CS 30 Lab 3 — Manipulating Strings in Python

This lab lets you practice using strings in Python. You are encouraged to work with a partner, and to talk things over with your lab-mates. Don't hesitate to call me or Dan over for help or answers to questions.

---

1. Write a program that counts the number of words in a sentence entered by the user. Hints: Use the string library function `string.split()` to break up a sentence into a list of its individual words. You can then use the built-in `len` function to find out the length of the list. Also, don't forget to use `raw_input` instead of `input` when asking the user for a sentence.

2. Write a program that calculates the average word length in a sentence entered by the user.

3. The built-in Python function `ord` can be used to convert a single letter or other character into an equivalent numeric value called that character's *ASCII code*, which is just an integer in the range 0-126. Similarly, the function `chr` converts an ASCII code number back into the equivalent character. This technique can be used to encode and decode messages as lists of numbers.

   Work through sections 4.4.2 and 4.4.3 of your textbook (pages 89-94) by typing in the example programs `text2numbers` and `numbers2text`. Test out the programs in Python and make sure you understand how they work.

4. Write a program called `grade()` that accepts a number in the range 0-12 as input and prints out the corresponding letter grade based on Pomona's 12-point grading scale. To do this, keep track of the possible letter grades in a list, where the position of a grade in the list corresponds to its 12-point value:

   ```
   letterGrades = ['F', 'F', 'D-', 'D', 'D+', 'C-', 'C', 'C+',
                   'B-', 'B', 'B+', 'A-', 'A']
   ```

   Use the number entered by the user as an index into this list in order to retrieve the appropriate grade.

5. Do problem 6 on page 119 of the textbook. You'll have to convert upper or lower case letters in the range a to z into numbers in the range 1 to 26. One way is to first convert the letter to uppercase using `string.upper()`, then convert that to an ASCII code using `ord`, and then subtract 64, since the ASCII code for A is 65, B is 66, C is 67, and so on.

Zip Codes

For faster sorting of letters, the US Postal Service encourages companies that send large volumes of mail to use a bar-code encoding of zip codes. For example, a typical address label is shown below:

```
* * * * * * * * * * ECRLOT * * CO57
CODE C671RTS2
JOHN DOE                CO57
1009 FRANKLIN BLVD
SUNNYVALE      CA 95014

||.|...|.|.||......||.|..|...|||
```

(continued on back)

Each zip code digit is encoded as a sequence of five tall or short bars (we'll represent the short bars as periods):

```
 0  ||...    2  ..|.|    4  .|..|    6  .||..    8  |..|.
 1  ...||    3  ..||.    5  .|.|.    7  |...|    9  |.|..
```

In addition, there are full-height frame bars on each end of the bar code, plus an extra correction digit which is computed as follows: Add up all digits, and choose the correction digit to make the sum a multiple of 10. For example, the zip code 95014 has sum of digits 19, so the correction digit is 1 to make the sum equal to 20.

The complete code for 95014 is shown below, with spaces inserted to make it easier to see the individual digit codes and the frame bars:

```
|  |.|..  .|.|.  ||...  ...||  .|..|  ...||  |
```

Your job is to write a program that asks the user for a zip code and prints out the corresponding bar code. You should represent zip codes as strings instead of numbers so that zip codes with leading zeros won't cause problems. Here is an outline of what your program should do:

1. Get a zip code string from the user. Hint: Use `raw_input` instead of `input`.

2. Create a list of integers containing the individual digits of the zip code. Hints: Use a for-loop to loop through the individual digits in the zip code string, appending them to your list one at a time. In the process, you'll also need to convert each digit from a string to an integer using Python's `int` function (the `eval` function will work too).

3. Using another for-loop, build up a second list consisting of the individual bar codes corresponding to each of the digits in the first list. Hint: Store the bar codes for 0-9 in a list called `table`, as shown below, and then index into this table using the digits.

```
table = ["||...", "...||", "..|.|", "..||.", ".|..|",
         ".|.|.", ".||..", "|...|", "|..|.", "|.|.."]
```

4. Compute the correction digit. To do this, first add up all of the digits in the zip code using another for-loop. Call this value *sum*. To compute the correction digit, use the following Python formula:

$$(10 - (sum \% 10)) \% 10$$

   The % operator divides one value by another and returns the remainder (it has nothing to do with percentages). For example, 17 % 5 is 2, since 5 goes into 17 3 times with 2 left over.

5. Append the bar code for the correction digit to the list of bar codes created in step 3.

6. Build up the complete barcode string from the list of individual digit bar codes.
   Hint: Use another for-loop.

7. Add a frame bar `|` to each end of the string and print the final result.