In [ ]:
```
# regx
# oops

import re
re.search() -><Object>/None - validation
re.findall() ->[result]/[]
-----------------------------------------------------------------------
^Pattern
Pattern$
^Pattern$
.
.*
[A-Z] [a-z] [0-9] [A-Za-z0-9] [^A-Za-z0-9\s] -->[^\w\s]
^$
|
()
+
{}
------------------------------------------------------------------
```

In [ ]:
```
# substitute ==> re.sub()

# re.sub("OldPattern","replacedstring","inputstring") -> 'string'
#                                                             |__replaced string
#                                                             |__original string
#

>>> import re
>>>
>>> s='101,ram,sales,pune,1000'
>>>
>>> re.sub('sales','prod',s)
'101,ram,prod,pune,1000'
>>>
>>> re.sub('QA','prod',s)
'101,ram,sales,pune,1000'
>>>
>>> re.sub('sales','prod','ram,sales,101,sales,sales,sales,sales,pune')
'ram,prod,101,prod,prod,prod,prod,pune'
>>>
>>> re.sub('sales','prod','ram,sales,101,sales,sales,sales,sales,pune',1)
'ram,prod,101,sales,sales,sales,sales,pune'
>>> re.sub('sales','prod','ram,sales,101,sales,sales,sales,sales,pune',2)
'ram,prod,101,prod,sales,sales,sales,pune'
>>> 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.

>>> re.sub('Sales','prod','ram,sales')
'ram,sales'
>>> re.sub('Sales','prod','ram,sales',0,re.I)
'ram,prod'
>>>
>>> with open("D:\\emp.csv") as FH:
...     for var in FH.readlines():
...             r=re.sub("sales","ADMIN",var)
...             print(r.strip())
...
ram,ADMIN,pune,1000
ashi,prod,bglore,2345
xerox,ADMIN,chennai,45900
yahoo,prod,pune,32450
anu,HR,hyd,4560
biju,prod,bglore,4567
vijay,hr,chennai,3453
theeb,ADMIN,hyd,5678
nithin,prod,pune,1236
>>>
>>> with open("D:\\emp.csv") as FH:
```

```
...     for var in FH.readlines():
...             if(re.search("sales",var)):
...                     r=re.sub("sales","ADMIN",var)
...                     print(r.strip())
...
ram,ADMIN,pune,1000
xerox,ADMIN,chennai,45900
theeb,ADMIN,hyd,5678
>>>
>>> re.sub('^sales','ADMIN','ram,sales,pune')
'ram,sales,pune'
>>> re.sub('sales$','ADMIN','ram,sales,pune')
'ram,sales,pune'
>>>
>>> re.sub('sales$','ADMIN','ram,sales,pune,sales')
'ram,sales,pune,ADMIN'
>>>
>>> re.sub('^[A-Z]','-','OracleLinux')
'-racleLinux'
>>> re.sub('[A-Z]','-','OracleLinux')
'-racle-inux'
>>>
>>> re.sub('[A-Z]','','OracleLinux')
'racleinux'
>>> re.sub('sales','','101,ram,sales,pune,1000')
'101,ram,,pune,1000'
>>> re.sub('sales.','','101,ram,sales,pune,1000')
'101,ram,pune,1000'
>>>
>>> r=re.sub('sales.','','101,ram,sales,pune,1000')
>>>
>>> r
'101,ram,pune,1000'
>>>
```

In [3]:

```
>>> import re
>>>
>>> import os
>>>
>>> for v in os.popen("ps -f").readlines():
...     print(re.sub("bash","KSH",v.strip()))
...
UID         PID  PPID  C STIME TTY          TIME CMD
apelix     2557  2550  0 08:40 pts/0    00:00:00 KSH
apelix     3244  2557  0 09:48 pts/0    00:00:00 python
apelix     3303  3244  0 09:48 pts/0    00:00:00 sh -c ps -f
apelix     3304  3303  0 09:48 pts/0    00:00:00 ps -f
>>> with open("process.log","w") as WH:
...     for v in os.popen("ps -f").readlines():
...             print(re.sub("bash|sh","KSH",v.strip()))
...             WH.write(re.sub("bash|sh","KSH",v)) # writing data to FILE
...
UID         PID  PPID  C STIME TTY          TIME CMD
apelix     2557  2550  0 08:40 pts/0    00:00:00 KSH
apelix     3244  2557  0 09:48 pts/0    00:00:00 python
apelix     3313  3244  0 09:50 pts/0    00:00:00 KSH -c ps -f
apelix     3314  3313  0 09:50 pts/0    00:00:00 ps -f
>>>


STEP 1: read input file (F1) line by line
STEP 2: insert ls -l at the beginning
STEP 3: (Linux) -> execute the command(ls -l <filename>) use os module
STEP 4: (Linux) -> write/append commnd results to newFILE

apelix@krosumlabs:~$ cat -n F1
     1  /etc/passwd
     2  /etc/pam.conf
     3  /etc/fstab
     4  /etc/nail.rc
     5  /etc/gai.conf
apelix@krosumlabs:~$ cat -n e1
     1  import re
     2  import os
     3  WH=open("result.log","a")
     4
     5  with open("F1") as FH:
     6      for var in FH.readlines():
     7          r=re.sub("^","ls -l ",var.strip())
     8          result=os.popen(r).read()
     9          WH.write(result)
    10
    11  WH.close()
```

Out[3]:

```
'\nimport re\n\nimport os\n\nfor v in os.popen("ps -f").readlines():\n    pri
nt(re.sub("bash","KSH",v.strip()))\n\nUID         PID  PPID  C STIME TTY
TIME CMD\napelix     2557  2550  0 08:40 pts/0    00:00:00 KSH\napelix     3244
2557  0 09:48 pts/0    00:00:00 python\napelix     3303  3244  0 09:48 pts/0
00:00:00 sh -c ps -f\napelix     3304  3303  0 09:48 pts/0    00:00:00 ps -f\n
```

```
with open("process.log","w") as WH:\n    for v in os.popen("ps -f").readlines
():\n            print(re.sub("bash|sh","KSH",v.strip()))\n            WH.wri
te(re.sub("bash|sh","KSH",v)) # writing data to FILE\n\nUID        PID  PPID
C STIME TTY          TIME CMD\napelix    2557  2550  0 08:40 pts/0    00:00:0
0 KSH\napelix    3244  2557  0 09:48 pts/0    00:00:00 python\napelix    3313
3244  0 09:50 pts/0    00:00:00 KSH -c ps -f\napelix    3314  3313  0 09:50 p
ts/0    00:00:00 ps -f\n\n\n\nSTEP 1: read input file (F1) line by line\nSTEP
2: insert ls -l at the beginning \nSTEP 3: (Linux) -> execute the command(ls
-l <filename>) use os module\nSTEP 4: (Linux) -> write/append commnd results
to newFILE\n'
```

In [ ]:

```
echo "one"
echo "Two"
echo "Three"
# echo "Four"
echo # empty line
uptime # display loadbalance
# empty line
# -------------
# ------------
echo "Today:`date +%D`" # Today date MM/DD/YYYY Format
echo "Six"


STEP 1: delete all the comment lines
STEP 2: ignore/delete all empty lines
STEP 3: create a new shellscript(p1.sh) - write non-emptylines to newfile(p1.sh)
STEP 4: using os module - execute p1.sh script
STEP 5: create a newresult file -write data to FILE

import re,os

WH=open("p1.sh","w")
with open("pt.sh") as FH:
    for var in FH.readlines():
        s=re.sub("#.*","",var)
        if(re.search("^$",s)):
            continue
        else:
            print(s.strip()) # display to STDOUT
            WH.write(s) # writing data to FILE
WH.close()

with open("result.log","w") as WH:
    WH.write(os.popen("/bin/bash p1.sh").read())
```

In [4]:
```
# recap
# s.split() ->[ ]
# |_string(str)

import re
# re.split() ->[]

# re.split("RegxPattern","inputString") ->[]

s1="root:x:bin:bash"
print(s1.split(":"))

print(re.split(":",s1))
```

```
['root', 'x', 'bin', 'bash']
['root', 'x', 'bin', 'bash']
```

In [6]:
```
s2="root:x-bin,bash"

print(re.split("[:,]",s2))

print(re.split("[^\w\s]",s2))
```

```
['root', 'x-bin', 'bash']
['root', 'x', 'bin', 'bash']
```

In [9]:
```
s="101:ram:sales:pune:prod:bglore:1002:3004:code"

print(re.split("sales|prod",s))

print(re.split("\d+",s))
```

```
['101:ram:', ':pune:', ':bglore:1002:3004:code']
['', 'ram:sales:pune:prod:bglore:', '', 'code']
```

In [ ]:

```
apelix@krosumlabs:~$ cut -d, -f 2 emp.csv
sales
prod
sales
prod
HR
prod
hr
sales
prod
apelix@krosumlabs:~$ python
Python 2.7.2+ (default, Oct  4 2011, 20:03:08)
[GCC 4.6.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import re
>>> with open("emp.csv") as FH:
...     for var in FH.readlines():
...             print(re.split(",")[1])
...
Traceback (most recent call last):
  File "<stdin>", line 3, in <module>
TypeError: split() takes at least 2 arguments (1 given)
>>>     for var in FH.readlines():
  File "<stdin>", line 1
    for var in FH.readlines():
    ^
IndentationError: unexpected indent
>>>
>>>
>>> with open("emp.csv") as FH:
...     for var in FH.readlines():
...             print(re.split(",",var)[1])
...
sales
prod
sales
prod
HR
prod
hr
sales
prod
>>> with open("emp.csv") as FH:
...     for var in FH.readlines():
...             L=re.split(",",var)
...             print("{}\t{}".format(L[1],L[-1]))
...
sales   1000

prod    2345

sales   45900

prod    32450

HR  4560
```

```
prod    4567

hr  3453

sales   5678

prod    1236

>>>
>>>
>>> with open("IP1") as FH:
...     for var in FH.readlines():
...             L=re.split("[^\w\s]",var)
...             L=re.split("[^\w\s]",var.strip())
...             print("{}\t{}".format(L[0],L[-1]))
...
data1   data4
data5   data8
data9   dataB
>>>
>>> for v in os.popen("ps -f").readlines():
...     L=re.split("\s",v.strip())
...     print("{}\t{}".format(L[0],L[-1]))
...
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'os' is not defined
>>>
>>> import os
>>>
>>> for v in os.popen("ps -f").readlines():
...     L=re.split("\s",v.strip())
...     print("{}\t{}".format(L[0],L[-1]))
...
UID CMD
apelix  bash
apelix  python
apelix  -f
apelix  -f
>>> for v in os.popen("ps -e").readlines():
...     if(re.search("bash|python|init|java|mysql",v)):
...             L=re.split("\s",v.strip())
...             print("{}\t{}".format(L[-1],L[0]))
...
init    1
mysqld  1112
bash    2557
bash    3415
python  4059
bash    4109
```

In [ ]:
```
>>> import re
>>>
>>> type(re)
<class 'module'>
>>> re
<module 're' from 'C:\\Users\\Karthikeyan\\AppData\\Local\\Programs\\Python\
hon37-32\\lib\\re.py'>
>>>
>>> re.search("sales","sdfsd sales")
<re.Match object; span=(6, 11), match='sales'>
>>>
>>> re.compile("sales")
re.compile('sales')
>>>
>>> pobj=re.compile("sales")
>>> pobj.search("ssd sales asdfsd")
<re.Match object; span=(4, 9), match='sales'>
>>> pobj.findall("asd dssales sdfs")
['sales']
>>> pboj.split("sdfs sales asfs sales sfsd")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'pboj' is not defined
>>>
>>> pobj.split("sdfs sales asfs sales sfsd")
['sdfs ', ' asfs ', ' sfsd']
>>>
>>> re.match("sales","sales sdfsd ")
<re.Match object; span=(0, 5), match='sales'>
>>>
>>> re.match("sales","emp sales sdfsd ")
>>> #######################################  re.search("^pattern","input")
>>>

>>> re.search("sales","ram,sales,pune")
<re.Match object; span=(4, 9), match='sales'>
>>>
>>> p='sales'
>>>
>>> re.search(p,'ram,sales,pune')
<re.Match object; span=(4, 9), match='sales'>
>>>
>>> p=re.compile('sales')
>>>
>>> p.search('ram,sales,pune')
<re.Match object; span=(4, 9), match='sales'>
>>>
```

In [ ]:
```
# oops
# -----
# class
# object
# method
class - type - code block - template - blueprint of an object
object - entity - value

class classname:
    attr1
    attr2
    attr3
```

In [13]:
```
class box:
    var=10
    f=1.34
    s='data'
    L=['d1','d2']
    d={"K1":"V1"}
# class member (or) class attributes
print(box.var)
print(box.L[0])
print(box.d['K1'])
print(box.s)
box.s='server1'
print(box.s)
```

```
10
d1
V1
data
server1
```

In [15]:
```
class box:
    name='root'

print(box.name) # we can access existing class attribute
box.name="admin" # we can modify existing class attribute
box.port=1020 # we can create newvariable(member)
print(box.port)
```

```
root
1020
```

In [17]:
```
# dir(<classname>)->[class attrs]

dir(box)
box.__dict__
```

Out[17]:
```
mappingproxy({'__module__': '__main__',
              'name': 'admin',
              '__dict__': <attribute '__dict__' of 'box' objects>,
              '__weakref__': <attribute '__weakref__' of 'box' objects>,
              '__doc__': None,
              'port': 1020})
```

In [18]:
```
# class - blueprint of an object
#  |                        |
# type                     value

# +--------------------------------------+
# |  [ ]    [  ]                [  ]     |  white
# |                                      |  blueprint(class)
# |______________ ___  _   ______________ |
#   |       |          |          ...    |
#   |       |          |                 |
# B1        B2         B3                Bn <== building (object)
# 1st       2nd        3rd               nth <== address(memory)
# green     itemA
object=classname()
```

In [19]:
```
class box:
    name="root"
    port=1230
obj=box()
print(obj.name)
print(obj.port)
```

```
root
1230
```

In [ ]:

```
>>> class box:
...     name="root"
...
>>> box
<class '__main__.box'>
>>> box()
<__main__.box object at 0x00741ED0>
>>> box()
<__main__.box object at 0x00741E30>
>>> obj1=box()
>>> obj2=box()
>>>
>>> type(obj1)
<class '__main__.box'>
>>> type(obj2)
<class '__main__.box'>
>>> obj1.name
'root'
>>> obj2.name
'root'
>>> box.name
'root'
>>> box.name='admin'
>>> obj1.name
'admin'
>>> obj2.name
'admin'
>>> box.name='server'
>>> obj1.name
'server'
>>> obj2.name
'server'
>>> obj1.name="Green"
>>> obj2.name
'server'
>>> box.name
'server'
>>> obj2.name="Yellow"
>>> obj2.name
'Yellow'
>>> obj1.name
'Green'
>>> box.name="XYZ"
>>> obj1.name
'Green'
>>> obj2.name
'Yellow'
>>>
```

In [24]:
```
class serverinfo:
    name="default-server"

s1=serverinfo()
s2=serverinfo()

print(s1.name) # default-server
print(s2.name) # default-server
s1.name="Linux"
s2.name="Unix"
print(s1.name) # Linux
print(s2.name) # Unix

s3=serverinfo()
print(s3.name) # default-server
serverinfo.name="Sunos" # blueprint(class) changes
print(s3.name) # Sunos
print(s2.name) # Unix
print(s1.name) # Linux
s3.name="Aix"
serverinfo.name='10.20.30.40'
print(s3.name) # Aix
s4=serverinfo()
print(s4.name)
```

```
default-server
default-server
Linux
Unix
default-server
Sunos
Unix
Linux
Aix
10.20.30.40
```

In [26]:
```
# filesysteminfo
#  |   |   |
# obj1obj2obj3

class fsinfo:
    fstype=''
    findex=0
    fmount="/"

obj1=fsinfo()
obj1.fstype="xfs"
obj1.findex=1000
obj1.fmount="/D1"

obj2=fsinfo()
obj2.fstype="btrfs"
obj2.findex=2000
obj2.fmount="/D2"

print(obj1.fstype,obj1.findex,obj1.fmount)
print(obj2.fstype,obj2.findex,obj2.fmount)
print(fsinfo.fstype,fsinfo.findex,fsinfo.fmount)
```

```
xfs 1000 /D1
btrfs 2000 /D2
 0 /
```

In [ ]:
```
>>> class Box:
...     __var=100
...
>>> Box.__var
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: type object 'Box' has no attribut
>>> obj=Box()
>>> obj.__var
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Box' object has no attribute '__
>>>
```

In [27]:
```
class student:
    name=''
    usn=''
    dept=''
obj1=student()
obj1.name='arun'
obj1.dept='CSE'
obj1.usn='1abc001'

obj2=student()
obj2.name='vijay'
obj2.dept='MECH'
obj2.usn='3b13454'

obj3=student()
obj3.name='anu'
obj3.dept='Maths'
obj3.usn='4G3433'

print("Name:{}\tDept:{}\tUSN:{}".format(obj1.name,obj1.dept,obj1.usn))
print("Name:{}\tDept:{}\tUSN:{}".format(obj2.name,obj2.dept,obj2.usn))
print("Name:{}\tDept:{}\tUSN:{}".format(obj3.name,obj3.dept,obj3.usn))
```

```
Name:arun       Dept:CSE        USN:1abc001
Name:vijay      Dept:MECH       USN:3b13454
Name:anu        Dept:Maths      USN:4G3433
```

In [29]:
```
obj3.dept='Civil'
print("Name:{}\tDept:{}\tUSN:{}".format(obj3.name,obj3.dept,obj3.usn))

student.bgroup='' # adding new attribute to existing class

obj1.bgroup="A+Ve"
obj2.bgroup="AB-Ve"
obj3.bgroup="O+ve"
print("Name:{} bgroup is:{}".format(obj1.name,obj1.bgroup))
print("Name:{} bgroup is:{}".format(obj2.name,obj2.bgroup))
print("Name:{} bgroup is:{}".format(obj3.name,obj3.bgroup))
```

```
Name:anu        Dept:Civil      USN:4G3433
Name:arun bgroup is:A+Ve
Name:vijay bgroup is:AB-Ve
Name:anu bgroup is:O+ve
```

In [33]:
```
def f1():
    print("Hello")
#f1(10)
```

In [35]:
```
class Box:
    def f1(self):
        print("Welcome")

obj=Box()
#obj.f1() # method call  --> f1(obj)
obj.f1()
# obj.f1(10,20,30,40) ==>f1(obj,10,20,30,40) -->def f1(self,a1,a2,*a3):
```

```
Welcome
```

In [41]:
```
# class student:
#     attr
#     method1 ->f1() ->initialization
#     method2 ->f2() ->display student info
#     method3 ->f3(dept) ->update student dept
#

class student:
    name=''
    dept=''
    usn=''
    def f1(self,a1,a2,a3):
        self.name=a1
        self.dept=a2
        self.usn=a3
    def f2(self):
        print("NAME:{}\tDEPT:{}\tUSN:{}".format(self.name,self.dept,self.usn))
    def f3(self,a1):
        self.dept=a1

s1=student()
s1.f1("Arun","CSE","1abc22323")
s1.f2()

s2=student()
s2.f1("Vijay","MECH","3dfd332")
s2.f2()

s3=student()
s3.f1("Anu","Maths","4F23131")
s3.f2()
```

```
NAME:Arun       DEPT:CSE        USN:1abc22323
NAME:Vijay      DEPT:MECH       USN:3dfd332
NAME:Anu        DEPT:Maths      USN:4F23131
```

In [43]:
```
class student:
    __name=''
    __dept=''
    __usn=''
    def f1(self,a1,a2,a3):
        self.__name=a1
        self.__dept=a2
        self.__usn=a3
    def f2(self):
        print("NAME:{}\tDEPT:{}\tUSN:{}".format(self.__name,self.__dept,self.__us
    def f3(self,a1):
        self.__dept=a1
```

In [ ]:
```
>>> import e1
>>> e1.student()
<e1.student object at 0x00571AB0>
>>>
>>> e1.student
<class 'e1.student'>
>>>
>>> class box:
...     pass
...
>>> box
<class '__main__.box'>
>>>
>>> obj1=e1.student()
>>> obj1.f1("Arun","CSE","1as34334")
>>> obj1.f2()
NAME:Arun       DEPT:CSE        USN:1as34334
>>> obj1.f3("MECH")
>>> obj1.f2()
NAME:Arun       DEPT:MECH       USN:1as34334
>>> obj2=e1.student()
>>> obj2.f1("Vijay","ECE","412123")
>>> obj2.f2()
NAME:Vijay      DEPT:ECE        USN:412123
>>>
>>> obj1.f2()
NAME:Arun       DEPT:MECH       USN:1as34334
>>>
>>> from e1 import student
>>> obj=student()
>>> obj.f1("anu","maths","3343242")
>>> obj.f2()
NAME:anu        DEPT:maths      USN:3343242
>>>
>>> obj.f3("Civil")
>>> obj.f3()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: f3() missing 1 required positional argument: 'a1'
>>>
>>> obj.f2()
NAME:anu        DEPT:Civil      USN:3343242
>>>
```

In [ ]:
```
# constructor - special method -> __init__()
```

In [47]:
```
class student:
    __name=''
    __dept=''
    __usn=''
    def __init__(self,a1,a2,a3):
        self.__name=a1
        self.__dept=a2
        self.__usn=a3
    def f2(self):
        print("NAME:{}\tDEPT:{}\tUSN:{}".format(self.__name,self.__dept,self.__us
    def f3(self,a1):
        self.__dept=a1

s1=student("Arun","CSE","1sdfs3343")
s1.f2()

s2=student("Vijay","MECH","4f433223")
s2.f2()

s3=student("Anu","Maths","5FA3343")
s3.f2()

s3.f3("Civil")
s3.f2()
```

```
NAME:Arun       DEPT:CSE        USN:1sdfs3343
NAME:Vijay      DEPT:MECH       USN:4f433223
NAME:Anu        DEPT:Maths      USN:5FA3343
NAME:Anu        DEPT:Civil      USN:5FA3343
```

In [50]:
```
class Box:
    def __init__(self,a1,a2,a3):
        self.VAR1=a1
        self.VAR2=a2
        self.VAR3=a3
    def f1(self):
        return self.VAR1,self.VAR2,self.VAR3

obj1=Box(10,20,30)
print(obj1.f1())
#print("-->{}".format(obj1.VAR1))
obj2=Box("D1","D2","D3")
print(obj2.f1())
```

```
(10, 20, 30)
-->10
('D1', 'D2', 'D3')
```

In [51]:
```
class fsinfo:
    def __init__(self,fstype='ext',fsmount="/",fsize="0KB"):
        self.f1=fstype
        self.f2=fsmount
        self.f3=fsize
    def display(self):
        print("Mounted filesystem details:-")
        print("{}\t{}\t{}".format(self.f1,self.f2,self.f3))

obj1=fsinfo("xfs","/D1","120KB")
obj2=fsinfo("btrfs","/D2","4343MB")
obj3=fsinfo("ext4","/D3","34334GB")

obj1.display()
obj2.display()
obj3.display()
```

```
Mounted filesystem details:-
xfs     /D1     120KB
Mounted filesystem details:-
btrfs   /D2     4343MB
Mounted filesystem details:-
ext4    /D3     34334GB
```

In [52]:
```
obj4=fsinfo()
obj4.display()
```

```
Mounted filesystem details:-
ext     /       0KB
```

In [ ]:
```
#          X
#          |
# --------------------
# |    |    |    |   |
# V1  V2   V3   V4 ..Vn <== vendor
#
#  billing():
#       create a newfile(vendorinfo.log)
#       append all the vendor billing details to vendorinfo.log file
#
# vendorname,vendorID,Product,Cost,Qty
# -------------------------------- ====
#
```

In [75]:
```
class vendorinfo:
    def __init__(self,vn,vi,gst,contact):
        self.vname=vn
        self.vID=vi
        self.vGST=gst
        self.contact=contact
    def billing(self,pname,price,qty=0):
        self.total=0
        self.price=price
        self.total=self.price*qty
        self.product=pname
    def FILEOPERATION(self):
        with open("D:\\vendor_info.log","a") as WH:
            WH.write("-"*45+"\n")
            WH.write("VendorNAME:{}\tVendorID:{}\tGST:{}\tBillAmount:{}\n\n".forn
            WH.write("-"*45+"\n")
```

In [76]:
```
v1=vendorinfo("ABC","V-001","GST-V1",990230304)
v1.billing("Product-A",1000,5)
v1.FILEOPERATION()

v2=vendorinfo("XYZ","V-002","GST-V2","080-343323")
v2.billing("Product-B",1234,3)
v2.FILEOPERATION()
```

In [ ]:
```
'''
class classname:
    datamember(or) attribute

classname.<datamember> # blueprint access

obj=classname()
obj.<datamember>

obj.function() # method call -> function(obj)

class classname:
    def function(self,a1,a2=0,*a3,**a4):
        ....
        ....
class classname:
    def __init__(self,....):

import bs4
obj=bs4.BeautifulSoup("webpage")
obj.find("p")
obj.find("a")
obj.find("b")
'''
```

In [ ]:
```
class - type

a=10 -------------> class int:
                          def __init__(self,a=0):
                                self.a=a
s1='' -------------> class str:
                          def __init__(self,a=''):
                                self.a=a

f=1.34 -----------> class float:
                           def __init__(self,a=0.0):
s2="Hello"
```

In [78]:
```
a=10 # procedure style
b=int(10) # oops - constructor call
print(a,type(a))
print(b,type(b))
c=1.354
d=float(1.456)
print(c,type(c))
print(d,type(d))
```

```
10 <class 'int'>
10 <class 'int'>
1.354 <class 'float'>
1.456 <class 'float'>
```

In [79]:
```
a=int()  # obj=classname()
print(a)

#object - parent
#   int,float,str,bool,list, ... //subclass
```

```
0
```

In [ ]:
```
type(10) type(11) type(20)
|        |           |
int      int         int

int
|_10,11,20 ....
```

In [ ]:
```
>>> a=10
>>> b=3+7
>>>
>>> id(a)
1606968608
>>> id(b)
1606968608
>>> hex(id(b))
'0x5fc86520'
>>>
>>> hex(id(a))
'0x5fc86520'
>>> class Box:
...     pass
...
>>> obj1=Box()
>>> obj2=Box()
>>> obj3=Box()
>>>
>>> type(obj1)
<class '__main__.Box'>
>>> type(obj2)
<class '__main__.Box'>
>>> type(obj3)
<class '__main__.Box'>
>>>
>>> type(10)
<class 'int'>
>>> type(20)
<class 'int'>
>>> type(11)
<class 'int'>
>>> type(0)
<class 'int'>
>>> hex(id(obj1))
'0xd81eb0'
>>> hex(id(obj2))
'0xd81e70'
>>> hex(id(obj3))
'0xd81df0'
>>>
>>>
>>> a=10
>>> b=3+7
>>> c=4+6
>>> d=1+9
>>>
>>> hex(id(a))
'0x5fc86520'
>>> hex(id(b))
'0x5fc86520'
>>> hex(id(c))
'0x5fc86520'
>>> hex(id(d))
'0x5fc86520'
>>>
```

```
>>> hex(id(10))
'0x5fc86520'
>>>
>>> hex(id(11))
'0x5fc86530'
>>> hex(id(12))
'0x5fc86540'
>>> hex(id(13))
'0x5fc86550'
>>>
>>>
>>>
>>> s='45'
>>> i=int(s)
>>>
>>> i
45
>>> s=set()
>>>
>>> L=[]
>>> L=list()
>>> L
[]
>>> d=dict()
>>>
>>> d=dict()
>>> d
{}
>>> s=''
>>> s=str()
>>> s
''
>>> s=str('abc')
>>> print(s)
abc
>>> obj=str("abc")
>>> obj.upper()
'ABC'
>>> obj.title()
'Abc'
>>> help(obj.upper)
Help on built-in function upper:

upper() method of builtins.str instance
    Return a copy of the string converted to uppercase.

>>> help(str.upper)
Help on method_descriptor:

upper(self, /)
    Return a copy of the string converted to uppercase.

>>> help(list.append)
Help on method_descriptor:

append(self, object, /)
    Append object to the end of the list.
```

```
>>> # class list:
>>> #   def append(self,a1):
>>> #            ....
>>> #   def insert(self,index,a1):
>>> #
>>> #   def pop(self,index=-1);
>>> #        ...
>>>
>>> L=list()
>>> L.append("D1") # append(L,"D1")
>>> L.insert(1,"D2") # insert(L,1,"D2")
>>> L
['D1', 'D2']
>>>
```

In [84]:
```
# inheritance
# ------------------

# class Childname(Parentname):   vs    def  function(args):

class P1:
    def f1(self):
        print("F1block-P1 class")
class P2(P1): # inheritance - single inheritance
    def f2(self):
        print("F2block-P2 class")

obj=P2()
obj.f1()
obj.f2()
```

```
F1block-P1 class
F2block-P2 class
```

In [85]:
```
# obj1.f2() # AttributeError
#obj2.f1() # AttributeError
```

In [ ]:
```
# python 2.7,3.x         python 2.6
class Box:               class Box(object):
    var=100                    var=100
```

In [86]:
```
class P1:
    def f1(self):
        print("Welcome")
class P2(P1):
    def f1(self):
        print("Testcode")
obj=P2()
obj.f1()
obj.f1()
```

```
Testcode
Testcode
```

In [87]:
```
class P1:
    def f1(self):
        print("Welcome")
class P2(P1):
    def f1(self):
        print("Hello")
        super(P2,self).f1() # calling parent block
        print("exit from child")

obj=P2()
obj.f1()
```

```
Hello
Welcome
exit from child
```

In [88]:
```
class P1:
    def __init__(self):
        print("Parent block constructor")
class P2(P1):
    def f1(self):
        print("Childblock")

obj=P2()
```

```
Parent block constructor
```

In [90]:
```
class P1:
    def __init__(self):
        print("Parent block constructor")
class P2(P1):
    def __init__(self):
        print("Childblock")
        super(P2,self).__init__()

obj=P2()
```

```
Childblock
Parent block constructor
```

In [91]:
```
class A:
    def f1(self):
        print("ParentClass")

class B(A):
    def f1(self):
        print("ChildClass")
        A.f1(self) # calling parent block
obj=B()
obj.f1()
```

```
ChildClass
ParentClass
```

In [ ]:
```
# File: ab.py                                   file:sab.py
# -----------------                            ============
# class box():                                    import ab
#         def f1(self):                            class Fax(ab.box):
#         def f2(self):                                   def f4(self):
#         def f3(self):                                   def f5(self):
#                                                         def f3(self):
#                                                              pass


import sab
obj=sab.Fax()
obj.f1()
obj.f2()
obj.f3()
obj.f4()
obj.f5()
```

In [92]:
```
class Fax:
    def __init__(self,*args):
        print("Parentclass")
        print(args)
class Box(Fax):
    def __init__(self,*args):
        print("Child class")
        print(args)
        super(Box,self).__init__("A1","A2","A3") # calling parent class

obj=Box("D1","D2","D3","D4")
```

```
Child class
('D1', 'D2', 'D3', 'D4')
Parentclass
('A1', 'A2', 'A3')
```

In [93]:
```
class Product1:
    pname="P-A"
    pcost=1000
class Product2:
    ptag="Pxyz"

class Box(Product1,Product2):
    pass

obj=Box()
print(obj.pname)
print(obj.pcost)
print(obj.ptag)
```

```
P-A
1000
Pxyz
```

In [94]:
```
# P1
# |
# P2
# |
# P3
# |
# P4
# ..
class version1:
    pname="PA"
    pcost=100

class version2(version1):
    count=100
class version3(version2):
    cname="abc"
obj=version3()
print(obj.pname)
print(obj.pcost)
print(obj.count)
print(obj.cname)
```

```
PA
100
100
abc
```

In [ ]:
```
>>> __var=100
>>> __var
100
>>> class box:
...     __p=234
...
>>> box.__p
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: type object 'box' has no attribute '__p'
>>>
>>> class box:
...     __a__=34
...
>>> box.__a__
34
>>> help(list)
Help on class list in module builtins:

class list(object)
 |  list(iterable=(), /)
 |
 |  Built-in mutable sequence.
 |
 |  If no argument is given, the constructor creates a new empty list.
 |  The argument must be an iterable if specified.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return key in self.
 |
 |  __delitem__(self, key, /)
 |      Delete self[key].
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(...)
 |      x.__getitem__(y) <==> x[y]
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __iadd__(self, value, /)
 |      Implement self+=value.
 |
 |  __imul__(self, value, /)
 |      Implement self*=value.

>>>
>>> class box:
...     pass
...
>>>
>>> dir(box)
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '_
at__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__i
ubclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduc
 '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__su
shook__', '__weakref__']
>>>
>>>
>>>
>>> __name__
'__main__'
>>>
>>> import cgi
>>> dir(cgi.FieldStorage)
['FieldStorageClass', '_FieldStorage__write', '__bool__', '__class__', '__c
ns__', '__del__', '__delattr__', '__dict__', '__dir__', '__doc__', '__enter
'__eq__', '__exit__', '__format__', '__ge__', '__getattr__', '__getattribut
 '__getitem__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__i
', '__le__', '__len__', '__lt__', '__module__', '__ne__', '__new__', '__red
', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__
asshook__', '__weakref__', 'bufsize', 'getfirst', 'getlist', 'getvalue', 'k
 'make_file', 'read_binary', 'read_lines', 'read_lines_to_eof', 'read_lines
uterboundary', 'read_multi', 'read_single', 'read_urlencoded', 'skip_lines'
>>>
>>> "__name__" in dir(cgi.FieldStorage)
False
>>> __name__
'__main__'
>>>
>>>
>>> __name__
'__main__'
>>>
>>> import cgi
>>> cgi.FieldStorage
<class 'cgi.FieldStorage'>
>>>
>>> cgi.FieldStorage.__name__
'FieldStorage'
>>> cgi.FieldStorage.__module__
'cgi'
>>> box
<class '__main__.box'>
>>> box.__name__
'box'
>>> box.__module__
'__main__'
>>>
>>> # module.<member>
>>> # ------
>>> # __main__.box
>>>
>>>
```