This is part 2 of working with files. See the previous handout for part 1.

1 Writing to files

Recall that the built-in function `open(filename, mode)` takes two parameters: a filename and a mode. To write to a file, the file mode must be either `'w'` (to open for writing), or `'a'` (to open for appending to what is already in the file).

For example, this statement opens a file called `'Copy of RoadNotTaken.txt'` in writing mode.

```
frost_copy = open('Copy of RoadNotTaken.txt', 'w')
```

If this file does not already exist, then `open` creates the file. However, if the file already exists, then `open` *erases* the contents of the file! If you want to keep the current contents of the file and add to the end of it, then open in `'a'` mode.

We can use the `write` method to write to a file. In the example below, we use it to copy the contents of the `'RoadNotTaken.txt'` to our new file `'Copy of RoadNotTaken.txt'`.

Before copying over the contents, we want to write the word "Copy" as the first line of the file. To do this we must write the string `'Copy\n'` because, unlike `print`, the `write` method does not automatically add the newline.

```
frost_file = open('RoadNotTaken.txt')
frost_copy = open('Copy of RoadNotTaken.txt', 'w')
frost_copy.write('Copy\n')
for line in frost_file:
    frost_copy.write(line)
frost_file.close()
frost_copy.close()
```

Suppose we later decide when want to add the phrase "End of copy" to the end of the file. In this case, we should open the file in append mode, like this:

```
frost_copy = open('Copy of RoadNotTaken.txt', 'a')
frost_copy.write('End of copy\n')
frost_copy.close()
```

2 Exercises

Some solutions are presented in class and also included in the moodle version of this handout. The files used in these examples are available on moodle.

1. Open `'wwf_scores.txt'` and print all lines that contain data – i.e., don't print any line that starts with `'#'`.

Solution:

```
f = open('wwf_scores.txt')
for line in f:
    if not line.startswith('#'):
        print line,
f.close()
```

2. Open `'wwf_scores.txt'`, scan through all of the games, find Nancy's highest score, and print it. (Remember that every line of a file is a string even if it contains digits. For example, Nancy's score of 361 is actually the string `'361'`. You will need to convert it to an `int` to find the maximum.)

Solution:

```
f = open('wwf_scores.txt')
highest = -1
for line in f:
    if not line.startswith('#'):
        data = line.strip().split(',')
        date = data[0]
        nancy = int(data[1])  # be sure to convert to an int!
        michael = data[2]
        if nancy > highest:
            highest = nancy
f.close()
print "Highest score is", highest
```

3. Continuing the previous problem, print the dates of games where Nancy achieved her highest score. The program should print a *single* line, such as

```
Highest score occurred on these dates: 2013-10-20, 2013-10-12.
```

Solution:

```
f = open('wwf_scores.txt')
highest = -1
highest_dates = []

for line in f:
    if not line.startswith('#'):
        data = line.strip().split(',')
        date = data[0]
        nancy = int(data[1])    # be sure to convert to an int!
        if nancy > highest:
            highest = nancy
            highest_dates = []  # throw out old dates
        if nancy == highest:
            highest_dates.append(date)
f.close()
print "Highest score occurred on these dates:",
print ', '.join(highest_dates)
```

4. Continuing the previous problem, in addition to printing each date, write the dates to another file called `'BestGames.txt'`.

Solution:

```
# let's assume highest_dates has the correct
# dates
highest_dates = ['2013-10-20', '2013-10-12']
f = open('BestGames.txt', 'w')
for date in highest_dates:
    f.write(date + '\n')
f.close()
```

5. Modifying problem 3, print the dates of games where the highest overall score is achieved, *regardless of whether the player was Nancy or Michael*. The highest score overall is 474 and the dates are 2013-10-12 and 2013-10-20 and also 2013-10-09.

Solution:

```
f = open('wwf_scores.txt')
highest = -1
highest_dates = []

for line in f:
    if not line.startswith('#'):
        data = line.strip().split(',')
        date = data[0]
        winner = max(int(data[1]), int(data[2]))
        if winner > highest:
            highest = winner
            highest_dates = []  # throw out old dates
        if winner == highest:
            highest_dates.append(date)
f.close()
print "Highest score occurred on these dates:",
print ', '.join(highest_dates)
```

6. Instead of calculating the highest score, open 'wwf_scores.txt' and calculate the average difference in score between Nancy and Michael. For this file, the correct answer is

$$\frac{(361-341)+(440-366)+\ldots+(300-474)}{8} = -2.875$$

indicating that on average Michael beats Nancy by 2.875 points.

Solution:

```
f = open('wwf_scores.txt')
# skip the header
f.readline()
f.readline()
f.readline()

diff = 0
count = 0
for line in f:
    data = line.strip().split(',')
    nancy = int(data[1])
    michael = int(data[2])
    diff += nancy - michael
    count += 1
f.close()

print 'Average difference is', float(diff)/count
```

Some material adapted from Gries and Campbell.