In [1]:
```
import os
os.system("whoami") -> int (exit satus code)
os.popen("command") ->instance
os.listdir("command")-> []

import sys
sys.version
sys.path
sys.modules

import pprint
pprint.pprint()
```

Out[1]:
```
0
```

In [5]:
```
import sys
print(type(sys.argv))
```

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

In [8]:
```
import os
for i in os.environ.keys():
    print(i)
```

```
ALLUSERSPROFILE
APPDATA
COMMONPROGRAMFILES
COMMONPROGRAMFILES(X86)
COMMONPROGRAMW6432
COMPUTERNAME
COMSPEC
CONDA_PREFIX
DRIVERDATA
EFC_4580_1592913036
EFC_4580_2283032206
HOMEDRIVE
HOMEPATH
IPY_INTERRUPT_EVENT
JPY_INTERRUPT_EVENT
JPY_PARENT_PID
JPY_SESSION_NAME
LOCALAPPDATA
LOGONSERVER
NUMBER_OF_PROCESSORS
ONEDRIVE
OS
PATH
PATHEXT
PROCESSOR_ARCHITECTURE
PROCESSOR_IDENTIFIER
PROCESSOR_LEVEL
PROCESSOR_REVISION
PROGRAMDATA
PROGRAMFILES
PROGRAMFILES(X86)
PROGRAMW6432
PSMODULEPATH
PUBLIC
SESSIONNAME
SYSTEMDRIVE
SYSTEMROOT
TEMP
TMP
USERDOMAIN
USERDOMAIN_ROAMINGPROFILE
USERNAME
USERPROFILE
VBOX_MSI_INSTALL_PATH
WINDIR
ZES_ENABLE_SYSMAN
PYDEVD_USE_FRAME_EVAL
TERM
CLICOLOR
FORCE_COLOR
CLICOLOR_FORCE
PAGER
GIT_PAGER
MPLBACKEND
```

In [9]:
```
if "DB" in os.environ.keys():
    print("DB variable exist")
else:
    print("DB not exist")
```

```
DN not exist
```

In [ ]:
```
Regular Expression
-----------------
re module
|--search
|--substitute
|--split

in Linux shell script---> grep;sed ; awk // commands
in winx - findstr  // command
in python - module(re); functions-> conditional, looping, filehandling // script

search--> re.search() ; re.findall()
substitute --> re.sub()
split ---> re.split()


search
-------
 -> search pattern in input - character based search

re.search()===> re.search("pattern", "InputString")

re.search("bash", "root:bin:bash") --> <class-instance>   / None
           |             |
        Pattern         Input str


re.findall()=>    re.findall("pattern", "inputStr") --> [result]  / []
```

In [ ]:
```
read data from file   # file handling
search pattern from <inputfile> => re.search("pattern","inputstring")
conditional statemnets-> display matched pattern line # if only
```

In [14]:
```
import re
with open("D:\\emp.csv") as FH:
    for var in FH.readlines():
        if re.search("sales",var):
            print(var.strip())
```

```
101,Arun,sales,pune,2000
102,Vishnu,sales,hyderabad,3000
```

In [ ]:
```
# in linux   grep "sales" emp.csv
```

In [15]:
```
re.search("bash","root:x:bin:bash")
```

Out[15]:
```
<re.Match object; span=(11, 15), match='bash'>
```

In [17]:
```
print(re.search("HR","root:bin:bash"))
```

```
None
```

In [18]:
```
re.findall("bash","root:bin:bash:x:bash")
```

Out[18]:
```
['bash', 'bash']
```

In [19]:
```
re.findall("bash","root:bin:bash:x:bash:bash-shell:bash")
```

Out[19]:
```
['bash', 'bash', 'bash', 'bash']
```

In [20]:
```
re.findall("bash","root:bin:bash:x:Bash")
```

Out[20]:
```
['bash']
```

In [22]:
```
re.findall("bash","root:bin:bash:x:Bash1", re.I)  # ignores case
```

Out[22]:
```
['bash', 'Bash']
```

In [23]:
```
re.search("bash","dsadugsdBASHlkjdlsajdls",re.I)
```

Out[23]:
```
<re.Match object; span=(8, 12), match='BASH'>
```

In [ ]:
```
BRE
---
^   ==> ^pattern -->  re.search("bash","root:bash")---> OK
                      re.search("^bash","root:bash") -> None
                        re.search("^bash","bash:root:bin") --> OK
```

```
$   ==> pattern$    --> re.search("bash$","gdisagdjsadbash")--> OK
                        re.search("bash$","hjhdshjsabashgdhsd")-> None

^pattern$  ==> line has only pattern  -> re.search("^bash$","bash")-> OK
      -pattern only                             re.search("^bash$","bash ") -> None


.  single character
.*
[]   p[1234] --> p1  p2 p3 p4 p5  -> ok
^[]
[^]
[]$
^$  - empty line
--------------------------------

ERE
---
|
()
+
{n}
{n,}
{n,m}
    -----------------------------> regx char
```

In [24]:
```
re.search("^bash","root:x:bin:bash")
re.search("^5","5gsjdhjgg67687")
```
Out[24]:
```
<re.Match object; span=(0, 1), match='5'>
```

In [25]:
```
re.search("00$" , "dhjgdsdjhdlklkhkdg00")
```
Out[25]:
```
<re.Match object; span=(18, 20), match='00'>
```

In [26]:
```
re.search("^sales$", "sales")
```
Out[26]:
```
<re.Match object; span=(0, 5), match='sales'>
```

In [27]:
```
re.search("^sales$", "sales ")
```

In [28]:
```
re.search("^sales dept$", "salesdept")
```

In [29]:
```
re.search("^salesdept$", "salesdept")
```
Out[29]:
```
<re.Match object; span=(0, 9), match='salesdept'>
```

In [30]:
```
re.search("^sales.", "sales ")
```
Out[30]:
```
<re.Match object; span=(0, 6), match='sales '>
```

In [31]:
```
re.search("^sales.", "sales,")
```
Out[31]:
```
<re.Match object; span=(0, 6), match='sales,'>
```

In [32]:
```
re.search("^sales.*","salesgjnm ,$#")
```
Out[32]:
```
<re.Match object; span=(0, 13), match='salesgjnm ,$#'>
```

In [33]:
```
re.search("^sales.*sales$", "salesbbmmmmmsales")
```
Out[33]:
```
<re.Match object; span=(0, 17), match='salesbbmmmmmsales'>
```

In [34]:
```
re.findall("^sales.*sales$", "salesbbmmmmmsales")
```
Out[34]:
```
['salesbbmmmmmsales']
```

In [36]:
```
re.search("[A-Za-z0-9]","hghghABBHHJM6788 %")
```
Out[36]:
```
<re.Match object; span=(0, 1), match='h'>
```

In [38]:
```
re.findall("[A-Za-z0-9]","hghghABBHHJM6788 %")
```

```
Out[38]: ['h',
          'g',
          'h',
          'g',
          'h',
          'A',
          'B',
          'B',
          'H',
          'H',
          'J',
          'M',
          '6',
          '7',
          '8',
          '8']
```

```
In [41]: re.findall("\w","223243546%$# 88")
```

```
Out[41]: ['2', '2', '3', '2', '4', '3', '5', '4', '6', '8', '8']
```

```
In [42]: re.findall("\s","223243546%$# 88")   # single white space
```

```
Out[42]: [' ']
```

```
In [45]: re.findall("\s$","223243546%$# 88 ")   # single white space
```

```
Out[45]: [' ']
```

```
In [46]: s="root:x:bin,-0%test test123"   # special char
         re.findall("[^\s\w]", s) # other A-Z a-z 0-9 white space
```

```
Out[46]: [':', ':', ',', '-', '%']
```

```
In [47]: # Multiple Pattern Search
         #  Pattern1  | Pattern2   | Pattern3
         # like logical or
         #
         re.search("sales|QA|admin" , "101,raj,sales,pune,23232")
```

```
Out[47]: <re.Match object; span=(8, 13), match='sales'>
```

```
In [48]: # grouping ()
         # pattern1 | Pttern 2| Pattern 3-- anyone is matched--> matched
         #(Pattern1) (Pattern2) (Pattern3)   - all patterns to be matched--> matched
         re.search("bash|ksh", "working shell is bash")
```

```
Out[48]: <re.Match object; span=(17, 21), match='bash'>
```

```
In [49]: re.search("bash|ksh", "working shell is ksh")
```

```
Out[49]: <re.Match object; span=(17, 20), match='ksh'>
```

```
In [50]: re.search("(bash)(ksh)", "working shell is bash and ksh")
```

```
In [51]: re.search("(bash)(ksh)", "working shell is ksh  and bash")
```

```
In [52]: re.search("(bash)(ksh)", "working shell is bashksh")
```

```
Out[52]: <re.Match object; span=(17, 24), match='bashksh'>
```

```
In [53]: re.search("(bash)(ksh)", "working shell is kshbash")
```

```
In [54]: re.search("(bash).*(ksh)", "working shell is bash and ksh")
```

```
Out[54]: <re.Match object; span=(17, 29), match='bash and ksh'>
```

```
In [55]: re.search("(bash).*(ksh)", "working shell is ksh and bash")
```

```
In [ ]: # task
        ======
        # starts with   'http'  or 'https'  -> ok
        # end  withs 'org'   or 'com'  ->  ok

        #http://www.abc.com  -> ok
        #https://www.abc.com  -> ok
        #http://www.abc.org  -> ok
        #https://www.abc.org  -> ok
        #http://www.abc.edu  ->  not ok
```

```
#http://www.abc.in  -> not ok
#ftp://www.abc.com  -> not ok
```

In [57]:
```
re.search("^http|^https.*org$|com$", "ftp://abc.com")
#         ------- ------------ ----
#        pattern1   pattern2    pattern3
```

Out[57]:
```
<re.Match object; span=(10, 13), match='com'>
```

In [58]:
```
re.search("(^http|^https).*(org$|com$)", "ftp://abc.com")
```

In [59]:
```
re.search("(^http|^https).*(org$|com$)", "http://abc.com")
```

Out[59]:
```
<re.Match object; span=(0, 14), match='http://abc.com'>
```

In [60]:
```
re.search("(^http|^https).*(org$|com$)", "https://abc.com")
```

Out[60]:
```
<re.Match object; span=(0, 15), match='https://abc.com'>
```

In [61]:
```
re.search("(^http|^https).*(org$|com$)", "http://abc.org")
```

Out[61]:
```
<re.Match object; span=(0, 14), match='http://abc.org'>
```

In [62]:
```
re.search("(^http|^https).*(org$|com$)", "http://abc.in")
```

In [ ]:
```
+---> 1 or more
<pattern>+
|--------------------1 time max more time (no limit)

a+   => a   aaaaaaaaa
ab+   => ab  abbbbbbbbbbbbbbbb
ab+c  => abc  abbbbbbbbbbbbbbc   // matched
```

In [ ]:
```
^[A-Z][a-z]+[0-9]$   => Abhgghjkj0       Gb2  // matched
```

In [ ]:
```
\s+     => finding 1 or more space
```

In [ ]:
```
{}- range style
<pattern>{n}  -> n times
ab{2}c   => abbc  // match
AB3333bbb--->   ^[A-Z][A-Z][0-9][0-9][0-9][0-9],a-z][a-z]$  --> ^[A-Z]{2}[0-9]{4},a-z]{2}$
```

In [ ]:
```
<pattern>{n,}   min n times max no limit
ab{2,}   abc    abbbbbc   abbbbbbbbbbbbbbbbbbbbbbbc   //matched
abc  // not matched

ab+c    ==>   ab{1,}c  // same

<pattern>{n,m}  -> min n times max m times
ab{2,4}c ---> abbc    abbbc     abbbbc    // matched
             abbbbbbbbbbbbbbbc // not matched

|  () {} +  -> ERE
```

In [64]:
```
url ="http://www.orcle.com"
re.search("(^http|^https).*(com$|org$)" , url)
```

Out[64]:
```
<re.Match object; span=(0, 20), match='http://www.orcle.com'>
```

In [65]:
```
url ="http://www.orcle.com"
re.findall("(^http|^https).*(com$|org$)" , url)
```

Out[65]:
```
[('http', 'com')]
```

In [66]:
```
url ="http://www.orcle.com"
re.findall("(^http|^https)(.*)(com$|org$)" , url)
```

Out[66]:
```
[('http', '://www.orcle.', 'com')]
```

In [67]:
```
url ="http://www.orcle.com"
L=re.findall("(^http|^https)(.*)(com$|org$)" , url)
```

In [70]:
```
L   # List of  tuple
print(L[0][0])
print(L[0][1])
print(L[0][-1])
```

```
http
://www.orcle.
com
```

In [ ]:
```
re.sub()
re.sub("OldpatternString","NewpatternStr","InputStr")-> replacedstr  / Inputstr
```

In [71]:
```
var="root:x:bin:bash"
re.sub("bash","ksh",var)
```

Out[71]:
```
'root:x:bin:ksh'
```

In [72]:
```
var="root:x:bin:zsh"
re.sub("bash","ksh",var)
```

Out[72]:
```
'root:x:bin:zsh'
```

In [74]:
```
var="root:x:bin:bash"
re.sub("^bash","ksh" , var)
```

Out[74]:
```
'root:x:bin:bash'
```

In [75]:
```
with open("D:\\emp.csv") as FH:
    for var in FH.readlines():
        s=re.sub("sales","ADMIN",var)
        print(s.strip())
```

```
101,Arun,ADMIN,pune,2000
102,Vishnu,ADMIN,hyderabad,3000
103,Vijay,prod,Pune,2000
104,Raghav,Hr,pune,3000
105,sam,Hr,bglore,8000
```

In [77]:
```
WH= open("D:\\r1.txt","w")
with open("D:\\emp.csv") as FH:
    for var in FH.readlines():
        s=re.sub("sales","ADMIN",var)
        WH.write(s)
WH.close()
```

In [78]:
```
with open("D:\\r1.txt") as FH:
    print(FH.read())
```

```
101,Arun,ADMIN,pune,2000
102,Vishnu,ADMIN,hyderabad,3000
103,Vijay,prod,Pune,2000
104,Raghav,Hr,pune,3000
105,sam,Hr,bglore,8000
```

In [79]:
```
with open("D:\\emp.csv") as FH:
    for var in FH.readlines():
        if(re.search("sales",var)):
            s=re.sub("sales","ADMIN",var)
            print(s)
```

```
101,Arun,ADMIN,pune,2000

102,Vishnu,ADMIN,hyderabad,3000
```

In [81]:
```
WH= open("D:\\r1.txt","w")
with open("D:\\emp.csv") as FH:
    for var in FH.readlines():
        if re.search("sales",var):
            s=re.sub("sales","ADMIN",var)
            WH.write(s)
WH.close()
```

In [82]:
```
with open("D:\\r1.txt") as FH:
    print(FH.read())
```

```
101,Arun,ADMIN,pune,2000
102,Vishnu,ADMIN,hyderabad,3000
```

In [83]:
```
var="4555,hari,sales,pune,122222"
# delete sales word

re.sub("sales",'',var)
```

Out[83]:
```
'4555,hari,,pune,122222'
```

```
In [84]: var="4555,hari,sales,pune,122222"
         # delete sales word

         re.sub("sales.",'',var)

Out[84]: '4555,hari,pune,122222'
```

```
In [85]: re.sub("sales","ADMIN","101,sales,sales,sales,pune")

Out[85]: '101,ADMIN,ADMIN,ADMIN,pune'
```

```
In [86]: re.sub("sales","ADMIN","101,sales,sales,sales,pune",1)

C:\Users\Lenovo\AppData\Local\Temp\ipykernel_8376\335233139.py:1: DeprecationWarning: 'count' is passed as posit
ional argument
  re.sub("sales","ADMIN","101,sales,sales,sales,pune",1)

Out[86]: '101,ADMIN,sales,sales,pune'
```

```
In [87]: re.sub("sales","ADMIN","101,sales,sales,sales,pune",1),2)

C:\Users\Lenovo\AppData\Local\Temp\ipykernel_8376\476900409.py:1: DeprecationWarning: 'count' is passed as posit
ional argument
  re.sub("sales","ADMIN","101,sales,sales,sales,pune",2)

Out[87]: '101,ADMIN,ADMIN,sales,pune'
```

```
In [88]: re.sub("sales","ADMIN","101,Sales,pune")

Out[88]: '101,Sales,pune'
```

```
In [89]: re.sub("sales","ADMIN","101,Sales,pune",0,re.I)

C:\Users\Lenovo\AppData\Local\Temp\ipykernel_8376\3195416774.py:1: DeprecationWarning: 'count' is passed as posi
tional argument
  re.sub("sales","ADMIN","101,Sales,pune",0,re.I)

Out[89]: '101,ADMIN,pune'
```

```
In [90]: help(re.sub)

Help on function sub in module re:

sub(pattern, repl, string, count=0, flags=0)
    Return the string obtained by replacing the leftmost
    non-overlapping occurrences of the pattern in string by the
    replacement repl.  repl can be either a string or a callable;
    if a string, backslash escapes in it are processed.  If it is
    a callable, it's passed the Match object and must return
    a replacement string to be used.
```

```
In [91]: #re.split("regpattern","inputstr")

         s="root:bin:bash"

         s.split(":")   # split of str

Out[91]: ['root', 'bin', 'bash']
```

```
In [92]: s="root:x,bin-bash~data%sh"
         re.split("[^\w\s]",s)     # Vs   s.split(singledelimetr)

Out[92]: ['root', 'x', 'bin', 'bash', 'data', 'sh']
```

```
In [93]: s="programming data java 11 and python 3.6 list "
         re.split("\d+\s+",s)

Out[93]: ['programming data java ', 'and python 3.', 'list ']
```

```
In [94]: import os
         for var in os.popen("powershell Get-Process").readlines():
             if(re.search("^\s+\d+.*(notepad|zoom|python)",var,re.I)):
                 print(var.strip())
```

```
1628      57    64740     138068     168.56   5768   2 Notepad
191      12     2896      14412       4.23   7544   2 Notepad
191      12     2892      14368       4.41  11208   2 Notepad
191      12     2852      14620       4.27  16348   2 Notepad
467      95   155052      62392     265.45   1056   2 python
152      14    12876      12676       0.16   4760   2 python
114      11    11972      19192      45.13   5760   2 python
357     101   109488      34424       6.34   8376   2 python
1874     151   446100     321476   9,220.13   8524   2 Zoom
898      57    88292      94568   1,283.42  10780   2 Zoom
```

In [ ]:

```
import os
for var in os.popen("ps -e").readlines():
    if(re.search("^\s+\d{3,}.*gnome.*[a-e]$",var,re.I)):
        print(var.strip())
```

In [ ]:

```
Inventory.txt
=============
ItemNo:salesCount
101:10,20,30,40
January Details
102:30,20,10,3
103:20,100,400,10
February Details
104:30,20,10,3
105:20,100,400,10

write a program

step 1: read the inventory file
step 2: split itemno, sales count
step 3: calculate sum of sales count
step 4: display each Item No and total sales count

ex: Item No: 101         sales count : 100
```

In [97]:

```
import re
F=open("D:\\Inventory.txt")
for var in F.readlines():
    if re.search("^\d",var):
        print(var.strip())
F.close()
```

```
101:10,20,30,40
102:30,20,10,3
103:20,100,400,10
104:30,20,10,3
105:20,100,400,10
```

In [101…

```
import re
F=open("D:\\Inventory.txt")
for var in F.readlines():
    if re.search("^\d",var):   # filtering line starts with digit
        var=var.strip()    # remove \n char
        L=re.split("[:,]",var)   # split based on : and ,- multiple delimeter
        print(L)
        t=0
        for v in L[1:]:  # from 1st index to  list of all
            t=t+int(v)
        else:
            print("ITEM NO : {}\t TOTAL SALES COUNT : {}".format(L[0],t))

F.close()
```

```
['101', '10', '20', '30', '40']
ITEM NO : 101    TOTAL SALES COUNT : 100
['102', '30', '20', '10', '3']
ITEM NO : 102    TOTAL SALES COUNT : 63
['103', '20', '100', '400', '10']
ITEM NO : 103    TOTAL SALES COUNT : 530
['104', '30', '20', '10', '3']
ITEM NO : 104    TOTAL SALES COUNT : 63
['105', '20', '100', '400', '10']
ITEM NO : 105    TOTAL SALES COUNT : 530
```

In [ ]:

```
L=['100K','200GB','700G', '6TB']
# calculate sum of list?
# '100k'--> '100'
#'100' -> int('100')-> 100
'''
delete alpha char
s="100K"

re.sub('[A-Z]','',s)
```

In [103…
```
L=['100K','200GB','700G', '6TB']
t=0
for var in L:
    r=re.sub("[A-Za-z]+","",var)   # delete alpha
    t=t+int(r)
else:
    print("Sum of storage size:{} ".format(t))
```
```
Sum of storage size:1006
```

In [ ]:
```
Functional Style :
=================

Procedural style --> statements  + block
Functional style -->  expression (or) single line code - > ML DL

procedural style---> 10 lines of code   ---> single line===> functional style
```

In [ ]:
```
lambda
comprehension
|
map,filter,reduce
```

In [ ]:
```
lambda  - python keyword
|------------------------> unnamed function - function call with arg -- return value
syntax:-
=======
    lambda list of args : expression
        VS
    def f1(a1,a2,a3):
        ..
        ..  // block style code

```

In [107…
```
def f1(a1,a2):
    return a1+a2
f1(10,20)
f1("hari","govind")
```
Out[107…
```
'harigovind'
```

In [109…
```
# lambda list of args : expression
f1=lambda a1,a2 : a1+a2
# function call
f1(1,2)
f1("Hari","Govind")
```
Out[109…
```
'HariGovind'
```

In [110…
```
f3=lambda a,b: a>b
f3(30,100)
```
Out[110…
```
False
```

In [111…
```
f4=lambda a: fx(a)

def fx(a):
    return a+100
f4(100)
```
Out[111…
```
200
```

In [112…
```
f5= lambda a: a.upper()
f5('hari')
```
Out[112…
```
'HARI'
```

In [113…
```
L=[] # empty list
for var in range(5):
    r=var+100
    L.append(r)

L
```
Out[113…
```
[100, 101, 102, 103, 104]
```

In [114…
```
# List Comprehension
#---------------------

# [ value  for  iterable  ]
#          |---(1)--------|
# --(2)----

[  var+100   for var in range(5) ]    # 0  1 2 3 4
```

```
Out[114… [100, 101, 102, 103, 104]
```

```
In [117… # based on condition
L=[]
for var in range(10):
    if var> 5:
        r= var + 500
        L.append(r)
    else:
        r= var + 100
        L.append(r)
L
```

```
Out[117… [100, 101, 102, 103, 104, 105, 506, 507, 508, 509]
```

```
In [118… # syntax:-
#[true_exp if condition else false_expression for iterable ]
[var+500  if  var>5  else  var+100  for var in range(10) ]
```

```
Out[118… [100, 101, 102, 103, 104, 105, 506, 507, 508, 509]
```

```
In [119… [var.upper()   for var in open("D:\\emp.csv").readlines() ]
```

```
Out[119… ['101,ARUN,SALES,PUNE,2000\n',
 '102,VISHNU,SALES,HYDERABAD,3000\n',
 '103,VIJAY,PROD,PUNE,2000\n',
 '104,RAGHAV,HR,PUNE,3000\n',
 '105,SAM,HR,BGLORE,8000\n']
```

```
In [121… [ re.sub("sales","ADMIN",var) for var in open("D:\\emp.csv").readlines()]
```

```
Out[121… ['101,Arun,ADMIN,pune,2000\n',
 '102,Vishnu,ADMIN,hyderabad,3000\n',
 '103,Vijay,prod,Pune,2000\n',
 '104,Raghav,Hr,pune,3000\n',
 '105,sam,Hr,bglore,8000\n']
```

```
In [122… with open("D:\\emp.csv") as FH:
    for var in FH.readlines():
        s=re.sub("sales","ADMIN",var)
        print(s.strip())
```

```
101,Arun,ADMIN,pune,2000
102,Vishnu,ADMIN,hyderabad,3000
103,Vijay,prod,Pune,2000
104,Raghav,Hr,pune,3000
105,sam,Hr,bglore,8000
```

```
In [127… L=[ re.sub("sales","ADMIN",var)  if re.search("sales",var) else None for var in open("D:\\emp.csv").readlines()
L
```

```
Out[127… ['101,Arun,ADMIN,pune,2000\n',
 '102,Vishnu,ADMIN,hyderabad,3000\n',
 None,
 None,
 None]
```

```
In [126… for i in L:
    if(i):
        print(i)
```

```
101,Arun,ADMIN,pune,2000

102,Vishnu,ADMIN,hyderabad,3000
```

```
In [ ]: map
filter
reduce
-------
|-----------High order function ---> function(function)// fn accept another fn as arg
map(function, collection)  # performs on every elmnt
filter(function, collection) # testing--> true (matching)
reduce(function,collection)  # compute--> single value
         |          |
        lambda     comprehension

python 3.x                                       Python 2.x
==========                                      ============
map()-> <Address>                              map()---> []
filter() -> <Address>                          filter()-->[]
functools.reduce() -> single value             reduce()---> Singlevalue
```

In [128…
```
L=[]
for var in range(5):
    t=var+100
    L.append(t)

L
```

Out[128…
```
[100, 101, 102, 103, 104]
```

In [129…
```
#map(function, collection)
map(lambda a: a+100 , range(5))
```

Out[129…
```
<map at 0x1e6a767da80>
```

In [130…
```
#map(function, collection)
list(map(lambda a: a+100 , range(5)))
```

Out[130…
```
[100, 101, 102, 103, 104]
```

In [133…
```
L=[]
L.append(list(map(lambda a:a+100, range(5))))
d={}
d["k1"]=list(map(lambda a:a+100, range(5)))
print(L,d)
```

```
[[100, 101, 102, 103, 104]] {'k1': [100, 101, 102, 103, 104]}
```

In [134…
```
L=[]
def f1(a):
    if(a>5):
        return True
    else:
        return False

for var in [10,2,55,6,77,18]:
    rv=f1(var)
    L.append(rv)

L
```

Out[134…
```
[True, False, True, True, True, True]
```

In [135…
```
list(map(lambda a: a>5,  [10,2,55,6,77,18]))
```

Out[135…
```
[True, False, True, True, True, True]
```

In [137…
```
list(map(lambda  a: a.upper(), open("D:\\emp.csv")))
```

Out[137…
```
['101,ARUN,SALES,PUNE,2000\n',
 '102,VISHNU,SALES,HYDERABAD,3000\n',
 '103,VIJAY,PROD,PUNE,2000\n',
 '104,RAGHAV,HR,PUNE,3000\n',
 '105,SAM,HR,BGLORE,8000\n']
```

In [138…
```
list(filter(lambda a: a>5,  [10,2,55,6,77,18]))  # returns only the matched elemnt
```

Out[138…
```
[10, 55, 6, 77, 18]
```

In [139…
```
filter( lambda a: a>30 ,[10,20,30,40,50])
```

Out[139…
```
<filter at 0x1e6a767cca0>
```

In [140…
```
list(filter( lambda a: a>30 ,[10,20,30,40,50]))
```

Out[140…
```
[40, 50]
```

In [141…
```
list( filter (lambda a: "sales" in a,["raj,sales,pune,1111","arun,prod,pune,2322"]))
```

Out[141…
```
['raj,sales,pune,1111']
```

In [142…
```
# filter doesnot supports  arithmetic opr
list(filter(lambda a: a+100, [10,20,30,40,50]))
```

Out[142…
```
[10, 20, 30, 40, 50]
```

In [143…
```
# map supports  arithmetic opr
list(map(lambda a: a+100, [10,20,30,40,50]))
```

```
Out[143… [110, 120, 130, 140, 150]
```

```
In [ ]:  python 3.x
         ==========
             map()---> iterator--> typecast --> list
             filter() ---> iterator --> tpecast--> list
```

```
In [ ]:  functools.reduce()   # reduce(function,collection)---> Singlevalue
```

```
In [144… L=[10,20,30,40,50]
         a=0
         for var in L:
             a=a+var
         else:
             print(a)

150
```

```
In [145… import functools
         functools.reduce(lambda a,b: a+b,   L)     # python 2.x   reduce(...)
         '''
         10        20        30        40       50
         ----------|
                  30
                  ------------|
                              60
                               -----------|
                                          100
                                           ---------|
                                                    150
         '''

Out[145… 150
```

```
In [154… # sum of sales emp cost

         fobj=open("D:\\emp.csv")     # map
         L=fobj.readlines()
         t=0
         for v in L:
             if "sales" in v.lower():   # filter
                # print(E)
                 E=v.split(",")
                 cost=E[-1]
                 t=t+int(cost)           #   reduce
         print("Sum of sales emp cost :{}".format(t))

Sum of sales emp cost :5000
```

```
In [148… list(map(lambda a: a, open("D:\\emp.csv")))

Out[148… ['101,Arun,sales,pune,2000\n',
          '102,Vishnu,sales,hyderabad,3000\n',
          '103,Vijay,prod,Pune,2000\n',
          '104,Raghav,Hr,pune,3000\n',
          '105,sam,Hr,bglore,8000\n']
```

```
In [149… list(filter(lambda a: "sales" in a,list(map(lambda a: a, open("D:\\emp.csv")))))

Out[149… ['101,Arun,sales,pune,2000\n', '102,Vishnu,sales,hyderabad,3000\n']
```

```
In [152… list(map(lambda a: a.split(",")[-1],list(filter(lambda a: "sales" in a,list(map(lambda a: a, open("D:\\emp.csv"

Out[152… ['2000\n', '3000\n']
```

```
In [153… functools.reduce(lambda a,b: int(a)+int(b),
                          list(map(lambda a: a.split(",")[-1],
                                   list(filter(lambda a: "sales" in a,
                                               list(map(lambda a: a, open("D:\\emp.csv"))))))))

Out[153… 5000
```

```
In [156… sum=functools.reduce(lambda a,b: int(a)+int(b),
                          list(map(lambda a: a.split(",")[-1],
                                   list(filter(lambda a: "sales" in a,
                                               list(map(lambda a: a, open("D:\\emp.csv"))))))))
         print("Sum of Sales Emp cost :{}".format(sum))

Sum of Sales Emp cost :5000
```

```
In [ ]:  Exception Handling
         ===================
             |------------> in programming --> class
             |------------> in Os --> pythncode(Process)----signal -- process ---> Exit state
         Logical error

         code block/statement
         -------------------
         try
         except
         else
         finally
```

In [ ]:
```
try:
    initialization/monitoring block
except:
    Handle this Error statement
else:
    There is no Exception
finally:
    Always running block
```

In [157…
```
var=100
print(VAR)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[157], line 2
      1 var=100
----> 2 print(VAR)

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

In [158…
```
try:
    var=100
    print(VAR)
except NameError:    #Exception
    print("Undefined Variable")
    print("Exeception has Occured")
else:
    print(VAR+100)
finally:
    print("Thankyou")

    # try-> except-> finally
```

```
Undefined Variable
Exeception has Occured
Thankyou
```

In [162…
```
try:
 #   var=100
   VAR=100
   print(VAR)
except NameError:    #Exception
    print("Undefined Variable")
    print("Exeception has Occured")
else:
    print(VAR+100)
finally:
    print("Thankyou")

    # try-> else-> finally
```

```
100
200
Thankyou
```

In [164…
```
try:
    L=[]
    print(L[4])
except NameError:
    print("Error")
except IndexError:
    print("Error")
except TypeError:
    print("Handle typeerror")
```

```
Error
```

In [165…
```
try:
    L=[]
    print(L[4])
except Exception:
    print("Exception")
```

```
Exception
```

In [168…

```
try:
    L=[]
    print(l[4])
except Exception as eobj:
    print("Exception: {}".format(eobj))
```

```
Exception: name 'l' is not defined
```

In [169…

```
L=[]
print(l[4])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[169], line 2
      1 L=[]
----> 2 print(l[4])

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

In [ ]:

```
exception happens--> try-> except->finally
exception doesnot occur -> try-> else -> finally
```