Chapter 11

Files & Exceptions

Jie Tian, PhD

Clark University

Outline

- Read files
- Write files
- Format outputs
- Write Modules
- Deal with Exceptions

Working with Files

- Get input data from files (.txt, .shp, etc)
- Write output or results into files.
- Reading and writing files are important.

Filenames & Paths

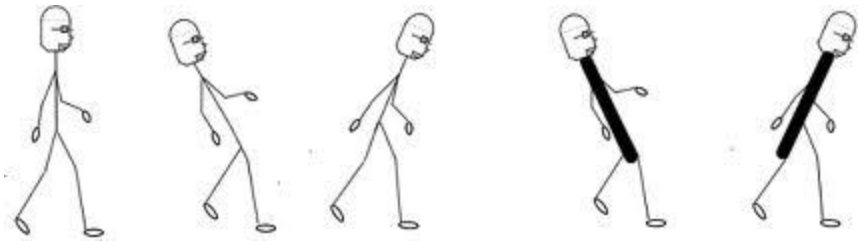
```
strPath =
'd:\\PythonForGIS\\2011fall\\ch11_FilesAndExceptions\\'
strName = 'mytestfile.txt'


strPathName = strPath+strName
f=open(strPathName, 'r')
f.close()
```

strPath can also be written as (not the convention for this class):

```
strPath  =  'd:/PythonForGIS/ch11_FilesAndExceptions/'
```

- Remember to use **** to represent **** in a path string



Slash vs. Back Slash

- Alternative is to place an **r** before the string

```
strFolderPath = r'd:\Intro2Python\Fall2012\Files\'
```

Read an Existing Text File

- Like how you read a file:

  open () → read () → close()

```
strFolderPath = 'd:\\Intro2Python\\Files\\'
strFileName = 'ReadMe.txt'
strFullPath = strFolderPath + strFileName
f2 = open(strFullPath, 'r')
str  = f2.read() # read the whole file
f2.close()

print str
```

Read-only Mode

Output

```
Clark University is in Worcester.
Worcester is in Massachusetts.
```

Read a given number of characters

```
strFolderPath = 'd:\\Intro2Python\\Files\\'
strFileName = 'ReadMe.txt'
strFullPath = strFolderPath + strFileName

f2 = open(strFullPath,'r')

str1  = f2.read(5) #only read five characters
str2  = f2.read()  #read the rest from the file
f2.close()

print "str 1 is: " + str1
print "str 2 is: " + str2
```

Output

```
str 1 is: Clark
str 2 is:  University is in Worcester.
And Worcester is in Massachusetts.
```

Read a Text File Line by Line

```
strFolderPath = 'd:\\Intro2Python\\Fall2012\\Files\\'
strFileName = 'ReadMe.txt'
strFullPath = strFolderPath + strFileName

f2 = open(strFullPath,'r')

line1=f2.readline() # Read the first line
line2=f2.readline() # Read the second line

print "Line 1:", line1
print "Line 2:", line2
```

Output:
Line 1: Clark University is in Worcester
Line 2: Worcester is in Massachusetts

```
linesList = f2.readlines() # Read lines into a list

print linesList
['Clark University is in Worcester\n','Worcester is in Massachusetts']
```

Write a New Text File

- Like how you write a file:

  open () → write () → close()

```
strFolderPath = 'd:\\Intro2Python\\Fall2012\\Files\\'
strFileName = 'NewFile.txt'

strFullPath = strFolderPath + strFileName

f=open(strPathName, 'w')   #'w' is for 'writing' mode.
                           # if a file with the same name exits,
                           # its contents are removed.


f.write('Clark University is in Worcester.\n')
f.write('Worcester is in Massachusetts.')
f.close()
```

Copy a Text File

```
def copyFile(oldFile, newFile):
    f1 = open(oldFile, "r")
    f2 = open(newFile, "w")

    text = f1.read(50)
    while text != "":
        f2.write(text)
        text = f1.read(50)

    f1.close()
    f2.close()
```

open() → read() → close()

open() → write() →close()

Alternative?

```
text = f1.read()
f2.write(text)
```

```
strPath = 'd:\\PythonForGIS\\ch11_FilesAndExceptions\\'
strName = 'mytestfile.txt'
strFullName = strPath+strName
strNewName = strPath + 'copy.txt'
copyFile(strFullName, strNewName)
```

Formatting Output

- It is only allowed to write strings into a file.
- Convert other data types into strings.

Question:

```
>>> f=open("c:\\test.txt", 'w')
>>> f.write(str(12.3))
>>> f.write(str([1,2,3]))
>>> f.close()
```

What is in the file:

12.3[1, 2, 3]

Formatting Output (cont'd)

- Can also use the **format operator**, **%**

  `%d` for outputting integers

  `%f` for outputting floating numbers

  `%s` for outputting strings

```
>>> name = ‘good’
>>> '%s' % name
‘good'

>>> students = 18
>>> '%d' % students
18

>>> number = 18
>>> '%f' % number
18.000000

>>> "%6d"  %   62
    62

>>> "%-6d"  %   62
62

>>> '%12f' %  6.1
    6.100000
>>> '%12.2f'  %  6.1
```

***Sample Code*: Nicely Formatted Output**

```
def report (wages):
    students = wages.keys()
    students.sort()
    for student in students :
        print "%-20s %12.2f" % (student, wages[student])

wages = {'mary': 6.238, 'joe': 5.45, 'john': 4.25}
report (wages)

# output


joe                             5.45
john                            4.25
mary                            6.24
```

Writing Modules

- Any file that contains Python code can be imported as a module.
- For example, suppose you have a file named wc.py with the following code:

```
def linecount(filename):
    count = 0
    for line in open(filename):
        count += 1
    return count

print linecount('c:\\wc.py')
```

```
>>> import wc
7

>>> wc.linecount('c:\\wc.py ')
7
```

Once imported, the module code runs automatically.

Now, you have the linecount () function available.

```
def linecount(filename):
    count = 0
    for line in open(filename):
        count += 1
    return count

if __name__ == '__main__':
    print linecount(' c:\\wc.py ')
```

Importing this module will not execute anything but make the function available

- __name__ is a built-in variable that is set when the program starts.
- If the program is running as a script, __name__ has the value __main__; the test code is executed.
- Otherwise, if the module is being imported, the test code is skipped.

PythonPath

```
import sys

if "C:\\My_Python_Lib" not in sys.path:

      sys.path.append("C:\\My_Python_Lib")
```

Exceptions (errors)

```
>>> print 55/0
ZeroDivisionError: integer division or modulo

>>> a = []
>>> print a[5]
IndexError: list index out of range

>>> b = {}
>>> print b['what']
KeyError: what

>>> f = open("Idontexist", "r")
IOError: [Errno 2] No such file or directory:
'Idontexist'
```

try…except…

- It is better to go ahead and try, and deal with problems if they happen
- Python starts by executing the try clause. **If all goes well**, it **skips** the except clause and proceeds. **If an exception occurs**, it **jumps out** of the try clause and **executes the except clause**.

```
filename = raw_input('Enter a file name:')
try:
    f = open (filename, "r")
    f.read()
    f.close()
except IOError:
    print "There is no file named ", filename
```

http://docs.python.org/2/library/exceptions.html

Sample Code: putting pieces together

Problem:

A text file contains donors' IDs and the amount donated in a comma separated value (CSV) file. **1)** Get the file name; **2)** read the data into a dictionary; **3)** output the donors' IDs (left-aligned) and the amount donated (right-aligned)

The data file contains the following contents:

1, 32.5, 2, 42.3, 3, 89.9, 4, 0.5, 5, 0.7, 6, 4.1, 7, 5.8, 8, 8.2

```
import string, sys

def exists(filename):
    try:
        f = open(filename)
        f.close()
        return True
    except IOError: #an build-in exception in Python.
        return False

def ReadInTextFile(fname):
    f=open(fname,'r')
    str = f.read()
    return str

def StrToDic(strInput):
    list = string.split(strInput, ',')
    dic={}
    i=0
    while i<len(list):
        if (i % 2 == 0):
            key=list[i]
        else:
            val = list[i]
            dic[int(key)] = float(val)
        i = i +1
    return dic
```

```
def PrintDic(dicInput):
    for (key, value) in dicInput.items():
        print "%-5d %8.2f" % (key, value)

filename = sys.argv[1]
if exists(filename):
    mystr = ReadInTextFile(filename)
    mydic = StrToDic(mystr)
    PrintDic(mydic)
else:
    print "File "+filename+" does not exists."
```

Summary

- Open a file before reading it and don’t forget to close it afterwards.
- Use open() to create a new file and then write content.
- Format operator % can be very handy.
- Use try…except… & raise to catch exceptions.
- You can write your own Python modules.