THE UNIVERSITY OF ARIZONA.
DEPARTMENT OF COMPUTER SCIENCE

CSc 120: Fall 2019

Assignment 01: Part 2

Start: Wed Aug 28, 2019
Due: 8:00 PM, Thu Aug 29 2019

Important: Follow the directions given *exactly*: your code will be graded by a software script, so any deviation from the program specification can result in a significant loss of credit. If you are unsure about something, ask for clarification in office hours or Piazza (however, see the class's academic integrity policy about sharing your code or using code written by others).

Please pay attention to the programming style guidelines for this class. For this assignment you will be notified of style violations but not penalized for them; style violations will be penalized in subsequent assignments.

Note
There are two problems in this assignment, which together require that you submit two files: **swap.py** (problem 1) and **population.py** (problem 2). Because of the way the auto-grader script works, in order to get full credit for your work you have to submit *both* these files each time you want to resubmit a solution to either problem.

Problem 1.

Write a program, in a file named **swap.py**, that behaves as follows:

1. it reads in a string using the Python statement

   ```
   input("input: ")
   ```

2. if the length of the string read in is even, it constructs a new string by swapping the first and second halves of the string.
3. if the length of the string read in is odd, then let the first part of the string be the portion of the string from the beginning up to, but not including, the middle character; and the last part of the string be the portion of the string that starts after the middle character and goes all the way to the end. Your program constructs a new string by swapping the first and last parts of the string but leaving the middle character in the same position.
4. It prints out the new string constructed.

Examples:

- If the input string is **abcd** the output should be **cdab**
  - In this case, the length is 4, which is even. The first half of the input string is **ab** and the second half is **cd**. Swapping these, we get the output string **cdab**.
- If the input string is **12345** the output should be **45312**
  - In this case, the length is 5, which is odd. The program behaves as follows:
    - the middle character of the input string is **3**.

- the first part of the string is the character sequence starting from the beginning and going up to, but not including, the middle character. In this case, this is **12**.
- the last part is the character sequence starting just after the middle character and going all the way to the end. In this case, this is **45**.
- The program swaps the last part (**45**) and the first part (**12**) and leaves the middle character **3** in the same position. So the new string is **45312**.

Problem 2.

This problem involves the analysis of state-level population data; it is based on data available in https://simple.wikipedia.org/wiki/List_of_U.S._states_by_population.

Write a program, in a file named **population.py**, as specified below.

1. Write a function **sum_pop(fobj)** that takes a single argument **fobj** that is a file object. (A file object is the value returned by **open( )**). This function behaves as follows:
   a. for each line in **fobj**, it extracts the population figure from each line (see below under "**Input file format"** for the format for each line), and computes the sum of all the population figures so obtained;
   b. it returns the sum of the population figures so computed.
2. Write a function **main()** that behaves as follows:
   a. it reads in a string using the Python statement

      ```
      input("file: ")
      ```

   b. it opens the resulting file for reading;
   c. it calls the function **sum_pop()**, passing the file object obtained in the previous step as an argument;
   d. it prints out the value returned by **sum_pop()** using the statement

      ```
      print(value_returned)
      ```

Input file format

The input file is a text file where each line consists of the name of a US state or territory together with its estimated 2019 population, e.g.:

```
Oklahoma           3943079
Oregon             4190713
South Dakota           882235
Pennsylvania           12807060
Northern Mariana Islands   55194
```

You can assume that the file does not contain any blank lines.

Programming hints and gotchas to watch for:

- Consider using **split()** to break input lines into its components.
- Note that, as in the example above, some state/territory names may consist of more than one word. The population will always be the last entry of each line.
- Remember that input values read from a file are strings; in order to do arithmetic on them you will have to convert them to numbers where necessary.

Here are a few input files you can use for trying out your code (feel free to also create your own from this master data set):

File name	Total population
pop1.txt	87969191
pop2.txt	48488880
pop3.txt	40065652
pop4.txt	88130818
pop5.txt	35825447