fax.faxno=9999
print(obj1.faxno)
print(obj2.faxno)
```

```
F134
9999
```

In [64]:
```
obj2.faxno='F454'
Fax.faxno=5555
print(obj1.faxno)
print(obj2.faxno)
```

```
F134
F454
```

In [70]:
```
# def functionNAME():                          class classNAME:
#         ...                                          ...
# functionNAME() <-- function call   Vs         obj=classNAME()
#
# functionNAME() <-- function call symbol
#
# variable=()<-- tuple (data type)
#
#  '' b'' [] () {}
type('5$% gH[10,20,30]')
type(['data',True,-1,2])
type([])
type(())
type({})
```

Out[70]:
```
dict
```

In [71]:
```
class serverinfo:
    sname='default-server'

s1=serverinfo()
s2=serverinfo()
s3=serverinfo()

s1.sname="Unix-Server"
s2.sname="Linux-Server"
s3.sname="Winx-Server"

print(s1.sname)
print(s2.sname)
print(s3.sname)
```

```
Unix-Server
Linux-Server
Winx-Server
```

In [72]:
```
serverinfo.sname="10.20.30.40"
print(s1.sname,s2.sname,s3.sname)
```

```
Unix-Server Linux-Server Winx-Server
```

In [73]:
```
s4=serverinfo()
print(s4.sname)
```

```
10.20.30.40
```

In [74]:
```
s4.sname='SunOS'
print(s4.sname)
```

```
SunOS
```

In [78]:
```
class EMP:
    NAME=''
    DEPT=''
```

In [79]:
```
e1=EMP()
e2=EMP()
e3=EMP()
print(type(e1),type(e2),type(e3))
```

```
<class '__main__.EMP'> <class '__main__.EMP'> <class '__main__.EMP'>
```

In [80]:
```
e1.NAME='arun'
e1.DEPT='sales'
e2.NAME='vijay'
e2.DEPT='Prod'
e3.NAME='kumar'
e3.DEPT='HR'
```

In [82]:
```
print("Name:{}\t Working dept:{}".format(e1.NAME,e1.DEPT))
print("Name:{}\t Working dept:{}".format(e2.NAME,e2.DEPT))
print("Name:{}\t Working dept:{}".format(e3.NAME,e3.DEPT))
```

```
Name:arun        Working dept:sales
Name:vijay       Working dept:Prod
Name:kumar       Working dept:HR
```

In [84]:
```
len(dir(EMP)) # 28
"DEPT" in dir(EMP)
```

Out[84]:
```
True
```

In [85]:
```
"PLACE" in dir(EMP)
```

Out[85]:
```
False
```

In [86]:
```
EMP.PLACE=""
"PLACE" in dir(EMP)
```

Out[86]:
```
True
```

In [87]:
```
len(dir(EMP)) # 29
```

Out[87]:
```
29
```

In [89]:
```
e1.code
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-89-ff88ba44af64> in <module>
----> 1 e1.code

AttributeError: 'EMP' object has no attribute 'code'
```

In [90]:
```
EMP.code=0
```

In [93]:
```
e1.code
e2.code
e3.code
```

Out[93]:
```
0
```

In [94]:
```
e1.PLACE
```

Out[94]:
```
''
```

In [96]:
```
e1.PLACE="City1"
e2.PLACE="City2"
e3.PLACE="City3"
print("""About {} details:-
------------------------------
Name:{}
-----------
Working dept:{}
------------------------------
Working city:{}
------------------------------""".format(e1.NAME,e1.NAME,e1.DEPT,e1.PLACE))
```

```
About arun details:-
------------------------------
Name:arun
-----------
Working dept:sales
------------------------------
Working city:City1
------------------------------
```

In [97]:
```
class Box:
    v1=100

obj1=Box()
obj2=Box()
```

In [98]:
```
obj1.v1 # 100
obj2.v1 # 100
Box.v1=200
obj1.v1 # 200
obj2.v1 # 200
```

Out[98]:
```
200
```

In [99]:
```
obj1.v1='DATA1'
obj2.v1='DATA2'
obj1.v1 # DATA1
obj2.v1 # DATA2
Box.v1 # 200
```

Out[99]:
```
200
```

In [100]:
```
obj3=Box()
obj3.v1 # 200
```

Out[100]:
```
200
```

In [101]:
```
obj1.Code=123
obj2.Temp='/tmp'
```

In [106]:
```
#Box.Code # AttributeError
#obj2.Code # AttributeError
obj3.Code # AttributeError
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-106-6b62017e2142> in <module>
      1 #Box.Code # AttributeError
      2 #obj2.Code # AttributeError
----> 3 obj3.Code # AttributeError

AttributeError: 'Box' object has no attribute 'Code'
```

In [107]:
```
class Box1:                                    #   [     ] class
    v1=100                                     # [DEPT] [DEPT]
                                               #   obj1   obj2
obj1=Box1()
obj2=Box1()
obj1.DEPT='sales'
obj2.DEPT='prod'
print(obj1.DEPT,obj2.DEPT)
```

```
sales prod
```

In [109]:
```
obj3=Box1()
print(obj3.DEPT)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-109-e03798a3f9f8> in <module>
      1 obj3=Box1()
----> 2 print(obj3.DEPT)

AttributeError: 'Box1' object has no attribute 'DEPT'
```

In [112]:
```
class Box2:                                     # [   DEPT ='' ] Class
    DEPT='xyz'                                  #
obj1=Box2()                                     # [ DEPT]   [DEPT]   [ ]
obj2=Box2()

obj1.DEPT='Maths'
obj2.DEPT='Physics'
print(obj1.DEPT,obj2.DEPT)
```

```
Maths Physics
```

In [113]:
```
obj3=Box2()
print(obj3.DEPT)
```

```
xyz
```

In [ ]:
```
'''
file: property.txt
--------------------
interface='eth0'
onboot=none
IP='10.20.30.40'
prefix=24
DNS1='145.665.534.456'

STEP 1: empty dict

STEP 2: use fileHandling - read data from <FILE> -line by line
        split each line into multiple fields(value)//list
        K,V=string.split("=")
step 3: add splited data to dict

step 4: display key/value

step 5: update onboot ->dhcp;
               IP->'12.35.66.45'
        ADD    DNS2 ->135.544.343.535

step 6: display updated dict details

step 7: create a newFILE/Write data to FILE (newproperty.txt) file

'''
```

In [122]:
```
d={} # empty dict
with open("D:\\property") as FH:
    for var in FH.readlines(): # reading data from <FILE> line by line
        var=var.strip() # remove \nchar
        K,V=var.split("=") # split single ->multiple value
        d[K]=V # adding new data to existing dict

print(d)
d['interface']='eth1'
d['onboot']='dhcp'
d['IP']='12.34.66.44'
d['DNS2']='343.223.444.555'
print("")
for var in d:
    print("{}\t{}".format(var,d[var]))

with open("D:\\newproperty.txt","w") as WH:
    for var in d:
        WH.write("{}={}\n".format(var,d[var]))
```

```
{'interface': "'eth0'", 'onboot': 'none', 'IP': "'10.20.30.40'", 'prefix': '2
4', 'DNS1': "'145.665.534.456'"}

interface       eth1
onboot  dhcp
IP      12.34.66.44
prefix  24
DNS1    '145.665.534.456'
DNS2    343.223.444.555
```

In [123]:
```
class Box:
    var=100

obj=Box()
obj.var
```

Out[123]:
```
100
```

In [125]:
```
class Box:
    var=100
    def f1():
        print("f1 block from Box class")

# Box is a class
# ---
#   |_attributes - var f1
#
obj=Box()
#obj.var
# obj.f1() -->f1(obj)
# obj.f1(10,20) -------> f1(obj,10,20)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-125-12882f3ae8b1> in <module>
     10 obj=Box()
     11 obj.var
---> 12 obj.f1()

TypeError: f1() takes 0 positional arguments but 1 was given
```

In [127]:
```
def f1():
    print("Hello")
f1()
f1(10) # Error
```

```
Hello

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-127-b0b2310fbbf1> in <module>
      2     print("Hello")
      3 f1()
----> 4 f1(10)

TypeError: f1() takes 0 positional arguments but 1 was given
```

In [130]:
```
class Box:
    var=10
    def f1(self):
        print(self.var) # obj.var
        self.var=20 # obj.var=20
obj=Box()
print(obj.var) # 10
obj.f1() # f1(obj)
print(obj.var) # 20
```

```
10
10
20
```

In [133]:
```
class Box:
    var=10
    def f1(self,a1):
        self.var=a1
obj1=Box()
obj2=Box()
print(obj1.var,obj2.var) # (A)
obj1.f1(100) # f1(obj1,100) # obj1.var=100  ->print(obj1.var) ->100
obj2.f1(200) # f1(obj2,200) # obj2.var=200  ->print(obj2.var) ->200
```

```
10 10
```

In [132]:
```
print(obj1.var) # (B)
print(obj2.var) # (C)
```

```
100
200
```

In [137]:
```
# help(list.append) # append(self,object)
#      -----------
#        |__ class list
#                  def append(self,object):
#                        .... |
# L=list()                   |
# L.append("D1")   # append(L,"D1")
# help(str.upper)  # upper(self)
#        |__class str:
#               def upper(self):
#                       ...
#
```

In [138]:
```
L=[]
L.append("data")
L.append(["D2","D3"])
L.append(34)

# python 2.x
# help(list.append) -> def append(object):
#                      L.append(object)
#
s='abc'
s.upper()
# help(str.upper) --> S.upper()
```

Out[138]:
```
'ABC'
```

In [140]:
```
L=list()   # obj=classname()
print(type(L),len(L))
L
```

```
<class 'list'> 0
```

Out[140]:
```
[]
```

In [141]:
```
s=str()
s
```

Out[141]:
```
''
```

In [142]:
```
L=list()              #  obj=classname()
L.append("D1")        #  obj.method1("arg")
                      #  obj.method2("Args")
L.insert(1,"D2")
L.append("D3")
L
```

Out[142]:
```
['D1', 'D2', 'D3']
```

In [ ]:
```
'''
classname -> student
                attributes = sname,USN,place
                  |__method1() - initialization sname,USN,place
                  |__method2() - display student details
                  |__method3() - update place
                  |__display updated details //method2()
'''
```

In [143]:
```
class student:
    sname=''
    USN=0
    place=''
    def f1(self,a1,a2,a3):
        self.sname=a1
        self.USN=a2
        self.place=a3
    def f2(self):
        print("Name:{}\tUSN:{}\tCityName:{}".format(self.sname,self.USN,self.plac
    def f3(self,a1):
        self.place=a1
```

In [144]:
```
obj1=student()
obj1.f1("arun","1sbf3434",'city1')
obj1.f2()
```

```
Name:arun       USN:1sbf3434    CityName:city1
```

In [145]:
```
obj2=student()
obj2.f2()
```

```
Name:   USN:0   CityName:
```

In [146]:
```
obj2.f1("Vijay","4GH434334","city2")
obj2.f2()
```

```
Name:Vijay      USN:4GH434334   CityName:city2
```

In [147]:
```
obj3=student()
obj3.f1("kumar","3FDDFDF","city3")
obj2.f2()
```

```
Name:Vijay      USN:4GH434334   CityName:city2
```

In [150]:
```
obj1.f2()
obj2.f2()
obj3.f2()
```

```
Name:arun       USN:1sbf3434    CityName:city1
Name:Vijay      USN:4GH434334   CityName:city2
Name:kumar      USN:3FDDFDF     CityName:city3
```

In [151]:
```
obj2.f3("pune")
obj2.f2()
```

```
Name:Vijay      USN:4GH434334   CityName:pune
```

In [152]:
```
obj1.f2()
obj2.f2()
obj3.f2()
```

```
Name:arun       USN:1sbf3434    CityName:city1
Name:Vijay      USN:4GH434334   CityName:pune
Name:kumar      USN:3FDDFDF     CityName:city3
```

In [156]:
```
print(obj1.sname)
print(obj2.sname)
obj3.sname
```

```
arun
Vijay
```

Out[156]:
```
'kumar'
```

In [157]:
```
class student:
    __sname=''
    __USN=0
    __place=''
    def f1(self,a1,a2,a3):
        self.__sname=a1
        self.__USN=a2
        self.__place=a3
    def f2(self):
        print("Name:{}\tUSN:{}\tCityName:{}".format(self.__sname,self.__USN,self.
    def f3(self,a1):
        self.__place=a1
```

In [159]:
```
obj1=student()
obj1.f1("ANU",'1434334','bglore')
obj1.f2()
obj1.__sname
```

```
Name:ANU        USN:1434334     CityName:bglore

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-159-d0bb768cfcbd> in <module>
      2 obj1.f1("ANU",'1434334','bglore')
      3 obj1.f2()
----> 4 obj1.__sname

AttributeError: 'student' object has no attribute '__sname'
```

In [ ]:
```
# class  fsdetails:
#            fstype,findex,fpartition,fmount
#
#          initialization
#          display details
#          change partition,mount directory
#          display updated details
```

In [163]:
```
class fsdetails:
    __fstype=''
    __findex=0
    __fpart=''
    __fmount='/'
    def f1(self,*args):
        if(len(args) == 0):
            print("Empty argument")
            exit() # exit from script
        self.__fstype=args[0]
        self.__findex=args[1]
        self.__fpart=args[2]
        self.__fmount=args[-1]
    def f2(self):
        return self.__fstype,self.__findex,self.__fpart,self.__fmount
    def f3(self,a1,a2):
        self.__fpart=a1
        self.__fmount=a2

obj1=fsdetails()
obj1.f1("xfs",1001,"/dev/sda1","/D1")
rv1=obj1.f2()
obj1.f3("/dev/xvdb1",'/home/D1')

obj2=fsdetails()
obj2.f1("ext4",1023,"/dev/sda2","/D2")
rv2=obj2.f2()
obj2.f3("/dev/mapper","/mnt")
```

In [164]:
```
print(rv1)
print(rv2)
```

```
('xfs', 1001, '/dev/sda1', '/D1')
('ext4', 1023, '/dev/sda2', '/D2')
```

In [165]:
```
obj1.f2()
```

Out[165]:
```
('xfs', 1001, '/dev/xvdb1', '/home/D1')
```

In [166]:
```
obj2.f2()
```

Out[166]:
```
('ext4', 1023, '/dev/mapper', '/mnt')
```

In [ ]:
```
class DB:
    def f1(self,....):
        dbinitialization task
        self.var=value
    def f2(self):
        Query
    def f3(self):
        operation1
    def f4(self):
        operation2

obj1=DB()    Vs   obj2=DB()
obj1.f1()         obj2.f2()
obj1.f2()
```

In [169]:
```
class Box:
    def f1(self):
        print("non-constructor")
obj=Box()
obj.f1()
```

```
non-constructor
```

In [171]:
```
class Box:
    def __init__(self):
        print("constructor")

obj=Box()
```

```
constructor
```

In [174]:
```
class Box:
    def __init__(self,a1,a2):
        self.v1=a1
        self.v2=a2
        print("constructor block")
obj1=Box(10,20)
print(obj1.v1,obj1.v2)
```

```
constructor block
10 20
```

In [ ]:
```
class DB:
    def __init__(self,....):
        dbinitialization task
        self.var=value
    def f2(self):
        Query
    def f3(self):
        operation1
    def f4(self):
        operation2

obj1=DB(dbname,user,IP,port,....)    Vs   obj2=DB() <==Error
                                          obj2.f2()
obj1.f2()
```

In [176]:
```
class Box:
    def __init__(self,a1,a2,a3=0,*args,**kwargs):
        print(a1,a2,a3,args,kwargs)

obj=Box(10,20,30,40,50,60,db='sql',name='root')
```

```
10 20 30 (40, 50, 60) {'db': 'sql', 'name': 'root'}
```

In [182]:
```
def f1(*args,**kwargs):
    print("Hello")
    print(args)
    print(kwargs)


f1(10,20,var=1234)
f1()
```

```
Hello
(10, 20)
{'var': 1234}
Hello
()
{}
```

In [185]:
```
class Box:
    def __init__(self,*args,**kwargs):
        print("Constructor")
        print(args)
        print(kwargs)
obj1=Box()
obj2=Box(10,20)
obj3=Box(sh='/bin/bash')
obj4=Box(100,200,sh='/usr/bin/ksh')
```

```
Constructor
()
{}
Constructor
(10, 20)
{}
Constructor
()
{'sh': '/bin/bash'}
Constructor
(100, 200)
{'sh': '/usr/bin/ksh'}
```

In [189]:
```
class Emp:
    def __init__(self):
        print("Hello")
obj=Emp()
```

```
Hello
```

In [190]:
```
class fsdetails:
    def __init__(self,*args):
        if(len(args) == 0):
            print("Empty argument")
            exit() # exit from script
        self.__fstype=args[0]
        self.__findex=args[1]
        self.__fpart=args[2]
        self.__fmount=args[-1]
    def f2(self):
        return self.__fstype,self.__findex,self.__fpart,self.__fmount
    def f3(self,a1,a2):
        self.__fpart=a1
        self.__fmount=a2

obj1=fsdetails("xfs",1001,"/dev/sda1","/D1")
#obj1.f1("xfs",1001,"/dev/sda1","/D1")
rv1=obj1.f2()
obj1.f3("/dev/xvdb1",'/home/D1')
rv1
```

Out[190]:
```
('xfs', 1001, '/dev/sda1', '/D1')
```

In [191]:
```
obj1.f2()
```

Out[191]:
```
('xfs', 1001, '/dev/xvdb1', '/home/D1')
```

In [ ]:
```
                      Running
                      |
python3 (parent)   - Waiting    <---------------
 |                                              |
subprocess(command-child) - R+ (Running)        |
 |                                 |__Exit_____|
```

In [ ]:
```
import os
os.system("command") -->STDOUT
|
return status code

s=os.popen("command").read() / L=os.popen("command").readlines()

import subprocess

subprocess.check_output("command") ->''/b''
                        --------
                        "command -option",shell=True
                        ["command","options"]

subprocess.Popen
            |
           piping one process to another
            open/troubleshoot - intermediate processs
           |
           SSH
```

In [ ]:

```
>>> wh=open("D:\\r1.log","w")
>>>
>>> subprocess.call("powershell get-process",shell=True,stdout=wh)
0
>>> subprocess.Popen(["powershell","get-process","zoom"])
<subprocess.Popen object at 0x00AD8A90>
>>>
Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
    540      51    83008      70960      17.44    4428   1 Zoom
    684     163   381496     356224   1,270.53    6296   1 Zoom



>>> # class object methods
>>> # inhertiance
>>> # @classmethod
>>> # |
>>> # decorator
>>> # iterator ->generator
>>> # regx
>>> #   |__Search;Substitute;split(awk) + inputValidation
>>>
```