Lab 8: Files

Feb 20, 2018

Main Event

1. Reading and writing skills

1. Create a Python program that writes ten random numbers between zero and 50 to a file.
2. Create a *second* Python program that opens that same file, sums its values, and prints the results (normally, not necessarily to another file).

Solution:

```
import random

fp = open('ten-randoms', mode='w')
for i in range(10):
    r = random.randrange(1, 100)
    print(r, file=fp)
fp.close()

total = 0
fp = open('ten-randoms')
for i in fp:
    total += int(i)
fp.close()
print(total)
```

2. Persistent phone book

These exercises extends the phone book we made in the last lab:

1. Build a database of names and phone numbers. Continuously ask the user for their friends name and phone number. When the name that is given is the

empty string, stop. Instead of saving each entry into a dictionary, write them to a file—each name/number combination should be a line in the file, with a comma separating the name from the number:

```
Donald Trump,1-202-123-4567
Narendra Modi,91-80-3443-3443
Xi Jinping,86-456-223-3443
```

You can name the file whatever you want.

2. Read in the file and create a dictionary of name/numbers. Remember, split is your friend.

3. Continuously ask the user for a name. If that name is in the dictionary, print the phone number; otherwise, print a notification that it is not. Again, when the empty string is entered, stop the interaction.

Solution:

```
fp = open('friends', 'w')
while True:
    name = input("Enter your friend's name: ")
    if not name:
        break
    number = input("Enter your friend's number: ")
    print(name, number, sep=',', file=fp)
fp.close()

fp = open('friends')
friends = {}
for line in fp:
    parts = line.strip().split(',')
    name = parts[0]
    number = parts[1]

    friends[name] = number

while True:
    name = input("Enter your friend's name: ")
    if not name:
        break

    if name in friends:
```

```
        print(friends[name])
    else:
        print(name, 'is not your friend')
```

3. Read a board from file

Consider a file consisting of a matrix of letters. In such a file the number of letters on a given line would be the columns; the number of lines in the file would be the rows. For example, a 3-row, 4-column matrix would look like the following:

```
ABAB
CDCD
EFEF
```

1. The file board.txt represents such a matrix. Download this file and write a program that puts its contents into a list-of-lists (lines are sublists).
2. Using code from previous work, print the matrix in a grid:

   ```
   A|B|A|B
   -*-*-*-
   C|D|C|D
   -*-*-*-
   E|F|E|F
   ```

   Note that this format is a slight deviation from the past. Updating your logic is good practice.
3. Once you have the matrix in memory, continuously ask the user for coordinates, along with a letter:

   ```
   >>> Please enter two coordinates (row,col): 2,3
   >>> Please enter a letter: P
   ```

   Replace elements of the matrix with the letter entered. In this case, the “F” in the lower right corner of the original board.txt would be replaced with “P”.

   Continue this interaction until the user asks to “stop”. At that point, write the file back to disk using the same file name (board.txt).

If you have done this correctly, you should be able to run the program again and see a new matrix, updated based on the user interaction.

Solution:

```
board_file = 'board.txt'

# 1. Read the board
board = []
fp = open(board_file)
for i in fp:
    line = i.strip()
    board.append(list(line))

# 2. Print to grid
i = 0
while i < len(board):
    row = list(board[i])
    print('|'.join(row))
    if i < len(board) - 1:
        sep = ['-'] * len(row)
        print('*'.join(sep), sep='')
    i += 1

# 3. Coordinate replacement
while True:
    coordinates = input('Please enter two coordinates (row,col): ')
    if coordinates == 'stop':
        fp = open(board_file, 'w')
        for i in board:
            print(''.join(i), file=fp)
        fp.close()
        break
    (row, col) = map(int, coordinates.split(','))

    if 0 < row <= len(board) or 0 < col <= len(board[0]):
        letter = input('Please enter a letter: ')
        if len(letter) == 1:
            board[row][col] = letter
```

Additional Practice

1. Fun with file encryption

A Caesar cipher is an encryption technique that involves replacing letters with other letters a fixed position down the alphabet. For example, a “right shift” of five would turn ‘a’ into ‘f’; a “left-shift” of 3 would turn ‘b’ into ‘y’. Although we now know the Caesar cipher is no match for the NSA, it is still fun to implement.

First, a quick review of ASCII: A widely used character encoding is the American Standard Code for Information Interchange (ASCII). The standard assigns numeric values to characters in the English alphabet; thus, you can use it to treat characters as if they were numbers. Python provides two functions to do ASCII conversion: ord takes a character and returns a number; chr takes a number and returns a character:

```
>>> ord('a')
97
>>> chr(97)
'a'
>>> chr(ord('a')) == 'a'
True
```

ASCII values are sequential: “b” is 98, “c” is 99, and so on. ASCII values range from 0 to 127. If you try to print values outside of this range, you may get an error.

Encrypt/Decrypt

- Download either original-dos.txt or original-unix.txt depending on your operating system (Windows users the former, other OS’s the latter). Read the file, right-shift each character by 17 spaces (ROT17), then write the shifted character to encrypted.txt.
- Read encrypted.txt and left-shift each alphabetic character by 17 positions. Write the output to unencrypted.txt. Ensure that the original and unencrypted files are the same directory.

You only need to shift characters that are letters—do not shift any other character (whitespace or punctunation, for example)! You can easily identify a character in this range by looking at it’s ASCII value. Also you can assume all characters are lower-case.

Check your work

To test whether you have done things correctly by comparing original.txt and unencrypted.txt. In Python, that can be done using the filecmp module:

```
import filecmp
diff = filecmp.cmp('original.txt', 'unencrypted.txt', False)
```

The value of `diff` should be `True`.

Solution:

```
import filecmp

rot = 17

lower_bound = ord('a') - 1
upper_bound = ord('z') + 1
bounds = upper_bound - lower_bound

original = 'original.txt'
encrypted = 'encrypted'
decrypted = 'unencrypted.txt'

# encrypt
infile = open(original)
outfile = open('encrypted.txt', mode='w')
while True:
    c = infile.read(1)
    if not c:
        break

    ascii_value = ord(c)
    if lower_bound < ascii_value < upper_bound:
        scaled_letter = ascii_value - lower_bound       # scale
        rotated_letter = (scaled_letter + rot) % bounds # rotate
        scaled_letter = rotated_letter + lower_bound    # unscale

        c = chr(scaled_letter)
    outfile.write(c)
outfile.close()
infile.close()
```

```
# decrypt
infile = open('encrypted.txt')
outfile = open(decrypted, mode='w')
while True:
    c = infile.read(1)
    if not c:
        break

    ascii_value = ord(c)
    if lower_bound < ascii_value < upper_bound:
        scaled_letter = ascii_value - lower_bound       # scale
        rotated_letter = (scaled_letter - rot) % bounds # rotate
        scaled_letter = rotated_letter + lower_bound    # unscale

        c = chr(scaled_letter)
    outfile.write(c)
outfile.close()
infile.close()

# verify
print(filecmp.cmp(original, decrypted, False))
```

Introduction to Computer Science

Introduction to Computer Science
jerome.white@nyu.edu

jerome-white

Learning computer science concepts through practice.