ORGANIZING FILES

CS 3030: Python

Instructor: Damià Fuentes Escoté

University of Colorado Colorado Springs

Next class – March 5th – Quiz 3

- Object Oriented Programming
- Iterators and Generators
- Decorators
- Regular expressions
  - *The slides “Review of regex symbols 1 and 2” will be on the screen*

Previous lesson

- Reading and writing files
  - *Change the current working directory*
  - *Creating new directories*
  - *Absolute vs relative paths*
  - *File sizes and folder contents*
  - *Checking path validity*
  - *Open, reading and writing to files*
  - *Shelve module*

Organizing files

- Maybe you’ve had the experience of going through a folder full of dozens, hundreds, or even thousands of files and copying, renaming, moving, or compressing them all by hand.

The shutil module

- The shutil (or shell utilities) module has functions to let you copy, move, rename, and delete files in your Python programs.
- To use the shutil functions, you will first need to use
  - *import shutil*

Copying Files and Folders

- shutil.copy(source, destination)
  - *copy the file at the path source to the folder at the path destination.*
  - *If destination is a filename, it will be used as the new name of the copied file.*
  - *This function returns a string of the path of the copied file.*

- shutil.copytree(source, destination)
  - *copy the folder at the path source, along with all of its files and subfolders, to the new folder at the path destination.*

Moving and Renaming Files and Folders

- shutil.move(source, destination)
  - *Move the file or folder at the path source to the path destination and will return a string of the absolute path of the new location.*
  - *If there is a file with the same name in destination, it would be overwritten.* ***Since it’s easy to accidentally overwrite files in this way, you should take some care when using move().***
  - *The destination path can also specify a filename.*
  - *But if there is no destination folder, then move() will rename the source file to the destination name.*

Permanently Deleting Files and Folders

- os.unlink(path)
  - *will delete the file at path.*
- os.rmdir(path)
  - *will delete the folder at path. This folder must be empty of any files or folders.*
- shutil.rmtree(path)
  - *will remove the folder at path, and all files and folders it contains will also be deleted.*

BE CAREFUL WHEN USING THESE FUNCTIONS!!
They permanently delete the files and folders.

Permanently Deleting Files and Folders

```
import os

for filename in os.listdir():
    if filename.endswith('.txt'):
        os.unlink(filename)
```

Permanently Deleting Files and Folders

```
import os

for filename in os.listdir():
    if filename.endswith('.txt'):
        #os.unlink(filename)
        print(filename)
```

Do this before, to see which files are going to be deleted.

Safe Deletes with the send2trash Module

- Using send2trash is much safer than Python’s regular delete functions, because it will send folders and files to your computer’s trash or recycle bin instead of permanently deleting them.
- If a bug in your program deletes something with send2trash you didn’t intend to delete, you can later restore it from the recycle bin.
- You may have to install the module
  - *pip install send2trash*
- It cannot pull files out of the trash.

Safe Deletes with the send2trash Module

```
import send2trash

baconFile = open('bacon.txt', 'a’) # creates the file
baconFile.write('Bacon is not a vegetable.’)
baconFile.close()

send2trash.send2trash('bacon.txt')
```

Walking a Directory Tree with os.walk(path)

```
import os

for folderName, subfolders, filenames in os.walk('.’):
    print('The current folder is ' + folderName)

    for subfolder in subfolders:
        print('SUBFOLDER OF ' + folderName + ': ' + subfolder)

    for filename in filenames:
        print('FILE INSIDE ' + folderName + ': '+ filename)

    print('')
```

Compressing Files with the zipfile Module

- Compressing a file reduces its size, which is useful when transferring it over the Internet.
- And since a ZIP file can also contain multiple files and subfolders, it’s a handy way to package several files into one.
- Your Python programs can both create and open (or extract) ZIP files using functions in the zipfile module.

Compressing Files with the zipfile Module - Reading

```
import zipfile, os

path = os.path.join('.','lectures','lecture12')
os.chdir(path) # move to the folder with example.zip
exampleZip = zipfile.ZipFile('example.zip')
print(exampleZip.namelist())       # ['example/', 'example/folder2/',
                                   # 'example/folder1/', 'example/image1.png',
                                   # 'example/image2.png', 'example/image3.png']
spamInfo = exampleZip.getinfo('example/image1.png')
print(spamInfo.file_size)          # 215872 bytes
print(spamInfo.compress_size)      # 190101 bytes
exampleZip.close()
```

Compressing Files with the zipfile Module - Extracting

```
import zipfile, os


path = os.path.join('.','lectures','lecture12’)
os.chdir(path) # move to the folder with example.zip
exampleZip = zipfile.ZipFile('example.zip’)
exampleZip.extractall()
exampleZip.close()
```

Compressing Files with the zipfile Module - Extracting

```
import zipfile, os


path = os.path.join('.','lectures','lecture12’)
os.chdir(path) # move to the folder with example.zip
exampleZip = zipfile.ZipFile('example.zip’)
# Extract specific file
exampleZip.extract('example/image1.png’)
# Extract to specific location
newPath = os.path.join('..','lecture11’)
exampleZip.extract('example/image1.png', newPath)
exampleZip.close()
```

Compressing Files with the zipfile Module - Compressing

```
import zipfile

newZip = zipfile.ZipFile('ziptest.zip', 'w’)
newZip.write('test.py', compress_type=zipfile.ZIP_DEFLATED)
newZip.close()
```

Compressing Files with the zipfile Module - Compressing

```
import zipfile


newZip = zipfile.ZipFile('ziptest.zip', 'w’)
newZip.write('test.py', compress_type=zipfile.ZIP_DEFLATED)
newZip.close()
```

Notice that, just as with writing to files, write mode will erase all existing contents of a ZIP file. If you want to simply add files to an existing ZIP file, pass 'a' as the second argument to zipfile.ZipFile() to open the ZIP file in append mode .

Homework 6

1. Multiclipboard
2. Renaming Files with American-Style Dates to European-Style Dates
3. Selective Copy
4. Deleting Unneeded Files (Extra points)