```
#!/usr/bin/env python
# Filename : helloworld.py
print 'Hello World'
```

```
# Filename : var.py
i=5
print i
i=i+1
print i

s='''This is a multi-line string.
This is the second line.'''
print s
```

```
#!/usr/bin/env python
# Filename: expression.py

length=5
breadth=2
area=length*breadth
print 'Area is',area
print 'Perimeter is',2*(length+breadth)
```

```
#!/usr/bin/env python
# Filename: if.py

number=23
guess=int(raw_input('Enter an integer : '))

if guess==number:
	print 'Congratulations, you guessed it.' # New block starts here
	print "(but you do not win any prizes!)" # New block ends here
elif guess<number:
	print 'No, it is a little higher than that' # Another block
	# You can do whatever you want in a block ...
else:
	print 'No, it is a little lower than that'
	# you must have guess > number to reach here

print 'Done'
# This last statement is always executed, after the if statement is executed
```

```
#!/usr/bin/env python
# Filename: while.py

number=23
running=True

while running:
	guess=int(raw_input('Enter an integer : '))

	if guess==number:
		print 'Congratulations, you guessed it.'
		running=False # this causes the while loop to stop
	elif guess<number:
		print 'No, it is a little higher than that'
	else:
		print 'No, it is a little lower than that'
else:
	print 'The while loop is over.'
	# Do anything else you want to do here

print 'Done'
```

```
#!/usr/bin/env python
# Filename: break.py

while True:
        s=raw_input('Enter something : ')
        if s=='quit':
                break
        print 'Length of the string is',len(s)
print 'Done'
```

```
#!/usr/bin/env python
# Filename: continue.py

while True:
        s=raw_input('Enter something : ')
        if s=='quit':
                break
        if len(s)<3:
                continue
        print 'Input is of sufficient length'
        # Do other kinds of processing here...
```

```
#!/usr/bin/env python
# Filename: function1.py

def sayHello():
	print 'Hello World!' # block belonging to the function

sayHello() # call the function
```

```
#!/usr/bin/env python
# Filename: func_param.py

def printMax(a,b):
        if a>b:
                print a,'is maximum'
        else:
                print b,'is maximum'

printMax(3,4) # directly give literal values

x=5
y=7

printMax(x,y) # give variables as arguments
```

```
#!/usr/bin/env python
# Filename: func_local.py

def func(x):
        print 'x is',x
        x=2
        print 'Changed local x to',x

x=50
func(x)
print 'x is still',x
```

```
#!/usr/bin/env python
# Filename: func_global.py

def func():
        global x

        print 'x is',x
        x=2
        print 'Changed local x to',x

x=50
func()
print 'Value of x is',x
```

```
#!/usr/bin/env python
# Filename: func_default.py

def say(message,times=1):
	print message*times

say('Hello')
say('World',5)
```

```
#!/usr/bin/env python
# Filename: func_key.py

def func(a,b=5,c=10):
	print 'a is',a,'and b is',b,'and c is',c

func(3,7)
func(25,c=24)
func(c=50,a=100)
```

```
#!/usr/bin/env python
# Filename: func_return.py

def maximum(x,y):
        if x>y:
                return x
        else:
                return y

print maximum(2,3)
```

```
#!/usr/bin/env python
# Filename: func_doc.py

def printMax(x,y):
	'''Prints the maximum of two numbers.

	The two values must be integers.'''
	x=int(x) # convert to integers, if possible
	y=int(y)

	if x>y:
		print x,'is maximum'
	else:
		print y,'is maximum'

printMax(3,5)
print printMax.__doc__
```

```
#!/usr/bin/env python
# Filename: using_sys.py

import sys

print 'The command line arguments are:'
for i in sys.argv:
        print i

print '\n\nThe PYTHONPATH is',sys.path,'\n'
```

```
#!/usr/bin/env python
# Filename: using_name.py

if __name__=='__main__':
	print 'This program is being run by itself'
else:
	print 'I am being imported from another module'
```

```
#!/usr/bin/env python
# Filename: mymodule.py

def sayhi():
	print 'Hi, this is mymodule speaking.'

version='0.1'

# End of mymodule.py
```

```
#!/usr/bin/env python
# Filename: mymodule_demo.py

import mymodule

mymodule.sayhi()
print 'Version', mymodule.version
```

```
#!/usr/bin/env python
# Filename: mymodule_demo2.py

from mymodule import sayhi,version
# Alternative:
# from mymodule import *

sayhi()
print 'Version',version
```

```
#!/usr/bin/env python
# Filename: using_list.py

# This is my shopping list
shoplist=['apple','mango','carrot','banana']

print 'I have',len(shoplist),'items to purchase.'

print 'These items are:', # Notice the comma at end of the line
for item in shoplist:
	print item,

print '\nI also have to buy rice.'
shoplist.append('rice')
print 'My shopping list is now',shoplist

print 'I will sort my list now'
shoplist.sort()
print 'Sorted shopping list is',shoplist

print 'The first item I will buy is',shoplist[0]
olditem=shoplist[0]
del shoplist[0]
print 'I bought the',olditem
print 'My shopping list is now',shoplist
```

```
#!/usr/bin/env python
# Filename: using_tuple.py

zoo=('wolf','elephant','penguin')
print 'Number of animals in the zoo is',len(zoo)

new_zoo=('monkey','dolphin',zoo)
print 'Number of animals in the new zoo is',len(new_zoo)
print 'All animals in new zoo are',new_zoo
print 'Animals brought from old zoo are',new_zoo[2]
print 'Last animal brought from old zoo is',new_zoo[2][2]
```

```
#!/usr/bin/env python
# Filename: print_tuple.py

age=22
name='Swaroop'

print '%s is %d years old' %(name,age)
print 'Why is %s playing with that python?' %name
```

```
#!/usr/bin/env python
# Filename: using_dict.py

# 'ab' is short for 'a'ddress'b'ook

ab={    'Swaroop'       : 'swaroopch@byteofpython.info',
        'Larry'         : 'larry@wall.org',
        'Matsumoto'     : 'matz@ruby-lang.org',
        'Spammer'       : 'spammer@hotmail.com'
   }

print "Swaroop's address is %s" %ab['Swaroop']

# Adding a key/value pair
ab['Guido']='guido@python.org'

# Deleting a key/value pair
del ab['Spammer']

print '\nThere are %d contacts in the address-book\n' %len(ab)
for name,address in ab.items():
        print 'Contact %s at %s' %(name,address)

if 'Guido' in ab: # OR ab.has_key('Guido')
        print "\nGuido's address is %s" %ab['Guido']
```

```
#!/usr/bin/env python
# Filename: seq.py

shoplist=['apple','mango','carrot','banana']

# Indexing or 'Subscription' operation
print 'Item 0 is', shoplist[0]
print 'Item 1 is', shoplist[1]
print 'Item 2 is', shoplist[2]
print 'Item 3 is', shoplist[3]
print 'Item -1 is', shoplist[-1]
print 'Item -2 is', shoplist[-2]

# Slicing on a list
print 'Item 1 to 3 is', shoplist[1:3]
print 'Item 2 to end is', shoplist[2:]
print 'Item 1 to -1 is', shoplist[1:-1]
print 'Item start to end is', shoplist[:]

# Slicing on a string
name='swaroop'
print 'characters 1 to 3 is', name[1:3]
print 'characters 2 to end is', name[2:]
print 'characters 1 to -1 is', name[1:-1]
print 'characters start to end is', name[:]
```

```
#!/usr/bin/env python
# Filename: reference.py

print 'Simple Assignment'
shoplist=['apple','mango','carrot','banana']
mylist=shoplist # mylist is just another name pointing to the same object!

del shoplist[0]

print 'shoplist is',shoplist
print 'mylist is',mylist
# notice that both shoplist and mylist both print the same list without
# the 'apple' confirming that they point to the same object

print 'Copy by making a full slice'
mylist=shoplist[:] # make a copy by doing a full slice
del mylist[0] # remove first item

print 'shoplist is', shoplist
print 'mylist is', mylist
# notice that now the two lists are different
```

```
#!/usr/bin/env python
# Filename: str_methods.py

name='Swaroop' # This is a string object

if name.startswith('Swa'):
        print 'Yes, the string starts with "Swa"'

if 'a' in name:
        print 'Yes, it contains the string "a"'

if name.find('war')!=-1:
        print 'Yes, it contains the string "war"'

delimiter=' * '
mylist=['Brazil','Russia','India','China']
print delimiter.join(mylist)
```

```
#!/usr/bin/env python
# Filename: backup_ver1.py

import os
import time

# 1. The files and directories to be backed up are specified in a list.
source=['/home/swaroop/byte','/home/swaroop/bin']
# If you are using Windows, use source=[r'C:\Documents',r'D:\Work'] or something like that

# 2. The backup must be stored in a main backup directory
target_dir='/mnt/e/backup/' #Remember to change this to what you will be using

# 3. The files are backed up into a zip file
# 4. The name of the zip archive is the current date and time
target=target_dir+time.strftime('%Y%m%d%H%M%S')+'.zip'

# 5. We use the zip command (in Unix/Linux) to put the files in a zip archive
zip_command="zip -qr '%s' %s" %(target,' '.join(source))

# Run the backup
if os.system(zip_command)==0:
	print 'Successful backup to',target
else:
	print 'Backup FAILED'
```

```
#!/usr/bin/env python
# Filename: backup_ver2.py

import os
import time

# 1. The files and directories to be backed up are specified in a list.
source=['/home/swaroop/byte','/home/swaroop/bin']
# If you are using Windows, use source=[r'C:\Documents',r'D:\Work'] or something like that

# 2. The backup must be stored in a main backup directory
target_dir='/mnt/e/backup/' #Remember to change this to what you will be using

# 3. The files are backed up into a zip file
# 4. The current day is the name of the subdirectory in the main directory
today=target_dir+time.strftime('%Y%m%d')
# The current time is the name of the zip archive
now=time.strftime('%H%M%S')

# Create the subdirectory if it isn't already there
if not os.path.exists(today):
	os.mkdir(today) # make directory
	print 'Successfully created directory',today

# The name of the zip file
target=today+os.sep+now+'.zip'

# 5. We use the zip command (in Unix/Linux) to put the files in a zip archive
zip_command="zip -qr '%s' %s" %(target,' '.join(source))

# Run the backup
if os.system(zip_command)==0:
	print 'Successful backup to',target
else:
	print 'Backup FAILED'
```

```
#!/usr/bin/env python
# Filename: backup_ver3.py

import os
import time

# 1. The files and directories to be backed up are specified in a list.
source=['/home/swaroop/byte','/home/swaroop/bin']
# If you are using Windows, use source=[r'C:\Documents',r'D:\Work'] or something like that

# 2. The backup must be stored in a main backup directory
target_dir='/mnt/e/backup/' #Remember to change this to what you will be using

# 3. The files are backed up into a zip file
# 4. The current day is the name of the subdirectory in the main directory
today=target_dir+time.strftime('%Y%m%d')
# The current time is the name of the zip archive
now=time.strftime('%H%M%S')

# Take a comment from the user to create the name of the zip file
comment=raw_input('Enter a comment --> ')
if len(comment)==0: # check if a comment was entered
	target=today+os.sep+now+'.zip'
else:
	target=today+os.sep+now+'_'+
	comment.replace(' ','_')+'.zip'

# Create the subdirectory if it isn't already there
if not os.path.exists(today):
	os.mkdir(today) # make directory
	print 'Successfully created directory',today

# 5. We use the zip command (in Unix/Linux) to put the files in a zip archive
zip_command="zip -qr '%s' %s" %(target,' '.join(source))

# Run the backup
if os.system(zip_command)==0:
	print 'Successful backup to',target
else:
	print 'Backup FAILED'
```

```
#!/usr/bin/env python
# Filename: backup_ver4.py

import os
import time

# 1. The files and directories to be backed up are specified in a list.
source=['/home/swaroop/byte','/home/swaroop/bin']
# If you are using Windows, use source=[r'C:\Documents',r'D:\Work'] or something like that

# 2. The backup must be stored in a main backup directory
target_dir='/mnt/e/backup/' #Remember to change this to what you will be using

# 3. The files are backed up into a zip file
# 4. The current day is the name of the subdirectory in the main directory
today=target_dir+time.strftime('%Y%m%d')
# The current time is the name of the zip archive
now=time.strftime('%H%M%S')

# Take a comment from the user to create the name of the zip file
comment=raw_input('Enter a comment --> ')
if len(comment)==0: # check if a comment was entered
	target=today+os.sep+now+'.zip'
else:
	target=today+os.sep+now+'_'+\
			comment.replace(' ','_')+'.zip'
	# Notice the backslash!

# Create the subdirectory if it isn't already there
if not os.path.exists(today):
	os.mkdir(today) # make directory
	print 'Successfully created directory',today

# 5. We use the zip command (in Unix/Linux) to put the files in a zip archive
zip_command="zip -qr '%s' %s" %(target,' '.join(source))

# Run the backup
if os.system(zip_command)==0:
	print 'Successful backup to',target
else:
	print 'Backup FAILED'
```

```
#!/usr/bin/env python
# Filename: simplestclass.py

class Person:
	pass # An empty block

p=Person()
print p
```

```
#!/usr/bin/env python
# Filename: method.py

class Person:
	def sayHi(self):
		print 'Hello, how are you?'

p=Person()
p.sayHi()

# This short example can also be written as Person().sayHi()
```

```
#!/usr/bin/env python
# Filename: class_init.py

class Person:
        def __init__(self,name):
                self.name=name
        def sayHi(self):
                print 'Hello, my name is',self.name

p=Person('Swaroop')
p.sayHi()

# This short example can also be written as Person('Swaroop').sayHi()
```

```
#!/usr/bin/env python
# Filename: objvar.py

class Person:
        '''Represents a person.'''
        population=0

        def __init__(self,name):
                '''Initializes the person's data.'''
                self.name=name
                print '(Initializing %s)' %self.name

                #When this person is created, he/she adds to the population
                Person.population+=1

        def __del__(self):
                '''I am dying.'''
                print '%s says bye.' %self.name

                Person.population-=1

                if Person.population==0:
                        print 'I am the last one.'
                else:
                        print 'There are still %d people left.' %Person.population

        def sayHi(self):
                '''Greeting by the person.

                Really, that's all it does.'''
                print 'Hi, my name is %s.' %self.name

        def howMany(self):
                '''Prints the current population.'''
                if Person.population==1:
                        print 'I am the only person here.'
                else:
                        print 'We have %d persons here.' %Person.population

swaroop=Person('Swaroop')
swaroop.sayHi()
swaroop.howMany()

kalam=Person('Abdul Kalam')
kalam.sayHi()
kalam.howMany()

swaroop.sayHi()
swaroop.howMany()
```

```
#!/usr/bin/env python
# Filename: inherit.py

class SchoolMember:
        '''Represents any school member.'''
        def __init__(self,name,age):
                self.name=name
                self.age=age
                print '(Initialized SchoolMember: %s)' %self.name

        def tell(self):
                '''Tell my details.'''
                print 'Name:"%s" Age:"%s"' %(self.name,self.age),

class Teacher(SchoolMember):
        '''Represents a teacher.'''
        def __init__(self,name,age,salary):
                SchoolMember.__init__(self,name,age)
                self.salary=salary
                print '(Initialized Teacher: %s)' %self.name

        def tell(self):
                SchoolMember.tell(self)
                print 'Salary: "%d"' %self.salary

class Student(SchoolMember):
        '''Represents a student.'''
        def __init__(self,name,age,marks):
                SchoolMember.__init__(self,name,age)
                self.marks=marks
                print '(Initialized Student: %s)' %self.name

        def tell(self):
                SchoolMember.tell(self)
                print 'Marks: "%d"' %self.marks

t=Teacher('Mrs. Shrividya',40,30000)
s=Student('Swaroop',22,75)

print # prints a blank line

members=[t,s]
for member in members:
        member.tell() # works for both Teachers and Students
```

```
#!/usr/bin/env python
# Filename: using_file.py

poem='''\
Programming is fun
When the work is done
if you wanna make your work also fun:
        use Python!
'''

f=file('poem.txt','w') # open for 'w'riting
f.write(poem) # write text to file
f.close() # close the file

f=file('poem.txt')
# if no mode is specified, 'r'ead mode is assumed by default
while True:
        line=f.readline()
        if len(line)==0: # Zero length indicates EOF
                break
        print line,
        # Notice comma to avoid automatic newline added by Python
f.close() # close the file
```

```
#!/usr/bin/env python
# Filename: pickling.py

import cPickle as p
#import pickle as p

shoplistfile='shoplist.data'
# the name of the file where we will store the object

shoplist=['apple','mango','carrot']

# Write to the file
f=file(shoplistfile,'w')
p.dump(shoplist,f) # dump the object to a file
f.close()

del shoplist # remove the shoplist

# Read back from the storage
f=file(shoplistfile)
storedlist=p.load(f)
print storedlist
```

```
#!/usr/bin/env python
# Filename: try_except.py

import sys

try:
        s=raw_input('Enter something --> ')
except EOFError:
        print '\nWhy did you do an EOF on me?'
        sys.exit() # exit the program
except:
        print '\nSome error/exception occurred.'
        # here, we are not exiting the program

print 'Done'
```

```
#!/usr/bin/env python
# Filename: raising.py

class ShortInputException(Exception):
        '''A user-defined exception class.'''
        def __init__(self,length,atleast):
                Exception.__init__(self)
                self.length=length
                self.atleast=atleast

try:
        s=raw_input('Enter something --> ')
        if len(s)<3:
                raise ShortInputException(len(s),3)
        # Other work can continue as usual here
except EOFError:
        print '\nWhy did you do an EOF on me?'
except ShortInputException,x:
        print 'ShortInputException: The input was of length %d, \
was expecting at least %d' %(x.length,x.atleast)
else:
        print 'No exception was raised.'
```

```
#!/usr/bin/env python
# Filename: finally.py

import time

try:
    f=file('poem.txt')
    while True: # our usual file-reading idiom
        line=f.readline()
        if len(line)==0:
            break
        time.sleep(2)
        print line,
finally:
    f.close()
    print 'Cleaning up...closed the file'
```

```
#!/usr/bin/env python
# Filename: cat.py

import sys

def readfile(filename):
	'''Print a file to the standard output.'''
	f=file(filename)
	while True:
		line=f.readline()
		if len(line)==0:
			break
		print line, # notice comma
	f.close()

# Script starts from here
if len(sys.argv)<2:
	print 'No action specified.'
	sys.exit()

if sys.argv[1].startswith('--'):
	option=sys.argv[1][2:]
	# fetch sys.argv[1] but without the first two characters
	if option=='version':
		print 'Version 1.2'
	elif option=='help':
		print '''\
This program prints files to the standard output.
Any number of files can be specified.
Options include:
  --version : Prints the version number
  --help    : Display this help'''
	else:
		print 'Unknown option.'
	sys.exit()
else:
	for filename in sys.argv[1:]:
		readfile(filename)
```

```python
#!/usr/bin/env python
# Filename: list_comprehension.py

listone=[2,3,4]
listtwo=[2*i for i in listone if i>2]
print listtwo
```

```
#!/usr/bin/env python
# Filename: lambda.py

def make_repeater(n):
        return lambda s: s*n

twice=make_repeater(2)

print twice('word')
print twice(5)
```