```
# int float str list tuple dict set
# str , list ,tuple
# -----------------
#   |->Collection of item - index,slicing
#   |->List is mutable (append,insert,pop) //methods
#
#   |->tuple - record
# dict - collection of item //{key:value} //data
# ----- ==================   ----------- //mutable
# using Key ->we can add/modify/delete
# for var in dict_name:
#     list of keys

# list,dict - mutable
#  - add/modify/delete

# operators ->conditional //if statement
# looping while + for
# ---------------------------------------------

# File Handling
#  open(inputFile,'r') ->fobj
#  fobj.read() -->str Vs fobj.readlines() --> List
#  fobj.close()

#  To create a newfile
#  open('resultFile','w') ->wobj
#  wobj.write('Single String\n')
#  wobj.close()
# <or>
# with open() as fobj:
#    ...

# function - Code block - reusability
#   |-> simple FunctionCall
#   |-> Call with Arguments - required args ; default args ; variable length args; keyword args
#   |-> scope of variable
#   |-> global Vs return  ; return <value1>,<value2> //tuple

# Exception Handling
# try:
#   <code block>
# exception Exception as eobj:
#    <Handle the error>
# else:
#     there is no Error
# finally:
#      always running
# ------------------------------------

OOPs
 |--> class object methods

 class - type
 int float str ..list //class

 object - real value
 - each object - own state

 class cname:
   ...
    def method1(self):
      self is current object

 obj = cname()
 obj.method1() # method1(obj)

 ## special methods
 dunder methods
 built in methods
 -----------------

 __attributeName__
 -------------------
```

```
s="abc"
len(s)
```

```
3
```

```
dir(str)
```

Show hidden output

```
class myclass:
  def __init__(self,a=0):
    self.a = a
  def method1(self):
    return self.a

obj = myclass(15)
obj.method1()
```

```
15
```

```
obj1 = myclass()
obj1.method1()
```

```
0
```

```
len(obj)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-2739632275.py in <cell line: 0>()
----> 1 len(obj)

TypeError: object of type 'myclass' has no len()
```

Next steps: Explain error

```
i = 15
len(i)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-2287699442.py in <cell line: 0>()
      1 i = 15
----> 2 len(i)

TypeError: object of type 'int' has no len()
```

Next steps: Explain error

```
'__len__' in dir(str)
```

```
True
```

```
'__len__' in dir(int)
```

```
False
```

```
class myclass:
  def __init__(self,a=0):
    self.a = a
  def __len__(self):
    return self.a + 100

obj = myclass(15)
len(obj) # obj.__len__()
```

```
115
```

```
va = 10
vb = 20
va.__add__(vb)
```

```
30
```

```
s1="Python"
s2="programming"
```

```
s1.__add__(s2)
```

```
'Pythonprogramming'
```

```
# Inheritance
'''
ParentClass
 |
 | ISA
 |
 ChildClass --- object -> we can invoke parent attributes

class ParentClass:
    ...

class ChildClass(ParentClass): <== Vs def functionName(argument):
----- ========== -----------

'''

class product:
  pid = 101
  pname = 'pA'

class customer:
  cname = 'cusA'

obj = customer()
print(obj.cname)
print(obj.pid)
```

```
cusA
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipython-input-3595084175.py in <cell line: 0>()
     17 obj = customer()
     18 print(obj.cname)
---> 19 print(obj.pid)

AttributeError: 'customer' object has no attribute 'pid'
```

Next steps: Explain error

```
class product:
  pid = 101
  pname = 'pA'

class customer(product): # Inheritance
  cname = 'cusA'

obj = customer()
print(obj.cname)
print(f'{obj.pid},{obj.pname}')
```

```
cusA
101,pA
```

```
'''
file: ab.py            file: sab.py                  file: p1.py
------------           --------------                =============
class Fax:             import ab                     import sab
  fax_no = 123         class Mail(ab.Fax):           obj = sab.Mail()
  def f1(self):              def f4(self):           obj.f1()
  def f2(self):              def f5(self):           obj.f2()
  def f3(self):       ------------------             obj.f3()
                                                     obj.f4()  ;  obj.f5()
--------------
```

```
print(type(10),type(11),type(0),type(-1))
```

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

```
i = 10  ##

j = int(20) ## oops style of initialization

print(type(i),type(j))
```

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

```
i+j
```

```
30
```

```
f = float(15.32)
s = str("Hello")
print(f)
print(s)
```

```
15.32
Hello
```

```
'''
 M/s: CosTools - Trading
 +---------------+
 |               |
 +---------------+
   |   |   |     |
   |   |   |  .. VendorN
   |   |  Vendor3
   |  Vendor2
   Vendor1


1st - Vendor Enrollment (Registration) - vendorName,GST,contact
2nd - Billing_Process
              |->productName,cost,Qty
              |->calculate 18% and Total including Tax
              |
              |-> Create a new billing_file - append all the billing details
                  to vendor_info.log file //date/time //FileHandling
'''
```

```
import time
class Vendor:
  def __init__(self,vName,vGST):
    self.vName = vName
    self.vGST = vGST
    print(f'Vendor {self.vName} Enrollment is done!')
  def billing(self,pName,pCost=0.0,pQty=0):
    self.pName= pName
    self.pCost = pCost
    self.pQty = pQty
    self.total = self.pCost * self.pQty
    self.tax = self.total * 0.18
    self.GS = self.tax + self.total
    with open('VendorInfo.log','a') as wobj:
      S1=f'{self.vName}\t{self.vGST}\t{self.pName}\t'
      S2=f'{self.pCost}\t{self.GS}(including Tax)\t'
      S3=f'Purchase Date/time:{time.ctime()}\n\n'
      wobj.write(S1+S2+S3)
```

```
vobj1 = Vendor("KLabs",'1ABCD23FG')
```

```
Vendor KLabs Enrollment is done!
```

```
vobj1.billing('HDD',1200,5)
```

```
vobj1.billing('SSD',2345.31,3)
```

```
vobj2 = Vendor("CTPL","7CBDE34")
```

```
Vendor CTPL Enrollment is done!
```

```
vobj2.billing("pX",3450,12)
```

```
vobj2.billing("SSD",3200,5)
```

```
vobj1.billing('pY',3423,6)
```

```
import time
class Vendor:
  def __init__(self,vName,vGST):
    self.vName = vName
    self.vGST = vGST
    print(f'Vendor {self.vName} Enrollment is done!')
  def billing(self,pName,pCost=0.0,pQty=0):
    self.pName= pName
    self.pCost = pCost
    self.pQty = pQty
    self.total = self.pCost * self.pQty
    self.tax = self.total * 0.18
    self.GS = self.tax + self.total
    with open('VendorInfo.log','a') as wobj:
      S1=f'{self.vName}\t{self.vGST}\t{self.pName}\t'
      S2=f'{self.pCost}\t{self.GS}(including Tax)\t'
      S3=f'Purchase Date/time:{time.ctime()}\n\n'
      wobj.write(S1+S2+S3)

class sales(Vendor):
  def method1(self):
    pass
```

```
### Functional style programming (or) Expression style code   - Data Clean up
### ------------------------------------------------------------+ ML
1. lambda
2. list comprehension
3. generator
4. map
5. filter
6. enumerate

lambda
------
 - python keyword
 - unnamed function - functionCall with arguments and return some Value //

 Syntax:-
 - lambda <list of args> : operation
```

```
def f1(a1,a2):
  return a1+a2

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

```
30
```

```
lambda a1,a2:a1+a2
```

```
<function __main__.<lambda>(a1, a2)>
```

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

```
30
```

```
print(type(f2))
```

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

```
f3 = lambda a1 : a1+100
f3(10)
```

```
110
```

```
f3(15)
```

```
115
```

```
def f4(a):
  return a.upper()
```

```
f4('hello')
```

```
'HELLO'
```

```
f5 = lambda a:a.upper()
f5('hello')
```

```
'HELLO'
```

```
def f5(a1):
  return a1 > 100

print(f5(150))
print(f5(50))
```

```
True
False
```

```
F5 = lambda a1 : a1 > 100
print(F5(150))
print(F5(50))
```

```
True
False
```

```
rv = F5(150)
print(rv)
```

```
True
```

```
def fx(a):
  if(a >100 and a <200):
    return a+100
  elif(a >500 and a<600):
    return a+500
  else:
    return a+1000
```

```
fx(150)
```

```
250
```

```
fx(560)
```

```
1060
```

```
fx(10)
```

```
1010
```

```
f6 = lambda a:fx(a)
f6(150)
```

```
250
```

```
f6(560)
```

```
1060
```

```
f6(10)
```

```
1010
```

```
# 2. list comprehension
# ----------------------// list append operation

L = []
for var in range(1,6): # [1,2,3,4,5]
  r = var + 100
  L.append(r)

print(L)
```

```
[101, 102, 103, 104, 105]
```

```
[var+100 for var in range(1,6)]
```

```
[101, 102, 103, 104, 105]
```

```
# [ final_Value for iterable ]
#               ----(1)----
#   ---(2)----
```

```
d={}
d['K1'] = [var+100 for var in range(1,6)]
d
```

```
{'K1': [101, 102, 103, 104, 105]}
```

```
L = []
for var in range(1,6): # [1,2,3,4,5]
  if(var >3):
    r = var + 100
    L.append(r)
  else:
    r = var + 500
    L.append(r)

print(L)
```

```
[501, 502, 503, 104, 105]
```

```
# Syntax:-
# -----------
# [True if condition else False for iterable]
#                               ----(1)---

[var+100 if var > 3 else var + 500 for var in range(1,6)]
```

```
[501, 502, 503, 104, 105]
```

```
3. generator
--------------
 |--->Function - returns an address(iterator) // generator
      .........  ....... ================

  def fx():
    return <Value> //this  is not an address
    Vs
    yield <Value> //returns address - generator
```

```
def f1():
  return 10

def f2():
  yield 10
```

```
print(type(f1),type(f2))
```

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

```
print(type(f1()))
```

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

```
print(type(f2()))
```

```
<class 'generator'>
```

```
f2()
```

```
<generator object f2 at 0x782340980d50>
```

```
# open this generator (address)
#  1. next(generator) ->Value .... StopIteration
#  2. use for loop
#  3. typecast to list
```

```
def f1():
  return 10
  print("This line won't execute")
```

```
f1()
```

```
10
```

```
f1()
```

```
10
```

```
def f2():
  yield 10
  print("Next line of code")
  yield 20,30
  yield 100+200
  yield "D1","D2"
```

```
f2()
```

```
<generator object f2 at 0x78235be31a80>
```

```
genobj = f2()
```

```
print(next(genobj))
```

```
10
```

```
print(next(genobj))
```

```
Next line of code
(20, 30)
```

```
print(next(genobj))
```

```
300
```

```
print(next(genobj))
```

```
('D1', 'D2')
```

```
print(next(genobj))
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
/tmp/ipython-input-2725223541.py in <cell line: 0>()
----> 1 print(next(genobj))

StopIteration:
```

Next steps: Explain error

```
def f2():
  yield 10
  print("Next line of code")
  yield 20,30
  yield 100+200
  yield "D1","D2"
```

```
genobj = f2()
```

```
for var in genobj:
  print(var)
```

```
10
Next line of code
(20, 30)
300
('D1', 'D2')
```

```
genobj = f2()
```

```
list(genobj) # typecast to list
```

```
Next line of code
[10, (20, 30), 300, ('D1', 'D2')]
```

```
# Excel - A1:G8 -->generator
# DB - select *from table -->generator
# ....//collective data --> generator

C:\Users\karth>dir D1
 Volume in drive C is OS
 Volume Serial Number is 3E30-0B4D

 Directory of C:\Users\karth\D1

06-01-2026  19:31    <DIR>          .
12-02-2026  19:01    <DIR>          ..
06-01-2026  19:22                24 ab.py
06-01-2026  19:45    <DIR>          D2
28-11-2025  10:06               306 p1.log
06-01-2026  19:31    <DIR>          __pycache__
               2 File(s)            330 bytes
               4 Dir(s)  693,548,417,024 bytes free

C:\Users\karth>dir D1\D2
 Volume in drive C is OS
 Volume Serial Number is 3E30-0B4D

 Directory of C:\Users\karth\D1\D2

06-01-2026  19:45    <DIR>          .
06-01-2026  19:31    <DIR>          ..
28-11-2025  10:06    <DIR>          D3
06-01-2026  19:46                97 pa.py
               1 File(s)             97 bytes
               3 Dir(s)  693,548,482,560 bytes free

C:\Users\karth>dir D1\D2\D3
 Volume in drive C is OS
 Volume Serial Number is 3E30-0B4D

 Directory of C:\Users\karth\D1\D2\D3

28-11-2025  10:06    <DIR>          .
06-01-2026  19:45    <DIR>          ..
28-11-2025  10:06    <DIR>          D4
28-11-2025  10:06               306 test.log
               1 File(s)            306 bytes
               3 Dir(s)  693,548,482,560 bytes free

C:\Users\karth>dir D1\D2\D3\D4
 Volume in drive C is OS
 Volume Serial Number is 3E30-0B4D

 Directory of C:\Users\karth\D1\D2\D3\D4

28-11-2025  10:06    <DIR>          .
28-11-2025  10:06    <DIR>          ..
28-11-2025  10:06               306 r1.log
28-11-2025  10:06               306 test.log
               2 File(s)            612 bytes
               2 Dir(s)  693,548,482,560 bytes free

C:\Users\karth>python
Python 3.14.2 (tags/v3.14.2:df79316, Dec  5 2025, 17:18:21) [MSC v.1944 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> import os
>>>
>>> os.listdir("D1")
['ab.py', 'D2', 'p1.log', '__pycache__']
>>>
>>> ## os.listdir("DirectoryName") --> ls <directoryName> (or)  dir <directoryName>
>>>
>>> os.walk("D1") # ls -R DirectoryName
<generator object walk at 0x000001EA7FB53CA0>
>>>
>>>
>>> genobj = os.walk("D1")
>>>
>>> for var in genobj:
...     print(var)
...
('D1', ['D2', '__pycache__'], ['ab.py', 'p1.log'])
('D1\\D2', ['D3'], ['pa.py'])
('D1\\D2\\D3', ['D4'], ['test.log'])
('D1\\D2\\D3\\D4', [], ['r1.log', 'test.log'])
```

```
('D1\\__pycache__', [], ['ab.cpython-310.pyc'])
>>>
>>>
```

```
def mywalk():
  yield "D1",["D2","D2"],["F1","F2","F3"]

mywalk()
```

```
<generator object mywalk at 0x78234313e8c0>
```

```
for var in mywalk():
  print(var)
```

```
('D1', ['D2', 'D2'], ['F1', 'F2', 'F3'])
```

```
map()
-----
map(function,collection) -> generator
--- ======== ----------------------------//Higher order programming
```

```
# Block style Code
L=[]
def f1(a):
  return a + 100

for var in [10,20,30,40,50]:
  r = f1(var)
  L.append(r)

print(L)
```

```
[110, 120, 130, 140, 150]
```

```
list(map(lambda a:a + 100,[10,20,30,40,50]))
```

```
[110, 120, 130, 140, 150]
```

```
map(lambda a:a + 100,[10,20,30,40,50])
```

```
<map at 0x7823409e9ab0>
```

```
obj = map(lambda a:a + 100,[10,20,30,40,50])
list(obj)
```

```
[110, 120, 130, 140, 150]
```

```
obj = map(lambda a:a + 100,[10,20,30,40,50])
for var in obj:
  print(var)
```

```
110
120
130
140
150
```

```
obj = map(lambda a:a + 100,[10,20,30,40,50])
print(next(obj))
```

```
110
```

```
print(next(obj))
```

```
120
```

```
print(next(obj))
```

```
130
```

```
print(next(obj))
```

```
140
```

```
print(next(obj))
```

```
150
```

```
print(next(obj))
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
/tmp/ipython-input-396705727.py in <cell line: 0>()
----> 1 print(next(obj))

StopIteration:
```

Next steps: Explain error

```
open('emp.csv','r')
```

```
<_io.TextIOWrapper name='emp.csv' mode='r' encoding='utf-8'>
```

```
fobj = open('emp.csv','r')
####  Generator
for var in fobj:
  print(var.strip())
```

```
eid,ename,edept,ecity,ecost
101,raj,sales,pune,1000
102,leo,prod,bglore,2301
230,raj,prod,pune,2300
450,shan,sales,bglore,3401
542,anu,HR,mumbai,4590
321,bibu,sales,hyd,5419
651,ram,hr,bglore,3130
541,leo,admin,chennai,4913
652,karthik,sales,bglore,3405
```

```
import html_Template_code

html_Template_code.fx()
html_Template_code.fy()

import html_Template_code as ht
ht.fx()                       ---//user defined
ht.fy()

>>>
>>> import os
>>> os.getcwd()
'C:\\Users\\karth'
>>>
>>> getcwd()
Traceback (most recent call last):
  File "<python-input-4>", line 1, in <module>
    getcwd()
    ^^^^^^
NameError: name 'getcwd' is not defined
>>>
>>> os.getcwd()
'C:\\Users\\karth'
>>>
>>> # from <module> import <member>
>>> # -----
>>>
>>> from os import getcwd
>>> getcwd()
'C:\\Users\\karth'
>>>
```

```
list(map(lambda a:a+100,[10,20,30,40,50]))
```

```
[110, 120, 130, 140, 150]
```

```
# map(function,collection) ->Generator
# ---
#
# filter(function,collection) ->Generator
# ------//filter True value only
```

```
L = []
def f1(a):
  if(a >3):
```

```
    return True
  else:
    return False

for var in [1,2,3,4,5,6]:
  r = f1(var)
  L.append(r)

print(L)
```

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

```
list(map(lambda a: a>3,[1,2,3,4,5,6]))
```

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

```
list(filter(lambda a: a>3,[1,2,3,4,5,6]))
```

```
[4, 5, 6]
```

```
#help(enumerate)

enumerate("hello")
# enumerate(collection,start=0)
```

```
<enumerate at 0x78233a21b740>
```

```
for var in enumerate("hello"):
  print(var)
```

```
(0, 'h')
(1, 'e')
(2, 'l')
(3, 'l')
(4, 'o')
```

```
for var in enumerate("hello",1):
  print(var)
```

```
(1, 'h')
(2, 'e')
(3, 'l')
(4, 'l')
(5, 'o')
```

```
for var in enumerate("hello",5):
  print(var)
```

```
(5, 'h')
(6, 'e')
(7, 'l')
(8, 'l')
(9, 'o')
```

```
for var in enumerate("hello",10):
  print(var)
```

```
(10, 'h')
(11, 'e')
(12, 'l')
(13, 'l')
(14, 'o')
```

```
fobj = open('emp.csv','r')
for var in enumerate(fobj):
  print(f'{var[0]} - {var[1].strip()}')
```

```
0 - eid,ename,edept,ecity,ecost
1 - 101,raj,sales,pune,1000
2 - 102,leo,prod,bglore,2301
3 - 230,raj,prod,pune,2300
4 - 450,shan,sales,bglore,3401
5 - 542,anu,HR,mumbai,4590
6 - 321,bibu,sales,hyd,5419
7 - 651,ram,hr,bglore,3130
8 - 541,leo,admin,chennai,4913
9 - 652,karthik,sales,bglore,3405
```

```
fobj = open('emp.csv','r')
for var in enumerate(fobj,1):
  print(f'{var[0]} - {var[1].strip()}')
```

```
1 - eid,ename,edept,ecity,ecost
2 - 101,raj,sales,pune,1000
3 - 102,leo,prod,bglore,2301
4 - 230,raj,prod,pune,2300
5 - 450,shan,sales,bglore,3401
6 - 542,anu,HR,mumbai,4590
7 - 321,bibu,sales,hyd,5419
8 - 651,ram,hr,bglore,3130
9 - 541,leo,admin,chennai,4913
10 - 652,karthik,sales,bglore,3405
```

```
for line, data in enumerate(open('emp.csv','r'),1):
  print(f'{line} - {data.strip()}')
```

```
1 - eid,ename,edept,ecity,ecost
2 - 101,raj,sales,pune,1000
3 - 102,leo,prod,bglore,2301
4 - 230,raj,prod,pune,2300
5 - 450,shan,sales,bglore,3401
6 - 542,anu,HR,mumbai,4590
7 - 321,bibu,sales,hyd,5419
8 - 651,ram,hr,bglore,3130
9 - 541,leo,admin,chennai,4913
10 - 652,karthik,sales,bglore,3405
```

```
for line, data in enumerate(open('emp.csv','r'),1):
  if(line >3 and line <9):
    print(f'{line} - {data.strip()}')
```

```
4 - 230,raj,prod,pune,2300
5 - 450,shan,sales,bglore,3401
6 - 542,anu,HR,mumbai,4590
7 - 321,bibu,sales,hyd,5419
8 - 651,ram,hr,bglore,3130
```

```
for line, data in enumerate(open('emp.csv','r'),1):
  if(line >3 and line <9):
    continue
  else:
    print(f'{line} - {data.strip()}')
```

```
1 - eid,ename,edept,ecity,ecost
2 - 101,raj,sales,pune,1000
3 - 102,leo,prod,bglore,2301
9 - 541,leo,admin,chennai,4913
10 - 652,karthik,sales,bglore,3405
```

```
list(open('emp.csv','r')) # generator
```

```
['eid,ename,edept,ecity,ecost\n',
 '101,raj,sales,pune,1000\n',
 '102,leo,prod,bglore,2301\n',
 '230,raj,prod,pune,2300\n',
 '450,shan,sales,bglore,3401\n',
 '542,anu,HR,mumbai,4590\n',
 '321,bibu,sales,hyd,5419\n',
 '651,ram,hr,bglore,3130\n',
 '541,leo,admin,chennai,4913\n',
 '652,karthik,sales,bglore,3405']
```

```
d={}
d['CSV'] = list(open('emp.csv','r'))
d
```

```
{'CSV': ['eid,ename,edept,ecity,ecost\n',
  '101,raj,sales,pune,1000\n',
  '102,leo,prod,bglore,2301\n',
  '230,raj,prod,pune,2300\n',
  '450,shan,sales,bglore,3401\n',
  '542,anu,HR,mumbai,4590\n',
  '321,bibu,sales,hyd,5419\n',
  '651,ram,hr,bglore,3130\n',
  '541,leo,admin,chennai,4913\n',
  '652,karthik,sales,bglore,3405']}
```

```
##############
```

```
'''
read an emp.csv file  - 5th line to 8th line    # enumerate
```

```
                    |--> sales --> QA dept
                                    |--->display Emp Name,Dept,City
'''
```

```
import re
for line,data in enumerate(open('emp.csv','r'),1):
  if(line >=3 and line <=8):
    if(re.search('sales',data)):
      result = re.sub('sales','QA',data,1,re.I)
      print(result.strip())
```

```
450,shan,QA,bglore,3401
321,bibu,QA,hyd,5419
```

```
import sys
sys.platform
```

```
'linux'
```

```
import os
os.listdir(".")
for var in os.popen("cat -n emp.csv"):
  print(var.strip())
```

```
1       eid,ename,edept,ecity,ecost
2       101,raj,sales,pune,1000
3       102,leo,prod,bglore,2301
4       230,raj,prod,pune,2300
5       450,shan,sales,bglore,3401
6       542,anu,HR,mumbai,4590
7       321,bibu,sales,hyd,5419
8       651,ram,hr,bglore,3130
9       541,leo,admin,chennai,4913
10      652,karthik,sales,bglore,3405
```

```
for line,data in enumerate(open('emp.csv','r'),1): # enumerate - iterate line by line
  if(line >=3 and line <=8):
    if(re.search('sales',data)): # search sales pattern
      result = re.sub('sales','QA',data,1,re.I) # replace sales ->QA
      L = re.split(",",result)  # split each line into multiple value
      print(f'{L[0]}\t{L[1]}\t{L[2]}')
```

```
450     shan    QA
321     bibu    QA
```

```
s='101,raj,sales,sales'
re.sub('sales','QA',s,1)
```

```
'101,raj,QA,sales'
```

```
import sys
sys.version
```

```
'3.12.12 (main, Oct 10 2025, 08:52:57) [GCC 11.4.0]'
```

```
##
#  python <---------------> json
#

import json

net_info={}
net_info['interface'] = ['eth0','eth1','eth2']
net_info['config']=[{'f1':'/etc/network.cfg'},{'f1':'/etc/sys.cfg'},{'f1':'/etc/hosts'}]
net_info['onboot'] = 'yes'

print(type(net_info))
```

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

```
print(net_info)
```

```
{'interface': ['eth0', 'eth1', 'eth2'], 'config': [{'f1': '/etc/network.cfg'}, {'f1': '/etc/sys.cfg'}, {'f1': '/etc/hosts'}]
```

```
# from python --> json
#
```

```
# json.dumps(python_data)
json.dumps(net_info)
```

```
'{"interface": ["eth0", "eth1", "eth2"], "config": [{"f1": "/etc/network.cfg"}, {"f1": "/etc/sys.cfg"}, {"f1": "/etc/host
s"}], "onboot": "yes"}'
```

```
json_data = json.dumps(net_info) # from python --> to json
```

```
## from json ---> python
python_data = json.loads(json_data) # from json ->python
python_data
```

```
{'interface': ['eth0', 'eth1', 'eth2'],
 'config': [{'f1': '/etc/network.cfg'},
  {'f1': '/etc/sys.cfg'},
  {'f1': '/etc/hosts'}],
 'onboot': 'yes'}
```

```
print(json_data)
```

```
{"interface": ["eth0", "eth1", "eth2"], "config": [{"f1": "/etc/network.cfg"}, {"f1": "/etc/sys.cfg"}, {"f1": "/etc/hosts"}]
```

```
json.dumps(True)
```

```
'true'
```

```
net_info
```

```
{'interface': ['eth0', 'eth1', 'eth2'],
 'config': [{'f1': '/etc/network.cfg'},
  {'f1': '/etc/sys.cfg'},
  {'f1': '/etc/hosts'}],
 'onboot': 'yes'}
```

```
with open('test.json','w') as wobj:
  json.dump(net_info,wobj) #  Vs json.dumps(python_Data)
```

```
with open('test.json','r') as fobj:
  python_data = json.load(fobj) # from json file to python_structure
```

```
python_data
```

```
{'interface': ['eth0', 'eth1', 'eth2'],
 'config': [{'f1': '/etc/network.cfg'},
  {'f1': '/etc/sys.cfg'},
  {'f1': '/etc/hosts'}],
 'onboot': 'yes'}
```

```
def connect(arg):
  class Connection:
    def __init__(self,arg):
      self.msg=arg
    def method1(self):
      return self.msg
  obj = Connection(arg)
  return obj
```

```
connect("my-test-data")
```

```
<__main__.connect.<locals>.Connection at 0x78233a3c2fc0>
```

```
myobj = connect("my-test-data")
myobj.method1()
```

```
'my-test-data'
```

```
import sqlite3
```

```
sqlite3.connect("test1.db")
```

```
<sqlite3.Connection at 0x78233a266110>
```

```
conn = sqlite3.connect("test1.db")
```

```
sth = conn.cursor()
```

```
sth.execute("create table product(pid INT,pname Text,pcost INT)")
```

```
<sqlite3.Cursor at 0x782340a30fc0>
```

```
sth.execute("insert into product(pid,pname,pcost) values(101,'pA',1000)")
```

```
<sqlite3.Cursor at 0x782340a30fc0>
```

```
sth.execute("insert into product(pid,pname,pcost) values(102,'pB',2000)")
sth.execute("insert into product(pid,pname,pcost) values(103,'pC',3000)")
sth.execute("insert into product(pid,pname,pcost) values(104,'pD',4000)")
sth.execute("insert into product(pid,pname,pcost) values(105,'pE',5000)")
```

```
<sqlite3.Cursor at 0x782340a30fc0>
```

```
sth.execute("select *from product")
```

```
<sqlite3.Cursor at 0x782340a30fc0>
```

```
sth.fetchall()
```

```
[(101, 'pA', 1000),
 (102, 'pB', 2000),
 (103, 'pC', 3000),
 (104, 'pD', 4000),
 (105, 'pE', 5000)]
```

```
sth.execute("select *from product")
sth.fetchone()
```

```
(101, 'pA', 1000)
```

```
sth.fetchone()
```

```
(102, 'pB', 2000)
```

```
sth.fetchone()
```

```
(103, 'pC', 3000)
```

```
sth.execute("select *from product")
for var in sth: # generator
  print(var)
```

```
(101, 'pA', 1000)
(102, 'pB', 2000)
(103, 'pC', 3000)
(104, 'pD', 4000)
(105, 'pE', 5000)
```

```
sth.execute("select *from product")
list(sth) # typecast to list - generator
```

```
[(101, 'pA', 1000),
 (102, 'pB', 2000),
 (103, 'pC', 3000),
 (104, 'pD', 4000),
 (105, 'pE', 5000)]
```

```
conn.commit()
```

```
conn.close()
```

```
import sqlite3
conn = sqlite3.connect("test1.db")
sth = conn.cursor()
sth.execute("select *from product")
list(sth)
```

```
[(101, 'pA', 1000),
 (102, 'pB', 2000),
 (103, 'pC', 3000),
 (104, 'pD', 4000),
 (105, 'pE', 5000)]
```

```
sth.execute("insert into product(pid,pname,pcost) values(106,'pF',1345)")
```

```
<sqlite3.Cursor at 0x782340a32340>
```

```
va = 107
vb = 'pG'
vc = 3323

sth.execute("insert into product(pid,pname,pcost) values(?,?,?)",(va,vb,vc))
```

```
<sqlite3.Cursor at 0x782340a32340>
```

```
sth.execute("select *from product")
list(sth)
```

```
[(101, 'pA', 1000),
 (102, 'pB', 2000),
 (103, 'pC', 3000),
 (104, 'pD', 4000),
 (105, 'pE', 5000),
 (106, 'pF', 1345),
 (107, 'pG', 3323)]
```

```
sth.execute("select *from product")
records = list(sth)
```

```
d={}
d['SQLITE'] = records
d
```

```
{'SQLITE': [(101, 'pA', 1000),
  (102, 'pB', 2000),
  (103, 'pC', 3000),
  (104, 'pD', 4000),
  (105, 'pE', 5000),
  (106, 'pF', 1345),
  (107, 'pG', 3323)]}
```

```
json_data = json.dumps(d) # converting to json
```

```
python_data = json.loads(json_data) # from json to python
```

```
python_data
```

```
{'SQLITE': [[101, 'pA', 1000],
  [102, 'pB', 2000],
  [103, 'pC', 3000],
  [104, 'pD', 4000],
  [105, 'pE', 5000],
  [106, 'pF', 1345],
  [107, 'pG', 3323]]}
```

```
##
Recap Module
        |-->existing python file (filename.py)
                                    ---------

E:\> file: ab.py
    --------------
     def display():
         print("OK")
     myvar = 100
     ---------------
E:\> python ab.py
E:\>

                                                E:\> python {Enter}
E:\> file: test1.py                             >>>  import ab
 -------------------                            >>>
   def report():                                >>> ab.display()
        print("Test page")                       OK
   def view():                                  >>>  import test1
        print("View block")                     120
   var = 120                                    Test page
   print(var)                                   View block
   report()                                     >>>
   view()
  -----------------------
E:\> python test1.py {Enter}
```

```
E:\> python test1.py {Enter}
120
Test page
View block


__name__ ===> display current module

vi p1.py
var = 120         __main__.var | 120  <==
def f1():         -------------|---------
    print("OK")  __main__.f1   | 0x1234
-----------------==========================
:wq

vi p2.py
import p1           __main__.fname | 'emp.csv'
print(p1.var)      ----------------|----------
fname = "emp.csv"      p1.var      | 120  <===


file: mycode.py
==================
def f1():
  print("test-1")
def f2():
  print("Test-2")
def f3():
  print("Test-3")

if __name__ == '__main__':
  f1()
  f2()
  f3()
============================
E:\>python mycode.py {Enter}
test-1
Test-2
Test-3

E:\> python {Enter}
>>>
>>> import mycode
>>>
>>> mycode.f2()
Test-2
>>>

##############################################
project/
     p1.py p2.py p3.py p4.py .. p50.py
     Sub1/
        p51.py p52.py .. p60.py
     Sub2/
        p61.py
        Sub2-1/
            p62.py p63.py

=========================================
To run this project -> import all the 63 python files

Commandline Steps
====================
Step 1: Create a project directory (mkdir  myproject)
Step 2: Copy/collect all the python files to into myproject directory - copy py files -> myproject/

Step 3: Create packge initialization file (special file) ==> __init__.py

Step 4:  copy the external symbols(variable/function/class) to __init__.py
                from <module> import <member>
                from <myproject>.p1 import f1,f2,f3
                from <myproject>.p2 import *
                ..
                from myproject.Sub1.p51 import *
                                    ===        =====
Step 5:  Test your package/directory

 import myproject
 myproject.f1()
 myproject.f2()
 ..
```

```
>>> import ERP
>>> ERP.count()
Total no.of sales count:120
>>>
>>> ERP.fx()
production details
>>>
>>> ERP.fy()
'No.of product Count:500'
>>>
>>> ERP.display()
List of customer records
>>>
>>> import os
>>> os.__file__
'C:\\Python314\\Lib\\os.py'
>>> import json
>>>
>>> json.__file__
'C:\\Python314\\Lib\\json\\__init__.py'
>>>
>>> ERP.__file__
'C:\\Users\\karth\\ERP\\__init__.py'
>>>
>>>
>>> import ERP
>>> ERP
<module 'ERP' from 'C:\\Users\\karth\\ERP\\__init__.py'>
>>>
>>> import json
>>> json
<module 'json' from 'C:\\Python314\\Lib\\json\\__init__.py'>
>>>
>>> import re
>>> re
<module 're' from 'C:\\Python314\\Lib\\re\\__init__.py'>
>>> import os
>>> os
<module 'os' (frozen)>
>>> os.__file__
'C:\\Python314\\Lib\\os.py'
>>>

######################   End of Day5 session  ################################
```