Information Infrastructure II

INFO I211 – Spring 2014 – Sections 18530 & 22519

Lecture 11 – 2014.02.19 & 2014.02.20

Instructor:
Mitja Hmeljak,
http://mypage.iu.edu/~mitja
mitja@indiana.edu

Python Standard Library Documentation

http://docs.python.org/2.6/library/

A comprehensive guide to the built-in modules in 2.6 Python.

Modules

Every Python file ... can be a module.

Modules contain groups of related *functions*, e.g.

math contains a lot of useful mathematical code

Modules can also contain classes or constants, e.g.

math.pi

Normally when we import a module we then have to say **modulename.methodname()**

User-defined Modules: *example*

we can define a new module in a file named **mathFunc.py** :

```
def summation(x):
    total = 0
    for num in x:
        total += num
    return total

def mean(x):
    total = 0
    for num in x:
        total += num
    return total/len(x)
```

we can then use our newly written module in any code, by *importing* the module **mathFunc** (note: no ".py" after the name of the module). E.g. a program **calc.py** can be written thus:

```
import mathFunc

lst = [1,2,3,4]
total = mathFunc.summation(lst)
mean = mathFunc.mean(lst)
print "Total:", total
print "Mean value:", mean
```

Ways to Import:

```
# import by module name, then use module name as prefix:
import mathFunc
mathFunc.summation([1,2,3])

# import and give your module a 'nickname' prefix, i.e. a local name prefix for use just within your program:
import mathFunc as mf
mf.summation([1,2,3])

# import all functions from a module and make their name local , i.e. no need for a module prefix:
from mathFunc import *
summation([1,2,3])
# be careful – this way you can't have local functions with the same name!
```

Python Standard Library

1. Intro to the Library
2. Built-in Functions
3. Built-in Functions
4. Built-in Constants
5. Built-in Types
6. Built-in Exceptions

Python Standard Library

7. Handling Text
8. Advanced Data Types (queues, heaps, etc)
9. Numbers and Math
10. Files and Directories
11. Data persistence & Databases
12. File compression (zip & tar)
13. File Reading (csv files, etc)
14. Cryptography
15. OS, time, etc

Python Standard Library

18. Internet Data Handling
19. HTML/XHTML/XML
20. Network Protocols
21. Multimedia

etc.

System Module (sys)

sys.path is a variable that lists the paths Python searches when loading modules (in addition to the current working directory)

Run this code!

```
import sys

for i in sys.path:
    print (i)
```

Getting help

If you're not sure what a module has, try this:

print (dir([module name]))

This gives you a list with all the objects and functions in the module. Make sure you import the module first!

print (help([module name]))

This gives you a (very) detailed breakdown of the module. Try adding the first line to the end of the code, then the second!

Operating Systems

Directories are structured differently in Windows and Unix *(Unix includes Mac OS X, Linux, ...)*

C:\Users\Username\Desktop\file.py Windows

/nfs/nfs1/home/username/file.py Unix

Make sure you use the right slash characters (forwards vs. backwards)

Operating System Module (os)

To get the current working directory:

```
import os
print (os.getcwd())
```

To go up a folder:

```
myDirPath = os.getcwd()
path = myDirPath + "/folderName"      #unix
```

Or

```
path = myDirPath + "\\folderName"    #windows
os.chdir(path)#changes the directory Python 'sees'
```

Operating System Module (os)

List contents of a directory:

```
import os
print (os.listdir(os.getcwd()))
```

Try this!

It returns a list, which we can then manipulate…

Files & Directories

```
import os
myDirPath = os.getcwd()
```

Check if something is a directory:

```
os.path.isdir(myDirPath)
```
True or False

Check if something is a file:

```
os.path.isfile(myDirPath + "\\testfile.py")
```
True or False

Directory Listing (Group Work)

Write a function called directory_choice.

This function should print the names of all the directories in the cwd (but not the files), and then ask the user to choose one of them.

If the user inputs something that isn't valid (i.e. isn't a directory name), ask again until they do.

Then print the user's choice.

Directory Listing (Solution)

```
import os

def directory_choice():
    currentDir = os.getcwd()
    allEntries = os.listdir(currentDir)
    allDirs = []

    for oneEntry in allEntries:
        if os.path.isdir(oneEntry):
            allDirs.append(oneEntry)

    print "Current Directories:"
    for oneDir in allDirs:
        print oneDir

    userChoice = ""
    while userChoice not in allDirs:
        userChoice = raw_input("Please select a valid directory: ")

    return userChoice

d = directory_choice()
print "The user chose:", d
```