```
Regualr Expression
------------------
Linux -> grep sed awk //commands
Winx -> findstr //commands

# search a pattern
# substitute oldpattern ->replaceNewValue
|
In python - Regx (Search ; Substitute ; Split)

Search     -  Search pattern
Substitue  -  Find and replace
Split      -  Formatted style results

import re  <== module

re.search() //Function

re.search('Pattern_String','input_String',re.I) # re.IGNORECASE
|                                          ===== defaultArgs
|
re.search('Pattern_String','input_String') -> <pattern_object> / None
-----------------------------------------
   |-> bool(re.search('Pattern_String','input_String')) -><pattern_object>/ True
   |                                                    -> None /False
```

```
import re
```

```
re.search('sales','101,raj,sales,pune,1000')
```

```
<re.Match object; span=(8, 13), match='sales'>
```

```
re.search('prod','101,raj,sales,pune,1000')
```

```
bool(re.search('sales','101,raj,sales,pune,1000'))
```

```
True
```

```
bool(re.search('prod','101,raj,sales,pune,1000'))
```

```
False
```

```
if(re.search('sales','101,raj,sales,pune,1000')):
  print("Yes - pattern is matched")
else:
  print("Sorry - pattern is not matched")
```

```
Yes - pattern is matched
```

```
if(re.search('prod','101,raj,sales,pune,1000')):
  print("Yes - pattern is matched")
else:
  print("Sorry - pattern is not matched")
```

```
Sorry - pattern is not matched
```

```
if(re.search('SALES','101,raj,sales,pune,1000')):
  print("Yes - pattern is matched")
else:
  print("Sorry - pattern is not matched")
```

```
Sorry - pattern is not matched
```

```
if(re.search('SALES','101,raj,sales,pune,1000',re.I)): # Vs if('SALES' in 'sales') -> False
  print("Yes - pattern is matched")
else:
  print("Sorry - pattern is not matched")
```

```
Yes - pattern is matched
```

```
help(re.search)
```

```
Help on function search in module re:

search(pattern, string, flags=0)
```

```
    Scan through string looking for a match to the pattern, returning
    a Match object, or None if no match was found.
```

```
input_var = "dev server LB value is:1.53 cpu usage:98.44"

if(re.search("1.53",input_var)):
  print(f'Found - {input_var}')
else:
  print('Not-Found')
```

```
Found - dev server LB value is:1.53 cpu usage:98.44
```

```
input_var = "dev server LB value is:1.53 cpu usage:98.44"

if(re.search(1.53,input_var)):
  print(f'Found - {input_var}')
else:
  print('Not-Found')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-1875523430.py in <cell line: 0>()
      1 input_var = "dev server LB value is:1.53 cpu usage:98.44"
      2
----> 3 if(re.search(1.53,input_var)):
      4   print(f'Found - {input_var}')
      5 else:

                                    1 frames
/usr/lib/python3.12/re/__init__.py in _compile(pattern, flags)
    297             return pattern
    298         if not _compiler.isstring(pattern):
--> 299             raise TypeError("first argument must be string or compiled pattern")
    300         if flags & T:
    301             import warnings

TypeError: first argument must be string or compiled pattern
```

Next steps: Explain error

```
input_var = "dev server LB value is:1.53 cpu usage:98.44"

if(re.search(str(1.53),input_var)):
  print(f'Found - {input_var}')
else:
  print('Not-Found')
```

```
Found - dev server LB value is:1.53 cpu usage:98.44
```

```
re.search('sale','asdfdsadsfds2343242sd@#$#@ sales asfsadf')
```

```
<re.Match object; span=(27, 31), match='sale'>
```

```
import re # <== Step 1

if(re.search('pattern_string','input_string',re.I)):
  # True block operation
else:
  # False block operation
```

```
# grep sales emp.csv <== Linux Commandline
#
# findstr sales emp.csv <== winx commandline

# open a emp.csv file                     --> open('emp.csv','r') ->fobj
# read the file content - line by line    ->
# search a pattern from input line        -> re.search()
# print/display - matched pattern line only   -> if(re.search('pattern','')):

fobj = open('emp.csv','r')
L = fobj.readlines()
fobj.close()

for var in L:
  if(re.search('sales',var)):
    print(var.strip())
```

```
101,raj,sales,pune,1000
450,shan,sales,bglore,3401
321,bibu,sales,hyd,5419
652,karthik,sales,bglore,3405
```

```
def my_grep(pattern,fname):
  fobj = open(fname,'r')
  L = fobj.readlines()
  fobj.close()
  for var in L:
    if(re.search(pattern,var,re.I)):
      print(var.strip())
```

```
my_grep('sales','emp.csv')
```

```
101,raj,sales,pune,1000
450,shan,sales,bglore,3401
321,bibu,sales,hyd,5419
652,karthik,sales,bglore,3405
```

```
my_grep('pune','emp.csv')
```

```
101,raj,sales,pune,1000
230,raj,prod,pune,2300
```

```
my_grep('HR','emp.csv')
```

```
542,anu,HR,mumbai,4590
651,ram,hr,bglore,3130
```

```
input_var = 'raj,sales,pune,101,sales'
re.search('sales',input_var)
```

```
<re.Match object; span=(4, 9), match='sales'>
```

```
# re.search() -> re.search('pattern_string','input_string',re.I) --> <ack>/None
#
# Vs
#
# re.findall() -> re.findall('pattern_string','input_string',re.I) --> [matched pattern results] / []

input_var = 'raj,sales,pune,101,sales'
re.findall('sales',input_var)
```

```
['sales', 'sales']
```

```
s = '101,raj,sales,pune,1000'

# re.sub('oldpattern_str','replace_string','input_string') --> output_str
#                                                                |--> replaced string if old pattern is matched with input_
#                                                                |--> display original input_string if oldpattern is not ma

re.sub('sales','QA',s)
```

```
'101,raj,QA,pune,1000'
```

```
re.sub('Prod','QA',s)
```

```
'101,raj,sales,pune,1000'
```

```
re.sub('sales','QA','101,sales,raj,sales,pune,sales,X,sales,4404,sales,SALES') # default substitute is global substitue
#
```

```
'101,QA,raj,QA,pune,QA,X,QA,4404,QA,SALES'
```

```
# help(re.sub)
re.sub('sales','QA','101,sales,raj,sales,pune,sales,X,sales,4404,sales,SALES',1)
```

```
'101,QA,raj,sales,pune,sales,X,sales,4404,sales,SALES'
```

```
re.sub('sales','QA','101,sales,raj,sales,pune,sales,X,sales,4404,sales,SALES',2)
```

```
'101,QA,raj,QA,pune,sales,X,sales,4404,sales,SALES'
```

```
re.sub('sales','QA','101,SALES,raj,sales,pune,sales,X,sales,4404,sales,SALES',2)
```

```
'101,SALES,raj,QA,pune,QA,X,sales,4404,sales,SALES'
```

```
re.sub('sales','QA','101,SALES,raj,sales,pune,sales,X,sales,4404,sales,SALES',2,re.I)
```

```
'101,QA,raj,QA,pune,sales,X,sales,4404,sales,SALES'
```

```
# Regx Chars
# --------------------
'''
Regx
 |----> Basic Regular Expression (BRE) - Single pattern
 |----> Extended Regular Expression (ERE) - Multiple pattern Search
BRE
===
^ $  ^pattern$  .  .*  [] ^[] []$  [^]  ^$   \s  <=== Regx Chars

^pattern - matches the pattern line starts with

re.search('^sales',input_str)

sales,pune.... //OK
-----
raj,sales,bglore //Not-Matched

re.search('^\s',input_str) --> line starts with space chars

-------------------
pattern$ - matches the pattern line ends with

re.search('sales$',input_var)

sales,raj # not matched
raj,sales # match
asdfss sales # match
leo,sales, # not matched
        ---



>>> import os
>>> os.listdir('.')
['.anaconda', '.bash_history', '.cache', '.composio',
'.conda', '.condarc', '.continuum', '.env', '.gitconfig', '.idlerc', '.influxdb', '.influx_history',
'.ipynb_checkpoints', '.ipython', '.jupyter', '.lesshst', '.matplotlib', '.mem0', '.minttyr..]

>>> files = os.listdir('.')
>>>
>>> # How to filter - python files (.py) ?
>>>
>>> import os
>>> files = os.listdir('.')
>>> len(files)
283
>>> import re
>>>
>>> L=['p1.pdf','p2.py','p3.c','pq.py','index.html']
>>>
>>> # re.search('py$',var_str)
>>> for var in L:
...     if(re.search('py$',var)):
...         print(var)
...
p2.py
pq.py
>>>
>>> L=['p1.pdf','p2.py','p3.c','pq.py','index.html','pydata.txt','pyload']
>>>
>>> for var in L:
...     if(re.search('py$',var)):
...         print(var)
...
p2.py
pq.py
>>>
>>> for var in L:
...     if(re.search('py',var)):
...         print(var)
...
p2.py
pq.py
pydata.txt
pyload
```

```
>>>
>>> files = os.listdir(".")
>>> for var in files:
...     if(re.search('py$',var)):
...         print(var)
...
app.py
csv_rag_app.py
demo1.py
##################################

^pattern
pattern$

^pattern$  <--- pattern only
---->--
--<----



. (dot)
. - 1character
.. - 2chars

^sales.dept$  --> sales,dept //OK
                  sales dept //OK
                  sales   dept //Not-matched
                  sales<CHAR>dept //OK
'''
```

```
S1='raj,sales'
S2='sales,pune'
S3='sales,'
S4='sales:'
S5='sales'

print(re.findall('^sales$',S1))
print(re.findall('^sales$',S2))
print(re.findall('^sales$',S3))
print(re.findall('^sales$',S4))
print(re.findall('^sales$',S5))
```

```
[]
[]
[]
[]
['sales']
```

```
course="Unix Shell Script"
print(re.findall("^Unix\\sShell\\sScript$",course))
#
print(re.findall("^UnixShellScript$",course))
```

```
['Unix Shell Script']
[]
```

```
S1='raj,sales'
S2='sales,pune'
S3='sales,'
S4='sales:'
S5='sales'

print(re.findall('^sales.$',S1))
print(re.findall('^sales.$',S2))
print(re.findall('^sales.$',S3))
print(re.findall('^sales.$',S4))
print(re.findall('^sales.$',S5))
```

```
[]
[]
['sales,']
['sales:']
[]
```

```
'''
# ^..   <=== line starts with any two chars
# ..$   <=== line ends with any two chars
#
>>> with open("/etc/passwd","r") as fobj:
...     for var in fobj.readlines():
...             var=var.strip()
...             if(re.search("^s.*system.*manage.*n$",var,re.I)):
```

```
...                     print(var)
...
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
>>>

student@paka:~$ grep -i ^s.*system.*manage.*n$ /etc/passwd
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
'''
```

```
'''
[] - 1char

[Aav]run
---------   -> Arun arun vrun //OK

[Aa][rR]un
-------------> Arun ARun arun aRun
[a-z]  - any lower
[A-Z]  - any upper
[a-zA-Z] - any alpha
[0-9] - any digits
[a-zA-Z0-9] - any alpha number

re.search("^[a-zA-Z].*[0-9]$",input_var)
           -----------------
            |_ line stats with any alpha followed by anyText ends with any signle digit

^[A-Z] -- Line stats with any uppercase chars

 [A-Z]$ -- line ends with any uppercase chars

 [^A-Z] -- NOT matching any uppercase chars

[^a-zA-Z0-9] --> NOT matching alpha and number ==>Matches space,specialchars

[^a-zA-Z0-9\s]  --> Match specialchars only

[A-Za-z0-9] --> \w  re.search("\w",input_var)

[0-9] -->\d   ==> re.search("^\d\d",input_var)
                           --------//line starts with any two digits

[^a-zA-Z0-9\s]  --> Match specialchars only <== [^\w\s]

^$ --> Matches Empty line
'''
```

```
n = input('Enter any two digits:')
print(n)
```

```
Enter any two digits:asfdsfs
asfdsfs
```

```
n = input('Enter any two digits:')
if(re.search("^[0-9][0-9]$",n)):
  print("Valid-Format")
  total = int(n) + 100
  print(total)
else:
  print('Invalid-Format')
```

```
Enter any two digits:34
Valid-Format
134
```

```
n = input('Enter any two digits:')
if(re.search("^[0-9][0-9]$",n)):
  print("Valid-Format")
  total = int(n) + 100
  print(total)
else:
  print('Invalid-Format')
```

```
Enter any two digits:452
Invalid-Format
```

```
'''
student@paka:~$ cat -n process.log
     1      PID TTY          TIME CMD
     2
     3
     4      310 pts/0    00:00:00 bash
     5
     6
     7      901 pts/0    00:00:00 ps
student@paka:~$ python3
Python 3.12.3 (main, Nov  6 2025, 13:44:16) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> import re
>>>
>>> with open("process.log","r") as fobj:
...     for var in fobj.readlines():
...             if(re.search("^$",var)):
...                     print(var)
...







## Display Non-Empty contents

>>> with open("process.log","r") as fobj:
...     for var in fobj.readlines():
...             if(re.search("^$",var)):
...                     continue
...             else:
...                     print(var.strip()) # non-empty lines
...
PID TTY          TIME CMD
310 pts/0    00:00:00 bash
901 pts/0    00:00:00 ps
>>>
>>>
>>> with open("process.log","r") as fobj:
...     for var in fobj.readlines():
...             if(not re.search("^$",var)):
...                     print(var.strip())
...
PID TTY          TIME CMD
310 pts/0    00:00:00 bash
901 pts/0    00:00:00 ps
'''
```

```
n=input('Enter n value:')
re.findall('..',n) # any two chars
```

```
Enter n value:abcd
['ab', 'cd']
```

```
## Extend Regular Expression - ERE  -- Multiple pattern search
# linux grep -E (or) egrep
# ------------------------

#  | () + {}
# -------------//ERE

#  pattern1 | pattern2  -- any one pattern - any order - matched - OK  -- Like logical OR

var = '101,raj,sales,pune,1000'

re.findall('sales|QA|prod',var)
```

```
['sales']
```

```
# (pattern1) (pattern2) -- both pattern should match same order - like Logical AND
# ---------  -----------
var = '101,raj,sales,pune,1000'

re.findall("(sales)(pune)",var)
### salespune -> OK
```

```
[]
```

```
re.findall("(sales).(pune)",var)
## sales<CHAR>pune
```

```
[('sales', 'pune')]
```

```
# URL Test - url name starts with http (or) https followed any name/text ends with org (or) com

# http://www.abc.com  - OK
# http://www.abc.org  - OK
# https://www.abc.com - OK
# https://www.abc.org - OK

# https://www.abc.edu - NOT-OK
# http://www.abc.in -   NOT-OK
# ftp://www.abc.com  - NOT-OK

# pattern='^https|http.*org$|com$'

pattern = "^(https|http).*(org|com)$"

input_list=['http://www.abc.com','http://www.abc.org','https://www.abc.com',
'https://www.abc.org','https://www.abc.edu','ftp://www.abc.com']

for var in input_list:
  if(re.search(pattern,var)):
    print(var)
```

```
http://www.abc.com
http://www.abc.org
https://www.abc.com
https://www.abc.org
```

```
# +  --> 1 or more

# ab+c --> abc abbbbbbbbbbbbbbbbbbbbbbbbbbbbbc //OK
#
#  abbbbbbbbbbb,bbbbbbbbbbc //Not-Matched

"^[a-zA-Z][0-9]+[a-z]$"
# -------------=====______
# A5g
# A342423423432423423234234324237f
# --------------------------------//OK
```

```
LB="06:50:19 up  2:25,  1 user,  load average: 0.00, 0.00, 0.00 in YEAR 2026 Month of FEB"
re.findall("[0-9]+",LB) # filter 1 or more digits
```

```
['06', '50', '19', '2', '25', '1', '0', '00', '0', '00', '0', '00', '2026']
```

```
print(re.findall("[A-Z]",LB))
print("\n")
print(re.findall("[A-Z]+",LB))
```

```
['Y', 'E', 'A', 'R', 'M', 'F', 'E', 'B']


['YEAR', 'M', 'FEB']
```

```
# {} - range - IRE
# ---
# <Pattern>{n} -- n times
# ab{3}c ---------> abbbc //OK   abc abbc abbbbc //Not-Matched
'''
^[0-9]{2}$
-----------
^[A-Za-z][A-Za-z][A-Za-z][0-9][0-9][0-9][0-9][0-9][a-z][a-z]$ <<=== Break

^[A-Za-z]{3}[0-9]{5}[a-z]{2}$ <== ERE

# <Pattern>{n,} -- minimum 'n' times  - maximum - nolimit
#
ab{3,}c ---> abbbc abbbbbbbbbbbbbbbbbbbbbbc //OK   abc abbc //Not-Matched


# <Pattern>{n,m} -- minimum 'n' times  - maximum - 'm' times

ab{3,5}c --> abbbc abbbbc abbbbbc //OK ; abc abbc abbbbbbc //not-matched
'''
```

```
input_var = "raj,sales,pune"
re.sub("^sales","QA",input_var)
```

```
'raj,sales,pune'
```

```
input_var = "sales,pune"
re.sub("^sales","QA",input_var)
```

```
'QA,pune'
```

```
input_var = '101,raj,sales,pune,1000'
# delete sales word from given input

print(re.sub("sales","",input_var)) # target string is empty
```

```
101,raj,,pune,1000
```

```
print(re.sub("sales.","",input_var)) # target string is empty
```

```
101,raj,pune,1000
```

```
s="101,raj,sales,pune,1000"
s.split(",")
```

```
['101', 'raj', 'sales', 'pune', '1000']
```

```
s="101,raj:sales-pune(2000"
s.split(",")
```

```
['101', 'raj:sales-pune(2000']
```

```
# re.split() ---> re.split(pattern_string,input_string) -->output_list
s="101,raj:sales-pune(2000"
re.split(",",s)
```

```
['101', 'raj:sales-pune(2000']
```

```
# Filter list of chars
re.findall("[^\\w\\s]",s)
```

```
[',', ':', '-', '(']
```

```
re.sub("[^\\w\\s]","|",s) # replace all the specialchars --> |
```

```
'101|raj|sales|pune|2000'
```

```
# split - input text based on specialchars
re.split("[^\\w\\s]",s)  ## Vs s.split(<Any Single sep>)
```

```
['101', 'raj', 'sales', 'pune', '2000']
```

```
####
var = 120
print(VAR)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-2908241345.py in <cell line: 0>()
      1 ####
      2 var = 120
----> 3 print(VAR)

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

Next steps: Explain error

```
L=[10,20,30]
print(L[5])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/tmp/ipython-input-4092956144.py in <cell line: 0>()
      1 L=[10,20,30]
----> 2 print(L[5])

IndexError: list index out of range
```

Next steps: Explain error

```
print("A"+5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-623900353.py in <cell line: 0>()
----> 1 print("A"+5)

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

Next steps: Explain error

```
int("ab")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipython-input-3758442271.py in <cell line: 0>()
----> 1 int("ab")

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

Next steps: Explain error

```
d={'K1':'V1'}
d['k2']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/tmp/ipython-input-3995950004.py in <cell line: 0>()
      1 d={'K1':'V1'}
----> 2 d['k2']

KeyError: 'k2'
```

Next steps: Explain error

```
# Error
#    |-----> 1. Syntax Error -- not following python rules -- code won't execute
#    |
#    |-----> 2. Logical Error -- following python rules(syntax) - logical runtime Error ----> Exit State
'''
Exception Handling
----------------------
try
except
else
finally
------------//python keywords

try:
   <code - initialization/monitoring> # Any Logical Errors
except <LogicalErrorName>:
  Handle the Error
else:
  There is No LogicalError
finally:
  Always running


try:
   <code - initialization/monitoring> # Any Logical Errors
except <LogicalErrorName>:
  Handle the Error
else:
  There is No LogicalError
```

```
try:
   <code - initialization/monitoring> # Any Logical Errors
except <LogicalErrorName>:
  Handle the Error

'''
```

```
print("one")
Fname = "emp.csv"
try:
    print("File name is:",fname)
except Exception as eobj:
    print(eobj)

print("Test-1")
print("Test-2")
print("Test-3")
print("End of the line")
```

```
one
name 'fname' is not defined
Test-1
Test-2
Test-3
End of the line
```

```
var = 120
try:
  print(var)
except Exception as eobj:
  print(eobj)
else:
  print(var+100)
finally:
  print('Thank you')
```

```
120
220
Thank you
```

```
var = 120
try:
  print(Var)
except Exception as eobj:
  print(eobj)
else:
  print(var+100)
finally:
  print('Thank you')
```

```
name 'Var' is not defined
Thank you
```

```
'''
###
1. Read IP.txt file
2. kernel --> OS
3. replaced lines only - Write to newFile

####
1. Read IP.txt
2. Search pattern shell
3.  |->display matched pattern linesep
####
1. Read IP.txt
2. display formatted style result - use re.split()
'''
```

```
'''
1. Read IP.txt file
2. kernel --> OS
3. replaced lines only - Write to newFile
'''
fobj = open('IP.txt','r')
L = fobj.readlines()
fobj.close()
wobj = open('result.log','w')
for var in L:
  result = re.sub('kernel','OS',var)
  wobj.write(result)

wobj.close()
```

```
'''
####
1. Read IP.txt
2. Search pattern shell
3.  |->display matched pattern linesep
'''
fobj = open('IP.txt','r')
L = fobj.readlines()
fobj.close()
for var in L:
  if(re.search('shell',var)):
    print(var.strip())
```

```
[    0.211712] shell: hv_vmbus: registering driver hv_storvsc
[    0.212437] shell: PPP generic driver version 2.4.2
[    0.212888] shell: hv_vmbus: registering driver hv_netvsc
[    0.213387] shell: Fusion MPT SPI Host driver 3.04.20
```

```
'''
####
1. Read IP.txt
2. display formatted style result - use re.split()
'''
fobj = open('IP.txt','r')
L = fobj.readlines()
fobj.close()
for var in L:
  if(not re.search('^$',var)):
    L=re.split('[^\\w\\s]|\\s+',var)
    print(f'{L[2]}|{L[3]}|{L[5]}|{L[-1]}')
```

```
1|458787|kernel|
1|460821|kernel|
1|461313|kernel|
1|461680|systemd|
1|462147|kernel|
1|666680|journald|
0|208793|kernel|
0|210653|kernel|
0|210890|kernel|
0|211193|kernel|
0|211378|kernel|
0|211550|kernel|
0|211712|shell|
0|212437|shell|
0|212675|kernel|
0|212888|shell|
0|213144|kernel|
0|213255|kernel|
0|213387|shell|
0|213391|kernel|storvsc_host_t
```

```
'''
class
object
Method

class - Type - blueprint of object - template
------
|
object - value - real entity

class className:
  class attributes

How to access class attributes?
className.<attribute>

user defined python class - mutable
                            ---------
1. using class name we can add new attribute
2. using class name we can modify an existing attribute
'''
```

```
winx
```

```
class product:
  pid = 101
  pcost = 1000
  pname = 'pA'
  others_details = ['pV1','pV101','customer@gmail.com']

print(type(product))
print(product)
```

```
<class 'type'>
<class '__main__.product'>
```

```
print(type(int),type(float),type(str),type(list),type(dict))
print(type(product))
```

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

```
class product:
  pid = 101
  pcost = 1000
  pname = 'pA'
  others_details = ['pV1','pV101','customer@gmail.com']

print(product.pid) # To access class attribute -> className.attribute
print(product.pname) # To access class attribute
print(product.pcost)
print(product.others_details[0])
print(product.others_details[-2:])
```

```
101
pA
1000
pV1
['pV101', 'customer@gmail.com']
```

```
class box:
  box_id = 101
  box_name = "B1"

print(box.box_id)
print(box.box_name)
```

```
101
B1
```

```
box.box_name = "B2" # modification  - like dict logic
box.box_size = "2inches" # adding new attribute - like dict logic
```

```
print(box.box_name)
print(box.box_size)
```

```
B2
2inches
```

```
dir(box)
```

```
['__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 'box_id',
 'box_name',
 'box_size']
```

```
box.__dict__  # Vs  dir(box) ->[list of attributes]
```

```
mappingproxy({'__module__': '__main__',
              'box_id': 101,
              'box_name': 'B2',
              '__dict__': <attribute '__dict__' of 'box' objects>,
              '__weakref__': <attribute '__weakref__' of 'box' objects>,
              '__doc__': None,
              'box_size': '2inches'})
```

```
var = 10
print(var)
```

```
10
```

```
del(var)
```

```
print(var)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-2734745701.py in <cell line: 0>()
----> 1 print(var)

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

Next steps: Explain error

```
box.box_id
```

```
101
```

```
del(box.box_id)
```

```
'box_id' in dir(box)
```

```
False
```

```
'box_name' in dir(box)
```

```
True
```

```
print(box.box_name)
```

```
B2
```

```
print(box.Box_name)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipython-input-315689895.py in <cell line: 0>()
----> 1 print(box.Box_name)

AttributeError: type object 'box' has no attribute 'Box_name'
```

Next steps: Explain error

```
'''
    Building
+------------------------------+
|   []   |  |    [     ]       |
|                              |  Blueprint sheet  - class
|                       (Green)  |
+-------------|    |------------+


+--------+      +---------+    +----------+
| B1     |      |   B2    |    |   B3     |    -- Real Building - object
+--------+      +---------+    +----------+
 1st main        2nd main       3rd main   -------- memory /address
```

```
'''
```

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

print(product)
product() # classname() - constructor - object initialization
```

```
<class '__main__.product'>
<__main__.product at 0x7e473582d640>
```

```
product()
```

```
<__main__.product at 0x7e473582d790>
```

```
product()
```

```
<__main__.product at 0x7e473582d5b0>
```

```
obj1 = product()
obj2 = product()
print(type(obj1),type(obj2))
```

```
<class '__main__.product'> <class '__main__.product'>
```

```
print(type(10),type(20))
```

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

```
## type --- class
## value --- real value / object
```

```
'''
    Building
+-------------------------------+
|   []   |  |    [     ]        |
|                               |  Blueprint sheet  - class
|                      (Green)  |
+-------------|    |------------+


+-------+      +---------+    +------------+
| B1    |      |   B2    |    |  | B3 |   |    -- Real Building - object
+-------+      +---------+    +----------+
 1st main        2nd main       3rd main   -------- memory /address
                  |->White

'''
```

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

print(product.pname) # 'pA'

obj1 = product()
obj2 = product()

print(f'obj1 => {obj1.pname}') # 'pA'
print(f'obj2 => {obj2.pname}') # 'pA'
```

```
pA
obj1 => pA
obj2 => pA
```

```
obj1.pname = "Demo-1" ## object based initialization
```

```
print(product.pname)
print(f'obj1 => {obj1.pname}')
print(f'obj2 => {obj2.pname}')
```

```
pA
obj1 => Demo-1
obj2 => pA
```

```
product.pname = "pB" # class based initialization - blueprint
```

```
print(product.pname)
print(f'obj1 => {obj1.pname}')
print(f'obj2 => {obj2.pname}')
```

```
pB
obj1 => Demo-1
obj2 => pB
```

```
obj2.pname = "Demo-2" # object based initialization
```

```
print(product.pname)
print(f'obj1 => {obj1.pname}')
print(f'obj2 => {obj2.pname}')
```

```
pB
obj1 => Demo-1
obj2 => Demo-2
```

```
product.pname = "pC" # class based initialization - blueprint
```

```
print(product.pname)
print(f'obj1 => {obj1.pname}')
print(f'obj2 => {obj2.pname}')
```

```
pC
obj1 => Demo-1
obj2 => Demo-2
```

```
class Cname:
  var = 120

obj1 = Cname()
obj2 = Cname()

obj1.var = 500

Cname.var = 600

print("A:",obj1.var) # 500
print("B:",obj2.var) # 600
print("C:",Cname.var) # 600
```

```
A: 500
B: 600
C: 600
```

```
## function --- method

def f1():
  print("OK-1")

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

obj = box()
print(type(f1))  ##  f1()
print(type(obj.f2))  ## obj.f2() //methodCall
```

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

```
L=[]
L.append('D1') //methodCall
```

```
def fx():
  print("OK-fx block")

fx()
fx(10)
```

```
OK-fx block
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-1560304358.py in <cell line: 0>()
      3
      4 fx()
----> 5 fx(10)

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

Next steps: Explain error

```
def f1():
  print("OK-1")

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

f1()
obj = box()
# obj.f2() # obj.f2() --> f2(obj)  ;  obj.f2(10,20) -->f2(obj,10,20)
```

```
OK-1
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-1858684903.py in <cell line: 0>()
      9 f1()
     10 obj = box()
---> 11 obj.f2()

TypeError: box.f2() takes 0 positional arguments but 1 was given
```

Next steps: Explain error

```
Start coding or generate with AI.
```

```
class cname:
  def method1(self):
    print("self=",self)

obj1 = cname()
print("obj1=",obj1)
obj1.method1() ###  method1(obj1)

print('')
obj2 = cname()
print("obj2=",obj2)
obj2.method1() ####
```

```
obj1= <__main__.cname object at 0x7e4735820950>
self= <__main__.cname object at 0x7e4735820950>

obj2= <__main__.cname object at 0x7e4735b42360>
self= <__main__.cname object at 0x7e4735b42360>
```

```
def fx(a):
  print(a,type(a))

fx(True)
fx(10)
fx([])
fx({})
fx(0.0)
```

```
True <class 'bool'>
10 <class 'int'>
[] <class 'list'>
{} <class 'dict'>
0.0 <class 'float'>
```

```
Start coding or generate with AI.
```

```
class product:
  pid = 101
  def f1(self):
    print("B:",self.pid) # 101
    self.pid = 501       # self(obj1).pid = 501

obj1 = product()
print("A:",obj1.pid) # 101
obj1.f1() # methodCall --------> f1(obj1)
print("C:",obj1.pid) # 501
```

```
A: 101
B: 101
C: 501
```

```
class product:
  pid = 101
  def f1(self,a1):
    self.pid = a1

obj1 = product()
obj1.f1(501) ### f1(obj1,501)

obj2 = product()
obj2.f1(601) #### f1(obj2,601)
```

```
print(obj1.pid)
print(obj2.pid)
print(product.pid)
```

```
501
601
101
```

```
class product:
  pid = 101
  def f1(self,a1):
    self.pid = a1
  def f2(self):
    print(f"{self.pid}")

obj1 = product()
obj1.f1(501) ### f1(obj1,501)

obj2 = product()
obj2.f1(601) #### f1(obj2,601)

obj1.f2()
obj2.f2()
```

```
501
601
```

```
class Enrollment:
  NAME = ""
  DOB = ""
  Bgroup = ""
  def f1(self,ename,edob,ebg):
    '''initialization'''
    self.NAME = ename
    self.DOB = edob
    self.Bgroup = ebg
    print(f'Emp {self.NAME} enrollment is done!!!')

  def f2(self):
    '''display'''
    print(f'Emp name is:{self.NAME} DOB:{self.DOB} Bgroup:{self.Bgroup}')

eobj1 = Enrollment()
eobj1.f1("Arun","1st Jan","A+")

eobj2 = Enrollment()
eobj2.f1("Vijay","2nd Feb","O+")
print('')
eobj1.f2()
eobj2.f2()
```

```
Emp Arun enrollment is done!!!
Emp Vijay enrollment is done!!!

Emp name is:Arun DOB:1st Jan Bgroup:A+
Emp name is:Vijay DOB:2nd Feb Bgroup:O+
eobj3 = Enrollment()
eobj3.f2()
```

```
Emp name is: DOB: Bgroup:
```

```
class DBI:
  def connect(self,.....):
    ''' Establish DB connection '''
  def method1(self):
    '''Query1 '''
  def method2(self):
    '''Query2'''
  def method3(self):
    '''Query3'''


DataBase View
==============
1st - Database connection - OK - then we invoke a query

OOPs     View
------------------
obj = DB1()
obj.method1() ## ValidCall - OOPs point of view
------------
   |--->--- result DataBase operation Error

special method ==> __init__() <=== constructor

In python - __<name/variable/function>__  <== Pre-defined attributes


>>>
>>> class box:
...     def method1(self):
...         print("Non-Constructor")
...
>>> box()
<__main__.box object at 0x000001B07082B0E0>
>>> obj = box()
>>> obj.method1()
Non-Constructor
>>>
>>> class box:
...     def __init__(self):
...         print("Constructor block")
...
>>> box()
Constructor block
<__main__.box object at 0x000001B07082B4D0>
>>>
>>> obj = box()
Constructor block
>>>
>>>
>>> class box:
...     def __init__(self,a1,a2=0,*a3,**a4):
...         print("Constructor block")
...         print(a1,a2,a3,a4)
...
>>> obj1 = box()
Traceback (most recent call last):
  File "<python-input-13>", line 1, in <module>
    obj1 = box()
TypeError: box.__init__() missing 1 required positional argument: 'a1'
>>>
>>> ob1
Traceback (most recent call last):
  File "<python-input-15>", line 1, in <module>
    ob1
NameError: name 'ob1' is not defined. Did you mean: 'obj'?
>>> obj1
Traceback (most recent call last):
  File "<python-input-16>", line 1, in <module>
    obj1
NameError: name 'obj1' is not defined. Did you mean: 'obj'?
>>>
>>> obj1 = box(10)
Constructor block
10 0 () {}
>>> obj1
<__main__.box object at 0x000001B070C45F90>
```

```
>>> obj2 = box(100,200,300,400,"D1","D2")
Constructor block
100 200 (300, 400, 'D1', 'D2') {}
>>>
>>> obj2 = box(100,200,300,400,"D1","D2",shell="/bin/bash")
Constructor block
100 200 (300, 400, 'D1', 'D2') {'shell': '/bin/bash'}
>>>
```

```
class Enrollment:

  def __init__(self,ename,edob,ebg):
    '''initialization'''
    self.NAME = ename
    self.DOB = edob
    self.Bgroup = ebg
    print(f'Emp {self.NAME} enrollment is done!!!')

  def f2(self):
    '''display'''
    print(f'Emp name is:{self.NAME} DOB:{self.DOB} Bgroup:{self.Bgroup}')

eobj1 = Enrollment("Arun","1st Jan","A+")
#eobj1.f1("Arun","1st Jan","A+")

eobj2 = Enrollment("Vijay","2nd Feb","O+")
#eobj2.f1("Vijay","2nd Feb","O+")
print('')
eobj1.f2()
eobj2.f2()
```

```
Emp Arun enrollment is done!!!
Emp Vijay enrollment is done!!!

Emp name is:Arun DOB:1st Jan Bgroup:A+
Emp name is:Vijay DOB:2nd Feb Bgroup:O+
```

```
Eobj3 = Enrollment()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-1814123648.py in <cell line: 0>()
----> 1 Eobj3 = Enrollment()

TypeError: Enrollment.__init__() missing 3 required positional arguments: 'ename', 'edob', and 'ebg'
```

Next steps: Explain error

```
class product:
  '''product class description
  about product class - constructor as arguments'''
  def __init__(self,pid,pname,pcost=0.0):
    self.pid = pid
    self.pname = pname
    self.pcost = pcost
  def display(self):
    '''display product details'''
    return self.pid,self.pname,self.pcost # tuple
```

```
obj1 = product(101,'Laptop',10000)
obj2 = product(102,'Computer',12000)
obj3 = product(103,'mobile',12340.32)
```

```
obj1.display()
```

```
(101, 'Laptop', 10000)
```

```
for var in [obj1,obj2,obj3]:
  print(var.display())
```

```
(101, 'Laptop', 10000)
(102, 'Computer', 12000)
(103, 'mobile', 12340.32)
```

```
product.__doc__
```

```
'product class description\n  about product class - constructor as arguments'
```