CSCI 1100 — Computer Science 1 Homework 4
Loops and Lists

Overview

This homework is worth **90 points** total toward your overall homework grade (each part is 30 points), and is due Thursday, October 13, 2016 at 11:59:59 pm. There are three parts to the homework, each to be submitted separately. All parts should be submitted by the deadline or your program will be considered late.

The homework submission server URL is below for your convenience:

```
https://submit.cs.rpi.edu/index.php?course=csci1100
```

Your programs for each part for this homework should be named:

```
hw4Part1.py
hw4Part2.py
hw4Part3.py
```

respectively. Each should be submitted separately.

See the handout for Homework 3 for discussion of grading and for a discussion of what is considered excess collaboration. These rules will be in force for the rest of the semester.

Final note, you will need to use loops in this assignment. We will leave the choice of loop type to you. Please feel free to use `while` loops or `for` loops depending on the task and your own, personal preference.

Part 1: Words with alternating vowels and consonants

Write a program that reads in a word entered by the user and checks whether:

- the word has at least 8 characters,
- starts with a vowel,
- has alternating vowels and consonants, and
- the consonants are in increasing alphabetical order.

Note that you can check letters for alphabetical order using `<`.

For example:

```
>>> 'a' < 'b'
True
>>> 'z' < 'b'
False
```

Your program should work for words entered upper or lower case, and must use a single function `is_alternating(word)` that returns `True` if the word has the above pattern, and `False` otherwise. (**Hint.** Remember to write a loop that goes through each letter and check for the necessary

conditions. Using indexing is important here as you need to compare letters in different positions. Here is an example run of this program:

```
Enter a word => eLimiNate
eLimiNate
The word 'eliminate' is alternating
```

```
Enter a word => ageneses
ageneses
The word 'ageneses' is not alternating
```

```
Enter a word => ADAGE
ADAGE
The word 'adage' is not alternating
```

The word `eliminate` has this alternating pattern since we have that: `'l'<'m'<'n'<'t'`.

The word `adage` is not long enough, `ageneses` has two consonants that are identical and as a result not in strictly increasing alphabetical ordering.

Here is a little hint that will help: Given a letter stored in the variable x, the boolean expression:

```
x in 'aeiou'
```

is `True` if and only if `x` is a lower case vowel.

When you have tested your code, please submit it as `hw4Part1.py`. Be sure you use the correct filename, otherwise, Submitty will not be able to grade your submission.

Part 2: New York State Death Statistics

As part of an open government initiative, New York State releases a large number of health related statistics for researchers and developers to use. Among these are death statistics by region of the state at `https://health.data.ny.gov`. For this assignment, we have simplified the statistics to provide just the total number of deaths per 100,000 people. Our goal is to come up with a simple visualization comparing death trends between two different counties over the last few years.

Write a program to read in the death statistics for two different areas of NYS for the 11 years from 2003 to 2013. Remember that these are deaths per 100,000 people. We will take the naive view that areas with lower deaths per 100,000 are automatically healthier places to live, ignoring all other population demographics such as age. Compute the difference between the death rates of the areas for each year, and encode this as a string visualization using the symbols `=`, +, and `-`, where `=` implies the two areas are within +/- 50 deaths per 100,000 for that year, + implies the first area has more than 50 fewer deaths for that year, and `-` implies that the first area has more than 50 more deaths for that year. Compare the number of + and `-` symbols in your string to determine the healthier area with + favoring the first area and `-` favoring the second. Print out the trend line including a header for the data labeling the start and end years – this should be in reverse order from 2013 to 2003 – along with your determination of the healthier area.

The utility module provided for this homework will give you some help. Given a string containing a county name, it provides a function `read_deaths(county)` that returns you a list of 11 numbers. Try the following:

```
import hw4_util
cdata1 = hw4_util.read_deaths('Erie')
cdata2 = hw4_util.read_deaths('Cattaraugus')
print('Erie:', cdata1)
print('Cattaraugus:', cdata2)
```

would give:

```
Erie: [1061.0, 1032.0, 1047.0, 1020.0, 1040.0, 1037.0, 1029.4, 1010.0, 1043.0, 1014.0,
    1046.0]
Cattaraugus: [1005.0, 1089.0, 1061.0, 978.7, 972.7, 978.8, 1010.2, 1083.0, 1002.0,
    977.9, 990.0]
```

The difference (rounded to one decimal) is:

```
[56.0, -57.0, -14.0, 41.3, 67.3, 58.2, 19.2, -73.0, 41.0, 36.1, 56.0]
```

Running the program with these values would give:

```
Enter the first area to check => Erie
Erie
Enter the second area to check => Cattaraugus
Cattaraugus

       2013   2003
Trend: -==+=--==+-

I would rather live in Cattaraugus than Erie
```

Note that 2013 and 2003 are printed to line up with the margins of the trend data.

Here is another example where the number of ”+” and ”-” symbols are the same:

```
Enter the first area to check => Bronx
Bronx
Enter the second area to check => Queens
Queens

       2013   2003
Trend: ===========

Bronx and Queens are the same
```

The function **hw4_util.read_deaths** will return an empty list whenever the county cannot be found. Your program must give an error and exit when it encounters an invalid name as shown below.

```
Enter the first area to check => Errie
Errie
Errie is an invalid name
```

or

```
Enter the first area to check => Erie
Erie
Enter the second area to check => Wesstchaester
Wesstchaester
Wesstchaester is an invalid name
```

You can ignore any warnings you get from using sys.exit() that may show up after your error message.

When you have tested your code, please submit it as `hw4Part2.py`. Be sure to use the correct filename or Submitty will not be able to grade your submission.

Part 3: Pokemon GO

This is a little variation on the turtle walk we did last homework. This time, we place a Pokemon trainer at the center of an 11x11 grid at position (5,5). As in the previous homework (0,0) is in the upper left corner and (10,10) is in the lower right corner. The grid is populated by Pokemon and your job is to guide the pokemon trainer around the grid gathering up any Pokemon they find.

You will need to read the list of Pokemon and their locations on the grid using the `read_pokemon` function from `hw4_util`. By now you should be experts at this. For example, the program:

```
import hw4_util
pokemon, locations = hw4_util.read_pokemon()
print(pokemon)
print(locations)
```

Will produce:

```
['Pikachu', 'Bulbasaur', 'Charizard', 'Squirtle']
[(2, 3), (4, 1), (6, 3), (2, 4)]
```

The first list is the pokemon that are available on this particular grid, while the second list is their locations. You may assume without checking that no locations are repeated in this list. The pokemon trainer collects pokemon by walking around the grid one step at a time. The trainer can move in one of 4 directions N (up), S (down), E (right) and W (left). When the trainer lands on the same space as a pokemon, the pokemon is considered captured. We won't require you to animate the pokemon battles for this homework. This simulation has an indeterminate number of steps driven by user input.

The Program

This is a turn based simulation driven by user input. To set up the simulation, first use `read_pokemon` to load in the pokemon data. Immediately print out the list of Pokemon and their locations using a function `print_pokemon` that you will need to write. The function should only print out pokemon that have not been captured by the trainer. This will allow you to use it at additional places in the simulation.

Once you have the simulation set up you enter a loop. At the start of each iteration through the loop begin by asking the user for a command:

1. **N** moves up one step
2. **S** moves down one step
3. **E** moves right one step
4. **W** moves left one step
5. **print** print all active (uncaptured) pokemon and their locations
6. **end** stop the simulation and end the program

After each command other than end, print out the current location of the trainer and report if the trainer has captured a pokemon. A sample run is below:

```
Current pokemon:
     Pikachu at (2, 3)
     Bulbasaur at (4, 1)
     Charizard at (6, 3)
     Squirtle at (2, 4)

N,S,E,W to move, 'print' to list, or 'end' to stop ==> N
N
Currently at (5, 4)
N,S,E,W to move, 'print' to list, or 'end' to stop ==> n
n
Currently at (5, 3)
N,S,E,W to move, 'print' to list, or 'end' to stop ==> w
w
Currently at (4, 3)
N,S,E,W to move, 'print' to list, or 'end' to stop ==> W
W
Currently at (3, 3)
N,S,E,W to move, 'print' to list, or 'end' to stop ==> w
w
Currently at (2, 3)
You capture a Pikachu on turn 4
N,S,E,W to move, 'print' to list, or 'end' to stop ==> print
print
Current pokemon:
     Bulbasaur at (4, 1)
     Charizard at (6, 3)
     Squirtle at (2, 4)

Currently at (2, 3)
N,S,E,W to move, 'print' to list, or 'end' to stop ==> end
end
```

Or a second example:

```
Current pokemon:
     Pikachu at (2, 3)
     Bulbasaur at (4, 1)
     Charizard at (6, 3)
     Squirtle at (2, 4)
```

```
N,S,E,W to move, 'print' to list, or 'end' to stop ==> w
w
Currently at (4, 5)
N,S,E,W to move, 'print' to list, or 'end' to stop ==> w
w
Currently at (3, 5)
N,S,E,W to move, 'print' to list, or 'end' to stop ==> w
w
Currently at (2, 5)
N,S,E,W to move, 'print' to list, or 'end' to stop ==> w
w
Currently at (1, 5)
N,S,E,W to move, 'print' to list, or 'end' to stop ==> w
w
Currently at (0, 5)
N,S,E,W to move, 'print' to list, or 'end' to stop ==> w
w
Currently at (0, 5)
N,S,E,W to move, 'print' to list, or 'end' to stop ==> prrrint
prrrint
Currently at (0, 5)
N,S,E,W to move, 'print' to list, or 'end' to stop ==> print
print
Current pokemon:
     Pikachu at (2, 3)
     Bulbasaur at (4, 1)
     Charizard at (6, 3)
     Squirtle at (2, 4)

Currently at (0, 5)
N,S,E,W to move, 'print' to list, or 'end' to stop ==> end
end
```

Some notes:

1. Be careful not to let the trainer walk off the field
2. Make sure that your commands are case insensitive
3. We will use a different pokemon file when we test on Submitty, so make sure you test thoroughly
4. If you get an invalid command (such as prrrint in the second example) just ignore it, print the current position, and start the next turn

When you have tested your code, please submit it as `hw4Part3.py`. You must use this filename, or Submitty will not be able to run and grade your code.