New York University
School of Continuing and Professional Studies
Division of Programs in Information Technology

Introduction to Python
Exercises, Session 8

Ex. 8.1 Allow your Program to Accept Arguments. Accept two arguments at the command line, using the default list object sys.argv, a list that is automatically available after you execute the statement import sys. Sum the values and print out a formatted string with an "addition" formula. (As mentioned in the slides, the first argument is at sys.argv[1], the second at sys.argv[2], etc.)

Expected Output:

```
$ python myprog.py 5 19
5 + 19 = 24

$ python myprog.py 100 4
100 + 4 = 104
```

Ex. 8.2 Validate Arguments. Extend the above program to validate user input. Using try: and except:, trap an IndexError exception if one of the two arguments is missing - and have the program exit() with a Usage: string if the exception occurs (see output and note below).

Exceptions: trap an exception for missing arguments (IndexError when reading from sys.argv). (Remember to place as little code as possible in your try: block.) If possible, print the program name in the Usage string (as shown below) using sys.argv[0].

Expected Output:

```
$ python myprog.py
Usage: myprog.py [int1] [int2]

$ python myprog.py 5
Usage: myprog.py [int1] [int2]

$ python myprog.py 5 hello
Usage: myprog.py [int1] [int2]

$ python myprog.py 5 10
5 + 10 = 15
```

File and Directory Input / Output

Note on files and directories: anytime we ask Python to ask the OS to find a file or directory (through os.listdir(), os.path.isfile(), os.path.getsize(), etc.), the OS is starting from the present working directory (pwd), which is the directory from which we are running our Python program. If the OS can't find this file or directory in the pwd, it will not look elsewhere.

The straightforward way to find a file or directory is to use the absolute pathname, which is the full location of the file within the filesystem. In Unix/Linux/Mac, it begins with a forward slash, e.g. /Users/dblaikie/thisdir/myfile.txt; and in Windows it usually begins with C:\: as in C:\Documents\ and\ Settings\dblaikie\thisdir\myfile.txt, or C:\Users\dblaikie\thisdir\myfile.txt.

To find the absolute path of any file, go to the directory where it is listed, and type pwd at the prompt. Add the filename to this path and you should have the absolute path. You can confirm this is the case with one of the following commands: ls [pathname] (that's "ell ess") on Unix/Linux/Mac, or dir [pathname] on Windows.

Ex. 8.3 os.path.getsize(). Pick a file in the python_data directory (or any file) and save the path to the file in a string variable, like so:

```
filename = '../python_data/student_db.txt'      # this is the name of an existing file
```

Write a script that prints the size of the file in bytes.

Expected output (will vary depending on file chosen):

```
../python_data/student_db.txt:   333 bytes
```

Ex. 8.4 Validate filename. Continuing the previous program, take a filename from the user through the command line. Use os.path.isfile() to see if the submitted file is an existing file (and not a directory or link or other entity). If it is a file, then print the filename and size. If it is not a file, then print an error message.

Expected output (will vary depending on file chosen)

```
$ python myprog.py ../python_data/student_db.txt
../python_data/student_db.txt:  333 bytes

$ python myprog.py xxxfile.txt
error:  xxxfile.txt is not a file in this directory
```

Ex. 8.5 List a Directory. Accept a string argument that is the pathname of a directory. Print out all items in the directory listing using os.listdir(). Trap the exception that would result from an unreadable directory (i.e., if not exist or no permissions. To determine this exception, try to read a directory that doesn't exist, and note the exception that results). Using os.path.isfile(listing) and os.path.isdir(listing), where listing is one of the items returned from listdir(), identify whether the listing is a file or directory.

Note that while listdir returns filenames, it doesn't return file paths. So if the directory you're listing isn't in the same directory as your Python program, you'll need the whole path. You can construct a whole path and test a file this way:

```
filepath = os.path.join(dir, filename)
```

so you can do a file test this way:

```
if os.path.isfile(os.path.join(dir, filename)):
```

Exceptions: trap exception for missing argument (IndexError when trying to access sys.argv[1]) and unreadable directory (OSError when attempting to read directory using os.listdir()).

Program runs (output of course depends on the directory you supply):

```
$ python myprog.py
error:  please provide an argument

$ python myprog.py xxxxnonexistxxx
error:  directory does not exist or is not readable

$ python myprog.py ../python_data
access_log.txt (file)
ALL_DATA.zip (file)
dormouse.html (file)
ex3.txt (file)
... [etc.] ...
```

Ex. 8.6 List files and sizes in another directory. Continuing the previous exercise, output the file name and byte size (using os.path.getsize()) of each file.

```
$ python myprog.py ../python_data/
access_log.txt (file):  395062
ALL_DATA.zip (file):  2447578
dormouse.html (file):  832
... [etc.] ...
```