File Manipulation using Python

Jerry Lee

CTI One Corporation

Operating System Interface

The os module provides functions for interacting with the operating system:

```
import os    #imports the os interface module
dir(os)      #returns a list of module’s functions
help(os)     #returns an extensive manual page
```

Some os module functions:

```
os.getcwd()    # get current working directory
os.chdir('/usr/cs265') # change current working directory
os.system('mkdir tensor') # perform a mkdir in the system shell
os.path.join('path’) # Join one or more path components
    intelligently
```

Shutil module

- Library for file/directory management tasks

```
>>> import shutil
#copies data1.txt to data2.txt
>>> shutil.copyfile('data1.txt', ‘data2.txt')
#move(source, destination)
>>> shutil.move('/build/executables',
    'installdir')
```

Shutil Functions - Copy

- shutil.copy(*src*, *dst*)
  - Copy the file *src* to the <u>file or directory *dst*</u>
- shutil.copy2(*src*, *dst*)
  - Identical to copy(), and preserve file metadata
- shutil.copytree(*src*, *dst*, *symlinks=False*, *ignore=None*)
  - Recursively copy an entire directory tree rooted at *src*
- shutil.copyfile(*src*, *dst*)
  - Copy the contents (no metadata) of the file named *src* to a file named *dst*. *dst* must be the <u>complete target file name</u>
- shutil.copymode(*src*, *dst*)
- shutil.copystat(*src*, *dst*)

Shutil Functions – move & delete

- shutil.rmtree(*path*[, *ignore_errors*[, *onerror*]])
  - Delete an entire directory tree; *path* must point to a directory
- shutil.move(*src*, *dst*)
  - Recursively move a file or directory (*src*) to another location (*dst*)
- shutil.make_archive()
- shutil.get_archive_formats()

File Wildcards

- glob provides function to search file
- Returns file lists from directory wildcard searches

```
import glob
glob.glob('*.py')
['example1.py', 'example2.py',
   'example3.py']
```

Glob Library

- The glob module finds all the pathnames matching a specified pattern
- glob.glob(*pathname*)
  - Return a possibly-empty list of path names that match *pathname*

Command Line Arguments

- Command line arguments are stored in the *sys* module’s *argv* attribute as a list:
- >>> python demo.py one two three
- In the code:

```
import sys
print sys.argv
# output: ['demo.py', 'one', 'two', 'three']
```

Virtual Env in Python

- Virtualenv is a tool to create isolated Python environments
- To use the tensorflow environment
  - $ source ~/tensorflow/bin/activate
  - It will show (tensorflow) in the command line
- Recommend to use Jupyter Notebook
  - $ Jupyter Notebook
- To leave the environment
  - $ deactivate
- Reference
  - https://docs.python-guide.org/dev/virtualenvs/

Thank you