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 `break`s 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.

Exercises

Now modify the grades program from section Dictionaries so that is uses file I/O to keep a record of the students.

Solution

Now modify the grades program from section Dictionaries so that is uses file I/O to keep a record of the students.

```
assignments = ['hw ch 1', 'hw ch 2', 'quiz   ', 'hw ch 3', 'test']
students = { }

def load_grades(gradesfile):
    inputfile = open(gradesfile, "r")
    grades = [ ]
    while True:
        student_and_grade = inputfile.readline()
        student_and_grade = student_and_grade[:-1]
        if not student_and_grade:
            break
        else:
            studentname, studentgrades = student_and_grade.split(",")
            studentgrades = studentgrades.split(" ")
            students[studentname] = studentgrades
    inputfile.close()
    print("Grades loaded.")

def save_grades(gradesfile):
    outputfile = open(gradesfile, "w")
    for k, v in students.items():
        outputfile.write(k + ",")
        for x in v:
            outputfile.write(str(x) + " ")
        outputfile.write("\n")
    outputfile.close()
    print("Grades saved.")

def print_menu():
    print("1. Add student")
    print("2. Remove student")
    print("3. Load grades")
    print("4. Record grade")
    print("5. Print grades")
    print("6. Save grades")
    print("7. Print Menu")
    print("9. Quit")

def print_all_grades():
    if students:
        keys = sorted(students.keys())
        print('\t', end=' ')
        for x in assignments:
            print(x, '\t', end=' ')
        print()
        for x in keys:
            print(x, '\t', end=' ')
            grades = students[x]
            print_grades(grades)
    else:
        print("There are no grades to print.")

def print_grades(grades):
    for x in grades:
        print(x, '\t', end=' ')
    print()

print_menu()
menu_choice = 0
while menu_choice != 9:
    print()
    menu_choice = int(input("Menu Choice: "))
    if menu_choice == 1:
        name = input("Student to add: ")
        students[name] = [0] * len(assignments)
    elif menu_choice == 2:
        name = input("Student to remove: ")
        if name in students:
            del students[name]
        else:
            print("Student:", name, "not found")
    elif menu_choice == 3:
        gradesfile = input("Load grades from which file? ")
        load_grades(gradesfile)
    elif menu_choice == 4:
        print("Record Grade")
        name = input("Student: ")
        if name in students:
            grades = students[name]
            print("Type in the number of the grade to record")
            print("Type a 0 (zero) to exit")
            for i,x in enumerate(assignments):
                print(i + 1, x, '\t', end=' ')
            print()
            print_grades(grades)
            which = 1234
            while which != -1:
                which = int(input("Change which Grade: "))
                which -= 1
                if 0 <= which < len(grades):
                    grade = input("Grade: ") # Change from float(input()) to input() to avoid an error when saving
                    grades[which] = grade
                elif which != -1:
                    print("Invalid Grade Number")
        else:
            print("Student not found")
    elif menu_choice == 5:
        print_all_grades()
    elif menu_choice == 6:
        gradesfile = input("Save grades to which file? ")
        save_grades(gradesfile)
    elif menu_choice != 9:
        print_menu()
```