# Non-Programmer's Tutorial for Python 3/File IO

### File I/O

Here is a simple example of file I/O (input/output):

# Write a file

with open("test.txt", "wt") as out\_file:

out\_file.write("This Text is going to out file\nLook at it and see!")

# Read a file

with open("test.txt", "rt") as in\_file:

text = in\_file.read()

print(text)

The output and the contents of the file test.txt are:

This Text is going to out file

Look at it and see!

Notice that it wrote a file called test.txt in the directory that you ran the program from. The \n in the string tells Python to put a *n*ewline where it is.

An overview of file I/O is:

* Get a file object with the open function
* Read or write to the file object (depending on how it was opened)
* If you did not use with to open the file, you'd have to close it manually

The first step is to get a file object. The way to do this is to use the open function. The format is file\_object = open(filename, mode) where file\_object is the variable to put the file object, filename is a string with the filename, and mode is "rt" to *r*ead a file as *t*ext or "wt" to *w*rite a file as *t*ext (and a few others we will skip here). Next the file objects functions can be called. The two most common functions are read and write. The write function adds a string to the end of the file. The read function reads the next thing in the file and returns it as a string. If no argument is given it will return the whole file (as done in the example).

Now here is a new version of the phone numbers program that we made earlier:

def print\_numbers(numbers):

print("Telephone Numbers:")

for k, v in numbers.items():

print("Name:", k, "\tNumber:", v)

print()

def add\_number(numbers, name, number):

numbers[name] = number

def lookup\_number(numbers, name):

if name in numbers:

return "The number is " + numbers[name]

else:

return name + " was not found"

def remove\_number(numbers, name):

if name in numbers:

del numbers[name]

else:

print(name," was not found")

def load\_numbers(numbers, filename):

in\_file = open(filename, "rt")

while True:

in\_line = in\_file.readline()

if not in\_line:

break

in\_line = in\_line[:-1]

name, number = in\_line.split(",")

numbers[name] = number

in\_file.close()

def save\_numbers(numbers, filename):

out\_file = open(filename, "wt")

for k, v in numbers.items():

out\_file.write(k + "," + v + "\n")

out\_file.close()

def print\_menu():

print('1. Print Phone Numbers')

print('2. Add a Phone Number')

print('3. Remove a Phone Number')

print('4. Lookup a Phone Number')

print('5. Load numbers')

print('6. Save numbers')

print('7. Quit')

print()

phone\_list = {}

menu\_choice = 0

print\_menu()

while True:

menu\_choice = int(input("Type in a number (1-7): "))

if menu\_choice == 1:

print\_numbers(phone\_list)

elif menu\_choice == 2:

print("Add Name and Number")

name = input("Name: ")

phone = input("Number: ")

add\_number(phone\_list, name, phone)

elif menu\_choice == 3:

print("Remove Name and Number")

name = input("Name: ")

remove\_number(phone\_list, name)

elif menu\_choice == 4:

print("Lookup Number")

name = input("Name: ")

print(lookup\_number(phone\_list, name))

elif menu\_choice == 5:

filename = input("Filename to load: ")

load\_numbers(phone\_list, filename)

elif menu\_choice == 6:

filename = input("Filename to save: ")

save\_numbers(phone\_list, filename)

elif menu\_choice == 7:

break

else:

print\_menu()

print("Goodbye")

Notice that it now includes saving and loading files. Here is some output of my running it twice:

1. Print Phone Numbers

2. Add a Phone Number

3. Remove a Phone Number

4. Lookup a Phone Number

5. Load numbers

6. Save numbers

7. Quit

Type in a number (1-7): **2**

Add Name and Number

Name: **Jill**

Number: **1234**

Type in a number (1-7): **2**

Add Name and Number

Name: **Fred**

Number: **4321**

Type in a number (1-7): **1**

Telephone Numbers:

Name: Jill Number: 1234

Name: Fred Number: 4321

Type in a number (1-7): **6**

Filename to save: **numbers.txt**

Type in a number (1-7): **7**

Goodbye

1. Print Phone Numbers

2. Add a Phone Number

3. Remove a Phone Number

4. Lookup a Phone Number

5. Load numbers

6. Save numbers

7. Quit

Type in a number (1-7): **5**

Filename to load: **numbers.txt**

Type in a number (1-7): **1**

Telephone Numbers:

Name: Jill Number: 1234

Name: Fred Number: 4321

Type in a number (1-7): **7**

Goodbye

The new portions of this program are:

def load\_numbers(numbers, filename):

in\_file = open(filename, "rt")

while True:

in\_line = in\_file.readline()

if not in\_line:

break

in\_line = in\_line[:-1]

name, number = in\_line.split(",")

numbers[name] = number

in\_file.close()

def save\_numbers(numbers, filename):

out\_file = open(filename, "wt")

for k, v in numbers.items():

out\_file.write(k + "," + v + "\n")

out\_file.close()

First we will look at the save portion of the program. First it creates a file object with the command open(filename, "wt"). Next it goes through and creates a line for each of the phone numbers with the command out\_file.write(k + "," + v + "\n"). This writes out a line that contains the name, a comma, the number and follows it by a newline.

The loading portion is a little more complicated. It starts by getting a file object. Then it uses a while True: loop to keep looping until a break statement is encountered. Next it gets a line with the line in\_line = in\_file.readline(). The readline function will return an empty string when the end of the file is reached. The if statement checks for this and breaks out of the while loop when that happens. Of course if the readline function did not return the newline at the end of the line there would be no way to tell if an empty string was an empty line or the end of the file so the newline is left in what readline returns. Hence we have to get rid of the newline. The line in\_line = in\_line[:-1] does this for us by dropping the last character. Next the line name, number = in\_line.split(",") splits the line at the comma into a name and a number. This is then added to the numbers dictionary.

### Advanced use of .txt files

You might be saying to yourself, "Well I know how to read and write to a textfile, but what if I want to print the file without opening out another program?"

There are a few different ways to accomplish this. The easiest way does open another program, but everything is taken care of in the Python code, and doesn't require the user to specify a file to be printed. This method involves invoking the subprocess of another program.

Remember the file we wrote output to in the above program? Let's use that file. Keep in mind, in order to prevent some errors, this program uses concepts from the Next chapter. Please feel free to revisit this example after the next chapter.

import subprocess

def main():

try:

print("This small program invokes the print function in the Notepad application")

#Lets print the file we created in the program above

subprocess.call(['notepad','/p','numbers.txt'])

except WindowsError:

print("The called subprocess does not exist, or cannot be called.")

main()

The subprocess.call takes three arguments. The first argument in the context of this example, should be the name of the program which you would like to invoke the printing subprocess from. The second argument should be the specific subprocess within that program. For simplicity, just understand that in this program, '/p' is the subprocess used to access your printer through the specified application. The last argument should be the name of the file you want to send to the printing subprocess. In this case, it is the same file used earlier in this chapter.