



**CREDIT CARD VERIFIER**

**By**

**Valery Aime Nundoya (B0099110)**

***School of Informatics***

***Department of Informatics and Engineering***

***Institute of Technology, Blanchardstown***

***Dublin 15.***

**Actual Words**  **Bachelor of Science Computing in**

**Information Technology**

**21/04/2019**

SECURE COMMUNICATION

Table of Contents

[Introduction 2](#_Toc6755827)

[Task and Function 2](#_Toc6755828)

[Authentication 3](#_Toc6755829)

[BankCard 4](#_Toc6755830)

[main Method 5](#_Toc6755831)

[Conclusion 6](#_Toc6755832)

[References 7](#_Toc6755833)

# Introduction

This is a credit card verifier python challenge in which, the code to be written, has to take a credit card number as an input and output check if it is valid or not.

Literally this process from my research, happens to be the explanation of Luhn’s Methodology of Bank Card. (Luhn algorithm, n.d.) The Luhn algorithm was developed by German computer scientist Hans Peter Luhn in 1954. What the algorithm does is to calculate simple checksum formula used to validate identification numbers such as credit card numbers. Actually, it was designed to protect against accidental errors, such as a digit mistyping etc. (Payment card number, n.d.)

From (Coding Horror, n.d.) checksums are some kind of analogous to filesystem fingerprints which no two should ever be the same, and any modification to the file should change the checksum.

## Task and Function

A simple Card has 16-digit numbers

In a card, the first digit is list [0] and the last card is [15].

Literally, what happens here is, the loop will go through the list from right to left meaning, from list [15] to list [0].

The loop will then check that a position is an even number and if it’s even, it will multiply the value at that position by 2.

So, if the result of that multiplication is > 9, it will then add all the result and the values at odd positions. Below is a brief table explanation of what happens in the process:

“””

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 (index numbers for credit card)  
4 8 4 7 3 5 2 9 8 9 2 6 3 0 9 4 (credit card number)  
  
sum of odd placed numbers from right to left  
4 + 0 + 6 + 9 + 9 + 5 + 7 + 8 = 48 (for odd placed digits)  
  
sum of even placed numbers from right to left  
9 3 2 8 2 3 4 4  
  
double every even placed number (i.e multiply by 2)  
9 \* 2 3 \* 2 2 \* 2 8 \* 2 2 \* 2 3 \* 2 4 \* 2 4 \* 2  
18 6 4 16 4 6 8 8  
  
for every double digit that is arrived at, add the two digits  
1 8 6 4 1 6 4 6 8 8  
1+8 6 4 1+6 4 6 8 8  
9 + 6 + 4 + 7 + 4 + 6 + 8 + 8 = 52 (for even placed digits)  
  
add sum of odd and even placed digits together  
48 + 52 = 100  
  
for luhn check:  
 100 % 10 == 0 (valid)  
else  
 (invalid)  
"""

## Card Authentication

**import** random *#import library for ramdom number generation***class** Authentication: *# class create* **def** \_\_init\_\_(self): *# inititialsation method* self.sum = 0 *# assign zero to sum* **def** verify\_card(self, card\_num): *# method to verify card* total\_even\_i = 0 *#declare and initialize even numbers placements* total\_odd\_i = 0 *# declare and initialize oadd numbers placements* str\_list = str(card\_num) *# assign card numbers  
  
 #loop to look through the length of card* **for** i **in** range((len(str\_list) - 1), -1, -1):  
 **if** i % 2 == 0: *# checkfor even number placements* num\_even\_i = int(str\_list[i]) *# assign card numbes* num\_even\_i = num\_even\_i \* 2  
 *# check for even number placements* **if** num\_even\_i > 9:  
 num\_even\_i = (num\_even\_i - 9) *# assign even numbers* total\_even\_i += num\_even\_i *#add to variable* **else**:  
 total\_odd\_i += int(str\_list[i]) *#add up all odd number placements after converting to integer* self.sum = total\_even\_i + total\_odd\_i *# add together even and off number placements* **if** self.sum % 10 == 0:  
 **return "Valid"** *# return valid card* **else**:  
 **return "Invalid"** *# esle return invalid card  
  
 # checksum method here* **def** get\_checksum(self, first\_portion):  
 first\_portion = str(first\_portion) *# assign first didgit of card* valid\_card\_lenght = 16 *# assign card length to 16  
 #loop through card length minus the first digits* **for** x **in** range((valid\_card\_lenght-1) - len(first\_portion)):  
 other\_portion = str(random.randint(0, 9))*#generate random number for the rest of the digits* first\_portion += other\_portion *#add the two parts of the digits together* self.verify\_card(int(first\_portion)) *#verification of card type and validity* **if** self.sum % 10 == 0: *#if modulus 10 results in 0* checksum = 0 *#checksum is 0* **else**:  
 checksum = 10 - (self.sum % 10) *#subtract modulus result of sum of card digits from 10* full\_card = first\_portion + str(checksum) *#append digits together* print(**"Generated Card: "**, full\_card) *#card is generated* print(self.verify\_card(full\_card)) *#card is verified* **return** checksum checksum **is** returned

## BankCard

**from** authentication **import** Authentication *#athentication library import  
  
#class***class** BankCard:  
 **def** \_\_init\_\_(self, card\_num):*# initialization* self.card\_num = card\_num *#assign card number* self.vendor = self.get\_vendor(self.card\_num) *#assign vendor fro calling get\_vendor method  
#method* **def** get\_vendor(self, card\_num):  
 validity = Authentication().verify\_card(card\_num) *#call verify card method fron Authentication class and assign it to variable  
 #check for validity* **if** validity == **'Valid'**:  
 first\_digit = str(card\_num)[0] *#assign first digits of card number* industry = **''** *#declare which type of industry card belongs to* **if** first\_digit == **'1' or** first\_digit == **'2'**: *#if first digit of card is 1 or 2* industry = **'Airline'** *#it is in airline industry* **elif** first\_digit == **'3'**: *#if digit is 3* industry = **'Travel & Entertainment'** issuer = **'American Express'** *#amex card* **elif** first\_digit == **'4' or** first\_digit == **'5' or** first\_digit == **'6'**: *#if digits are 4 or 5 or 6* industry = **'Banking'** *# it is banking industry* **if** first\_digit == **'4'**: *#if digit is 4* issuer = **'Visa'** *#card is VISA* **elif** first\_digit == **'5'**: *#if digit is 5* issuer = **'MasterCard'** *#it is Mastercard* **else**:  
 issuer = **'Discover'** *#any other digit is Discover* **return '\tIndustry: {0} \n\tIssuer: {1}'**.format(industry, issuer)  
 **else**:  
 self.vendor = **"Invalid card has no Vendor"** *#otherwise card has no vendor* **return** self.vendor  
  
 **def** get\_card\_details(self):  
 **return "\nCARD DETAILS\nCard No.: {0} \nVendor Info: \n{1}"**.format(self.card\_num, self.vendor)

## Card main Method

**from** bankcard **import** \* *# import bankcard class***from** authentication **import** \* *# import authentication library  
#main method***def** main(): *#sample bank card numbers to test* card\_num\_list = [4319473018453644, 4319550212629259, 4319327412228129, 4319473018453644]  
 cnum = input(**"Please enter the card number"**)  
 print(cnum)  
 card\_num = card\_num\_list[3] *#assign list to card\_num* card\_num=int(cnum)  
 auth = Authentication() *#create a variable to call on Authentication class* card = BankCard(card\_num) *#create a variable to call on Bankcard class  
  
  
 # 1)Verify  
 # assign verification to variable* verify = auth.verify\_card(card\_num)  
 print(verify)  
  
  
 *#2) Vendor: Displaying Vendor information  
 #assign verification of vendor card details* details = card.get\_card\_details()  
 print(details) *#print vendor card details  
  
  
 #3) Calculate checksum* portion = input(**"Please enter the first "**) *#assign call to checksum method* checksum=str(fportion)  
 checksum = auth.get\_checksum(fportion)  
  
 print(**"Checksum: "**, checksum) *#print checksum  
  
  
  
 # 4) Generate random valid card* vendor = **"MasterCard"** *# declare and initialize vendor variable* **if** vendor == **'Visa'**: *# check if vendor variable is assigned VISA* first\_digit = **'4'** *# check if first digit contains 4* **elif** vendor == **'MasterCard'**: *#check for Mastercard* first\_digit = **'5'** *#check if first digit is 5* checksum = auth.get\_checksum(first\_digit) *#call checksum method and assign result to checksum* print(**"Checksum: "**, checksum) *#print out result of checksum  
  
#call main method***if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

# Conclusion

Going through this Credit Card Verifier lab was absolutely a very good thing to get knowledge some about. As a Digital Cyber security specialized I think it is important to always verify Credit Cards, to know if they are valid or not. Credit to German computer scientist Hans Peter Luhn for such a brilliant luhn algorithm on how to calculate simple checksum.

# References

*Coding Horror*. (n.d.). Retrieved from https://blog.codinghorror.com: https://blog.codinghorror.com/checksums-and-hashes/

*Luhn algorithm*. (n.d.). Retrieved from https://planetcalc.com: https://planetcalc.com/2464/

*Payment card number*. (n.d.). Retrieved from https://en.wikipedia.org: https://en.wikipedia.org/wiki/Payment\_card\_number