In [5]:
```
import sys
sys.path.append("F:\\")
#import <module>
```

In [6]:
```
import os
os.getcwd()
```

Out[6]:
```
'C:\\Users\\Raja'
```

In [7]:
```
getcwd()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[7], line 1
----> 1 getcwd()

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

In [8]:
```
from os import getcwd
getcwd()
```

Out[8]:
```
'C:\\Users\\Raja'
```

In [ ]:
```
project/
        |__ p1.py .. .p50.py
        |__ Sub1/pa.py pb.py ...pn.py

import p1,p2,p3

import p1
import p2
import p3

|->package - Folder <or> directory

Commandline steps:
    1. create a project folder/directory
    2. collect all the python files into directory
    3. create a package initialization file (file name: __init__.py)
    4. import all external symbols to __init__.py file
    5. Test your project/package =>import <folderName>
```

In [ ]:
```
File: pa.py                                    File: p1.py
-------------                                  -----------
port=5000                                       import pa
app="Flask"                                     print(pa.port) Vs print(port)
def f1():                                       var=120                    __name__.pc
    print("Web page")                           print(var)
-----------------                              --------------
```

In [ ]:
```
File: pb.py                                  import pb
-------------
port=5000
def f1():
    print("Welcome")
print(port)
f1()
-----------------
```

In [ ]:
```
file: pc.py                                      file: p1.py
------------                                  --> import pc
port=5000                                       pc.f1()
def f1():
    print("Welcome")

if __name__ == '__main__':
    f1()
```

In [ ]:
```
__main__.port   | 5000                        pa.port   | 5000
----------------------                        -----------------
__main__.app    | Flask                       __main__.var| 120
----------------------                        -------------------
__main__.f1     | 0x1345

from <dir>.<sub>.<module> import <members>
                                        *
```

In [9]:
```
import os
import json
import numpy
import re
```

In [10]:
```
os.__file__
```

Out[10]:
```
'C:\\ProgramData\\anaconda3\\Lib\\os.py'
```

In [11]:
```
json.__file__
```

Out[11]:
```
'C:\\ProgramData\\anaconda3\\Lib\\json\\__init__.py'
```

In [12]:
```
import os
import json
print(type(os),type(json))
```

```
<class 'module'> <class 'module'>
```

In [13]:
```
import sys
sys.version
```

Out[13]:
```
'3.11.5 | packaged by Anaconda, Inc. | (main, Sep 11 2023, 13:26:23) [MSC v.1
916 64 bit (AMD64)]'
```

In [14]:
```
sys.platform
```

Out[14]:
```
'win32'
```

In [4]:
```
import subprocess          --->   import subprocess as sp
subprocess.call("")               sp.call("")

import http.lib.request   -------> import http.lib.request as hp
http.lib.request.fx()              hp.fx()
```

```
  Cell In[4], line 1
    import subprocess          --->   import subprocess as sp
                               ^
SyntaxError: invalid syntax
```

In [ ]:
```
Activity:
----------
initialize a pin number (ex: pin=1234)
|
use while loop - limit is 3
     -> read a pin Number from <STDIN>
     -> test inputPin with an existing pin
               -------------------
                    |->Pin is matched - <count>
pin is blocked - if all 3 inputs are failed

|
create a new file ->pin_history.log - append mode
|
update user input pin details and date/time to pin_history.log file
            ---------------                    ===============

|
create a function - pin_test():
|
loadable - module ->open pythonshell ->import <pinModule>
                                       <pinModule>.pin_test() //
```

In [15]:
```
import time
time.ctime()
```

Out[15]:
```
'Wed Oct 23 16:00:22 2024'
```

In [16]:
```
try:
    fobj = open('InvalidFile','r')
except Exception:
    print(sys.exc_info())
```

```
(<class 'FileNotFoundError'>, FileNotFoundError(2, 'No such file or director
y'), <traceback object at 0x000001DAFF288900>)
```

In [17]:
```
try:
    fobj = open('InvalidFile','r')
except:
    print(sys.exc_info())
```

```
(<class 'FileNotFoundError'>, FileNotFoundError(2, 'No such file or director
y'), <traceback object at 0x000001DA80399840>)
```

In [18]:
```
try:
    fobj = open('E:\\emp.csv1')
    print(fobj.read())
except:
    print(sys.exc_info())
```

```
(<class 'FileNotFoundError'>, FileNotFoundError(2, 'No such file or director
y'), <traceback object at 0x000001DA802A5380>)
```

In [19]:
```
print(type(10))
```

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

In [20]:
```
print(type(int))
```

```
<class 'type'>
```

In [ ]:
```
class
object
|
method
   |->special methods
inheritance
decorator
    |->classmethod,staticmethod
```

In [ ]:
```
class <className>:
    <attribute>
```

In [21]:
```
class product:
    pid = 101
    pname = 'prodA'

print(type(product),type(int))
```

```
<class 'type'> <class 'type'>
```

In [22]:
```
# we can access class attribute -> using className.<attribute>
# we can modify an existing class attribute

product.pid
```

Out[22]:
```
101
```

In [23]:
```
product.Pid
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[23], line 1
----> 1 product.Pid

AttributeError: type object 'product' has no attribute 'Pid'
```

In [24]:
```
product.pid = 505
product.pid
```

Out[24]:
```
505
```

In [25]:
```
product.pcost = 1000 # we can create new attribute to an existing class
```

In [ ]:
```
  +---------------------+
  | [ ]      |  |  [][] |    - blueprint sheet - class
  |              white  |
  +---------|   |-------+


  [Building1] [Building2] .. [BuildingN] - object
    |           |_2nBlock
    1st Block
```

In [26]:
```
class Enrollment:
    Name = ''
    DOB = ''
```

In [27]:
```
obj1 = Enrollment()
obj1.Name = 'Arun'
obj1.DOB = '1st Jan'
```

In [28]:
```
obj2 = Enrollment()
obj2.Name = 'Anu'
obj2.DOB = '2nd Feb'
```

In [29]:
```
print(obj1.Name,obj1.DOB,obj2.Name,obj2.DOB)
```

```
Arun 1st Jan Anu 2nd Feb
```

In [30]:
```
obj1.Place
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[30], line 1
----> 1 obj1.Place

AttributeError: 'Enrollment' object has no attribute 'Place'
```

In [31]:
```
Enrollment.Place = '' # using classname - we can add new attribute to an class
```

In [32]:
```
obj1.Place # note - there is no AttributeError
```

Out[32]:
```
''
```

In [33]:
```
obj1.Place = 'City1' # object based initialization
obj2.Place = 'City2' # object based initialization
```

In [34]:
```
def fx():
    print('hello')

class cname:
    def fy():
        print('hello')

obj = cname()

print(type(fx),type(obj.fy))
```

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

In [35]:
```
"ab".upper() # string method
```

Out[35]:
```
'AB'
```

In [36]:
```
def f1():
    print("OK")

f1()
f1(10)
```

```
OK

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[36], line 5
      2     print("OK")
      4 f1()
----> 5 f1(10)

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

In [37]:
```
class cname:
    def fy():
        print("OK")

obj = cname()
obj.fy() # fy(obj)   ;  obj.fy(10,20,30) -> fy(obj,10,20,30)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[37], line 6
      3         print("OK")
      5 obj = cname()
----> 6 obj.fy()

TypeError: cname.fy() takes 0 positional arguments but 1 was given
```

In [38]:
```
class cname:
    def fy(self):
        print("self=",self)

obj1 = cname()
obj1.fy() # fy(obj1)
print("obj1=",obj1)
time.sleep(10) # dely 10secs
print('') # empty line
obj2 = cname()
obj2.fy() # fy(obj2)
print("obj2=",obj2)
```

```
self= <__main__.cname object at 0x000001DA8043B010>
obj1= <__main__.cname object at 0x000001DA8043B010>

self= <__main__.cname object at 0x000001DAFF1D33D0>
obj2= <__main__.cname object at 0x000001DAFF1D33D0>
```

In [39]:
```
class Enrollment:
    Name = ''
    DOB = ''
    Place = ''
    def f1(self,n,d,p):
        '''initialization'''
        self.Name = n
        self.DOB = d
        self.Place = p
        print('Enrollment is done')
    def f2(self):
        '''display emp details'''
        print(f'''About {self.Name} details:-
        Emp name:{self.Name} DOB:{self.DOB}
        Working City:{self.Place}''')
    def f3(self,p):
        '''update emp working Place'''
        self.Place = p
        print(f'{self.Name} updated working City is:{self.Place}')
```

In [40]:
```
obj1 = Enrollment()
obj1.f1('Arun','1st Jan','City1')
```

```
Enrollment is done
```

In [41]:
```
obj2 = Enrollment()
obj2.f1('Anu','2nd Feb','City2')
```

```
Enrollment is done
```

In [42]:
```
obj1.f2()
```

```
About Arun details:-
        Emp name:Arun DOB:1st Jan
        Working City:City1
```

In [43]:
```
obj2.f2()
```

```
About Anu details:-
        Emp name:Anu DOB:2nd Feb
        Working City:City2
```

In [44]:
```
obj1.f3('Hyderabad')
```

```
Arun updated working City is:Hyderabad
```

In [45]:
```
obj1.f2()
```

```
About Arun details:-
        Emp name:Arun DOB:1st Jan
        Working City:Hyderabad
```

In [46]:
```
class DBI:
    def connect(self):
        '''DB connection'''
        ...
    def method1(self):
        'Query1'
    def method2(self):
        'Query2'

obj = DBI()
obj.method2() # we can method2 directly -> DBOpertion Error
```

In [47]:
```
class Enrollment:
    def __init__(self,n,d,p):
        '''initialization'''
        self.Name = n
        self.DOB = d
        self.Place = p
        print('Enrollment is done')
    def f2(self):
        '''display emp details'''
        print(f'''About {self.Name} details:-
        Emp name:{self.Name} DOB:{self.DOB}
        Working City:{self.Place}''')
    def f3(self,p):
        '''update emp working Place'''
        self.Place = p
        print(f'{self.Name} updated working City is:{self.Place}')
```

In [48]:
```
obj1 = Enrollment('Arun','1st Jan','City-1')
```

```
Enrollment is done
```

In [49]:
```
obj2 = Enrollment('Anu','2nd Feb','City-2')
```

```
Enrollment is done
```

In [50]:
```
for var in [obj1,obj2]:
    var.f2()
    time.sleep(3)
```

```
About Arun details:-
        Emp name:Arun DOB:1st Jan
        Working City:City-1
About Anu details:-
        Emp name:Anu DOB:2nd Feb
        Working City:City-2
```

In [51]:
```
obj1.f3('Hyderabad')
```

```
Arun updated working City is:Hyderabad
```

In [53]:
```
obj1.f2()
```

```
About Arun details:-
        Emp name:Arun DOB:1st Jan
        Working City:Hyderabad
```

In [54]:
```
i = 10
```

In [55]:
```
j = int(10)
```

In [56]:
```
print(type(i),type(j))
print(i,j)
```

```
<class 'int'> <class 'int'>
10 10
```

In [57]:
```
s = str()
```

In [58]:
```
L = list()
```

In [59]:
```
d = dict()
```

In [60]:
```
k = int()
f = float()
print(k,f)
```

```
0 0.0
```

In [ ]:
```
#help(str)
```

In [61]:
```
class cname:
    def __init__(self):
        print('initialization')
    def __del__(self):
        print('deallocation')
```

In [62]:
```
obj = cname()
```

```
initialization
```

In [63]:
```
del(obj)
```

```
deallocation
```

In [ ]:
```
In python inside the class - any variable/function
name starts with doubleunderscore and endswith doubleunderscore
predefined python class attributes (or) builtin attributes
```

In [64]:
```
dir(int)
```

Out[64]:
```
['__abs__',
 '__add__',
 '__and__',
 '__bool__',
 '__ceil__',
 '__class__',
 '__delattr__',
 '__dir__',
 '__divmod__',
 '__doc__',
 '__eq__',
 '__float__',
 '__floor__',
 '__floordiv__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getnewargs__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__index__',
 '__init__',
 '__init_subclass__',
 '__int__',
 '__invert__',
 '__le__',
 '__lshift__',
 '__lt__',
 '__mod__',
 '__mul__',
 '__ne__',
 '__neg__',
 '__new__',
 '__or__',
 '__pos__',
 '__pow__',
 '__radd__',
 '__rand__',
 '__rdivmod__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rfloordiv__',
 '__rlshift__',
 '__rmod__',
 '__rmul__',
 '__ror__',
 '__round__',
 '__rpow__',
 '__rrshift__',
 '__rshift__',
 '__rsub__',
 '__rtruediv__',
 '__rxor__',
 '__setattr__',
 '__sizeof__',
```

```
 '__str__',
 '__sub__',
 '__subclasshook__',
 '__truediv__',
 '__trunc__',
 '__xor__',
 'as_integer_ratio',
 'bit_count',
 'bit_length',
 'conjugate',
 'denominator',
 'from_bytes',
 'imag',
 'numerator',
 'real',
 'to_bytes']
```

In [65]:
```
va = int(15)
vb = int(20)
va.__add__(vb)
```

Out[65]: 35

In [66]:
```
class Box:
    def __init__(self):
        pass

obj = Box()
obj()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[66], line 6
      3         pass
      5 obj = Box()
----> 6 obj()

TypeError: 'Box' object is not callable
```

In [67]:
```
callable(obj)
```

Out[67]: False

In [68]:
```
def fx():
    pass
print(type(fx))
```

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

In [69]:
```
callable(fx)
```

Out[69]: True

In [70]:
```
class Box:
    def __call__(self):
        print("OK")

obj = Box()
callable(obj)
```

Out[70]: True

In [71]:
```
obj() #obj.__call__()
```

```
OK
```

In [72]:
```
def f1():
    print('f1 block')
f1.__call__() # f1()
```

```
f1 block
```

In [73]:
```
def f2():
    print("OK")
```

In [74]:
```
d={}
d['K1']=f1
d['K2']=f2
d
```

Out[74]: {'K1': <function __main__.f1()>, 'K2': <function __main__.f2()>}

In [75]:
```
callable(d['K1'])
```

Out[75]: True

In [76]:
```
d['K1']()
```

```
f1 block
```

In [77]:
```
class box:
    def __init__(self,a=10):
        self.a = a
    def __str__(self):
        return str(self.a+100)
```

In [78]:
```
obj = box(45)
str(obj) # obj.__str__()
```

Out[78]: '145'

In [ ]:
```
In Side class - any variable/function name - starts with double __ (not ends wi
                  ------------------
                        |->user defined private attributes
```

In [79]:
```
class mylogin:
    name='root'
    __password='Welcome'
```

In [81]:
```
print(mylogin.name)
mylogin.__password
```

```
root

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[81], line 2
      1 print(mylogin.name)
----> 2 mylogin.__password

AttributeError: type object 'mylogin' has no attribute '__password'
```

In [82]:
```
class box:
    def __init__(self,bid,bname):
        self.__bid = bid
        self.__bname = bname
    def method1(self):
        print(self.__bid)
        print(self.__bname)
```

In [84]:
```
obj = box(101,'Box-1')
obj.method1()
```

```
101
Box-1
```

In [85]:
```
obj.__bid
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[85], line 1
----> 1 obj.__bid

AttributeError: 'box' object has no attribute '__bid'
```

In [87]:
```
import pandas as pd
print(pd.DataFrame)
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [89]:
```
len(dir(pd.DataFrame))
```

Out[89]:
```
427
```

In [90]:
```
pd.DataFrame.__dict__
```

```
cksort', na_position: 'str' = 'last', ignore_index: 'bool' = False, key: 'V
alueKeyFunc' = None) -> 'DataFrame | None'>,
              'sort_index': <function pandas.core.frame.DataFrame.sort_inde
x(self, *, axis: 'Axis' = 0, level: 'IndexLabel' = None, ascending: 'bool |
Sequence[bool]' = True, inplace: 'bool' = False, kind: 'SortKind' = 'quicks
ort', na_position: 'NaPosition' = 'last', sort_remaining: 'bool' = True, ig
nore_index: 'bool' = False, key: 'IndexKeyFunc' = None) -> 'DataFrame | Non
e'>,
              'value_counts': <function pandas.core.frame.DataFrame.value_c
ounts(self, subset: 'Sequence[Hashable] | None' = None, normalize: 'bool' =
False, sort: 'bool' = True, ascending: 'bool' = False, dropna: 'bool' = Tru
e) -> 'Series'>,
              'nlargest': <function pandas.core.frame.DataFrame.nlargest(se
lf, n: 'int', columns: 'IndexLabel', keep: 'str' = 'first') -> 'DataFram
e'>,
              'nsmallest': <function pandas.core.frame.DataFrame.nsmallest
(self, n: 'int', columns: 'IndexLabel', keep: 'str' = 'first') -> 'DataFram
e'>,
              'swaplevel': <function pandas.core.frame.DataFrame.swaplevel
(self, i: 'Axis' = -2, j: 'Axis' = -1, axis: 'Axis' = 0) -> 'DataFrame'>,
```

In [91]:
```
class product:
    pid = 101
    pname = 'prodA'

class customer:
    cname = 'cusA'
```

In [ ]:
```
class classname(<parentclassName>) <== Inheritance
```

In [92]:
```
class product:
    pid = 101
    pname = 'prodA'

class customer(product): # inheritance  Vs  def functionName(argument):
    cname = 'cusA'

obj = customer()
obj.pid
```

Out[92]:
```
101
```

In [93]:
```
import threading
class myclass(threading.Thread):
    ...
```

Out[93]:
```
threading.Thread
```

In [ ]:
```
GIL - enabled in python - Cpython -> multiprocessing
---
  va = 10
  vb = 10
  vc = 4+6

jython
ipython
pypthon
```

In [94]:
```
class product:
    pid = 101
class vendor(product):
    pid = 505
obj = vendor()
obj.pid
```

Out[94]:
```
505
```

In [95]:
```
class product:
    pid = 101
class vendor(product):
    va = product.pid
    pid = 505
obj = vendor()
obj.va,obj.pid
```

Out[95]:
```
(101, 505)
```

In [96]:
```
class pa:
    def display(self):
        print('display list of files')
class pb(pa):
    def display(self):
        print('display customer records')
```

In [97]:
```
obj = pb()
obj.display()
```

```
display customer records
```

In [100]:
```
class pa:
    def display(self):
        print('display list of files')
class pb(pa):
    def display(self):
        print('display customer records')
        super().display()
```

In [101]:
```
obj = pb()
obj.display()
```

```
display customer records
display list of files
```

In [ ]:
```
# Decorator - meta programming - adding new features to an existing code
Home About  News
               |->City1
               |->City2
               |->City3
```

In [ ]:
```
def decorator(arg):
    def wrapper():
        args()
    return wrapper

functionA = decorator(functionA)
functionA()
|
|
same as
@decorator
def functionA():
    ....
functionA()
```

In [102]:
```
def news(a):
    def wrapper():
        a()
    return wrapper
```

In [103]:
```
@news
def city1():
    print('City-1 news page')
```

In [104]:
```
@news
def city2():
    print('City-2 new page')
```

In [105]:
```
city1()
```

```
City-1 news page
```

In [106]:
```
city2()
```

```
City-2 new page
```

In [107]:
```
@news
def city3():
    print('City-3 news page')
```

In [108]:
```
city3()
```

```
City-3 news page
```

In [111]:
```
class box:
    def method1(self):
        print('instance method')
        self.method3()
    @classmethod
    def method2(cls):
        print('class method')
        cls.method3()
    @staticmethod
    def method3():
        print('this python static method')
```

In [110]:
```
box.method2()
```

```
class method
```

In [ ]:
```
xfs  ext4  btrfs  ...  - initialize - object =>  df -Th
```

In [113]:
```
class fsinfo:
    def __init__(self,fstype):
        self.fstype=fstype
        self.fs_status()
    @staticmethod
    def fs_status():
        print("staticmethod-block")
        os.system("df -Th")

obj1 = fsinfo("ext4")
obj2 = fsinfo("xfs")
obj3 = fsinfo("ntfs")
```

```
staticmethod-block
staticmethod-block
staticmethod-block
```