## **Absolute vs. Relative Paths**

There are two ways to specify a file path.

* An *absolute path*, which always begins with the root folder (c:\\)
* A *relative path*, which is relative to the program’s current working directory
* There are also the *dot* (.) and *dot-dot* (..) folders.
* These are **not real folders** but special names that can be used in a path.
* A single period (“dot”) for a folder name is shorthand for “this directory.” Two periods (“dot-dot”) means “the parent folder.”

>>> os.getcwd()

'C:\\Python34'

>>> os.chdir('**c:\\python34\\mytestfold'**) **# c:\\ absolute path**

os.chdir('c:\\python34\\mytestfold')

>>> os.chdir('../')

>>> os.chdir('mytestfold') # Relative path

>>>

c:\\python34\\Scripts'

>>> os.rmdir('../mydir') **# Relative path**

>>>

>>> os.chdir('c:\\windows\\system32')

>>> os.getcwd()

'C:\\windows\\system32'

>>> os.chdir('c:\\python34')

>>> os.getcwd()

'c:\\python34'

>>> os.chdir('..//windows//system32')

>>> os.getcwd()

'c:\\windows\\system32'

>>>

>>> os.chdir('c:\\python34')

>>> os.getcwd()

'C:\\python34'

>>> **os.chdir('./../windows/system32')**

>>> os.getcwd()

'c:\\windows\\system32'

>>>

## **Handling Absolute and Relative Paths**

* Calling os.path.abspath(*path*) will return a string of the absolute path of the argument. This is an easy way to convert a relative path into an absolute one.
* Calling os.path.isabs(*path*) will return True if the argument is an absolute path and False if it is a relative path.
* Calling os.path.relpath(*path, start*) will return a string of a relative path from the *start* path to *path*. If *start* is not provided, the current working directory is used as the start path.

>>> os.getcwd()

'C:\\Python34'

>>> os.path.abspath('.\\prem')

'C:\\Python34\\prem'

>>> st = 'c:\\python34\\MyFold'

>>> os.path.isabs(st)

True

>>>

## 

## **Finding File Sizes and Folder Contents**

* There are ways of handling file paths, start gathering information about specific files and folders.
* The os.path module provides functions for finding the size of a file in bytes and the files and folders inside a given folder.

Calling os.path.getsize(*path*) will return the size in bytes of the file in the *path* argument.

Calling os.listdir(*path*) will return a list of filename strings for each file in the *path* argument.

>>> os.path.**getsize('d:\\myfold\\names.txt') # File Size**

40

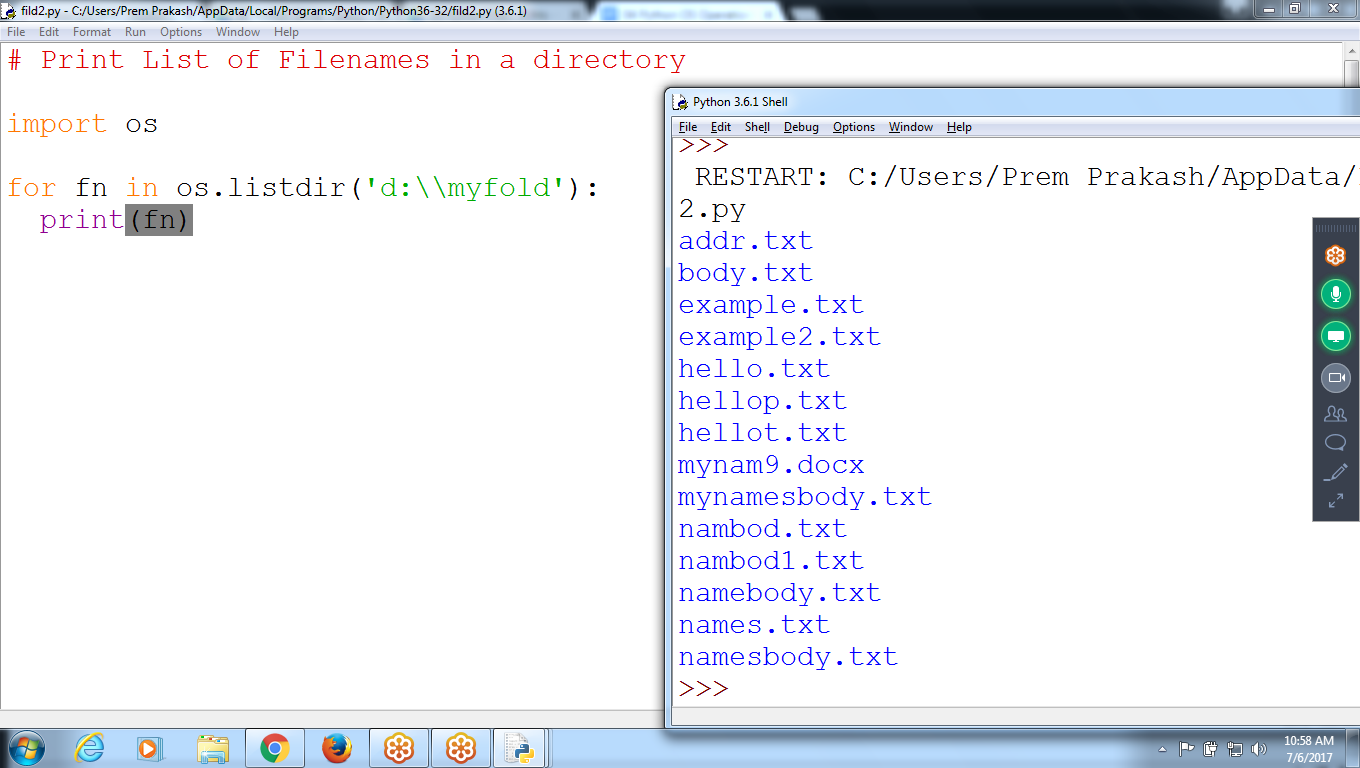
>>>

**>>> os.listdir('d:\\myfold') # List of Filenames**

['addr.txt', 'body.txt', 'example.txt', 'example2.txt', 'hello.txt', 'hellop.txt', 'hellot.txt', 'mynam9.docx', 'mynamesbody.txt', 'nambod.txt', 'nambod1.txt', 'namebody.txt', 'names.txt', 'namesbody.txt']

>>>

**Program :: Display list of filenames in a Directory**

****

**# Display list of Filenames in a folder**

**import os**

**st = 'c:\\python34'**

**for fn in os.listdir(st):**

**print(fn)**

**>>> for f in os.listdir('c:\\python34'):**

**print(f)**

**achi.py**

**body.txt**

**DLLs**

**Doc**

**exam**

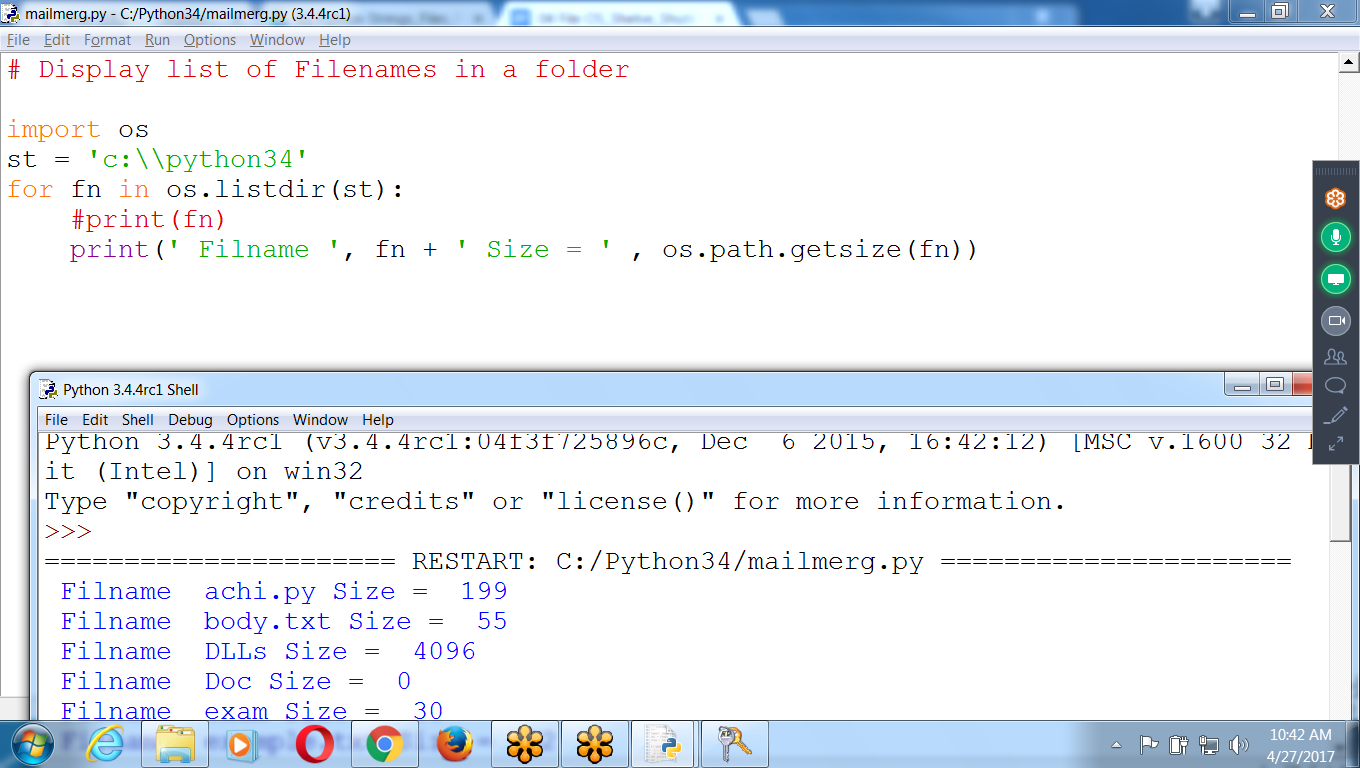
**example.txt**

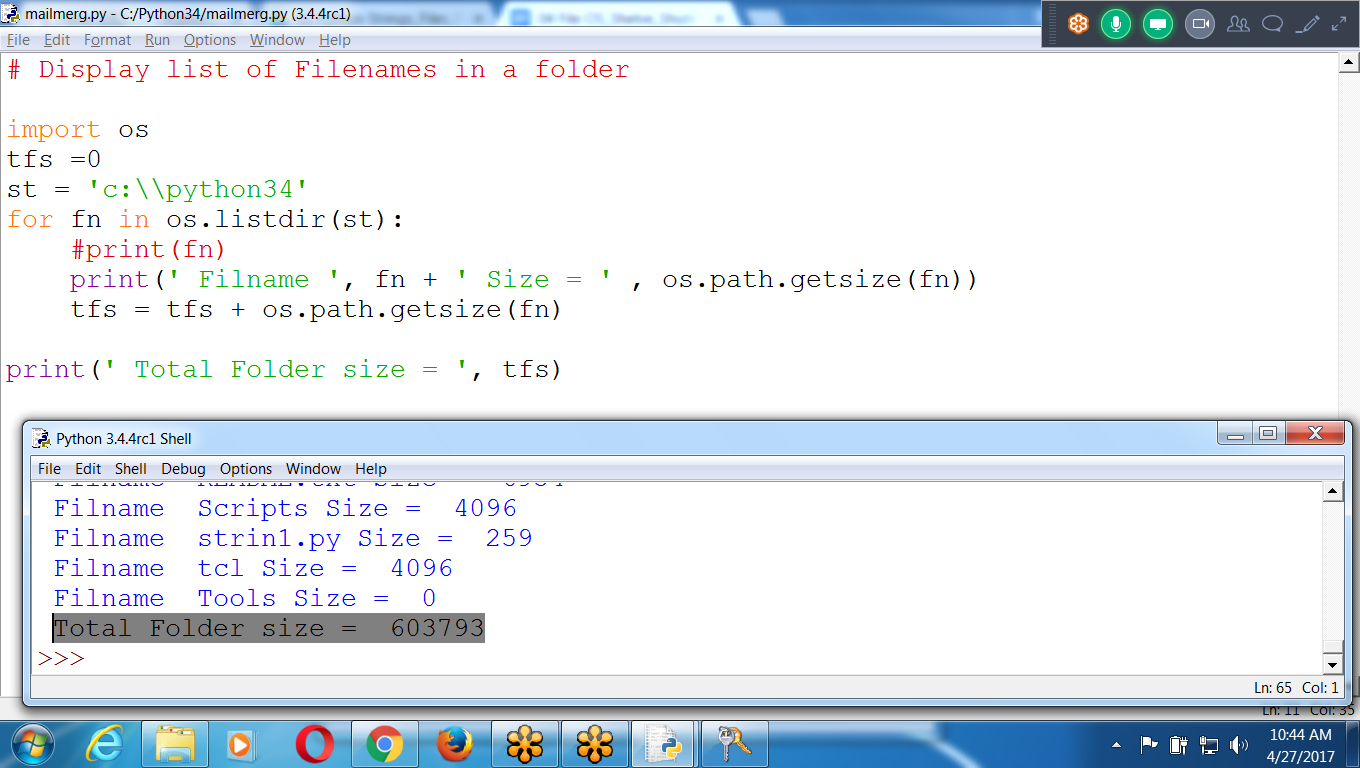
**example2.txt**

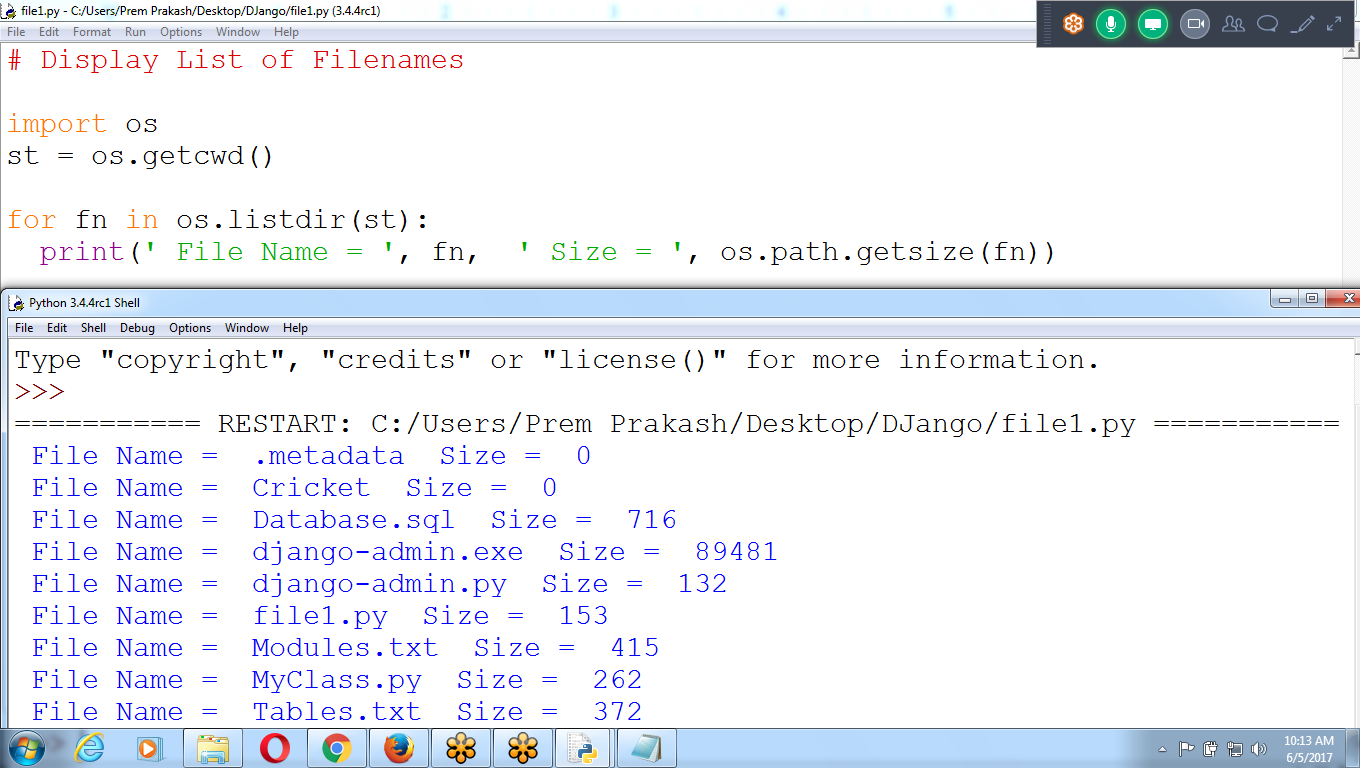
**f10.py**

**f21.py**

**Display Filename and Size**

****





**# Display list of Filenames in a folder**

**import os**

**tfs =0**

**st = 'c:\\python34'**

**for fn in os.listdir(st):**

**#print(fn)**

**print(' Filname ', fn + ' Size = ' , os.path.getsize(fn))**

**tfs = tfs + os.path.getsize(fn)**

**print(' Total Folder size = ', tfs)**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## **Checking Path Validity**

Calling os.path.exists(*path*) will return True if the file or folder referred to in the argument exists and will return False if it does not exist.

Calling os.path.isfile(*path*) will return True if the path argument exists and is a file and will return False otherwise.

Calling os.path.isdir(*path*) will return True if the path argument exists and is a folder and will return False otherwise

>>> **os.path.exists('C:\\Windows')**  
True

>>> **os.path.isdir('C:\\Windows\\System32')**  
True

>>> st = 'C:\\Windows\\System32'

>>> **os.path.isfile(st)**

False

# 

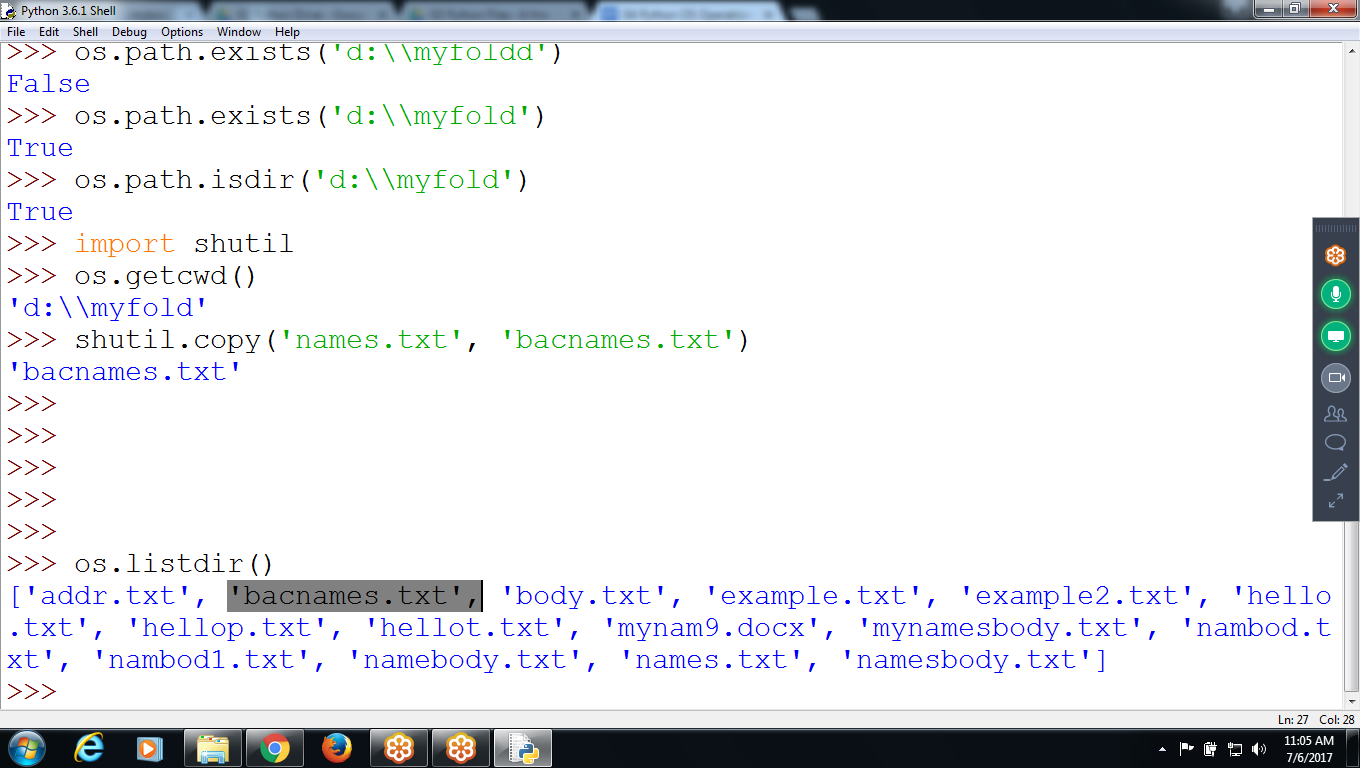
## **Copying Files and Folders**

* The **shutil module** provides functions for copying files, as well as entire folders.
* shutil.copy(*source, destination*) will copy the file at the path *source* to the folder at the path *destination*.
* **shutil.copy()** will copy a single file, **shutil.copytree()** will copy an entire folder and every folder and file contained in it

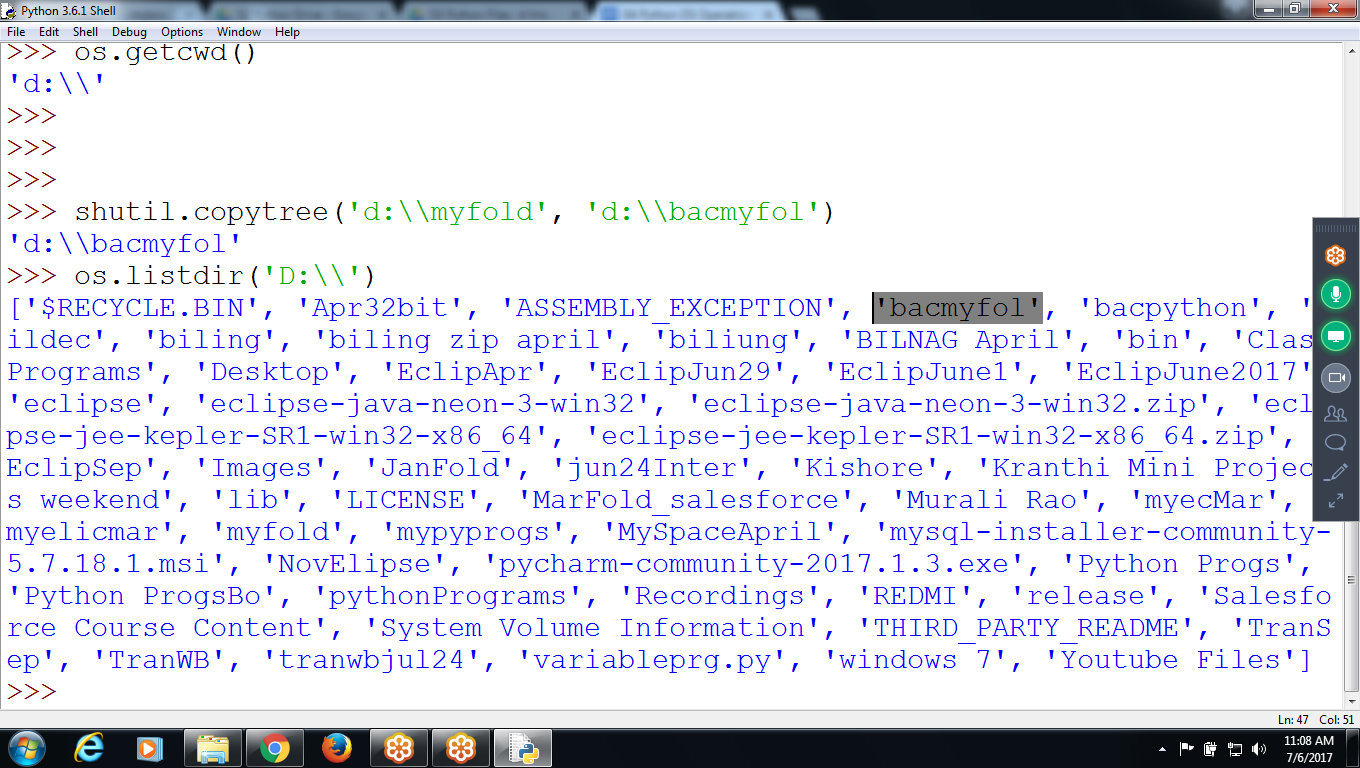
**>>> import shutil,os**

>>> os.getcwd()

'C:\\Python34'



Copying Complete folder files to another folder



>>> shutil.copytree('d:\\myfold', 'd:\\bacmyfol')

'd:\\bacmyfol'

>>> os.listdir('D:\\')

>>> os.listdir('d:\\bacmyfol')

['addr.txt', 'bacnames.txt', 'body.txt', 'example.txt', 'example2.txt', 'hello.txt', 'hellop.txt', 'hellot.txt', 'mynam9.docx', 'mynamesbody.txt', 'nambod.txt', 'nambod1.txt', 'namebody.txt', 'names.txt', 'namesbody.txt']

>>>

>>> import shutil

>>> **shutil.copy('body.txt', 'mybody123.txt')**

'mybody123.txt'

>>>

>>>

>>>

>>>

>>> **shutil.copytree('c:\\python34\\mydir', 'c:\\python34\\mydirbac')**

'c:\\python34\\mydirbac'

>>>

>>>

>>>

>>>

>>> for f in os.listdir():

print(f)

achi.py

bas2.py

body.txt

DLLs

Doc

>>> shutil.copy('example.txt', 'bacexample.txt')

'Bacexample.txt'

>>> os.path.exists('c:\\python34\\bacexample.txt')

True

>>> os.path.exists('bacexample.txt')

True

>>> fr = open('bacexample.txt')

>>> print(fr.read())

ramu

sita

lax

hanu

ravan

>>> shutil.copytree('c:\\python34', 'd:\\bacpython')

'd:\\bacpython'

>>

## 

## 

## **Moving and Renaming Files and Folders**

shutil.move(source, destination) will move the file or folder at the path source to the path destination

>>> shutil.move('example.txt', 'rajan.txt')

'rajan.txt'

>>> os.path.exists('example.txt') # File Doesnot exists

False

>>> os.path.exists('rajan.txt')

True

>>>

Shutil.copy

Shutil.copytree

shutill.move

## **Permanently Deleting Files and Folders**

Can delete a single file or a single empty folder with functions in the os module, whereas **to delete a folder and all of its contents,** you use the shutil module.

* Calling os.unlink(*path*) will **delete the file** at *path*.

>>> os.unlink('rajan.txt') # Removes File

>>> os.path.exists('rajan.txt')

False

>>>

* Calling os.rmdir(*path*) will delete the folder at *path*. **This folder must be empty of any files or folders.**

>>> os.path.isdir('c:\\python34\\mytestfold') # Checking Folder

True

>>> os.rmdir('mytestfold') # Removing Directory

>>> os.path.isdir('c:\\python34\\mytestfold')

False

>>>

* Calling shutil.rmtree(*path*) will **remove the folder** at *path*, **and all files and folders it contains will also be deleted.**

**>>> os.rmtree('d:\\bacmyfol') # Need to Remove Subfolder Files**

Traceback (most recent call last):

File "<pyshell#58>", line 1, in <module>

os.rmtree('d:\\bacmyfol')

AttributeError: module 'os' has no attribute 'rmtree'

>>>

>>>

>>> shutil.rmtree('d:\\bacmyfol') **# Removes subfolder files**

>>>

>>> shutil.rmtree('c:\\python34\\bacprem')

Delete all .py files

>>> for fn in os.listdir ():

if fn.endswith('.txt'):

os.unlink(fn)

# **>>> for f in os.listdir():**

# **if (f.endswith('py')):**

# **os.unlink(f)**

# 

# 

# 

# **>>> for f in os.listdir():**

# **print(f)**

# 

# 

# **bacexample.txt**

# **body.txt**

# **DLLs**

# **Doc**

# **exam99.txt**

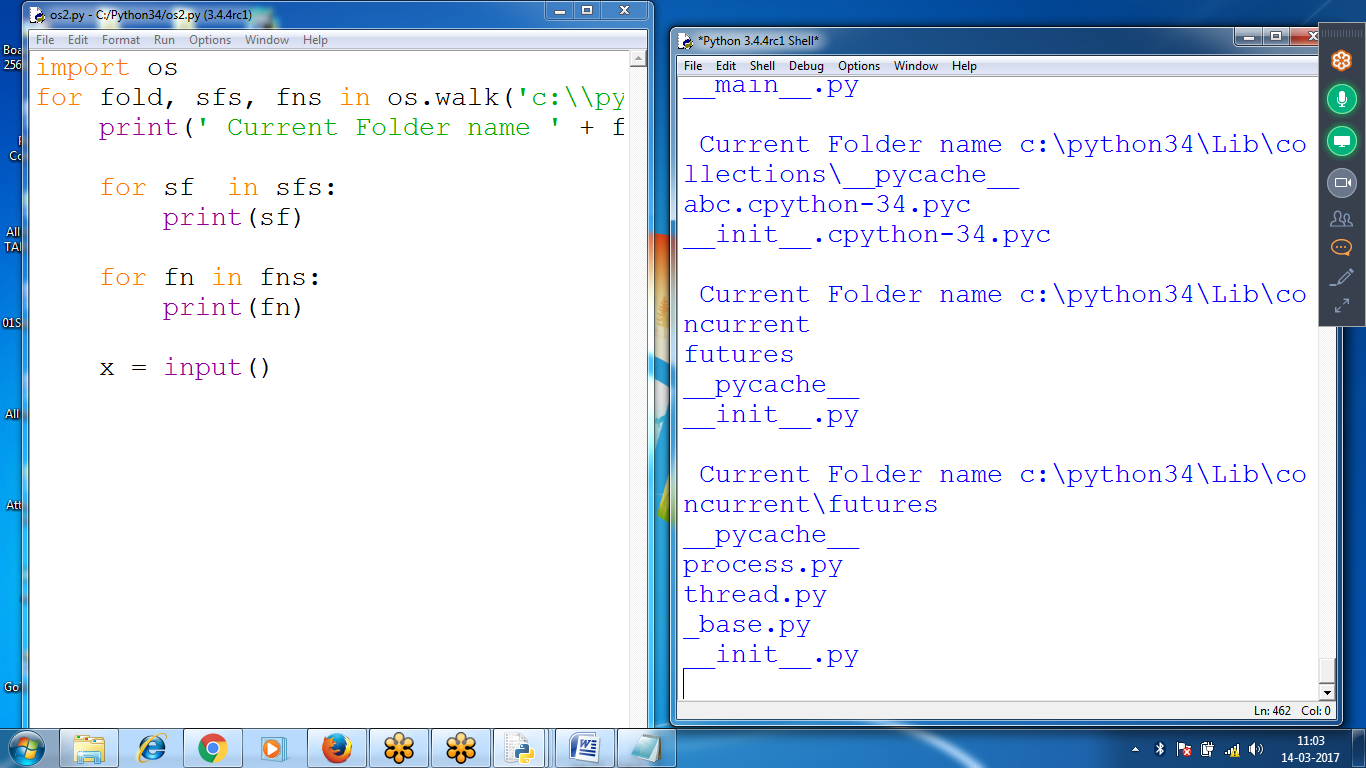
# **example2.txt**

# 

# 

# **Walking a Directory Tree**

To rename every file in some folder and also every file in every subfolder of that folder, walk through the directory tree, touching each file as you go.



import os

for fold, sfs, fns in os.walk('c:\\python34'):

print(' Current Folder name ' + fold)

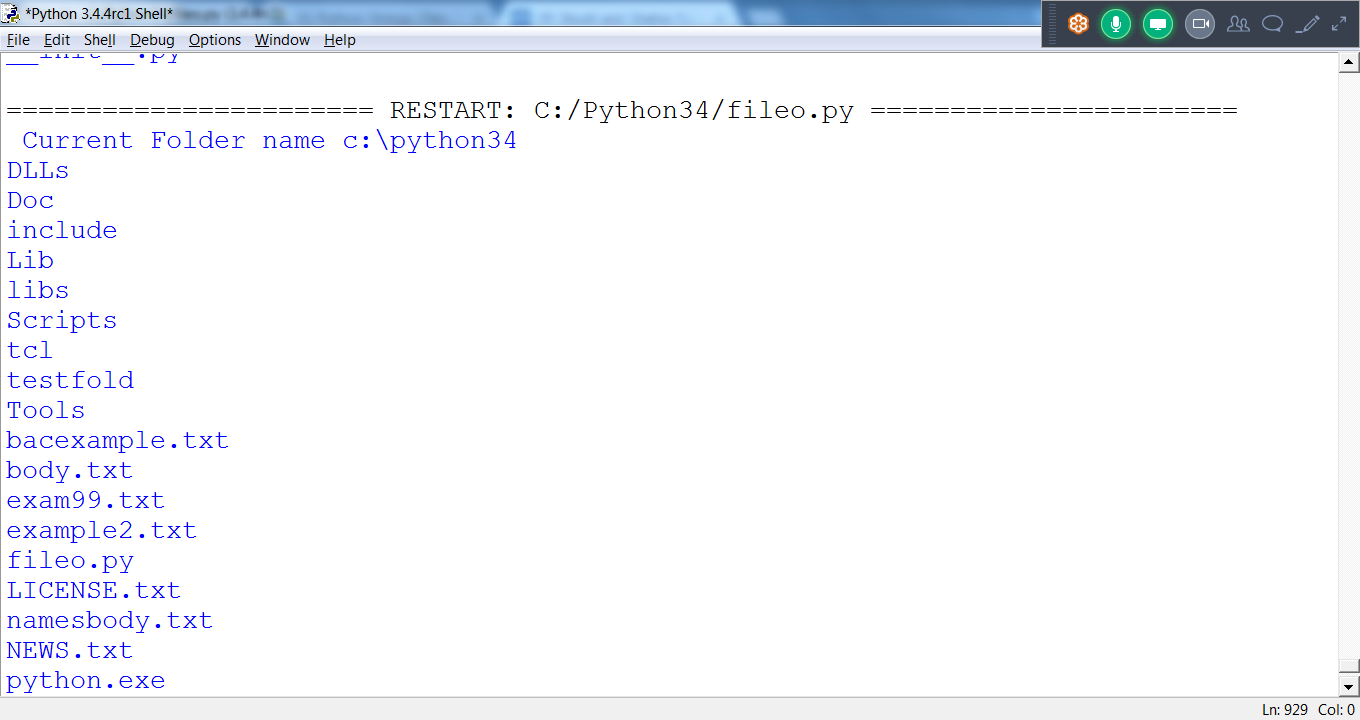
for sf in sfs:

print(sf)

for fn in fns:

print(fn)

x = input()



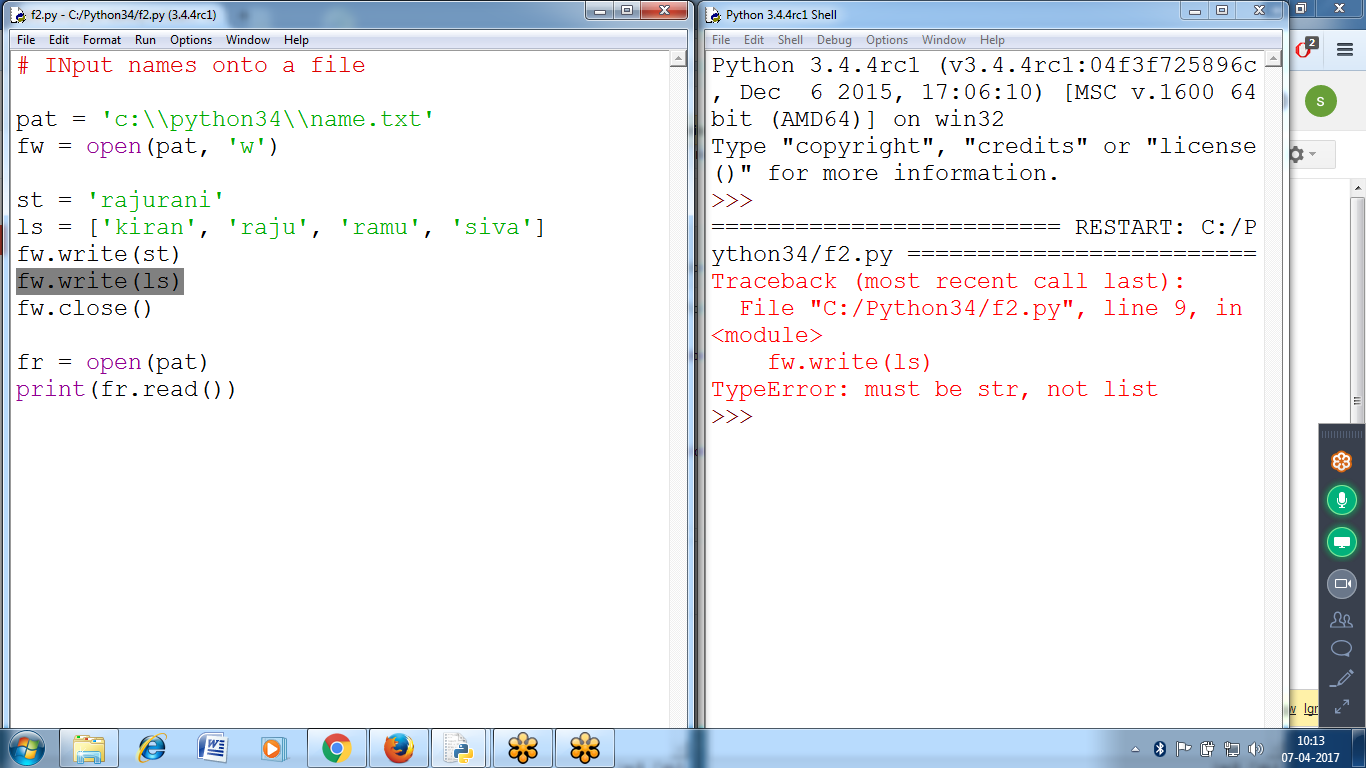
# 

# 

# 

# **To Store List Variables onto a file**

Strings can store onto a file, but not list variables



# **# INput names onto a file**

# 

# **pat = 'c:\\python34\\name.txt'**

# **fw = open(pat, 'w')**

# **st = 'rajurani'**

# **ls = ['kiran', 'raju', 'ramu', 'siva']**

# **fw.write(st)**

# **fw.write(ls)**

# **fw.close()**

# 

# **fr = open(pat)**

# **print(fr.read())**

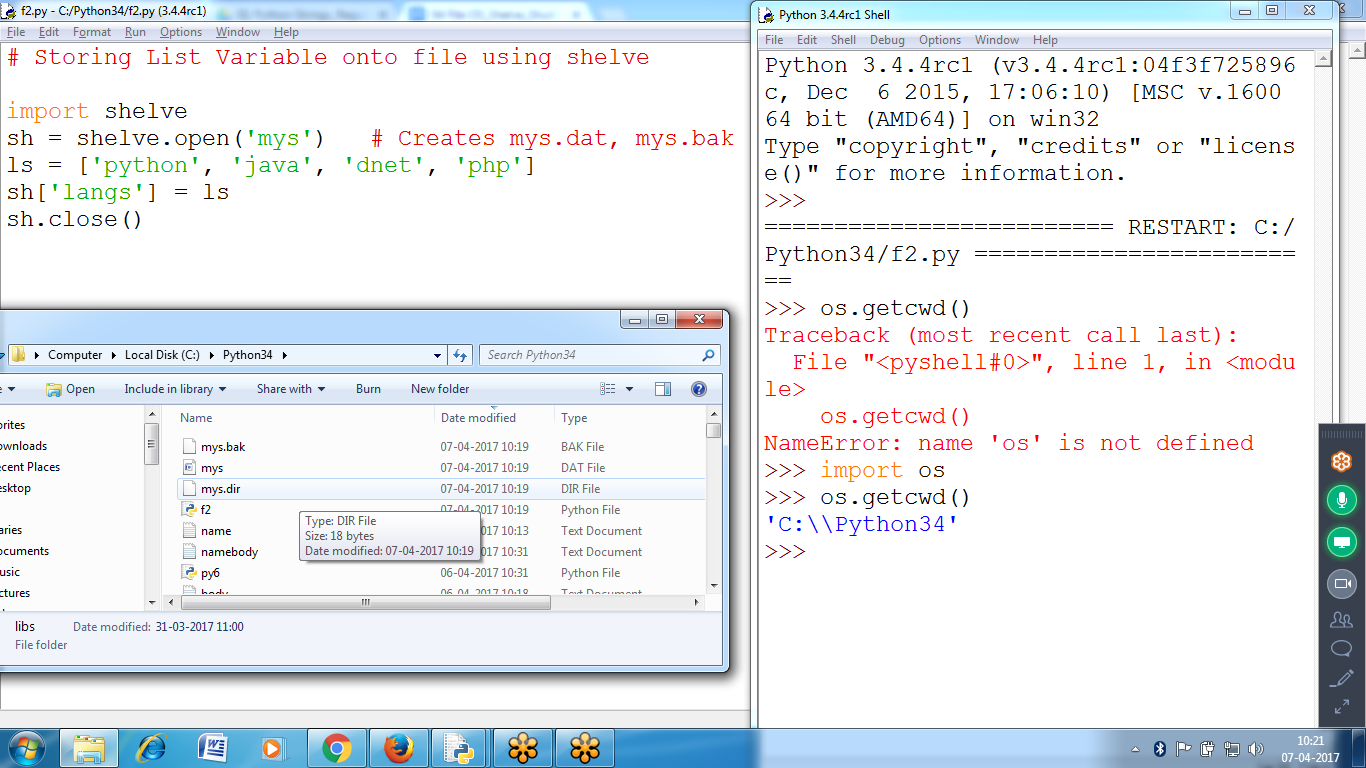
# 

# 

# 

# **Saving Variables like (LIST Type) with the shelve Module**

* To save variables in Python programs to binary files using the **shelve module**
* Program can restore data to variables from the hard drive



# Storing List Variable onto file using shelve

import shelve

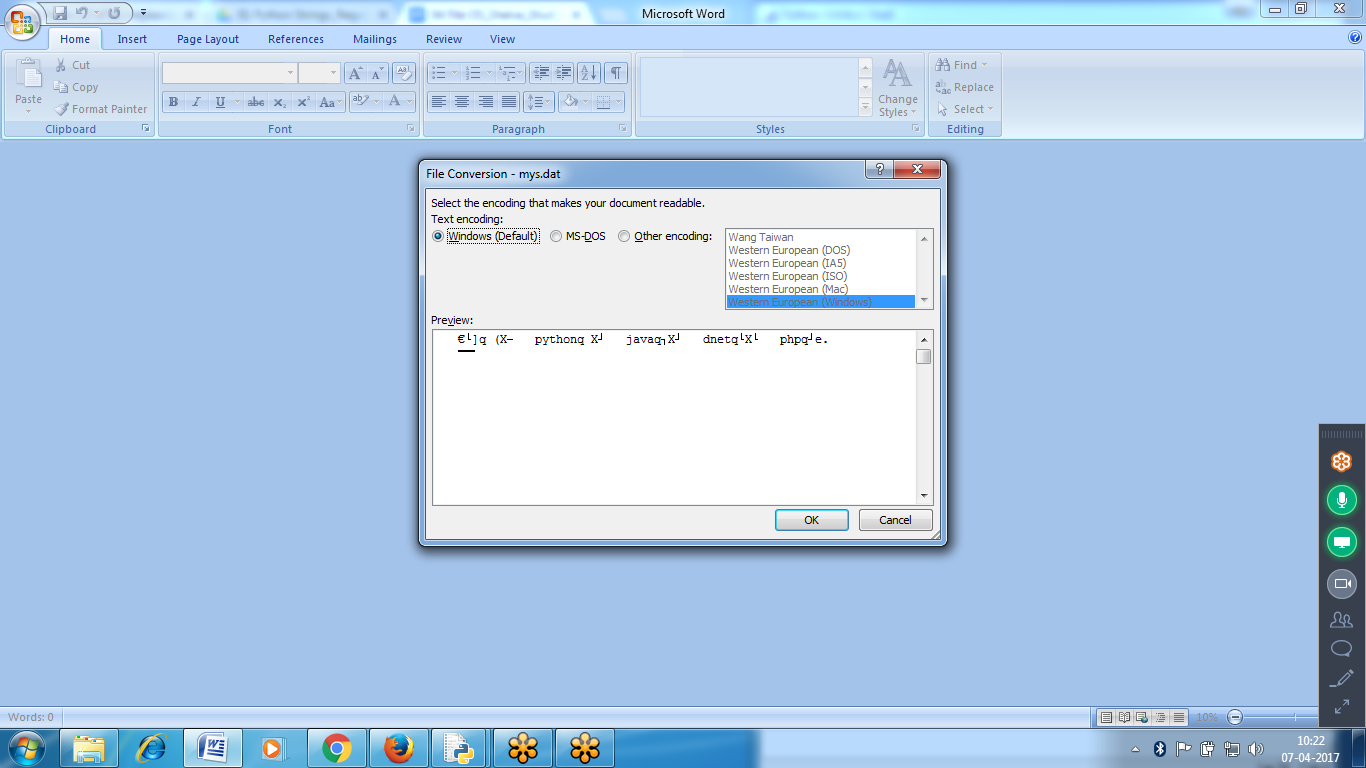
sh = shelve.open('mys') **# Creates mys.dat, mys.bak, mys.dir as binary files**

ls = ['python', 'java', 'dnet', 'php']

sh['langs'] = ls

sh.close()

Open mys.dat file



**Filename : mys , creates mys.dat, mys.dir, mys.bak**

Sh is object of mys binary file

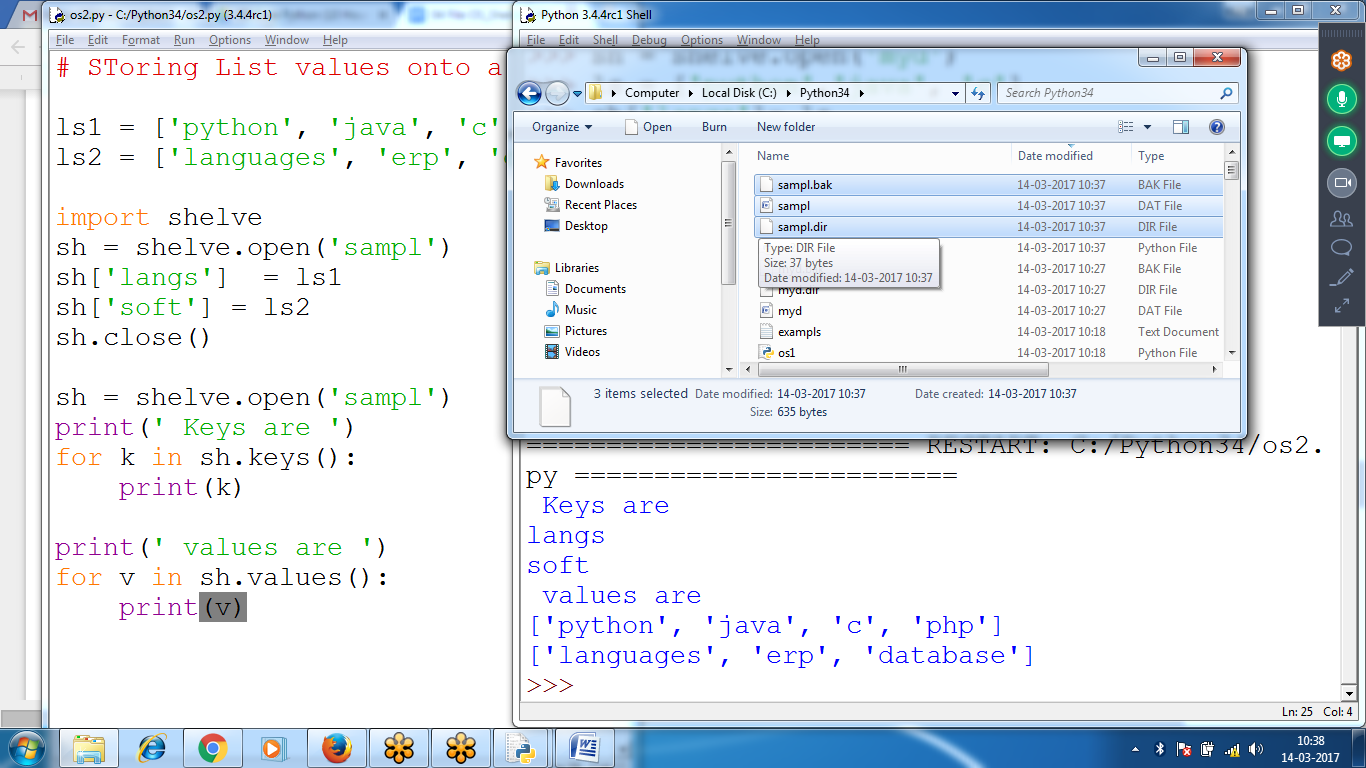
Langs is dictionay data type in myd file

If path not specified, default takes Current Working Directory :: c:\\pyton34  
Stores in 3 files:: bak, dat, dir

>>> sf = shelve.open('mydata')

>>> **type(sf)**

<class 'shelve.DbfilenameShelf'>



**#Program to restore data to variables from the hard drive**

ls1 = ['python', 'java', 'c', 'php']

ls2 = ['languages', 'erp', 'database']

import shelve

sh = shelve.open('sampl')

sh['langs'] = ls1

sh['soft'] = ls2

sh.close()

sh = shelve.open('sampl')

print(' Keys are ')

for k in sh.keys():

print(k)

print(' values are ')

for v in sh.values():

print(v)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# **Saving Variables with the pprint.pformat()**

>>> import pprint

>>> langs = [ {'lang1':'python', 'desc':'General Purpose'}, {'lang2':'java', 'desc':'Webbased'} ]

>>> print(langs)

[{'lang1': 'python', 'desc': 'General Purpose'}, {'lang2': 'java', 'desc': 'Webbased'}]

>>> pprint.pformat(langs)

"[{'desc': 'General Purpose', 'lang1': 'python'},\n {'desc': 'Webbased', 'lang2': 'java'}]"

>>> fobj = open('mylangs.py', 'w')

>>> fobj.write('Languages = ' + pprint.pformat(langs) + '\n')

101

>>> fobj.close()

>>>

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |