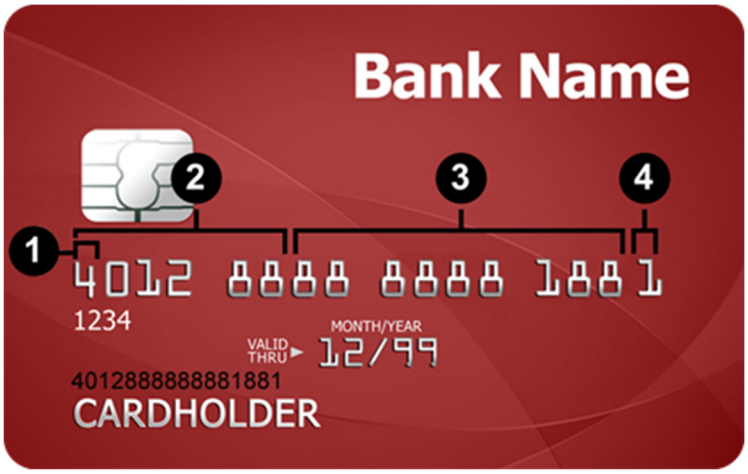
Validate a Credit Card using Python

The Luhn Algorithm

Credit card numbers may look random, but there is actually a hidden meaning behind each group of numbers.



In the above diagram:

1. **Major Industry Identifier (MII)** — identifies the industry of the card.
2. **Issuer Identification Number (IIN)** — identifies the issuer of the card. American Express starts with 34 or 37, MasterCard starts with 2221–2720 or 51–55, Visa starts with 4. This is especially useful for future updates if card issuers ever decide to expand their IIN ranges.
3. **Account Number** — identifies the customer's account
4. **Checksum — makes** sure that the account number is valid

The **Luhn Algorithm** determines the validity of a card using the account number and checksum (labels 3 and 4):

1. From the rightmost digit of your card number, double every other digit.
2. If the doubled digit is larger than 9 (ex. 8 * 2 = 16), subtract 9 from the product (16–9 = 7). OR add the resultant two digits together (16 → 1+6 = 7)
3. Sum the digits.
4. If there is no remainder after dividing by 10 (sum % 10 == 0), the card is valid.

Using the card from above, here is the Luhn Algorithm in action:

Card #	4	0	1	2	8	8	8	8	8	8	8	8	1	8	8	1
x2	8	0	2	2	16	8	16	8	16	8	16	8	2	8	16	1
x2 - 9	8	0	2	2	7	8	7	8	7	8	7	8	2	8	7	1

Summing up the last row gives us a value of 90, which is a multiple of 10. This card is valid!

An Algorithm

The first step in writing a program to validate a card is to design an algorithm, which is a procedure, or set of steps, that could be followed to solve the problem.

1. We'll need to get the **user to enter a card number**.
2. We need to **remove any spaces or dashes** "-".
3. Since the digits are doubled from the right hand side, and VISA/MasterCard have 16 digits, Amex has only 15 digits, hence, we should **reverse the order of all of the digits**.
4. Double every second digit. If the resulting number is greater then 9, either subtract 9 or add the two resultant digits together.
5. Sum all of the digits together and divide by 10.

```
# Python credit card validator program
#
# 1. Remove any '-' or ' '
# 2. Reverse the credit card digits - easier to work with.
# 3. Add all digits in the odd places from right to left
# 4. Double every second digit from right to left.
#      (If result is a two-digit number, subtract 9 to get a
#       single digit.)
# 5. Sum the totals of steps 2 & 3
# 6. If sum is divisible by 10, the credit card # is valid

sum_odd_digits = 0
sum_even_digits = 0
total = 0

# Step 1
card_number = input("Enter a credit card #: ")
card_number = card_number.replace("-", "")
card_number = card_number.replace(" ", "")
print(f"You have entered {card_number}.")

# Step 2
card_number = card_number[::-1] # <-- use indexing operator

# Step 3 - adds up the ODD placed digits
for x in card_number[::2]: # <-- iterates over the string
    # sum_odd_digits = sum_odd_digits + int(x)
    sum_odd_digits += int(x) # <-- better

# Step 4 - doubles, then adds the EVEN placed digits
for x in card_number[1::2]: # <-- iterates from position 2
    x = int(x) * 2
    if x > 9:
        sum_even_digits += (x-9)
```

```
    else:
        sum_odd_digits += x

# Step 5 - find the total
total = sum_odd_digits + sum_even_digits

# Step 6 - establish if the result is divisible by 10
if total % 10 == 0:
    print("The card is VALID")
else:
    print("The card is INVALID")
```