ECE-203 Programming for Engineers
Homework Week 3
Due Friday, Feb 3rd to BBLearn

(50 Points) The last digit of a credit card number is the *check digit*, which protects against transcription errors such as an error in a single digit or the transposition of two adjacent digits. The following method is used to verify actual credit card numbers but (for simplicity) we will describe it for a number with 8 digits instead of 16.

- Starting form the rightmost digit, form the sum of every other digit.
  - Example: if the card number is 4358 9795, then you form the sum: $5 + 7 + 8 + 3 = 23$
- Double each of the digits that were not included in the previous step. Add all the **digits** of the resulting numbers.
  - Example: for the card number 4358 9795, the not included digits were [9, 9, 5, 4]. Doubling these leads to [18, 18, 10, 8]. Summing the **digits** results in:
    $1 + 8 + 1 + 8 + 1 + 0 + 8 = 27$
- Add the sums from the previous two steps. If the last digit of the sum is zero, then the credit card number is valid.
  - Example: for the number 4358 9795, this step gives $23 + 27 = 50$. The last digit of 0 means the number is valid.

Write a program that accepts a number from the user via a prompt and perform the above algorithm to check if the number is valid or not. If the number is valid, print a message to the screen informing the user. Your program should be capable of performing the algorithm on a number consisting of any number of digits.

(*Hint: When isolating individual digits, it is useful work on the number as a string, which is just a sequence of characters. When performing arithmetic, you will need to convert characters into integers. Remember, the function `int()` is used to convert floats or strings containing number characters into integers. Likewise, the function `str()` is used to convert ints or floats into a string of number characters.*)

The output of your program should display the credit card number entered and its validity status. For example:

```
43589795 is a VALID credit card number.
```

Test your program with the following numbers (spaced provided simply to make reading easier, your program does not need to handle spaces.):

```
4358 9795
5979 8534 3339 4390
4561 3256 2991 3234
2349 1291 2347
3219 3231 3999
4092 6117 5811 4081
7591 7413 3009 6174 6711 9469 0997 4528
3432 1112 3223 0921
```

(*Hint: Using just what has been covered so far in lecture, this can be accomplished in as little as 14 lines of code.*)

Name your program **ccverify.py** and save its output as **ccverify.txt**. The output should show your interaction with the program testing the 8 provided test numbers. The easiest way to save this output is to simply highlight, copy, and paste into a plaintext file.

Submitting Your Assignment

Create a zip file with the following structure:

```
HW3_abc123.zip
├─ ccverify.py
└─ ccverify.txt
```

name your zip file using your Drexel user id. For example: `HW3_abc123.zip`. Upload your zip file to `learn.drexel.edu` using the assignment submission link found on the course website.

Points will be deducted for improperly packaged submissions.