HACKER\_RANK – PYTHON

Regex Substitution - Validating Credit Card Numbers

Words Score - Default Arguments

\*Lò Minh Thuận/B2D10

# Task 1. Regex Substitution(Thay thế văn bản)

Công cụ re.sub() (sub là viết tắt của sự thay thế) đánh giá một mẫu và đối với mỗi kết quả khớp hợp lệ, nó sẽ gọi một phương thức. Phương thức này gọi cho tất cả các kết quả khớp và có thể được sử dụng để sửa đổi chuỗi theo nhiều cách khác nhau. Phương thức re.sub() trả về chuỗi đã sửa đổi làm đầu ra.

Ex:

import re

# Squaring numbers

def square(match):

    number = int(match.group(0))

    return str(number\*\*2)

print(re.sub(r"\d+", square, "1 2 3 4 5 6 7 8 9"))

Đoạn mã Python trên sử dụng thư viện re (Regular Expressions) để thực hiện việc tìm và thay thế các số trong chuỗi đầu vào bằng bình phương của chúng.

Cụ thể, đoạn mã sử dụng hàm re.sub(pattern, repl, string) từ thư viện re để thực hiện việc tìm kiếm các chuỗi số trong chuỗi đầu vào và thay thế chúng bằng bình phương của số đó. Trong trường hợp này, pattern r"\d+" được sử dụng để tìm kiếm các chuỗi số trong chuỗi đầu vào.

Hàm square(match) được định nghĩa để nhận một đối tượng match (được trả về từ hàm re.sub) và trả về bình phương của số được tìm thấy.

**Đề bài:**

Trong một văn bản chứa N dòng. Văn bản chứa && và || biểu tượng. Nhiệm vụ của bạn là sửa đổi các ký hiệu đó như sau:

&& → and

|| → or

Cả && và || nên có một khoảng trống " " ở cả hai bên.

**Input:**

Dòng đầu tiên chứa số nguyên, N. Tiếp theo N mỗi dòng chứa một dòng văn bản.

Ràng buộc 0<N<100

Sao cho không có ký tự && cũng không || xảy ra ở đầu hoặc cuối mỗi dòng.

**Output:**

Xuất văn bản đã sửa đổi.

**Bài Lap:**

N = int(input())

for i in range(N):

line = input()

while (' && ' in line) or (' || ' in line):

line = line.replace(' && ', ' and ')

line = line.replace(' || ', ' or ')

print(line)

\*\*\*

- Chúng ta sẽ giải đơn giản như sau:

+ Chúng ta sử dụng vòng lặp for để đọc các dòng, và sử dụng vòng lặp while để tìm các chuỗi "&&" hoặc "||" mà có khoảng trắng xung quanh.

+ Khi thấy && hoặc || để thay thế các chuỗi tìm thấy bằng " and " hoặc " or " bằng lệnh replace().

+ Cuối cùng, chúng ta hiển thị các dòng văn bản đã được thay thế ra màn hình. Đoạn mã này đáp ứng các yêu cầu định dạng đầu vào và ràng buộc được nêu trong đề bài.

\*\*\*

# Task 2. Validating Credit Card Numbers(Check số tài khoản ngân hàng)

Bạn và Fredrick là bạn tốt. Hôm qua, Fredrick đã nhận được thẻ tín dụng N từ Ngân hàng ABCD. Anh ấy muốn xác minh xem số thẻ tín dụng của mình có hợp lệ hay không. Bạn tình cờ rất giỏi regex nên anh ấy đang yêu cầu sự giúp đỡ của bạn!

Thẻ tín dụng hợp lệ của ABCD Bank có các đặc điểm sau:

► Nó phải bắt đầu bằng 4, 5 hoặc 6.

► Nó phải chứa chính xác 16 chữ số.

► Nó chỉ phải bao gồm các chữ số (0-9).

► Nó có thể có các chữ số trong nhóm 4, cách nhau bởi một dấu gạch nối "-".

► Nó KHÔNG được sử dụng bất kỳ dấu phân cách nào khác như '', '\_', v.v.

► Nó KHÔNG được có 4 hoặc nhiều chữ số lặp lại liên tiếp.

**Input:**

Dòng đầu vào đầu tiên chứa một số nguyên N.

N dòng tiếp theo chứa số thẻ tín dụng.

**Ràng buộc**

0<N<100

**Output:**

In 'Valid' nếu số thẻ tín dụng hợp lệ. Nếu không, hãy in "Invalid". Không in dấu ngoặc kép.

**Bài làm:**

*- Để giải quyết bài toán này, bạn có thể sử dụng biểu thức chính quy (regex) trong Python để kiểm tra tính hợp lệ của số thẻ tín dụng. Dưới đây là một chương trình Python để thực hiện việc này:*

import re

def is\_valid\_credit\_card(card\_number):

    # Kiểm tra số thẻ có bắt đầu bằng 4, 5 hoặc 6 và có đúng 16 chữ số không

    if re.match(r"^[4-6]\d{3}-?\d{4}-?\d{4}-?\d{4}$", card\_number):

    # Loại bỏ các dấu gạch ngang và kiểm tra xem số còn lại có chứa hơn 1 số lặp liên tiếp không

        cleaned\_number = re.sub(r"-", "", card\_number)

        if re.search(r"(\d)(\1{3,})", cleaned\_number) is None:

            return "Valid"

    return "Invalid"

# Đọc số lượng số thẻ tín dụng từ đầu vào

N = int(input())

# Đọc N số thẻ tín dụng và kiểm tra tính hợp lệ của chúng

for \_ in range(N):

    card\_number = input().strip()

    print(is\_valid\_credit\_card(card\_number))

*- Trong chương trình này, hàm is\_valid\_credit\_card(card\_number) kiểm tra tính hợp lệ của số thẻ tín dụng bằng cách sử dụng biểu thức chính quy. Nếu số thẻ tín dụng thoả mãn các điều kiện đã cho, hàm sẽ trả về "Valid". Ngược lại, nó sẽ trả về "Invalid". Đọc số lượng số thẻ tín dụng từ đầu vào, sau đó kiểm tra và hiển thị kết quả cho mỗi số thẻ tín dụng.*

# Task 3. Words Score (Đếm số nguyên âm và tính điểm)

Trong thử thách này, nhiệm vụ là gỡ lỗi mã hiện có để thực thi thành công tất cả các tệp thử nghiệm được cung cấp.

Hãy xem xét rằng các nguyên âm trong bảng chữ cái là a, e, i, o, u và y.

Hàm score\_words lấy một danh sách các từ viết thường làm đối số và trả về điểm số như sau:

Điểm của một từ là 2 nếu từ đó chứa số nguyên âm chẵn. Nếu không, điểm của từ này là 1. Điểm số cho toàn bộ danh sách các từ là tổng điểm của tất cả các từ trong danh sách.

Gỡ lỗi hàm đã cho score\_words sao cho nó trả về điểm số chính xác.

Hàm của bạn sẽ được kiểm tra trong một số trường hợp bằng mã mẫu bị khóa.

**Input:** Đầu vào được đọc bởi mẫu mã bị khóa được cung cấp. Trong dòng đầu tiên, có một số nguyên n duy nhất biểu thị số từ. Trong dòng thứ hai, có n từ viết thường được phân tách bằng dấu cách.

**Ràng buộc**

1<N<20

Mỗi từ có tối đa 20 chữ cái và tất cả các chữ cái đều là chữ thường tiếng Anh

**Output**

Đầu ra được tạo ra bởi mẫu mã được cung cấp và khóa. Nó gọi hàm score\_words với danh sách các từ được đọc từ đầu vào làm đối số và in điểm trả về cho đầu ra.

*- Để giải quyết bài toán này, bạn cần viết lại hàm score\_words sao cho nó trả về điểm số đúng dựa trên các quy tắc đã cho. Dưới đây là phiên bản sửa lỗi của hàm score\_words:*

**Bài làm:**

def is\_vowel(letter):

    return letter in 'aeiouy'

def score\_words(words):

    score = 0

    for word in words:

        vowel\_count = sum(1 for letter in word if is\_vowel(letter))

        if vowel\_count % 2 == 0:

            score += 2

        else:

            score += 1

    return score

# Đọc số lượng từ từ đầu vào

n = int(input())

# Đọc danh sách từ từ đầu vào và chuyển thành danh sách

words = input().split()

# Tính điểm số và hiển thị kết quả

print(score\_words(words))

*- Trong chương trình này, hàm is\_vowel(letter) được sử dụng để kiểm tra xem một chữ cái có phải là nguyên âm hay không. Hàm score\_words(words) sau đó lặp qua từng từ trong danh sách words, đếm số nguyên âm trong mỗi từ, và tăng điểm số theo quy tắc đã cho. Cuối cùng, chương trình hiển thị tổng điểm số ra màn hình.*

# Task 4. Default Arguments( Đối số mặc định)

Trong thử thách này, nhiệm vụ là gỡ lỗi mã hiện có để thực thi thành công tất cả các tệp thử nghiệm được cung cấp.

Python hỗ trợ một khái niệm hữu ích về các giá trị đối số mặc định. Đối với mỗi đối số từ khóa của một hàm, chúng ta có thể gán một giá trị mặc định sẽ được sử dụng làm giá trị của đối số đã nói nếu hàm được gọi mà không có nó. Ví dụ: hãy xem xét hàm gia tăng sau:

def increment\_by(n, increment=1):

return n + increment

Các chức năng hoạt động như thế này:

>>> increment\_by(5, 2)

7

>>> increment\_by(4)

5

>>>

Gỡ lỗi hàm đã cho print\_from\_stream sử dụng giá trị mặc định của một trong các đối số của nó. Hàm có chữ ký sau:

def print\_from\_stream(n, stream)

Hàm này sẽ in các giá trị đầu tiên được trả về bởi phương thức get\_next() của đối tượng stream được cung cấp dưới dạng đối số. Mỗi giá trị này nên được in trong một dòng riêng biệt. Bất cứ khi nào hàm được gọi mà không có đối số luồng, nó nên sử dụng một thể hiện của lớp EvenStream được định nghĩa trong các sơ khai mã bên dưới làm giá trị của luồng. Hàm của bạn sẽ được kiểm tra trong một số trường hợp bằng mã mẫu bị khóa.

**Input:**

Đầu vào được đọc bởi mẫu mã bị khóa được cung cấp. Trong dòng đầu tiên, có một số nguyên q biểu thị số lượng truy vấn. Mỗi dòng q sau đây chứa một stream\_name theo sau là số nguyên n và nó tương ứng với một phép kiểm tra duy nhất cho hàm của bạn.

**Ràng buộc**

1<q<100

1<n<100

**Output:**

Đầu ra được tạo ra bởi mẫu mã được cung cấp và khóa. Đối với mỗi truy vấn (stream\_name, n), nếu stream\_name chẵn thì print\_from\_stream(n) được gọi. Mặt khác, nếu stream\_name là lẻ, thì print\_from\_stream(n, OddStream()) được gọi

**Bài làm:**

class EvenStream(object):

    def \_\_init\_\_(self):

        self.current = 0

    def get\_next(self):

        to\_return = self.current

        self.current += 2

        return to\_return

class OddStream(object):

    def \_\_init\_\_(self):

        self.current = -1

    def get\_next(self):

        self.current += 2

        return self.current

def print\_from\_stream(n, stream=None):

    if stream is None:

        stream = EvenStream()

    for \_ in range(n):

        value = stream.get\_next()

        print(value)

if \_\_name\_\_ == '\_\_main\_\_':

    queries = int(input())

    for \_ in range(queries):

        stream\_name, n = input().split()

        n = int(n)

        if stream\_name == 'even':

            stream = EvenStream()

        else:

            stream = OddStream()

        print\_from\_stream(n, stream)

Đoạn mã trên định nghĩa hai lớp EvenStream và OddStream để tạo ra dãy số chẵn và số lẻ:

Lớp EvenStream:

* Hàm \_\_init\_\_(self): Khởi tạo một đối tượng của lớp EvenStream với thuộc tính current được đặt là 0, bắt đầu từ số chẵn đầu tiên.
* Hàm get\_next(self): Trả về giá trị hiện tại của current, sau đó tăng current lên 2 để chuẩn bị cho lần gọi tiếp theo.

Lớp OddStream:

* Hàm \_\_init\_\_(self): Khởi tạo một đối tượng của lớp OddStream với thuộc tính current được đặt là -1, bắt đầu từ số lẻ đầu tiên.
* Hàm get\_next(self): Tăng current lên 2 để chuyển đến số lẻ tiếp theo, sau đó trả về giá trị hiện tại của current.

Hàm print\_from\_stream(n, stream=None):

Tham số n đại diện cho số lượng giá trị cần in ra.

Tham số stream là một đối tượng của lớp EvenStream hoặc OddStream (hoặc mặc định là None).

Nếu stream không được truyền vào (tức là stream bằng None), hàm sẽ sử dụng lớp EvenStream làm giá trị mặc định cho stream.

Hàm này sử dụng lệnh vòng lặp để gọi phương thức get\_next() của đối tượng stream và in ra giá trị được trả về n lần.

Phần if \_\_name\_\_ == '\_\_main\_\_':

Đoạn mã này kiểm tra xem chương trình đang được chạy trực tiếp hay được import vào một chương trình khác.

Nếu chương trình được chạy trực tiếp, nó đọc số lượng truy vấn từ đầu vào, sau đó đọc từng truy vấn (tên của dãy số và số lượng giá trị cần in ra). Dựa vào tên của dãy số, nó tạo ra một đối tượng EvenStream hoặc OddStream tương ứng và gọi hàm print\_from\_stream(n, stream) để in ra giá trị từ dãy số đó.

**Kết luận**

Từ 4 bài tập ứng dụng trên, chúng ta có thể rút ra một số kiến thức và bài học quan trọng từ thực tế như sau:

1. Sử dụng Regular Expressions (Regex):

Regex là công cụ mạnh mẽ để tìm kiếm, thay thế và xử lý chuỗi dữ liệu dựa trên mẫu cụ thể.

Trong bài toán "Regex Substitution", chúng ta đã thấy cách sử dụng re.sub() để thay thế các chuỗi theo mẫu xác định. Regex giúp xử lý dữ liệu phức tạp với các quy tắc và mẫu nhất định, giúp thay đổi ký tự 1 cách nhanh chóng hơn.

2. Xử lý dữ liệu văn bản và chuỗi:

Việc xử lý và thay đổi văn bản, chuỗi là một phần quan trọng trong lập trình. Trong bài tập thay thế các chuỗi "&&" và "||" bằng "and" và "or", chúng ta đã sử dụng phương pháp thay thế chuỗi để thay đổi văn bản theo yêu cầu cụ thể.

3. Sử dụng Regular Expressions trong kiểm tra dữ liệu:

Trong bài "Validating Credit Card Numbers", chúng ta đã sử dụng regex để kiểm tra tính hợp lệ của số thẻ tín dụng theo các tiêu chí đã cho. Regex có thể được sử dụng để kiểm tra và xác định tính đúng đắn của dữ liệu, như kiểm tra định dạng số thẻ tín dụng.

4. Sử dụng đối số mặc định trong hàm:

Trong bài "Default Arguments", chúng ta đã sử dụng giá trị mặc định cho đối số trong hàm để xử lý các trường hợp khác nhau. Sử dụng đối số mặc định giúp tăng tính linh hoạt và tái sử dụng của hàm.

*Ý nghĩa và bài học:*

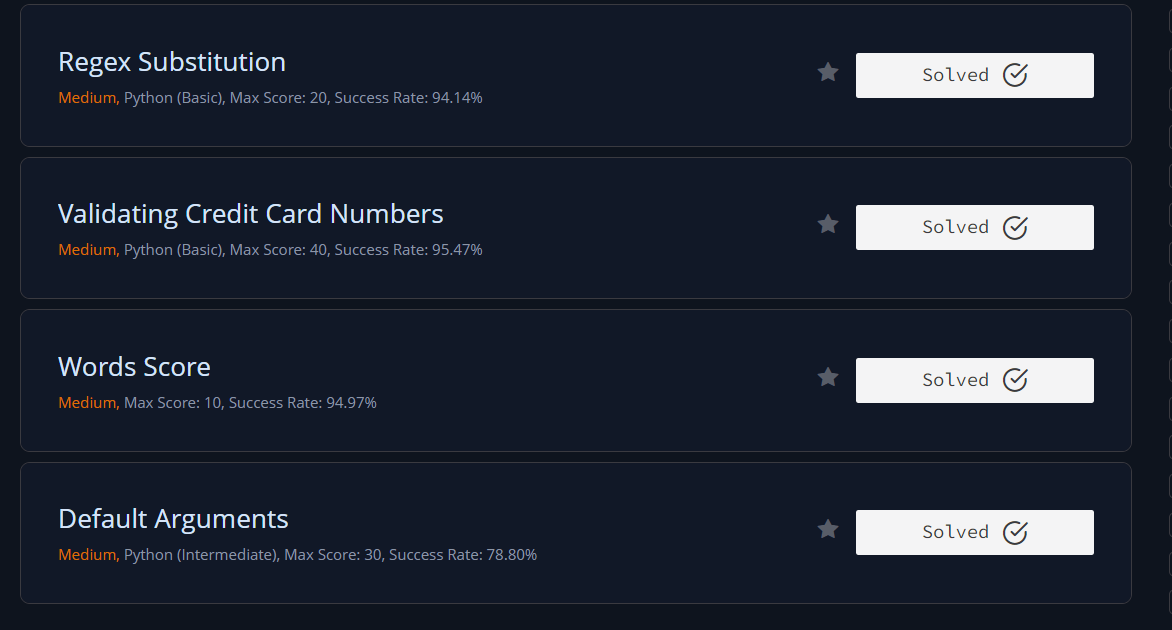
Việc giải quyết các bài toán thực tế bằng lập trình giúp cải thiện kỹ năng giải quyết vấn đề và logic.

Sử dụng các công cụ, thư viện thông dụng như Regular Expressions, xử lý chuỗi, đối số mặc định trong hàm giúp tăng khả năng xử lý dữ liệu và tối ưu hóa mã nguồn.

Từ 4 bài toàn với 4 cách giải quyết ta có thể ý thức được cách sử dụng công cụ linh hoạt và kỹ năng để giải quyết các vấn đề cụ thể một cách hiệu quả và hoàn thành tốt công việc một cách dễ dàng.

Việc thực hành và giải quyết các bài toán trong lập trình không chỉ giúp cải thiện kỹ năng lập trình mà còn rèn luyện khả năng phân tích vấn đề và tìm ra giải pháp tốt nhất cho từng tình huống cụ thể.

*Kết quả của 4 bài tập ứng dụng trên.*



Trong phạm vi khó có thể diễn đạt hết ý về mặt lý thuyết cũng như kỹ thuật, mặc dù đã cố gắng hoàn thành với tất cả sự nỗ lực của em, xong khó có thể tránh khỏi những thiếu sót. Kính mong nhận được những ý kiến đóng góp của thầy để em tiếp tục hoàn thiện kiến thức cũng như phương pháp của mình.

Em xin cam đoan những nội dung là sản phẩm của em tự nghiên cứu, những nội dung sao chép được chỉnh sửa theo ý hiểu. Nếu sai em xin hoàn toàn chịu trách nhiệm.

Em xin chân thành cảm ơn!