Python Training

1) Eng Stream
2) Classes - C++/Java/C#/Theory
3) Scripts - shell/perl/python/tcl/VBS/JS/Ruby/Go/R/Scala/

Scripter
Dev
Ana
Test
QA

Develop "C" - Fns
Develop "C++" - Class
Develop "C" - Modules
Develop "FP" - Inbuilt Fns

Terminology

1) Entity - Exists/Defined/Diff
2) Attributes - Data on Entity
3) Behvrs - Functionality
4) Class - generalized Entity
5) Object - Entity
6) Data mem - attrs
7) MEthod - behvrs
8) create obj - varname = new CLASSNAME
9) access dm - varname.datamember
10) invoke met - varname.methodname()
11) DEclare var in python
12) Type checking
13) MEmory Allocation
14) Python References
15) What is Reference counting - GC
16) what is Shallow copy/Deep copy - shallow

```
17) What is Data Pooling           - 2^8 or 2^16
18) what is Garbage Collector

Python Impl:-
=============
1) written using "C"            - CPython
2) written using "Java"         - Jython
3) written using "C#"           - IronPython
4) written using "C & CPYthon"  - Stackless python
5) written using "CPYTHON"      - pypy

                       python

         python 2.x                python 3.x
         2.4                       3.3
         2.6                       3.4
         2.7.x                     3.5
                                   3.6.x

>>syntax
>>Library
>>2to3
>>lib2to3

python2.x -> a=raw_input("Enter a value ")
          -> a=input("Enter a value : ")

          -> print "hello"

python3.x -> a=input("Enter a value ")
          -> print("Hello")
```

Basics(variables/std i/o)

Type Conversion:-

```
>> param Ctor
```

```
num1 = int(raw_input("Enter first  : "))
```

eg:-

```
num2 = raw_input("Enter second : ")
res = num1 + int(num2)   # Anon Objects
print "sum of %s and %s is %s" %(num1,num2,res)
```

>>int-class

===========

```
1)      num1=10     # decimal
        num2=0b1110  # BInary
        num3=0o130   # octal
        num4=0xffff  # hexa
2)
        print bin(num)
        print oct(num)
        print hex(num)

3)
        char="a"
        print ord(char)
        print chr(65)
```

Eg:-

```
        reslst = map(chr,xrange(ord("a"), ord("z")+1))
        print reslst
```

```
4)
        data="1010"
        #eq decimal of the binary string
        res = int(data,base=2)
        print res
```

```
5)
        num=12345
        print str(num)[-1]
```

```
6)
        num=12345
        # sum of digits
        print sum(map(int,str(num)))
```

str-class:-

```
===========
>> Collection of chars
>> im-mutable
>> default char set - ASCII
>> supports unicode - DB/WEB
```

a="hello"

a='hello'

a='''hello

world'''

```
a=r'10\n20\n30'    # raw string
a=R'10\n20\n30'    # raw string
define a str        :  a="hello world"
length              :  len(a)
first char          :  a[0]
last char           :  a[-1]
```

```
first 4 char        :  a[0:4]
except first 4      :  a[4:]
last 4 char         :  a[-4:]
except last 4       :  a[0:-4]
ex f4 & ex l4       :  a[4:-4]
alt chars           :  a[::2]
alt char            :  a[1::2]
reverse             :  a[::-1]
uppercase           :  b = a.upper()   # copy of the object
Concate             :  c = a+b
replace             :  b = a.replace("hai","bye")
trim                :  b = a.strip()   # a.lstrip() & a.rstrip()
count               :  b = a.count("hai")
get index substr    :  b = a.index("hai")
search              :  if "hai" in a:
split               :  flst = a.split("-")
```

```
Eg:
Given:-
-------
harish
Expected:-
----------
Harish
```

```
solution:-
----------
name = raw_input("Enter the string : ")
res = name[0].upper() + name[1:]
print res

Given:-
-------
hareesh
Expected:-
----------
ha-REE-sh

solution:-
----------
name = raw_input("Enter the string : ")
res = name[0:2]+"-"+name[2:-2].upper()+"-"+name[-2:]
print res
-----------------------------------------------------------------
a="10-may-2010"
flst = a.split("-")
print flst[0]
print flst[-1]
print flst[1:]




a="hello world of unix"
first word
last word
first words last char
last words first char
solution:-
----------
```

```
wlst = a.split()
print wlst[0]
print wlst[-1]
print wlst[0][-1]
print wlst[-1][0]

a="arun-cse-10,20,30"
find total marks
mlst = a.split("-")[-1].split(",")
mlst = [ int(elem) for elem in mlst ] # list compre
OR
mlst = map(int,mlst)                  # functional programming
print sum(mlst)

a="10 20,30,40 50 60"
incr every elem by 1
numlst = a.replace(","," ").split()
numlst = map(lambda x : int(x)+1, numlst)
print numlst
================================================================
====================
Given:-
-------
a="10-20-30-40-50"
Expected:-
----------
b="11-21-31-41-51"

solution:-
----------
numlst = a.split("-")
numlst[:] = map(lambda x: int(x)+1, numlst)
numlst[:] = map(str,numlst)
b = "-".join(numlst)
```

----

if-else:-

```
==========
>>there is no flower braces - instead we use indented blocks
>>else-if - elif
>>&&       - and
>>||       - or
>>!        - not
a=10
b=20
if a==b:
    a=a+5
    b=b+5
else:
    a=a-5
    b=b-5

print "a = ",a
print "b = ",b

Given:-
-------
yash/hari

Expected:
---------
u r named ends with CONSONANT
u r named ends with VOWEL

solution:-
----------
name = raw_input("Enter u r name : ").lower()
```

```
if name[-1] in "aeiou":
  print "u r names last letter is VOWEL"
else:
  print "u r names last letter is CONSO"


Given:-
-------
dob=12-10-2016
dob=12/10/2106

Expected:-
-----------
Happy Birthday

Belated/adv Wishes


Hint:-
-------
import time

print time.strftime("%d-%m-%Y")

or

import datetime

today = datetime.date.today()


solution:-
-----------
import time
```

```
dob = raw_input("Enter ur date of birth : ")
dob = dob.replace("/","-")

day,month,year = dob.split("-")  # list unpacking

cday = time.strftime("%d")
cmon = time.strftime("%m")

if int(day)==int(cday) and int(month)==int(cmon):
  print "HBD"
else:
  print "Belated/ADv Wishes"



OR
import time
import datetime

dob = raw_input("Enter ur date of birth : ")
dob = dob.replace("/","-")

day,month,year = dob.split("-")  # list unpacking

cday = datetime.date.today().day
cmon = datetime.date.today().month

if int(day)==cday and int(month)==cmon:
  print "HBD"
else:
  print "Belated/ADv Wishes"
```

How to generate natural nos in python:-
=======================================

```
range(start,stop-1,step)/range(stop-1)      - inbuilt fn
xrange(start,stop-1,step)/xrange(stop-1)    - class
```

```
[1,2,3,4,5,6,7,8] =  range(1,9)
[10,20,30,40,50]  =  range(10,60,10)
[0,2,4,6,8]       =  range(0,9,2)
[1,3,5,7,]        =  range(1,8,2)
[0,1,2,3,4]       =  range(5)
[10,9,8,7,6,5,4,3,2,1] = range(10,0,-1)
```

Iterators:-

```
===========
>> collection
>> auto start
>> auto increments
>> stops when in encounter the StopIteration

it = iter([10,20,30,40,50])

print next(it)
print next(it)
print next(it)
print next(it)
print next(it)

print next(it)   # exception
```

```
for-iterator:-
==============
>> forward iterator
>> const iterator


it = iter([10,20,30,40,50])

for elem in it:
   print elem



ex1:
# read only
numlst = [10,20,30,40,50]
for elem in numlst:
   print elem

ex2:
# read & write back
numlst = [10,20,30,40,50]
for index in xrange(len(numlst)):
   print  index, numlst[index]

ex3:
# read & write back - damn slow
numlst = [10,20,30,40,50]
for index,value in enumerate(numlst):
   print  index, value

ex4:
# read only

alst =[10,20,30,40]
blst =["a",b","c","d"]
for elem1,elem2 in zip(alst,blst):
  print elem1,elem2
```

tuple-class:-

```
=============
>> const collection
>> im-mutable
>> Faster
>> can be used as a DICT-Keys


define          :   a=(10,20,30,40,50)
                    a=10,20,30,40,50
length          :   len(a)
first elem      :   a[0]
compare 2       :   if a==b:
merge 2         :   c=a+b
search 40       :   if 40 in a:
iterate         :   for elem in a:
                      print elem


Tuple unpacking:-
-----------------
1) a,b,c = 10,20,30

2) a,b,c,d = 10,20,30,40,50,60

3) a,b,c,d = 10,20

4) a,b = "today","tomm"
   a,b = b,a

5) dob="12-may-2010"
```

```
   day,month,year = dob.split("-")  # list unpacking
6) print "HEllo %s and %s and %s" %(a,b,c)
```

list-class:-

```
============
>> collection
>> mutable
>> Bit slow

defin empty   : a=[]
define        : a=[10,20,30,40,50]
length        : len(a)

first elem    : a[0]
is it valid   : a[0] = 25
Add one elm   : a.append(60)   # actuals
Merge-2       : a.extend(b)
insert        : a.insert(0,value)

del val index 0 : a.pop(0)
del val 25      : a.remove(25)

sort asc order  : a.sort()   # in-place sort
sort desc order : a.sort(reverse=True)
reverse         : a.reverse()
stat-fns        : sum()/max()/min()
```

```
1) prepare a list with 10 elements
   every element set to ZERO
   alst=[0]*10

2) prepare a list with 10 elements
   every element set from 1 to 10
   alst=range(1,11)
   alst=list(range(1,11))

3) alst = [10,20,30,40,50,60,70]
   delete first 4 elements
   alst[:] = alst[4:]
   or
   del alst[0:4]

4) alst = [10,20,30,40,50,60,70,80]
   mid = len(alst)//2  # floor division
   alst[:mid],alst[mid:] = alst[mid:],alst[0:mid]

5) alst = [10,20,30,40,50,60]
   replace the last 4 by - zero
   alst[-4:] = [0]*4

6) import copy
   alst = [10,20,30,40,50]
   blst = alst[:]  # simple list deep copy
   blst = copy.deepcopy(alst) # universal deep copy
   alst[0:4] = [0]*4
   if alst == blst:
      print "yes"
   else:
      print "no"
```

```
Eg:
zonelst = ["south-blr-Q1,10,Q2,43,Q3,54,Q4,28",
           "north-
           "east-
           "west-
          ]
prompt the user to enter zone name
if that exists
name = south
city = blr
2ndbest = 43
Qtr     = Q2
Total   = ?

solution:-
----------
zonelst = ["south-blr-Q1,10,Q2,43,Q3,54,Q4,28",
           "north-blr-Q1,10,Q2,43,Q3,54,Q4,28",
           "east-blr-Q1,10,Q2,43,Q3,54,Q4,28",
           "west-blr-Q1,10,Q2,43,Q3,54,Q4,28",
          ]

zname = raw_input("Enter the zone name : ")
flag=False

for elem in zonelst:
  zone,city,vals = elem.split("-")
  if zone==zname:
     print "Zone name = ",zname
     print "City      = ",city
     vlst = vals.split(",")
     qlst = map(int,vlst[1::2])
     print "2nd best = ",sorted(qlst)[-2]
     print "2nd best = ",vlst[vlst.index(str(sorted(qlst)[-2]))-
1]
     flag=True
     break

if not flag:
  print "zone not found"
```

```
datalst=[
         ["arun","cse",58],
         ["ravi","cse",78],
         ["manu","ece",39],
         ["john","ise",74]
        ]

for elem in datalst:
   print elem[0],elem[-1]
```

Set Class

```
===========
>> collection of unique values
>> duplicates are automatically deleted
>> non-sequence
>> mutable
```

```
define a Empty set :  a = set()
define a set       :  a = {10,20,30,40,50}
                   :  b = {20,40,60,80}
```

```
length                :  len(a)
search 50             :  if 50 in a
add one elem          :  a.add(25)

union                 :  a|b
intersection          :  a&b
difference            :  a-b
uncommon              :  a^b
```

dict-class:-

```
============
>> collection of key-value pairs
>> Non-sequence
>> mutable



define a empty dict : a={}
define a dict       : colors={"red": 10, "blue":20 }

search for "pink"     : if "pink" in colors.keys()
get value of "red"    : res = colors.get("red",0)
                      : res = colors["red"]
add new "green"-50    : colors["green"] = 50
overwrite "red" - 55  : colors["red"]   = 55
delete red            : colors.pop("red")
keys lst              : klst = colors.keys()
vals lst              : vlst = colors.values()

iterate a dict:-
----------------
```

```
ex1:
for key in colors.keys():
  print key,colors[key]

ex2:
for key,value  in colors.items():
  print key,value

Given:-
--------
studs = {
          "arun"  :  [10,20,30,40,50],
          "ravi"  :  [20,43,65,34,65],
          "john"  :  [43,65,34,65,87]
        }

Expected:-
----------

arun - 2nd min  - without using sort()/sorted()
ravi - 2nd min
john - 2nd min



for key,value in studs.items():
    lowest = min(value)
    value.remove(lowest)
    print key,min(value)

================================================================
===================
```

Nested Data structure:-

```
-----------------------
a=[
    [10,20,30],
    [40,50,60],
    [70,80,90]
  ]

a={"arun" : [10,20],
   "ravi" " [30,40]
  }

a={"arun" : {"dept" : "sales", "loc" : "blr" },
   "ravi" : {"dept" : "sales", "loc" : "blr" },
  }



JSON
Ruby
QML
Perl
BSON




Extended Data structure:-
-------------------------
import array

a = array.array("i")

a.append(10)
a.append(20)
a.append("hello")

=================================================================
===================
```

Functions:-

```
-----------
>> Functions are objects in python
>> fns has to defined before its called - Interpreter
>> fns can return multiple values       - Tuple
>> var defined within fn by default local
>> There is no Type checking

>> call by value/call by reference
>> global keyword
>> posit/default/keyword
>> variable args
>> one liner fns - lambda expressions

ex:
def add2nos(a,b):
  ans = a+b
  city='chennai'

  print "CITY = ",city

  print "CITY = ",globals()["city"]




num1=10
num2=20
city = 'bengaluru'
add2nos(10,20)

Problem:-
=========

def fun():
  global num
  num=40
  print "Fun ",num

num=55
print "Main = ",num
fun()
print "Main = ",num
```

Guess:-
========

```
def callme(alst):
  alst[0:4] = [0] * 4


numlst = [10,20,30,40,50,60]

print numlst

callme(numlst)

print numlst   ##### here
```

A) [10,20,30,40,50,60]

B) Error im-mutable

C) [0,0,0,0,50,60]

D) Error

```
positional args:-
-----------------
>> strict in terms of no of args
>> strict in order of args

def addrecord(name,dept,loc,salary):
  pass


addrecord("arun","sales","Blr",25000)
addrecord("arun","sales","Blr")
addrecord("arun","sales")
addrecord("arun")
addrecord()


default args:-
-----------------
>> strict in order of args

def addrecord(name=None,dept="sales",loc="blr",salary=0):
  pass


addrecord("arun","sales","Blr",25000)
addrecord("arun","sales","Blr")
addrecord("arun","sales")
addrecord("arun")
addrecord()


keyword args:-
```

```
--------------
def addrecord(name=None,dept="sales",loc="blr",salary=0):
  pass


addrecord(dept="hrd")
addrecord(salary=14000,name="ranjith")
```

```
Guess:-
=======
def fun(lst=[],value=10):
   lst.append(value)
   print lst


fun()
fun()
fun()
alst = [1,2,3,4]
fun(alst,5)
fun()
fun()   #### what is the output here
```


Variable args:-

>> type of args - Tuple

```
def fun(*args):
  print args
```

```
fun(1,2,3,4,5)
fun(1,2,3)
fun("A","b")
fun()
fun(1,2,3,4)


>> type kwargs - dict

def fun(**kwargs):
   print kwargs
fun(a=10,b=20,c=30,d=40)
fun(now=1,later=2)
fun()
fun(older=1)



def fun(*args,**kwargs):  #most used semantic
  print args
  print kwargs
```

Nested Functions:-
==================

ex1:

```
def outer():
  print "hello from outer"
  def inner():
    print "hai from inner"
  inner()

outer()
```

```
ex2:
# i need to call the inner-function outside the OUTER-Fn

def outer():
  print "hello from outer"
  def inner():
    print "hai from inner"
  return inner

res = outer()
print res

if callable(res):
  res()
else:
  print "its not valid fn object"

ex3:
=====
1) what is the scope of FN-Arguments - local
2) when will fn ends return/last statement

# closures
# inspite of the outer fn exited,
# it still preserves the value of name
# until the scope of ur inner function

def outer(name):
  print "hello from outer",name
  def inner():
    print "hai from inner",name
  return inner
```

```
res = outer("Harish")
print res
res()
```

===
===================

Intro Classes:-

```
================
>> Run time Classes - metaprogramming
>> Monkey Patching
>> Ctor - def __init__
>> Dtor - def __del__    # optional
>> this pointer is "self"
>> python 2
   OLD Style classes - indpt class
   NEW Style classes - every class shld be a derived class of
"object"
>> Python 3
   NEW STYLE CLASSES
```

```
class Emps(object):

   def __init__(self,name,dept,salary):
      self.name = name
      self.dept = dept
      self.salary = salary
   def incr(self,value):
      self.salary +=value
   # tostring() eq of python
   def __str__(self):
   return "%s,%s,%s" %(self.name,self.dept,self.salary)
```

```
emp1 = Emps("arun","sales",15000)
emp1.incr(1000)
print emp1
```

solution:-

```
class Stack(object):

  def __init__(self,size):
     self.lst = []
     self.size = size
     self.top = -1

  def push(self,num):
     if self.top < self.size:
        self.top+=1
        self.lst.append(num)
     else:
        #raise Exception("Stack Over Flow")
        print "Errr"
  def pop(self):
     if self.top==-1:
        #raise Exception("Stack Under Flow")
        print "Err"
     else:
        self.top-=1
        print self.lst.pop()
  def peek(self):
     if self.top==-1:
        raise Exception("Stack Under Flow")
     else:
            print self.lst[top]

    def __str__(self):
        return ",".join(map(str,self.lst)))
```

```
class Stack(object):

    def __init__(self,size):
        self.lst = []
        self.size = size
        self.top = -1

    def push(self,num):
        if self.top < self.size-1:
            self.top+=1
            self.lst.append(num)
        else:
            #raise Exception("Stack Over Flow")
            print "Errr"
    def pop(self):
        if self.top==-1:
            #raise Exception("Stack Under Flow")
            print "Err"
        else:
            self.top-=1
            print self.lst.pop()
    def peek(self):
     if self.top==-1:
        raise Exception("Stack Under Flow")
     else:
        print self.lst[self.top]

  def __str__(self):
     return ",".join(map(str,self.lst))




stk1 = Stack(5)

print stk1

stk1.push(10)
stk1.push(20)
stk1.push(30)
stk1.push(40)
stk1.push(50)
stk1.push(60) # Stack OVer Flow

print stk1.peek()  #
print stk1

stk1.pop()
stk1.pop()
stk1.pop()
stk1.pop()
stk1.pop()
stk1.pop() # stack Underflow
```


```
slice()
vars()

hasattr()
getattr()
setattr()
isinstance(),
issubclass(),
super()

ex:
class Sample(object):

  def __init__(self):
    self.a=10
    self.b=20
    self.c=30

  def fun1(self):
    print "hello"

  def fun2(self):
    print "World"

  def fun3(self):
    print "of"

  def fun4(self):
    print "unix"

class Alpha(Sample):

 def fun1(self):
   super(Alpha,self).fun1()   # python 2.x
   #super().fun1()            # python 3.x
   print "Alpha"

#s1 = Sample()
#print vars(s1)
#print s1.__dict__
#print dir(s1)

#print Sample.__mro__
#print isinstance(s1,object)
#print isinstance(s1,Alpha)

aob1 = Alpha()
aob1.fun1()




ex1:
====
class Sample(object):
  def fun1(self):
    print "hello"

  def fun2(self):
    print "World"

  def fun3(self):
    print "of"

  def fun4(self):
    print "unix"



s1 = Sample()
```

```
fnlst = ["fun1","fun2","fun3","fun4"]
for elem in fnlst:
   if hasattr(s1,elem):
      getattr(s1,elem)()
-----------------------------------------------------------------
-------------
```

```
Functional programming:-
-----------------------
>> lambda expressions
>> map function       - expression returns a value
>> filter function    - expression returns a BOOLEAN
>> reduce             - recursion in fp

>> list compre
>> dict compre
>> partial Fns
```

```
python 2 - map returns a list     - map is a function
python 3 - map returns a iterator - map is a class
```

```
alst = map(lambda expression/fun-name, iterable)

prodlst = ["dvd-10","hdd-20","cpu-30","mon-40"]
Total Qty of prods

sol:

def myownfun(x):
   name,qty = x.split("-")
   return int(qty)
```

```
prodlst = ["dvd-10","hdd-20","cpu-30","mon-40"]
qtylst = map(myownfun , prodlst)
print qtylst
print sum(qtylst)

#OR

prodlst = ["dvd-10","hdd-20","cpu-30","mon-40"]
qtylst = map(lambda x: int(x.split("-")[1]) , prodlst)
print qtylst
print sum(qtylst)



emplst = [["arun","sales",15000],["ravi","accts",18000]]
total salary

sallst = map(lambda x : x[-1] , emplst)


datlst=["15-oct","21-dec","11-jan","10-feb","1-oct","25-oct"]
display only the dates which fall in current month

import time

currmonth = time.strftime("%b").lower()

datlst=["15-oct","21-dec","11-jan","10-feb","1-oct","25-oct"]

reslst = filter(lambda x: x.split("-")[1]==currmonth, datlst)

print relst
```

```
ex1:-
=====
numlst = [10,20,30,None,None,40,50,None,60,None]

numlst[:] = filter(None, numlst)

print numlst


ex2:
====
alst = [10,20,30,40,50]
blst = [1,2,3,4,5]


clst= map(lambda x,y : x+y , alst,blst)


ex3:-
=====
from functools import reduce

alst = [1,2,3,4,5]

res = reduce(lambda x,y: x+y, alst)

print res



ex4:
```

```
====
numlst = [1,2,54,3,76,34,76,32,31]

oddlst = []
for elem in numlst:
   if elem%2!=0:
     oddlst.append(elem)

OR

oddlst = [ elem for elem in numlst if elem%2!=0 ]
print oddlst

ex5:-
=====
alst = ["arun","hari","manu","yash"]
blst = [10,20,30,40]

prepare a dict in a such a manner that "arun" 10 value and so on

emps={elem1:elem2 for elem1,elem2 in zip(alst,blst)}
OR
emps=dict(zip(alst,blst))


ex6:-
=====
from functools import partial

def power(raisedto, num):
  res = num ** raisedto
  return res

square = partial(power,2)
```

```
cube   = partial(power,3)

print square(4)
print cube(5)


------------------------------------------------------------------
-------------------
```

python collections:-

```
--------------------

import collections

help(collections)


ex1:-
=====
import collections

citylst = ["blr","chn","hyd","tvm","blr","chn","blr"]

freqcnt = collections.Counter(citylst)

print freqcnt

freqcnt.update(["blr"])

print freqcnt

print freqcnt.most_common(2)
```

```
ex2:-
=====
import collections

emps = collections.OrderedDict()

emps['arun'] = 10
emps['basu'] = 20
emps['chet'] = 30
emps['dine'] = 40

print emps




OrderedDict:-
=============

import collections

a = collections.OrderedDict()

a["first"] = 10
a["second"] = 20
a["third"]  = 30
a["fourth"] = 40

print(a)

print(a.keys())
print(a.values())

a.move_to_end("first")
print(a)
a.move_to_end("first",last=False)
```

```
print(a)
```

DefaultDict:-

```
=============
import collections

def funfactory():
  return 0

emps = collections.defaultdict(funfactory,a=15,b=20)

print(emps)

print(emps["a"])
print(emps["b"])
print(emps["c"])
```

```
deque:-
========
import collections

a = collections.deque([20,50,10,30,40])

a.extendleft([10,20,30])
a.extend([20])

print(a)

a.pop()

print(a)
```

```
a.popleft()

a.rotate(-2)

a.rotate(2)

print(a)


array:-
========
import array

a = array.array("i")


heapq:-
========

import heapq

a = [10,40,20,50,30,15,45]

b=[]

for elem in a:
 heapq.heappush(b,elem)

for i in range(len(a)):
  print(heapq.heappop(b))


bisect:-
========
>> insort()



Files:-
========
>> data perissistance
>> text mode
>> binary mode
>> r/w/a/r+/w+/a+   - rb/wb/ab/rb+/wb+/ab+
>> BOF - 0
   CUR - 1
   EOF - 2
>> random-access
   fob.seek(no_of_bytes,REFPOINT)
   fob.tell()


How to open a file:-
--------------------
f1 = open("new.txt","w")

How to close a file:-
---------------------
f1.close()

How to write into the file:-
----------------------------

f1 = open("data.txt","w")

f1.write("hello\n")
f1.write("world\n")
f1.write("of\n")
```

```
f1.write("unix")

f1.close()


How to read from the file:-
----------------------------
with open("data.txt") as f1:
   print f1
   for elem in f1:
      print elem

print f1




Ex1:-
=====

fob = open("emps.txt","w+")

emplst=[
         "arun-sales-blr-18000\n",
         "ravi-accts-chn-17400\n",
         "john-purch-blr-12333\n",
         "hari-sales-chn-23233"
       ]

fob.writelines(emplst)



in each loc how many emps are there ?
total salary of each dept ?
sales - 412323
```

```
purch - 17400
accts - 12333


fob.close()


solution:-
----------
import collections

fob = open("emps.txt","w+")
emplst=[
         "arun-sales-blr-18000\n",
         "ravi-accts-chn-17400\n",
         "john-purch-blr-12333\n",
         "hari-sales-chn-23233\n",
         "guru-sales-blr-12345"
       ]

fob.writelines(emplst)
fob.seek(0)    # reset to BOF
freqcnt = collections.Counter(map(lambda x : x.split("-")[2],
fob))
print freqcnt


fob.seek(0,0)
subtotal={}
for elem in fob:
    name,dept,loc,salary = elem.split("-")
    if dept in subtotal.keys():
       subtotal[dept] = subtotal[dept] + int(salary)
    else:
       subtotal[dept] = int(salary)
```

```
fob.close()
print subtotal


ex2:-
=====

dept.txt:-
----------
501-sales
502-purch
503-hrd
504-accts
505-finan
506-mktg

emp.txt:-
---------
arun-503-blr-18000
ravi-501-chn-12345
elan-506-hyd-31321
john-505-blr-31231

out.csv:-
----------
name,did,dname,loc,salary
arun,503,hrd,blr,18000
ravi,501,sales,chn,12345


solution:-
----------
dfile = open("dept.txt")
```

```
print map(lambda x : (x.split("-")), dfile)

dfile.close()


file check:-
=============
import os

if os.path.isfile("one.txt"):
  print "File Exists"
else:
  print "File Not Found"


Other Fns:-
===========
strbuffer = fob.read()        # complete file
strbuffer = fob.read(1024)    # read a block of 1024 bytes
strbuffer = fob.readline()    # read only one line upto \n or EOF
lstbuffer = fob.readlines()   # complete file & Store it in a
LIST


-----------------------------------------------------------------
------------


itertools:-
===========
import itertools

c = itertools.count(10)

print(next(c))
print(next(c))
print(next(c))
print(next(c))

-----------------------------------------------------------------
------------
import itertools

a=[1,2,3]
b=[4,5,6]

it1 = iter(a)
it2 = iter(b)

c1 = itertools.chain(a,b)
#c2 = itertools.chain_from_iterables(it1,it2)

-----------------------------------------------------------------
------------
import itertools as it


a=[10,20,30]
b=["a","b","c","d","e","f"]

reslst = list(it.zip_longest(b,a,fillvalue=0))

print(reslst)

-----------------------------------------------------------------
------------

import itertools as it

alst=[10,20,30,40,50,60,70,80,90,100]
```

```
res = it.islice(alst,0,4)

for elem in res:
  print(elem)

------------------------------------------------------------------
------------
import itertools as it

alst=[1,2,3,4,5,6,7,8,9,10]
blst=["a","b","c"]

res = list(zip(alst,it.cycle(blst)))

print(res)

------------------------------------------------------------------
------------



alst = [10,20,30,40,50,60,70,80]

blst = [15,20,35,40,55,60,75,80]


vlst1 = map(lambda x : x + 5 , alst[0::2])
vlst2 = alst[1::2]

vlst3 = zip(vlst1,vlst2)

print vlst3


------------------------------------------------------------------
------------
```

```
ex:
import itertools as it

#alst = [10,20,30,40,50,60,70,80]

#vlst1 = map(lambda x : x + 5 , alst[0::2])
#vlst2 = alst[1::2]
#vlst3 = zip(vlst1,vlst2)
#print vlst3
#?

#print alst
#blst = map(lambda x: x[0]+x[1] ,zip(alst,it.cycle([0,5])))
#print blst

#####################################
import functools

@functools.lru_cache(10)
def factorial(num):
  if num==1:
    return 1
  else:
    return num*factorial(num-1)



for num in range(1,1001):
   factorial(num)
```

==
====================

Modules:-

==========

>> libs

>> collection of fns/classes/variables

```
>> file extension shld be .PY
>> every module will have its own namespace - same name as the
filename - __name__
>> include a module

   import modulename          - Fully Qualified Name
   from modulename import *   - Relative NAmes
   from modulename import fun/class/var - Relative NAmes

>> auto create PYTHON BYTE CODE .PYC/.PYO/.PYD
>> module search PATH
   import sys

   print sys.path
   OR

   PYTHONPATH

Note:
Any program we run such program namespace is set to "__main__"

mylib.py
=========

defaultstate=20

class Mylib(object):
   pass

def fun1(a,b):
   print "NS = ",__name__
   print a,b

def fun2():
   print "hello"
```

```
1) mylib.PYC   - Closed Source - portable

2) mylib.PY    - Open source   - portable

3) mylob.EXE   - closed Source - non-portable - windows
```

```
setup.py:-
==========
from distutils.core import setup

setup(name="mylib",
      version="1.0",
      py_modules=["mylib"]
     )
```

```
C:\> python setup.py  sdist
```

```
How to install a module:-
-------------------------
>> u shld have admin priv
>> clang-develop-toolkit   aka.ms/vcpython27
```

```
1)C:\python27\scripts>  pip install  c:\that\this\mylib-1.0.zip

2)C:\python27\scripts>  easy_install  c:\that\this\mylib-1.0.zip
```

```
3)got to c:\that\this>
  Extract the zip File
  cd mylib1.0
  python setup.py install


pip list

www.pypi.python.org
virtualenv




c:\python27\scripts> pip install numpy

c:\users\VIJAY\appdata\local\programs\python\python35\scripts>




docstrings:-
=============
def fun():
  '''this is the help of the function
  fun which will be displayed
  when we call help(fun)
  '''
  pass


help(fun)
print fun.__doc__
```

```
================================================================
====================
Packages:-
----------
>> collection of modules/subpackages
>> folder
>> every package shld have a compulsory file named __init__.py




================================================================
====================
unittest:-
==========
>> White Box Testing of the programs
>> programmer
>> unittest/pyunit
   pytest
   nose

listoper.py:-
=============
def addelem(value):
 pass
def delelem(index):
 pass
def increlem(value):
 pass
def modifyelem(newvalue):
 pass


test_listoper.py:-
==================
import unittest
import listoper as lst
class Test_list(unittest.TestCase):

   def setUp(self):
     pass

   def tearDown(self):
     pass

   def test_addelem1(self):
     lst.addelem(50)
     self.assertEquals(lst.numlst[-1],50)

   def test_delelem(self):
     lst.delelem(0)
     self.assertNotEquals(before,after)


if __name__ == "__main__":
   unittest.main()
```

doctest:-

=========

```
import re

def add(num1,num2):
  '''
  >>> add(10,20)
  30
  >>> add(10,"hello")
  '10hello'
  '''
  if re.search(r"^\d+$",str(num1)) and
re.search(r"^\d+$",str(num2)):
    return num1+num2
  else:
    return str(num1) + str(num2)


c:\> python -m  doctest  first.py
```

===
=================

pydoc

===
=================

pdb & inspect:-

===============

GUI Debug:-

CLI Debug:-

```
C:\> python -m pdb  first.py

l         - list the program with line number
l 20,25  -
b 20      -
b fun     -
```

===
=================

==
=====================

inspect module:-

================

```
 (name,suffix,mode,mtype) = 
inspect.getmoduleinfo("c:\\that\\lib\\sample.py")
 inspect.getdoc(module)
 inspect.getcomments(module)
 inspect.getsource(module.fun)
 inspect.getclasstree(module)
```

traceback :-

============

```
import traceback

try:
    block
except ValueError as e:
    print e
    traceback.print_exc()
```

timeit module:-

===============

```
>>

python -m timeit  "code"
```

sys module:-

============

```
>> sys is a interface b/w program & python Interpreter


import sys

sys.getrefcount(a)
sys.getsizeof(a)
sys.path - module search path
sys.argv - command line args
sys.maxint/sys.subversion  - only in python 2.x
sys.platform -
sys.stdin
sys.stdout
sys.stderr


ex:
import sys

old = sys.stdout

sys.stdout = open("out.txt","w")

print "My script name = ",__file__

print "My Script name = ",sys.argv[0]

sys.stdout = old

print "hai"


os module:-
===========
>> is an interface b/w python program & underlying KERNEL


import os

os.name
os.getcwd()
os.getpid()
os.listdir(".")
os.environ["PATH"]
os.system("command")
os.remove("one.txt")
os.mkdir("dir")

for path,files,dirs in os.walk(".",topdown=False):
  print(path)
  print(files)
  print(dirs)

os.stat("one.txt")
os.path.isfile("one.txt")
os.path.basename(path) - filename
os.path.dirname(path)  - dirpath


================================================================
================
import time

time.sleep(1)
time.strftime("%d %m %y")
time.localtime()
time.time()
time.clock()
```

```
start = time.time()
task()
end   = time.time()

print end-start    # no of seconds

================================================================
================
datetime modules:-
==================
date-class
time-class
datetime-class


from datetime import date,timedelta

print(date.today())

start = date(2017,1,1)

end   = date(day=15,month=8, year=2017)

res = end-start

print(res)

newdate= date.today() + timedelta(16)

print(newdate)


newdate.replace(month, newdate.month+1)
```

```
=================================================================
====
import random


for i in range(1,11):
  print(random.randrange(1,101))

a=[1,2,3,4]
random.shuffle(a)
print a

=================================================================
====

shutil:-
========
shutil.move("one.txt","c:\")
shutil.copy("one.txt","c:\")

shutil.movedir("dir","c:\")
shutil.copydir("dir","c:\")

shutil.rmtree("c:\\that")



Task:-
======

1) create a directory named "temp"
   if the folder already exists - delete it
   if there is a file named temp - delete it
```

2) copy all the .txt from the curr dir to folder "Temp"

3) zip the files in the temp folder

os-module
shutil-module
zipfile/tarfile-module
glob-module

```
import glob
import shutil
import zipfile
import os

if os.path.isfile("temp"):
  os.remove("temp")
elif os.path.isdir("temp"):
  shutil.rmtree("temp")

os.mkdir("temp")
for elem in glob.glob("*.txt")
  shutil.copy(elem,"temp")

zfile = zipfile.ZipFile("one.zip","w")
for elem in glob.glob("temp/*.txt"):
  zfile.write(elem)

zfile.close()

=================================================================
==========
import pprint

pprint.pprint(dict)



=================================================================
==========
hashlib:-
==========
>> hash code of FIPS standard
>> sha1/sha256/md5 & so on

import hashlib

h = hashlib.md5()

h.update("hello world of python".encode("ascii"))  # Python 3

h.update("hello world of python")                  # Python 2

print(h.hexdigest())

=================================================================
==========
>>optparse
>>argparse
>>shopts
```

```
argparse

python first.py --value1=10 --value2=10 --oper add/sub/mul/div

ex:

import argparse

parser = argparse.ArgumentParser(description="hello world")
parser.add_argument("--value1",action="store",type=int,dest="a")
parser.add_argument("--value2",action="store",type=int,dest="b")
parser.add_argument("--oper",action="store",type=str,dest="c")

res = parser.parse_args()

if res.c=="add":
  print(res.a+res.b)


logger:-
=========
>> 5 verbose level

DEBUG - 10
INFO  - 20
WARNING - 30       <<<<---- default level
ERROR   - 40
CRITICAL - 50


ex1:
import logging

logging.debug("a")
logging.info("b")
logging.warning("c")
logging.error("d")
logging.critical("e")
```

ex:

```
import logging


logging.basicConfig(level = logging.DEBUG, filename="new.log", \
                            format="%(asctime)s %(levelname)s %(message)s")

logging.debug("a")
logging.info("b")
logging.warning("c")
logging.error("d")
logging.critical("e")
```

ex:

```
import logging

log = logging.getLogger("anyname")
log.setLevel(logging.DEBUG)

fh = logging.FileHandler('anyname.log')
fh.setLevel(logging.DEBUG)
log.addHandler(fh)
log.debug("hello1")
log.info("hello2")
log.warning("hello3")
log.error("hello4")
log.critical("hello5")



=================================================================
=====================
Object Perssistance:-
=====================
>> ORM - SqlAlchemy , Django-ORM, PyPony
>> pickle
>> json
>> shelve


ex1:-
-----
import pickle

data = {
        "today" : [10,20],
        "yday"  : [30,40]
       }
fob = open("data.pickle","w")   # python2     # "wb"
pickle.dump(data,fob)
fob.close()


ex2:
-----
import pickle
```

```
fob = open("data.pickle","r")
res = pickle.load(fob)
print res
fob.close()
```

```
Concurreny in Python:-
======================

import multiprocessing - Process
import threading       - Thread
import subprocess
import concurrent


1) pthreads of "C"
2) java threads

from  multiprocessing import Process

def job1():
 pass

def job2():
 pass


if __name__ == "__main__":   # This check is compulsory
```

```
  p1 = Process(target=job1,args=())
  p2 = Process(target=job2,args=())

  p1.start()
  p1.start()

  p1.join()
  p2.join()
```

 - Thread/Process
Intro sockets & network automation
Intro Subprocess & CLI Automation

==
=====================

```
from multiprocessing import Process
import os
import time
```

```
def job1():
  time.sleep(5)
  print("Job1 = ",os.getpid())

def job2():
  time.sleep(8)
  print("Job2 = ",os.getpid())


if __name__ == '__main__':
  print("MAin = ",os.getpid())
  p1 = Process(target=job1,args=())
  p2 = Process(target=job2,args=())
  start = time.time()
  p1.start()
  p2.start()
  p1.join()
  p2.join()
  end  = time.time()
  print("Time taken = ",end-start)
```

ex1:

```
from multiprocessing import Process
from threading import Thread,currentThread
import os
import time

def job1():
  time.sleep(5)
  print("Job1 = ",os.getpid(),currentThread().getName())

def job2():
  time.sleep(8)
  print("Job2 = ",os.getpid(),currentThread().getName())


if __name__ == '__main__':
  print("MAin = ",os.getpid())
  p1 = Thread(target=job1,args=())
  p2 = Thread(target=job2,args=())
  start = time.time()
  p1.start()
  p2.start()
  p1.join()
  p2.join()
  end  = time.time()
  print("Time taken = ",end-start)
```

==
=====================

ex1:

```
from threading import Thread

class myClass(Thread):

   def __init__(self,name):
      self.name = name
      self.start()

   def run(self):
      act

c1 = myClass()
c2 = myClass()
c1.join()
c2.join()
```

```
=================================================================
=====================
-----------------------------------------------------------------
-----

subprocess CLI

import subprocess

p = subprocess.check_output("ipconfig",shell=True)

print(p)


=================================================================
=====================

>> import socket
>> import socketserver
>> import mutltiprocessing.Listener

>> import asyncore
>> import twisted

sockets:-
==========

import socket

ip = socket.gethostname()
port =12345

s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.bind((ip,port))
s.listen(5)
```

```
client,add = s.accept()
client.send(b"connect to server")
client.close()
s.close()


ex:
import socket

ip = socket.gethostname()
port =12345

s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((ip,port))
ans = s.recv(1024)
print(ans)
s.close()



Net Automation libs:-
======================
telnet  - import telnetlib
ftp     - import ftplib
ssh     - import paramiko / import fabric / import Exscript
mail    - import smtplib


ex1:
====
import telnetlib

hostname="localhost".encode("ascii")
```

```
user    ="root".encode("ascii")
pwd     ="root@123".encode("ascii")
cmds    ="uptime".encode("ascii")

tn = telnetlib.Telnet(hostname)

tn.read_until(b"login: ")
tn.write(user+b"\n")
tn.read_until(b"Password: ")
tn.write(pwd+b"\n")

tn.write(cmds+b"\n")
print(tn.read_all())
tn.close()

----------------------------------------------------------------
-----
import ftplib

host="ftp.cisco.com"

ftp = ftplib.FTP(host)
ftp.login()
ftp.dir()
ftp.cwd("/pub/mlibs")
ftp.dir()
ftp.close()

----------------------------------------------------------------
-----
import smtplib
import poplib

----------------------------------------------------------------
-----
```

```
import urllib2             # python 2.x

url="https://cisco.com"

resp = urllib2.urlopen(url)

print(resp.read())

-----------------------------------------------------------------
-----

import urllib.request     # python 3.x

url="https://cisco.com"

resp = urllib.request.urlopen(url)

print(resp.read())


-----------------------------------------------------------------
-----
```

Selenium WebDriver programming using Python