|  |  |  |  |
| --- | --- | --- | --- |
| **Instructor** |  | **Due Date** |  |

**PROJECT Using Functions : Luhn’s Formula for Credit Card Validation**

**Objective** To write a program that will validate credit card numbers using Luhn’s Method.

***PROJECT DESCRIPTION***

Construct a program that prompts the user to enter a credit card number as an integer. Then, display whether the number is valid or invalid.

You can use the following functions to design your program to:

**# if the card number is valid, this function returns true**

**def isValid(number) :**

**# obtain the result from Step 2**

**def sumOfDoubleEvenPlace(number) :**

**# if variable number is a single digit, then return the number**

**# otherwise, return number as the sum of the two digits**

**def getDigit(number) :**

**# this function returns the sum of odd place digits in variable number**

**def sumOfOddPlace(number) :**

**# if the digit d is a prefix for variable number, then return true**

**def prefixMatched(number, d) :**

**# this function returns the number of digits in variable d**

**def getSize(d) :**

**# returns the first k number of digits from variable number but if the**

**# number of digits in variable number is less than k, return number**

**def getPrefix(number, k) :**

***Information about This Project***

**[ Financial: Credit Card Number Validation: The Luhn’s Formula ]**

Credit card numbers follow these patterns: they must have between 13 and 16 digits, and the

number must start with a certain digit or digits.

• American Express credit cards begin with the number 37

• Discover cards begin with the number 6

• MasterCard credit cards begin with 5

• Visa cards begin with 4

**PROJECT Using Functions : Luhn’s Formula for Credit Card Validation**

Luhn’s Method is a procedure for validating credit card numbers. This method is useful to determine whether a card number is entered or scanned correctly.

Luhn’s Method uses a type of MOD 10 operation and is useful for generating valid credit card, as outlined below.  
  
 *sample card number* 4388 5760 1840 2626

**(1)** Double every second digit from right to left. If the doubling of a digit results in a two - digit number, sum the two digits to obtain a   
 single - digit number.

**(2)** Now add all single - digit numbers from Step 1 .

4 + 4 + 8 + 2 + 3 + 1 + 7 + 8 = 37

**(3)** Add all digits in the odd places from right to left in the card number.

6 + 6 + 0 + 8 + 0 + 7 + 8 + 3 = 38

**(4)** Sum the results from Step 2 and Step 3 .

37 + 38 = 75

**(5)** If the result from Step 4 is divisible by 10 , the card number is valid; otherwise, it is invalid. For example, the number 4388576018402626 is invalid, but the number 4388576018410707 is valid.

***Steps to Complete This Project***

**STEP 1**  **Open the Python IDLE, MS Visual Studio or VSCode**

Open the Python IDLE IDE ( Integrated Development Environment ) , VSCode or similar Python development environment on your computer.

**[ MS Visual Studio ]**

Open Microsoft Visual Studio, click [ Create a new project ] and at the next window, highlight the [ Python Application ] template and click on [ Next ] .

In the **Configure your new Project** dialog box, add in a desired Project name like Lab2 and check where is says ‘Place solution and project in the same directory’. Press [ Create ] to finish and proceed to the newly created Python source file, where you will write your program code.

**[ Python IDLE ]**

You will notice when you initially open Python, the default is an interpretive shell allowing only for single commands to be given. You really need to enter in a whole program then execute it to work any of the labs for the course. To start entering code into IDLE go to **File > New File** from your menu. This will allow you to enter your source code in an editor style format like Notepad.

**STEP 2**  **Write the Program Code for this Application**

Write your program statements that will satisfy the **Project Description** for this application.

Utilize the following steps to assist you to validate a user's credit card.

Use a sample 16 - digit combination for a credit card number other than what is provided in the example steps below.

**PROJECT Using Functions : Luhn’s Formula for Credit Card Validation**

**(1)** **Double the Value of Alternating Digits**

The first step is to double each of the alternating digits in the number.

But the trick is to start with the second digit from the right and work backwards.

Let us say we have a credit card number 1234 5678 1234 5670 .

We will start with the rightmost number 7 , double it, and then do the  
 same for every other digit.

**card #** **1234 5678 1234 5670**

This will give us the following values.

7 × 2 = 14 , 5 × 2 = 10 , 3 × 2 = 6 , etc.

**(2)** **Add the Separate Digits of all the Products**

Now we will total the separate digits of all the products and arrive at with a final sum.

(1 + 4) + (1 + 0) + 6 + 2 + (1 + 4) + (1 + 0) + 6 + 2 = 28

Be sure to add the digits, not just the number.

**(3)** **Add the Unaffected Digits**

We will now go back to the original number and add all the digits that we did not double.

We will still start from the right, but this time we will start from the rightmost number.

**card #** **1234 5678 1234 5670**

0 + 6 + 4 + 2 + 8 + 6 + 4 + 2 = 32

**(4)** **Add the Results and Divide by 10**

Finally, let us add both the results from steps **(2)** and **(3)** above and divide the answer by 10 .

28 + 32 = 60

60 is evenly divided by 10 , thus the credit card number is well formed and ready for further processing.   
  
 Issue a display statement if the card number checked is well - formed or not.

**PROJECT Using Functions : Luhn’s Formula for Credit Card Validation**

**STEP 3**  **Run and Test Your Program**

With your application compiling correctly and without any errors, run your program.

**[ MS Visual Studio ]**

To run your program, use [ Ctrl ] + [ F5 ] .

**[ Python IDLE ]**

To run your program, go to your menu and choose **Run > Run Module** or press [ F5 ] on your keyboard to run your program.

With your application compiling correctly and without any errors, run your program.

Test your program with each of these scenarios.

*a valid credit card number*

*an invalid credit card number*

**STEP 4**  **Submit Your Program**

Take a screen snapshot for each of the above scenarios and submit the snapshots for credit in an MS Word or equivalent document. Place your program source code into the same MS Word document and submit the document for credit into Blackboard in the appropriate area. Also please include your .py file as well.

**STEP 5 Questions and Answers Concerning this Computer Laboratory Project**

Answer the following questions in your own words.

Open MS Word and, within a new document, place your responses to each of these questions. Submit your completed MS Word document for credit.

1. Explain how placing your code into a separate module may benefit other python files involving credit card transactions.

**(2)** Investigate the link below. What are some other methods you can use in your project to verify a credit card transaction? Include at least three methods that may help validate user input of credit card data.

|  |
| --- |
| <https://www.codeproject.com/articles/2782/credit-card-validator-control-for-asp-net> |

**(3)** How would the use of regular expressions aid in reducing any bloated code to verify credit card data within an application? Give an example.

**(4)** Describe some ways to ultimately check your code to validate various card numbers for any faulty entry and / or invalid card numbers or even valid card numbers.

**(5)** Why would there be a reason to store any invalid numbers that were input by a particular user?