An Introduction to Python: File I/O

File I/O

Table of Contents

Let's do things with files!

File I/O in C/C++ required including several header files and defining a series of stream objects, or file objects. Python is similar, but opening files is a more basic function and generally much more straightforward. Let's create a file quickly and dump some text into, namely a list of names that we'll enter at the commandline.

```
# Let's create a file and write it to disk.
filename = "test.dat"
# Let's create some data:
done = 0
namelist = []
while not done:
    name = raw_input("Enter a name:")
    if type(name) == type(""):
        namelist.append(name)
    else:
        break

    # Create a file object:
    # in "write" mode
    FILE = open(filename,"w")
    FILE.writelines(namelist)

    # Alternatively
    # for name in namelist:
    #       FILE.write(name)

    FILE.close() # this is icing, you can just exit and this will be
    # handled automagically.
```

Notice that we simply have to use the open command to open a file, and give it the "r" argument to open the file for reading. The value returned from this function is a file object, which we can use to read (or write, if we had also(instead of) specified the "w" flag). From there you can either read the entire file with a fileobject.read() command, or read a line at a time with readline()(what I usually use).

Now let's go over some useful utility functions:

```
import os.path
```

```
# os.path - The key to File I/O
os.path.exists("bob.txt")
os.path.isfile("bob.txt") # Does bob.txt exist?  Is it a file, or a directory?
os.path.isdir("bob")
os.path.isabs("/home/me/bob.txt") # Is it an absolute path to this file?

# Creating cross platform paths
# This will be slightly different on each platform
currentdir = os.curdir
imagedir = os.path.join(currentdir, "images")

# Let's say I have a full path, and yet I want to store records based
# on the name of the file:
longpath = "/home/me/python/somefiles/junk/notjunk/blah/bingo.txt"
shortpath = os.path.basename(longpath)

# Get the type of shortpath:
print "Type of",shortpath,"is", os.path.splitext(shortpath)[1]

# os.path.walk can be used to traverse directories recursively
# to apply changes to a whole tree of files.
def callback( arg, dirname, fnames ):
    sum = 0
    for file in fnames:
        sum += os.path.getsize(file)
    arg.append(sum)

arglist = []
os.path.walk("./",callback,arglist)

sum = 0
for value in arglist:
    sum += value

print "Size of directory:",sum
```

Here is a more advanced example, which reads in the following file, "hw0grades.txt", which contains the current grades for the first (Zeroth) cs373 homework:

> 1883 46.5 Alpha Quadrant 62 Araldor 47 ASPWizKid 48.5 bamboozle 12.5 banana 42.5 Barney 47.5 Ben Kenobi 32.5 bfavre 49.5 Bib-B 47.5 Big-P 47.5 bikerider 38.5 brit1 42.5 buan 35.5 cheese 47.5 ChinDaddy 56 Clockwisefoot 47.5 clutch 38.5 crzyfk 51.5 Dave 32 dbrown 49.5 Eggman63 47.5 eidiki 47 elf2582 58 epsilonought 58 fats403 60 fuzzy 56 gilman 60 HealeySt 30 HockeyBoy 48.5 HUMAN#31B8A 62 Iconoclast 29 IlliniSgEp 24 Isles311 54.5 jorin 58 klee kvltbm 46.5 KWO 60.5 Lastone 18 lcoker 49.5 liren 42.5 ljn35 54.5 Lock 37 Mist1 58.5 Mist2 58.5 moses 42.5 NiceMan 60.5 og4l 51.5 photon 51 Pootie-Tang 32 Rollo Tamasi 27 Sammy 43.5 Shodan 37 skv9 56 Slayn 37 spidey 38.5 spoonman 43.5 spoonman2530 54.5 summy 43.5 The First Amigo 33 The Second

Amigo33 The Third Amigo 33 the257 44.5 TooDumb373 42.5 Very Sleepy 63.5 Very Sleepy 63.5 Very Sleepy 63.5 washu 60 xtreme 51 ziggy43 19 zorba 20.5 56.5 56.5 58.5 58.5 35.5 41

Now, since we know some basics of files, we can go ahead and read them a bit, and go ahead and compute somethings about my grade:

```
# Let's get the average and our grades:
US = {"Sammy":0,"summy":0,"spoonman":0}
HWAVE = 0
TOTAL = 0
FILES = ["hw0grades.txt"]

STUDENTS = 0

# Get the Data
for file in FILES:
    infile = open(file,"r")
    while infile:
        line = infile.readline()
        for person in US:
            if line.find(person) >= 0 and len(line.split()[0])==len(person):
                US[person] = float( line.split()[1] )
        s = line.split()
        n = len(s)
        if n == 0:
            break
        try:
            TOTAL += float( s[ n-1 ] )
        except:
            pass
        STUDENTS += 1

# Compute the Average
print TOTAL, STUDENTS
HWAVE = TOTAL / ( STUDENTS * (1.0) )

# Assume the average is C
# Define grade ranges:
C = HWAVE
Cmax = C + HWAVE * .05
Cmin = C - HWAVE * .05
Bmax = Cmax + HWAVE * .1
Bmin = Cmax
Amin = Bmax
Dmax = Cmin
Dmin = Cmin - HWAVE * .1
Emax = Dmin
# Print out some STATS:
print "The AVERAGE for this homework:", HWAVE
print "The A range:", ">="+str(Amin)
print "The B range:", Bmax,"-", Bmin
print "The C range:", Cmax,"-", Cmin
print "The D range:", Dmax,"-", Dmin
print "The E range:", "<"+str(Emax)

# Assign grades to US:
for person in US:
    if US[person] >= Amin:
        print person,"(",US[person],")","probably got an A on this assignment."
    elif Bmax > US[person] >= Bmin:
        print person,"(",US[person],")","probably got a B on this assignment."
    elif Cmax > US[person] >= Cmin:
        print person,"(",US[person],")","probably got a C on this assignment."
    elif Dmax > US[person] >= Dmin:
        print person,"(",US[person],")","probably got a D on this assignment."
    else:
        print person,"(",US[person],")","probably got a E on this assignment."
```

This example is a little more complicated, but it's definitely more real world. A very common use for Python is doing file and string parsing tasks like this. It's not as well known for this task as Perl, although it really is quite suitable for this sort of thing (and your chances of being able to maintain this code in the future is better).

Finally, I want to talk about pickling things. Python has a facility for compressing native objects into strings for transmission across the network, or to write to disk.

You might refer to this technique as a method of "Object Persistance" or something similar, and you'd be right. If you have data you'd like to save between sessions, like an objects personal state, just pickle it. This could be a simple way to create a basic file format for your programs datafiles, or it could enable faster crash recovery for a large program. You might even use it while playing at the commandline to save something temporarily.

There are quite a few modules besides Pickle that can be used to do this in Python, but Pickle is the fundamental building block for many of them. Larger projects with more data might decide to use Shelves, which are a lot like dictionaries that store pickled objects.

Follow along with this simple example, and remember that there is more to Pickling then what I've got below.

```
# Let's define a class quick:
class Student:
    def __init__(self, name="", school="", major="", year="", gpa=-1):
        self.name = name
        self.school = school
        self.major = major
        self.year = year
        self.gpa = gpa
    def _Print(self):
        string = "++++++++++++++++++++++" + \
                 "+"+name \
                 "+"+school \
                 "+"+major \
                 "+"+year \
                 "+"+gpa \
                 "++++++++++++++++++++++"
        return string

    def Print(self):
        print self._Print()

    def __str__(self):
        return self._Print()

# Now I'll create a student:
Mike = Student("Mike","UIUC","CS","Senior",0.6)
# Now, let's Pickle Mike
mike_the_pickle = pickle.dumps(Mike)
print mike_the_pickle
# Now let's clone mike!
Mike2 = pickle.loads(mike_the_pickle)
if type(Mike) == type(Mike2):
    print "Mike and Mike2 are of type:",(str(type(Mike)))

# Create A Pickler
P.pickle.Pickler("pickle.dat")
# Write to your Pickler
P.dump(Mike)
# OR, combine the last 2 lines
pickle.dump(Mike, "pickle.dat")

# Now, Create an Unpickler, and get Mike back
U = pickle.Unpickler("pickle.dat")
# Read from the pickler
Mike = U.load()
# Do it in one line
Mike2 = pickle.load("pickle.dat")

# Just something to keep in mind:
# For more sophisticated usage, you may want to use the Shelve module,
# or other Database modules.
```

---

hello.py
Table of Contents