**TRƯỜNG ĐẠI HỌC CẦN THƠ**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

**KHOA KHOA HỌC MÁY TÍNH**



**BÁO CÁO HP TIN HỌC LÝ THUYẾT (CT121)**

**ĐỀ TÀI:**

**Mô Phỏng Hoạt Động NFAe**

**Sinh viên: Trần Chí Hiếu STT 24**

**Mã số: B2207520**

**Khóa: K48**

HK1 2024-2025

***Cần Thơ, 11/2024***

[PHẦN 1: CÁC BUỔI THỰC HÀNH 1](#_Toc183444758)

[I. THỰC HÀNH BUỔI 1 1](#_Toc183444759)

[1. BÀI TẬP CƠ BẢN: 1](#_Toc183444760)

[2. BÀI TẬP NÂNG CAO: 6](#_Toc183444762)

[II. THỰC HÀNH BUỔI 2 10](#_Toc183444764)

[1. Chỉnh sửa chương trình cho phép nhận DFA Cứng. 10](#_Toc183444765)

[2. DFA File 10](#_Toc183444766)

[3. Dựa trên DFA đã xây dựng, mở rộng xây dựng lớp NFA và kiểm tra 1 chuỗi có thuộc ngôn ngữ sinh bởi NFA đã cho. 11](#_Toc183444767)

[III. THỰC HÀNH BUỔI 3 13](#_Toc183444768)

[1. Đọc file txt, và xây dựng btcq để tìm và in: 13](#_Toc183444769)

[PHẦN 2: Mô phỏng hoạt động của NFAε 23](#_Toc183444770)

[I. GIỚI THIỆU: 23](#_Toc183444771)

[1. Lý do thực hiện đề tài: 23](#_Toc183444772)

[2. Tính năng: 24](#_Toc183444773)

[II. MỤC TIÊU BÀI TOÁN: 24](#_Toc183444774)

[1. Hiểu và mô phỏng hoạt động của NFAε 24](#_Toc183444775)

[2. Tạo công cụ mô phỏng hỗ trợ học tập và nghiên cứu: 24](#_Toc183444776)

[III. PHƯƠNG PHÁP THỰC HIỆN: 24](#_Toc183444777)

[1. Thiết kế Bài toán: 24](#_Toc183444778)

[IV. CÀI ĐẶT: 27](#_Toc183444779)

[1. Định Nghĩa NFAe 27](#_Toc183444780)

[2. Các Thủ Tục: 27](#_Toc183444781)

[3. Cài đặt: 29](#_Toc183444782)

[V. KẾT QUẢ ĐẠT ĐƯỢC: 33](#_Toc183444783)

[1. Công cụ mô phỏng hoạt động NFAe 33](#_Toc183444784)

[2. Giao diện: 33](#_Toc183444785)

[VI. HƯỚNG PHÁT TRIỂN: 35](#_Toc183444786)

[1. Mở rộng tính năng của công cụ 35](#_Toc183444787)

[2. Tăng cường trực quan hóa: 35](#_Toc183444788)

[3. Ứng dụng trong Giáo dục 36](#_Toc183444789)

[VII. TÀI LIỆU THAM KHẢO: 36](#_Toc183444790)

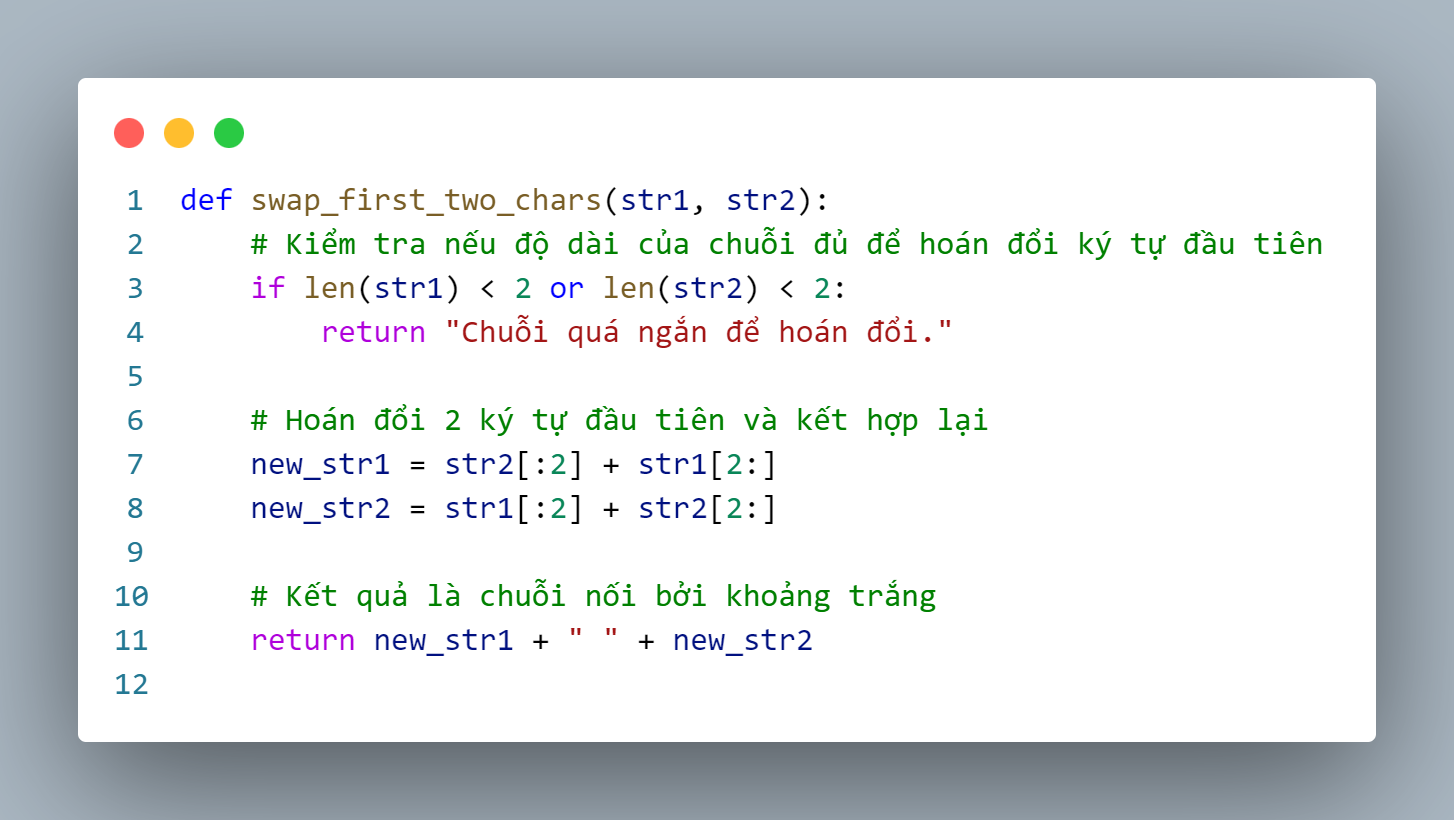
# CÁC BUỔI THỰC HÀNH

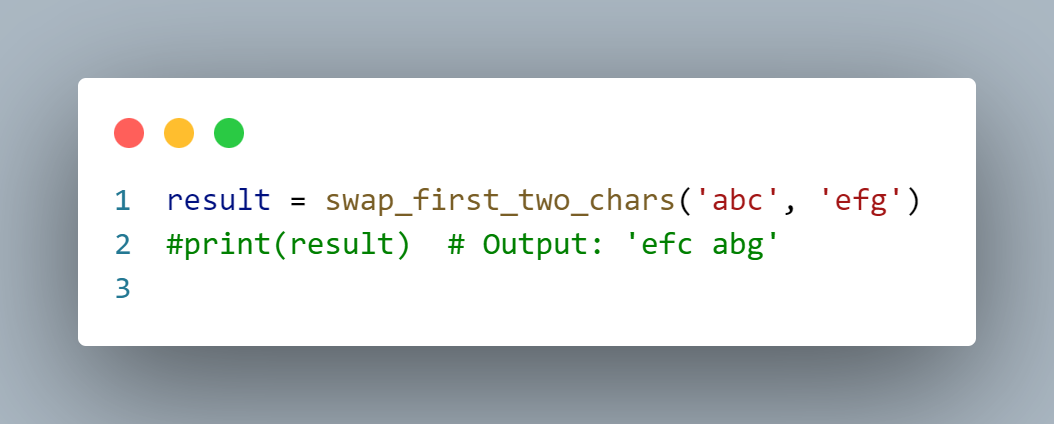
## THỰC HÀNH BUỔI 1

### BÀI TẬP CƠ BẢN:

#### Viết hàm hoán đổi 2 ký tự đầu tiên của 2 chuỗi đơn và sinh ra chuỗi mới từ 2 chuỗi kết quả (cách nhau bằng khoảng trắng).

Ví dụ: Input: ‘abc’ ‘efg’ Output: ‘efc abg’





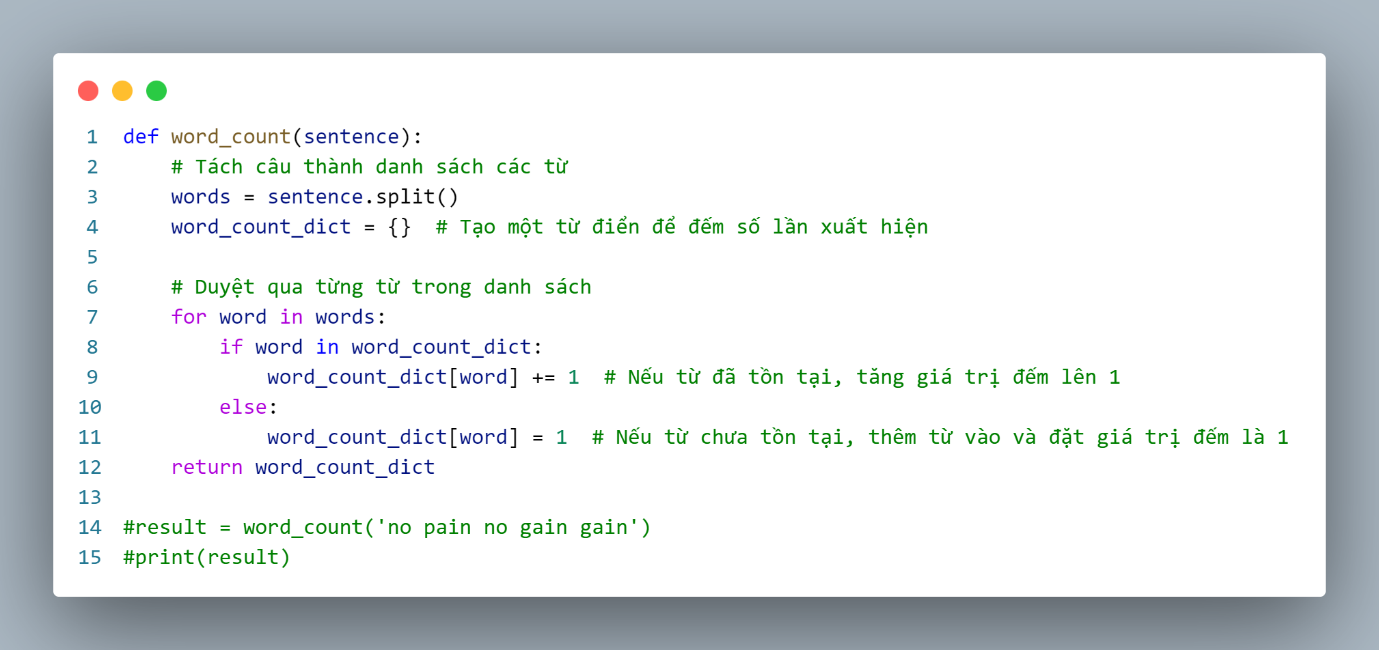
#### Viết hàm loại bỏ các ký tự tại vị trí index chẵn của một chuỗi cho trước

*Ví dụ: Input: ‘python’ Output: ‘yhn’*



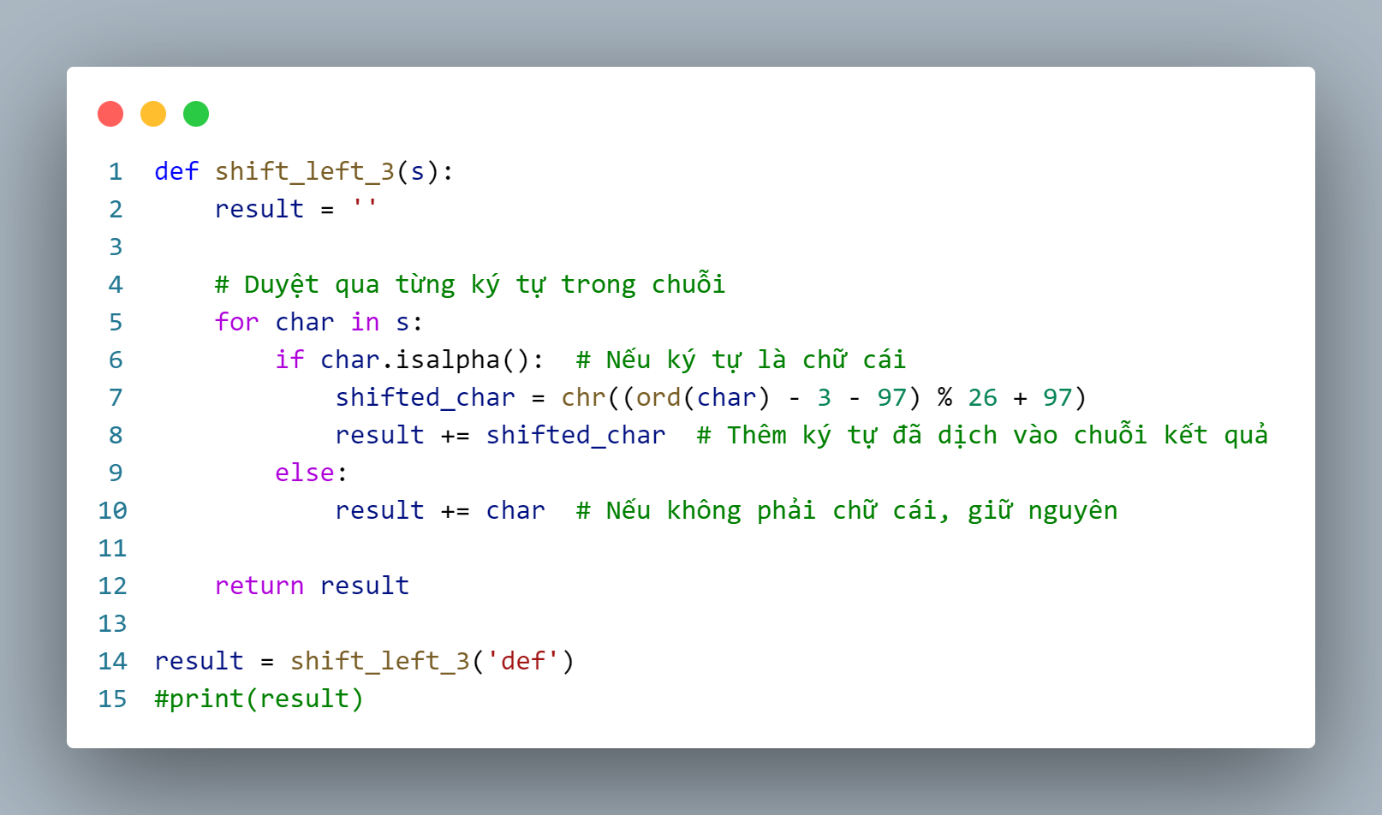
#### Viết hàm đếm số lần xuất hiện của các từ trong câu

*Ví dụ: Input: ‘no pain no gain’ Output: {‘no’: 2, ‘pain’: 1, ‘gain’: 1}*



#### Viết hàm mã hóa một chuỗi bằng cách dịch chuyển các ký tự sang trái 3 bước.

*Ví dụ: d->a, e->b, f->c Input: ‘def’ Output: ‘abc’*

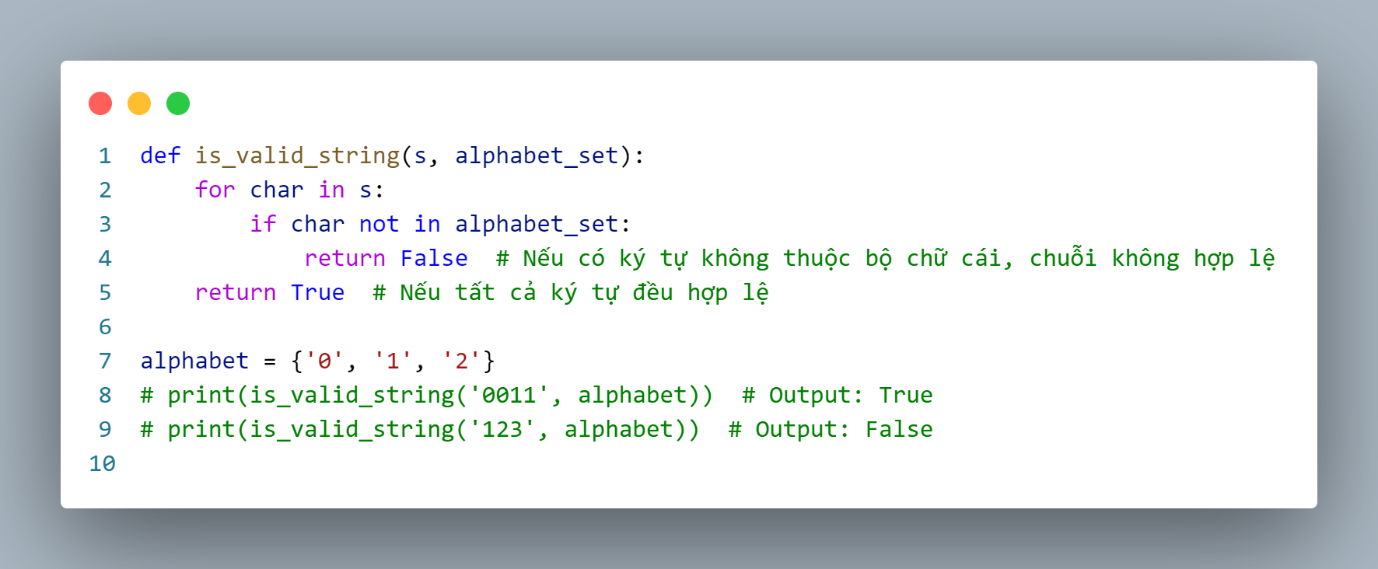


#### Viết hàm kiểm tra một chuỗi hợp lệ hay không? (có được sinh từ bộ chữ cái nhập)

*Ví dụ: bộ chữ cái nhập S = {0,1,2}*

*Các chuỗi hợp lệ: 0011, 12, 012, 122...*

*Các chuỗi không hợp lệ: 123, 24,...*



#### Viết hàm chuyển một chuỗi thành danh sách

*Ví dụ: Input: ‘This is a list’ Output: [‘This’, ‘is’, ‘a’, ‘list’]*



#### Viết hàm in ra ký tự không lặp lại đầu tiên trong chuỗi

*Ví dụ: Input: ‘abcdef’ Output: ‘a’*

*Input: ‘abcabcdef’ Output: ‘d’*

*Input: ‘aabbcc’ Output: ‘None’*



#### Viết hàm loại bỏ khoảng trắng trong chuỗi cho trước

*Ví dụ: Input: ‘a b c’ Output: ‘abc’*



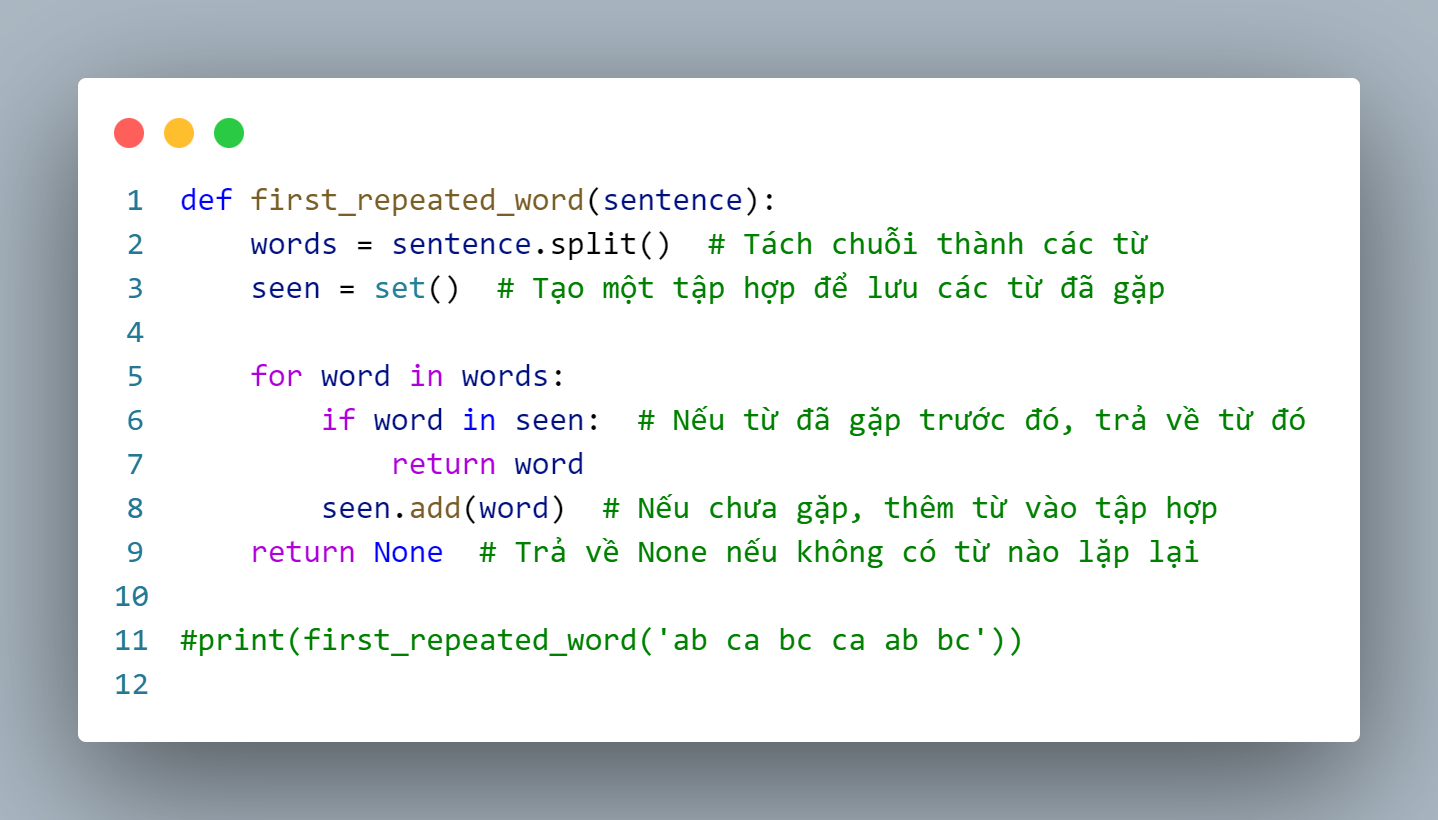
output: hieubaby

#### Viết hàm in ra từ được lặp lại đầu tiên trong chuỗi

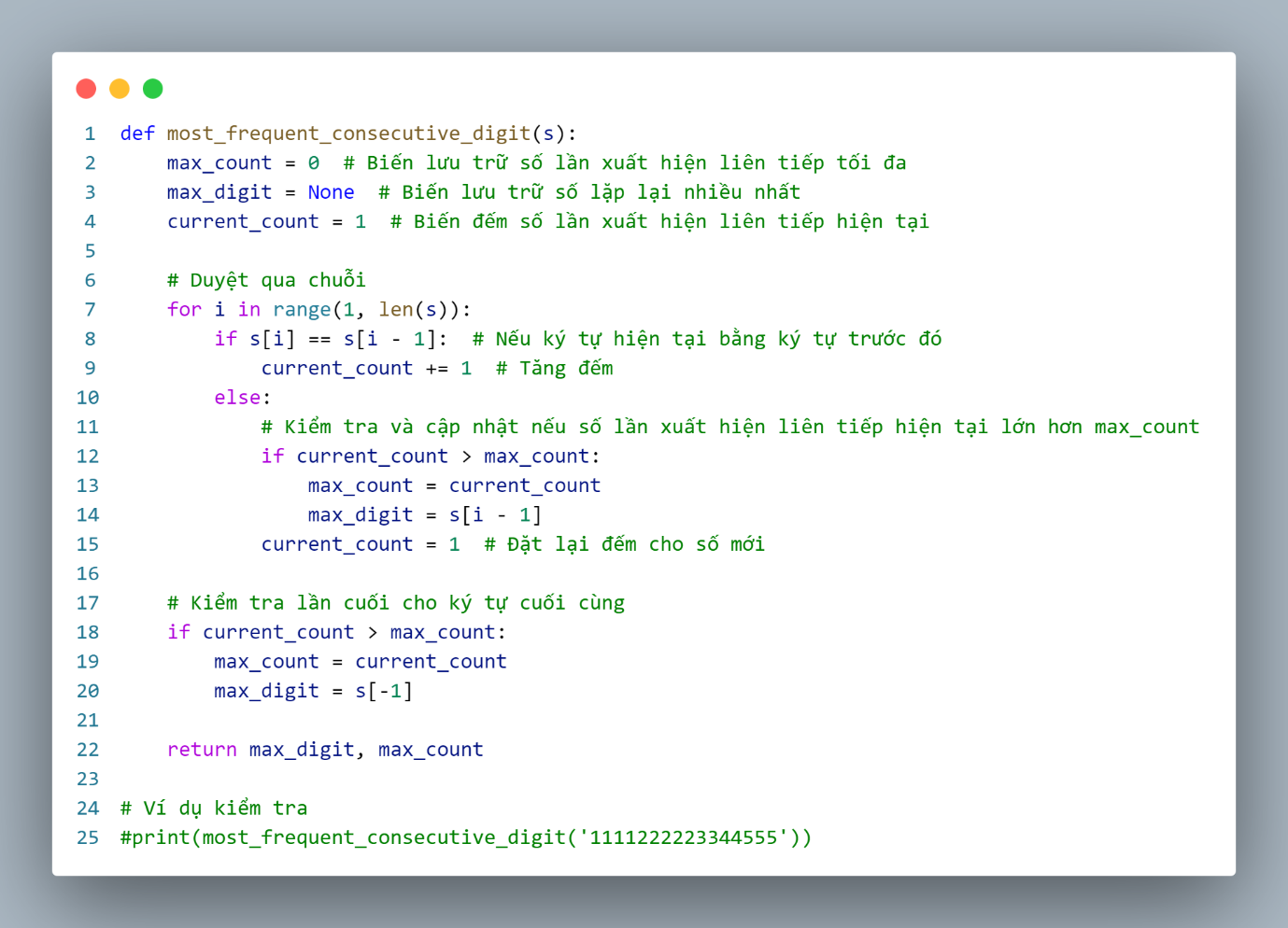
*Ví dụ:*

*Input: ‘ab ca bc ca ab bc’ Output: ‘ca’*

*Input: ‘ab ca bc ab’ Output: ‘ab’*



#### Viết hàm tìm độ dài tối đa của chuỗi con gồm các số 0 liên tiếp trong chuỗi nhị phân cho trước.



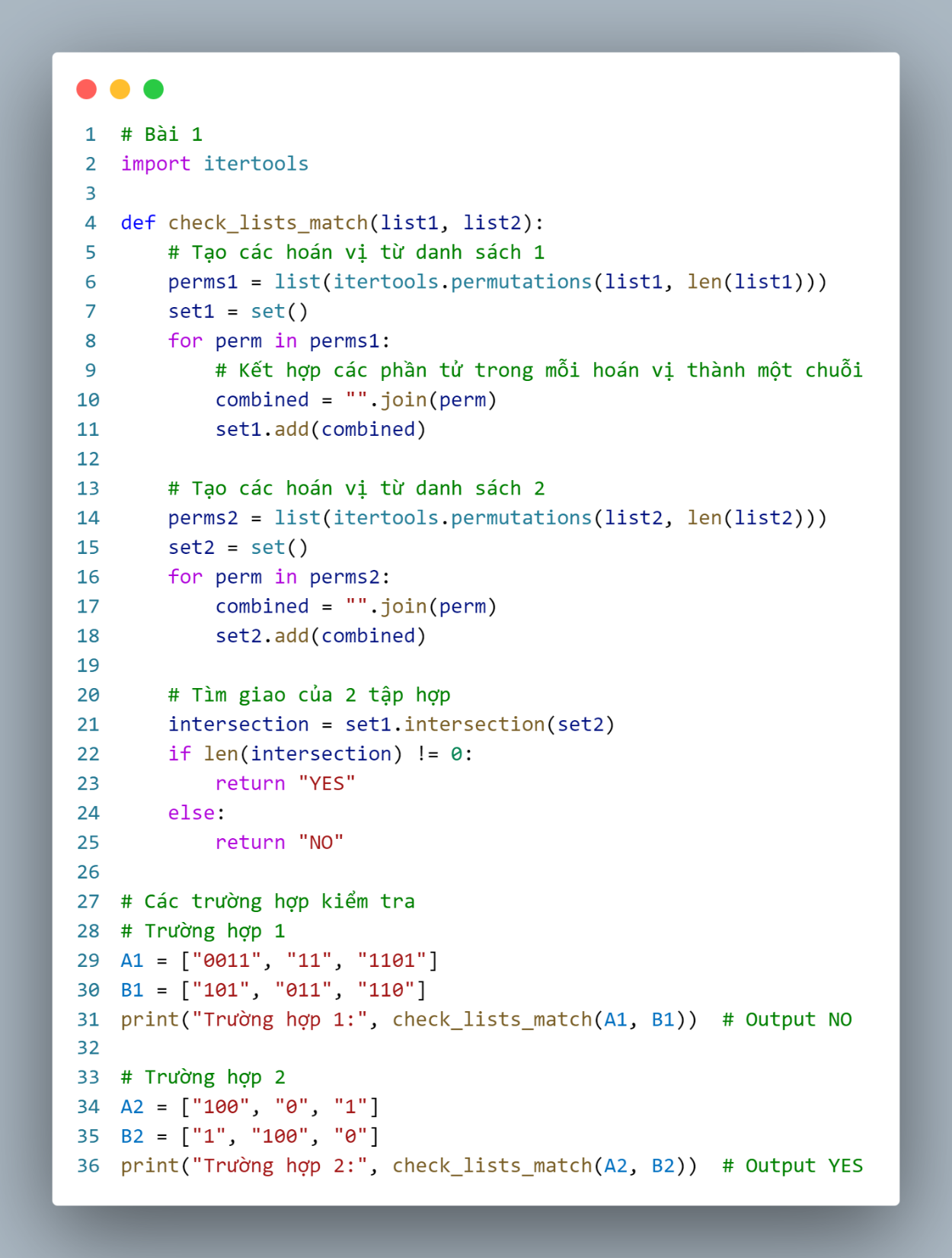
Output : ('2', 5)

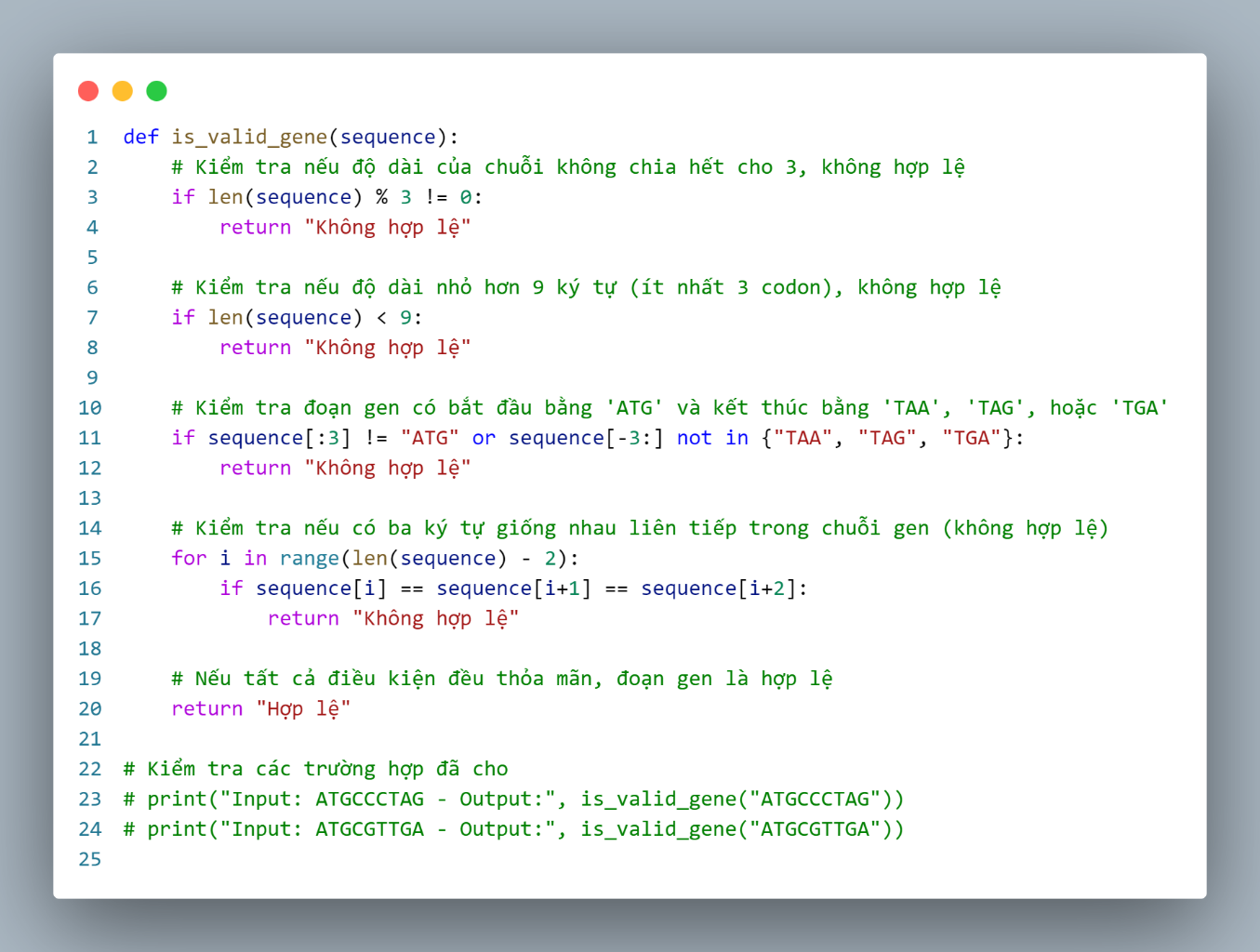
### BÀI TẬP NÂNG CAO:

#### Bài tập 1: Kiểm tra 2 danh sách khác nhau có thể sinh ra cùng một chuỗi hay không?

Inp*ut danh sách A = (s1, s2, ..., sn), danh sách B = (t1, t2, ...tm)*

*Output: Có,*





#### Bài tập 2:

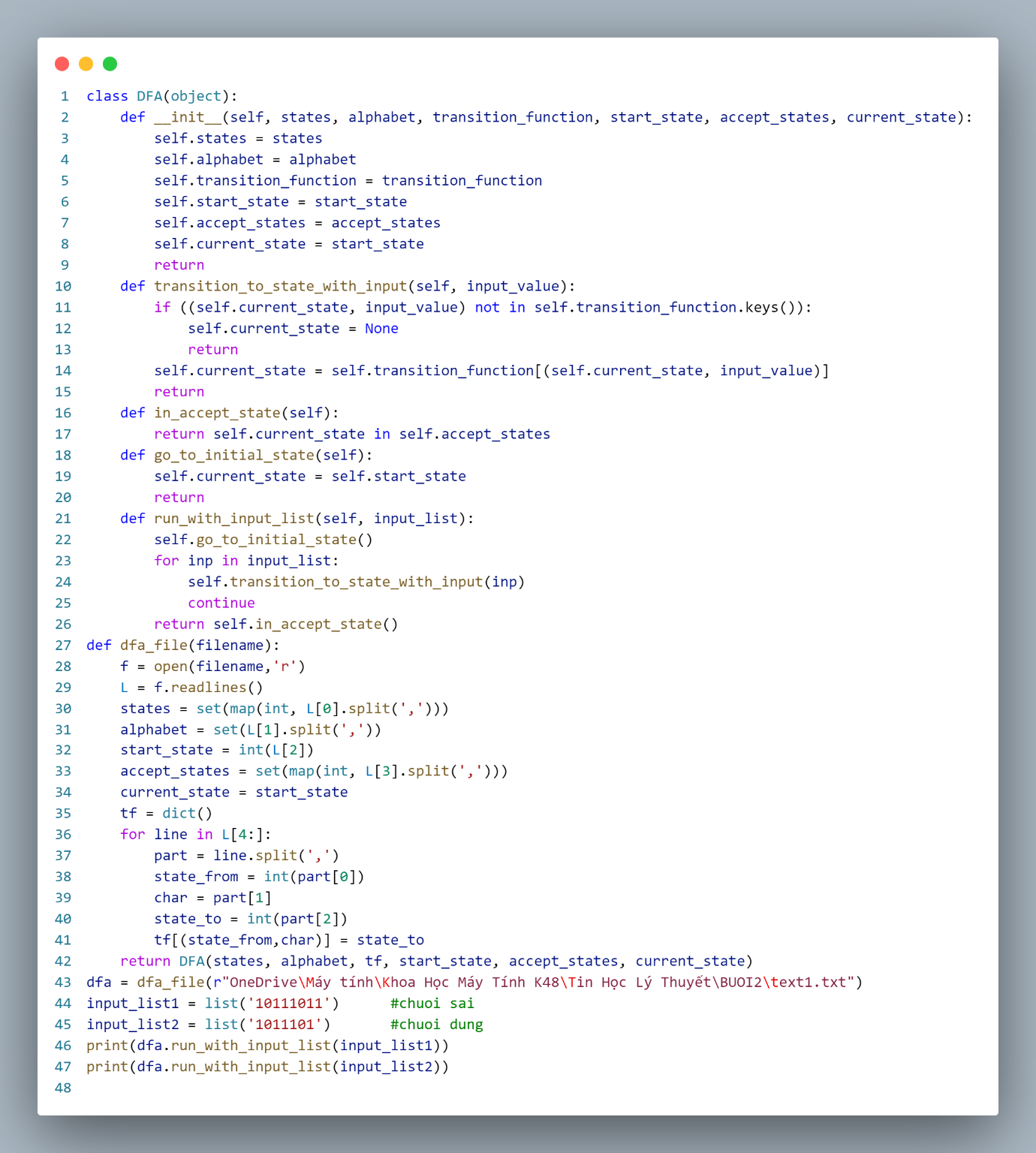
Output : không hợp lệ

Hợp lệ

## THỰC HÀNH BUỔI 2

### Chỉnh sửa chương trình cho phép nhận DFA Cứng.

### DFA File

******

### Dựa trên DFA đã xây dựng, mở rộng xây dựng lớp NFA và kiểm tra 1 chuỗi có thuộc ngôn ngữ sinh bởi NFA đã cho.

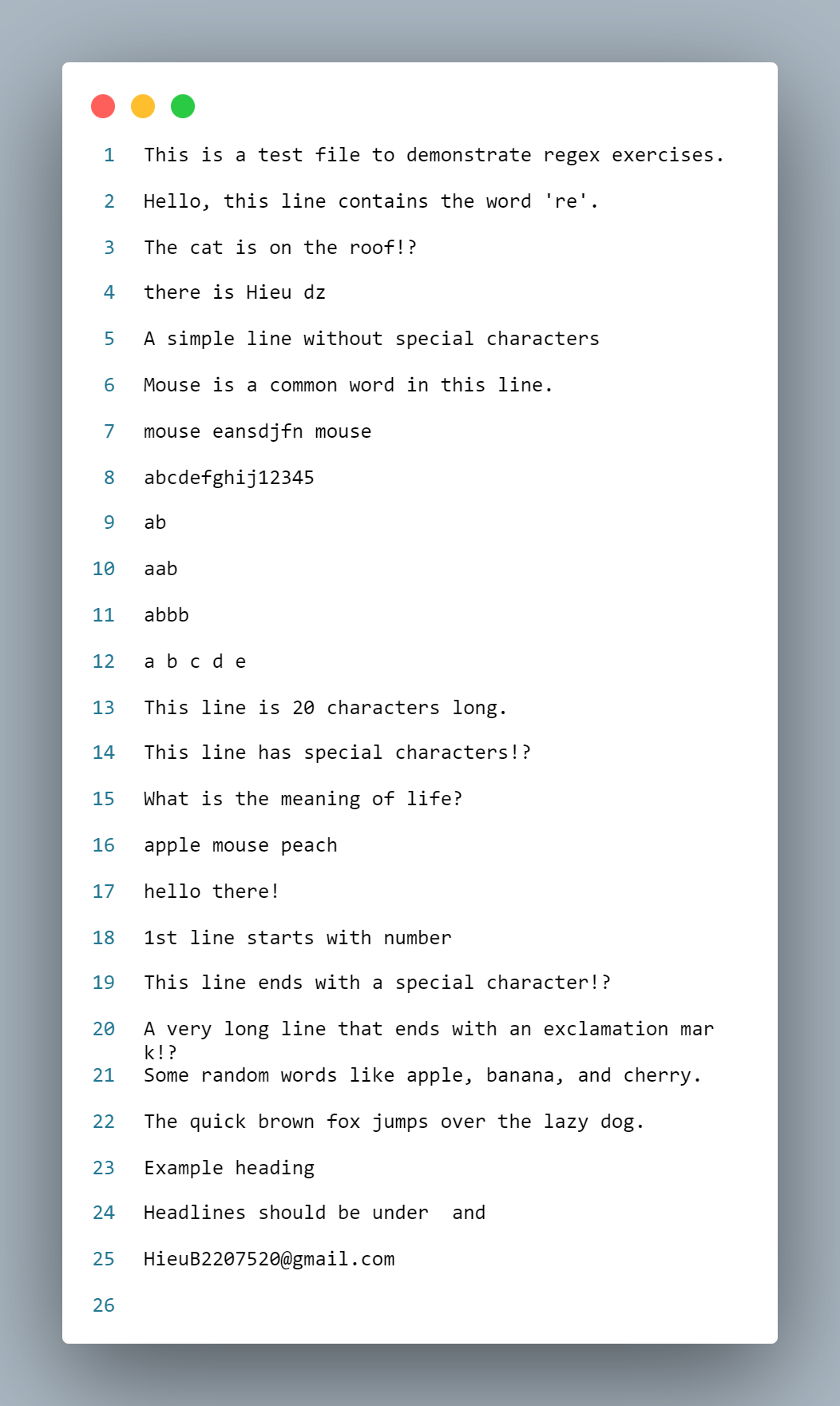


## THỰC HÀNH BUỔI 3

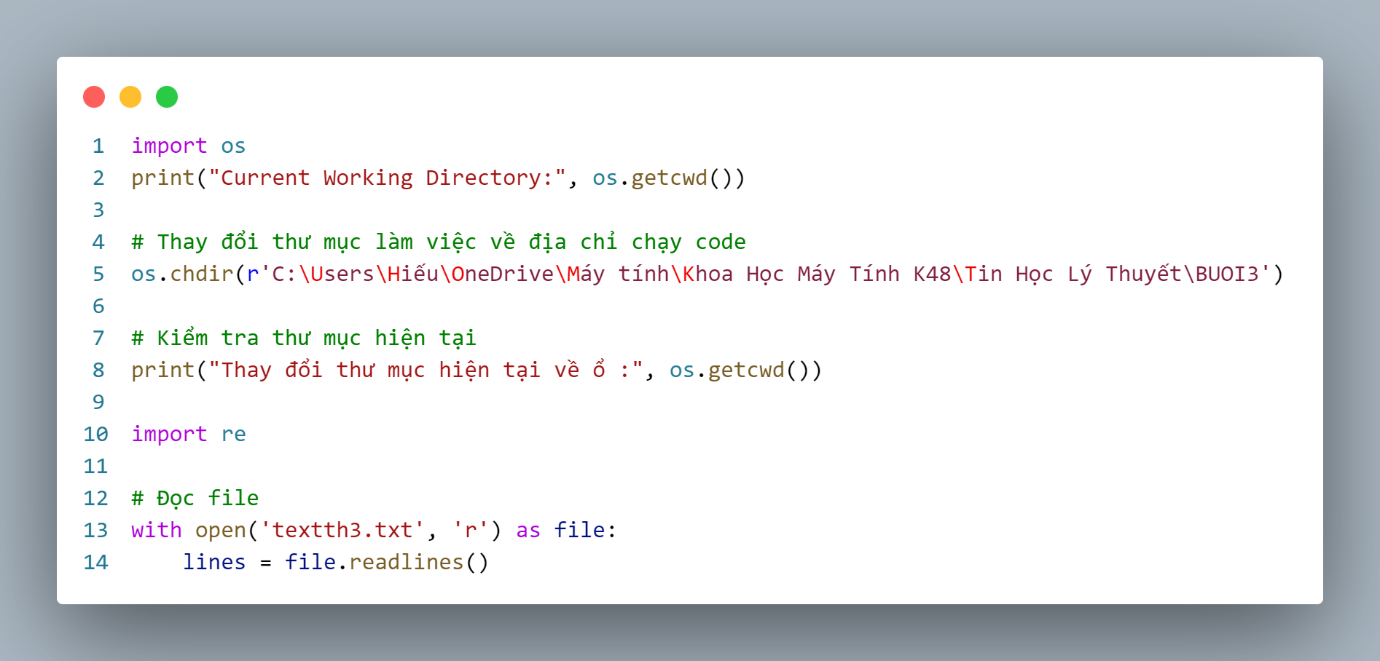
### Đọc file txt, và xây dựng btcq để tìm và in:

#### Xử lí file:

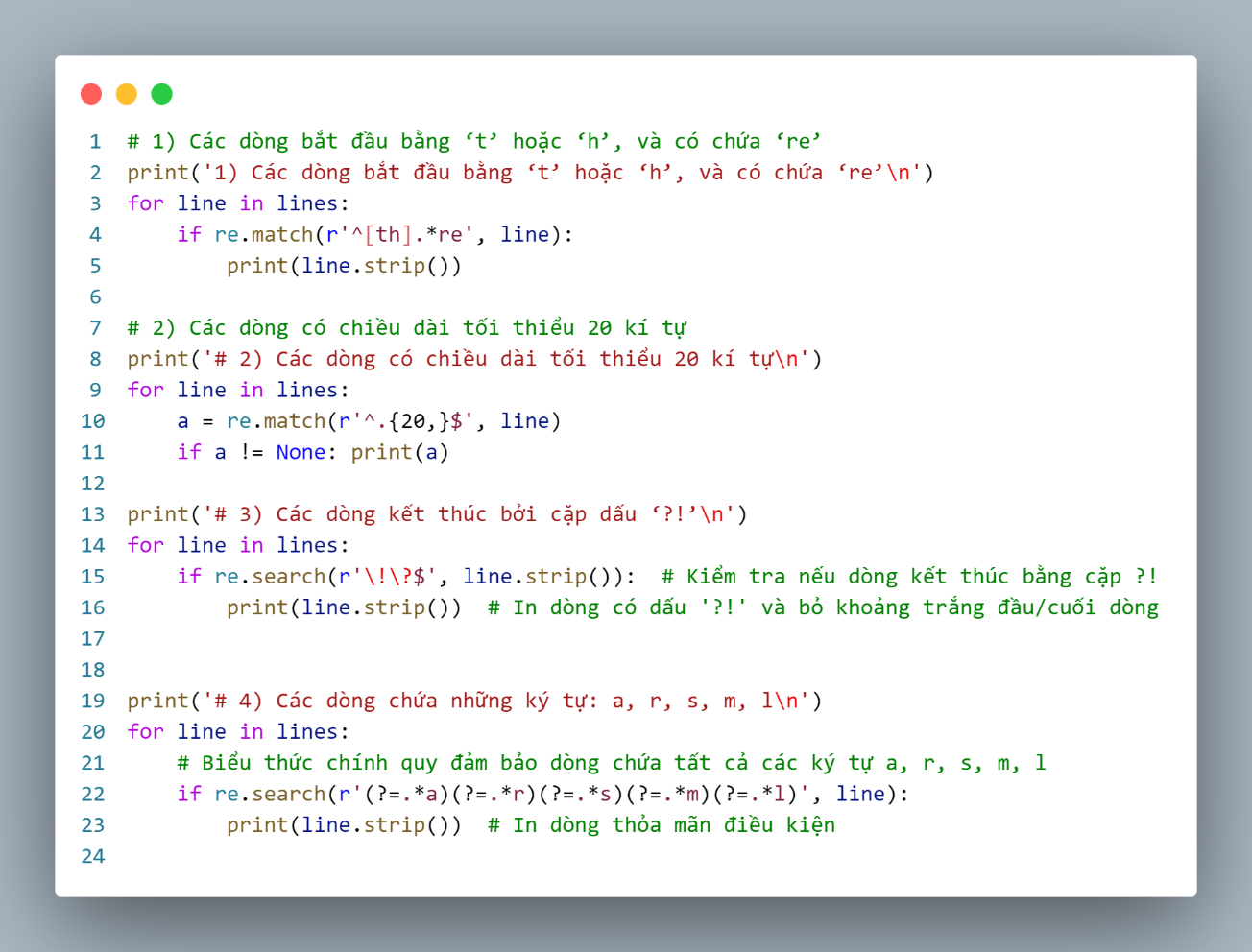
File text3.txt:



#### Hàm Đọc File

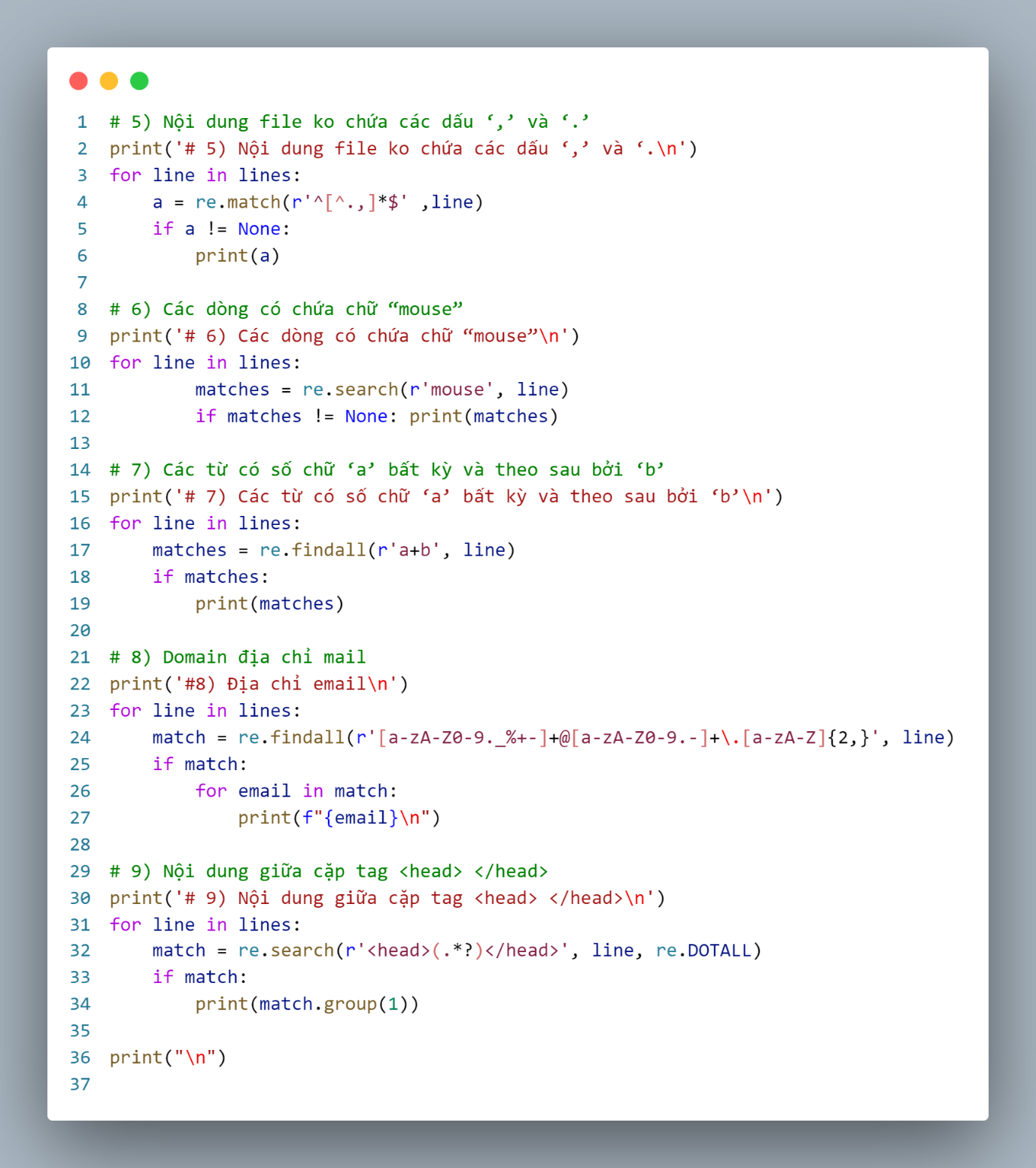


#### Câu 1 – 4:



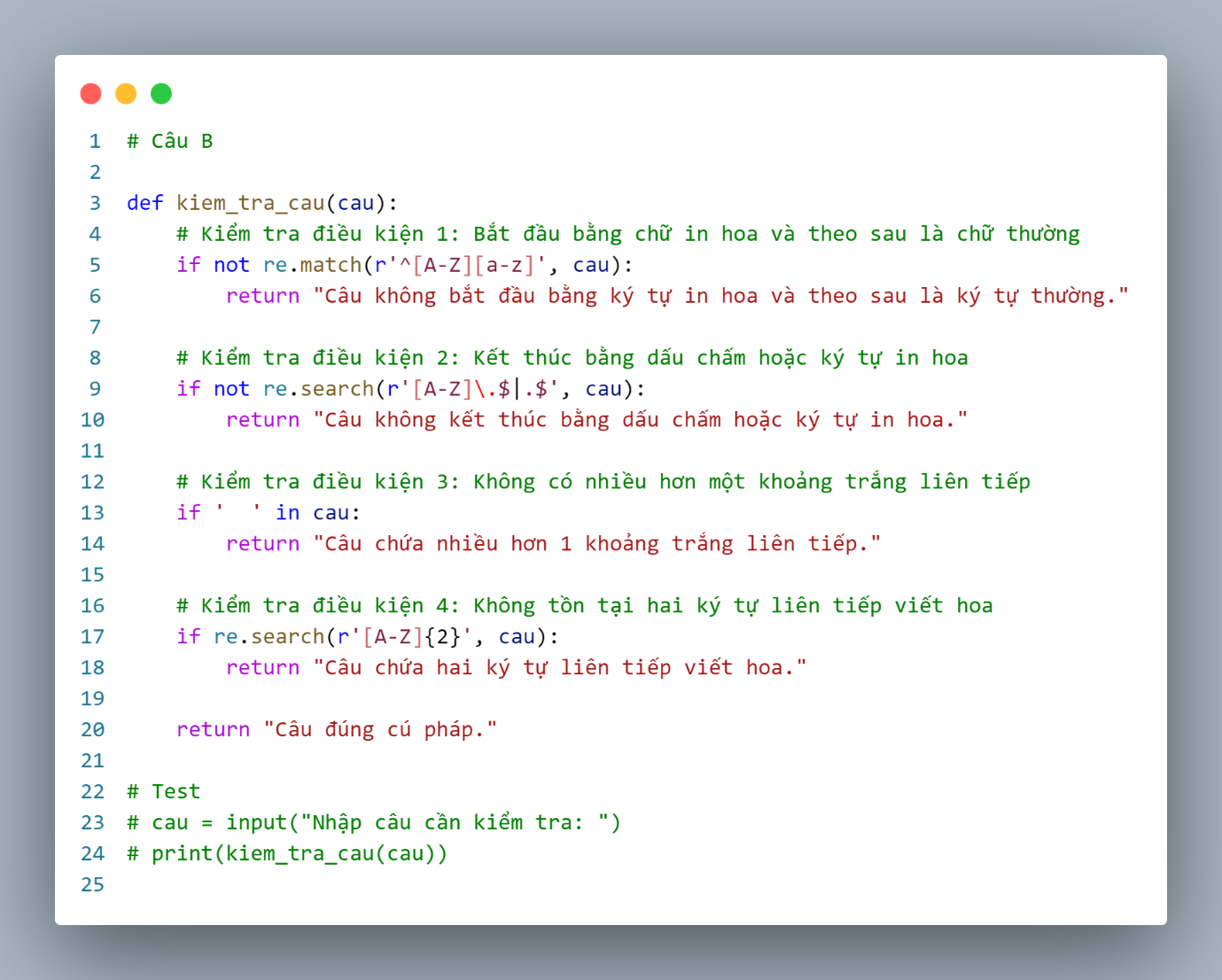
Output: 

#### Câu 5-9

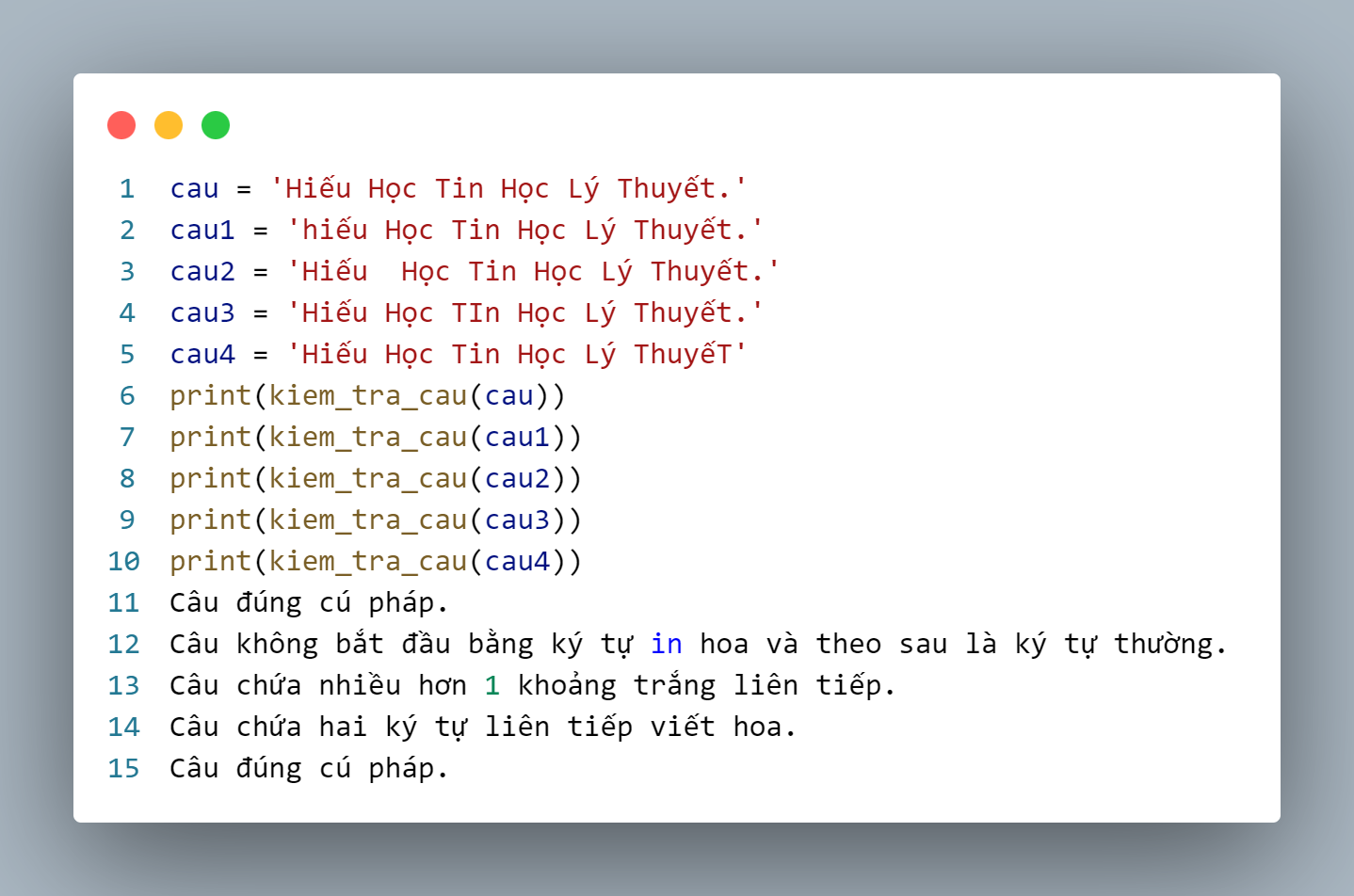


Output: 

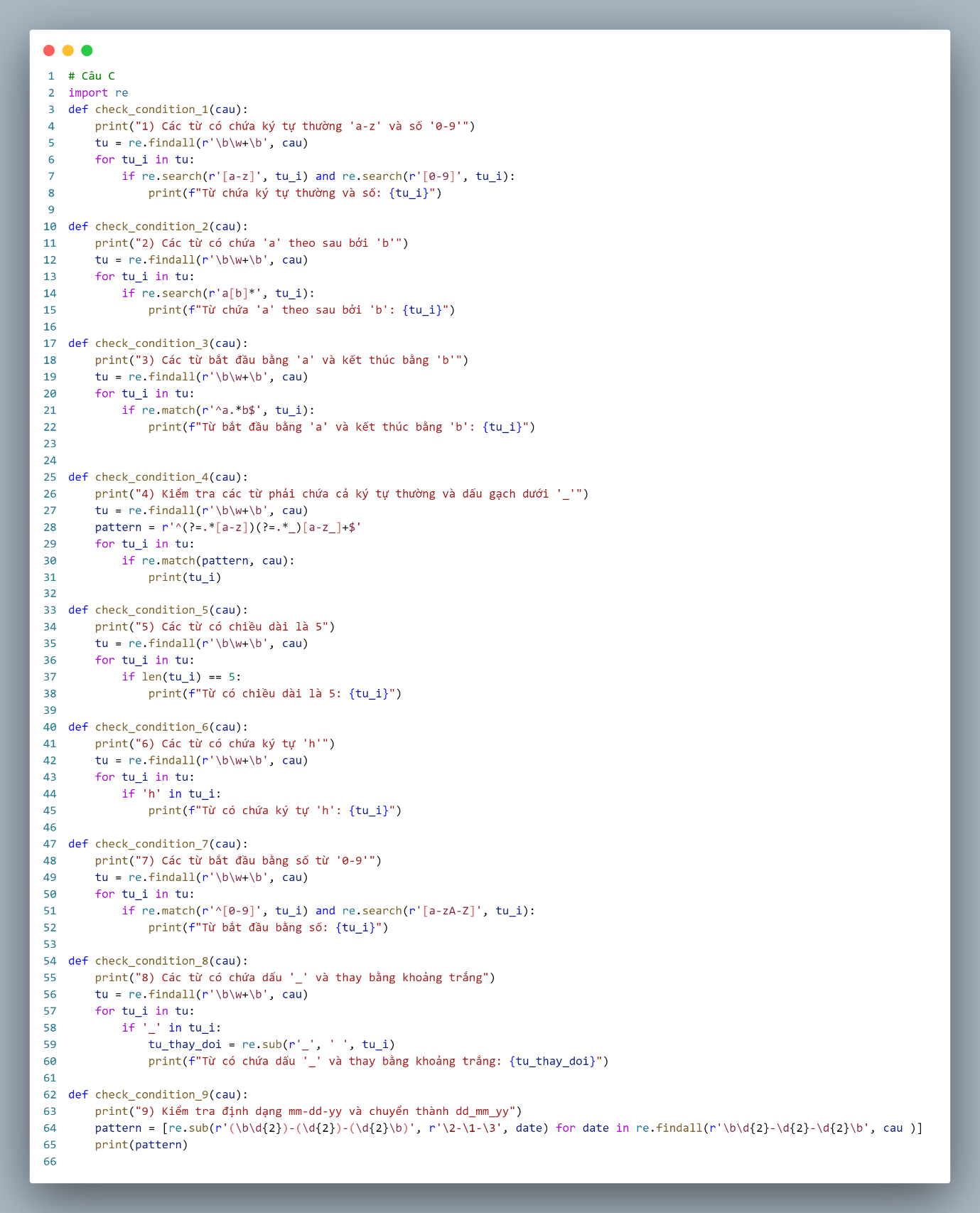
#### Câu B: 1-4



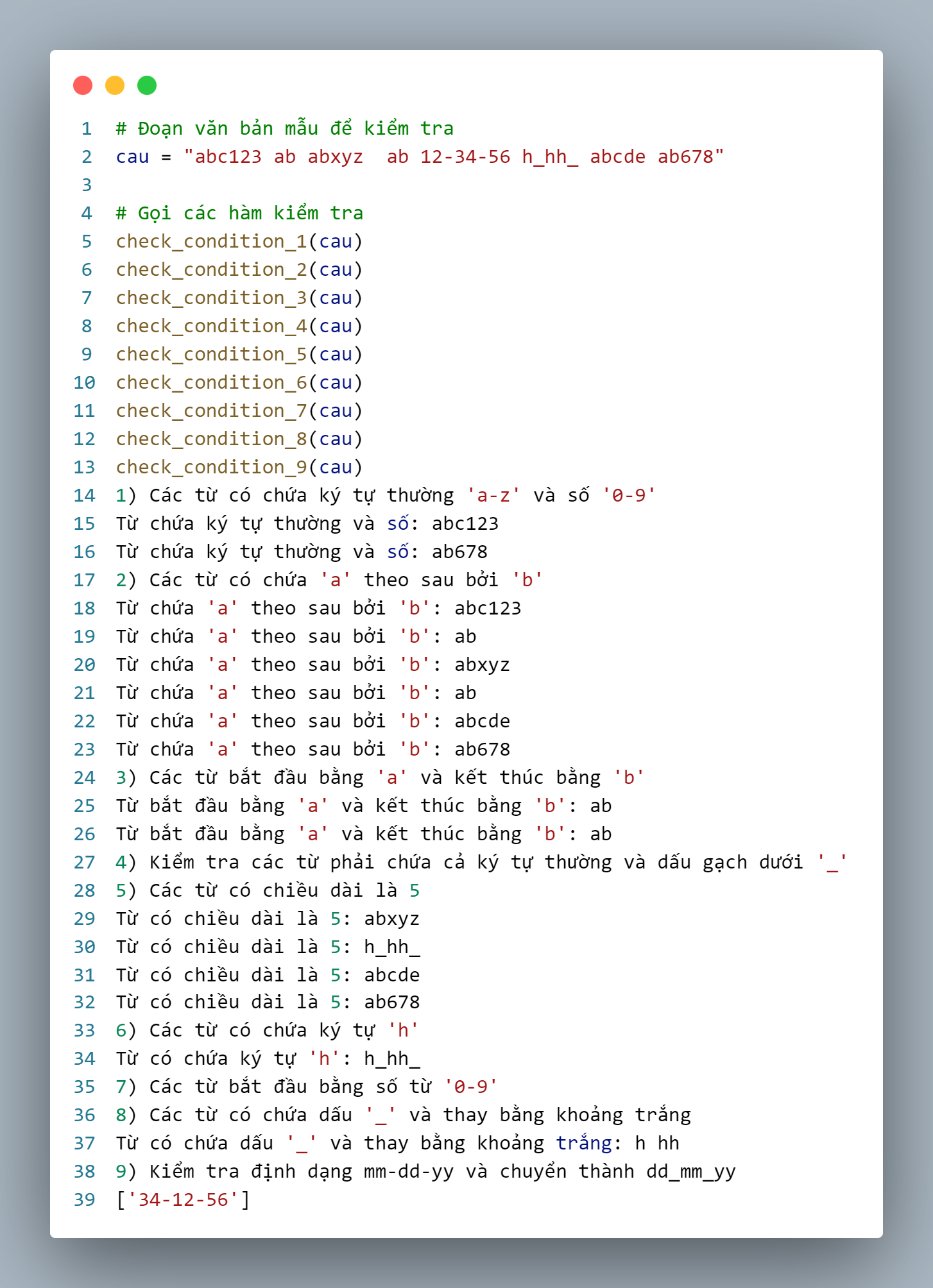
output:



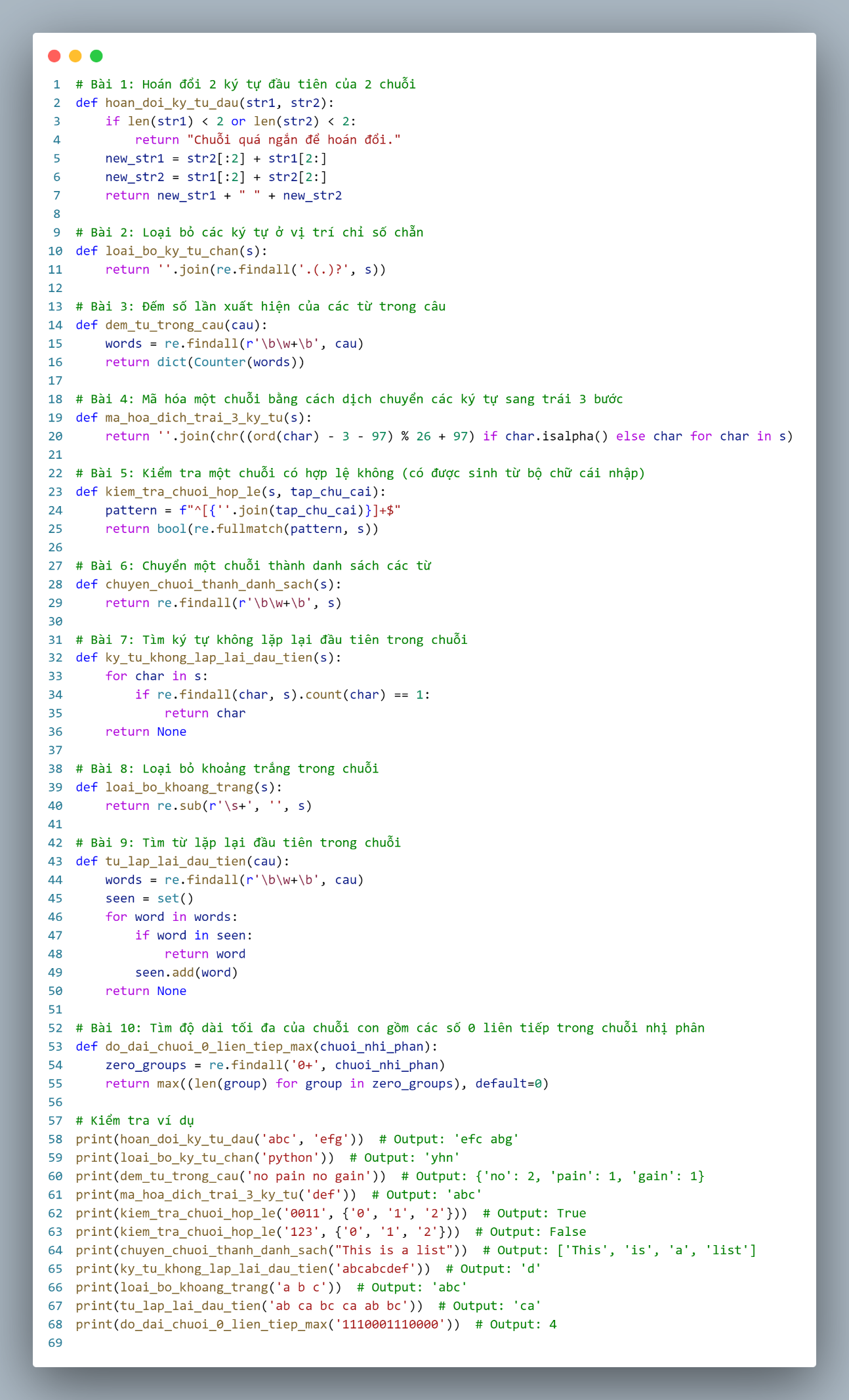
#### Câu C: 1- 9:



Output:



#### Câu D:



# Mô phỏng hoạt động của NFAε

## GIỚI THIỆU:

NFAε (Nondeterministic Finite Automaton with Epsilon Transitions) là một mô hình tự động hữu hạn không xác định, trong đó các trạng thái có thể chuyển đổi không cần dựa vào bất kỳ ký tự đầu vào nào, gọi là epsilon-transitions (ký hiệu là ε). Mô hình này được phát triển để mở rộng khả năng biểu diễn các ngôn ngữ chính quy và hỗ trợ chuyển đổi giữa các trạng thái một cách linh hoạt hơn.

NFAε là một mô hình tự động mạnh mẽ, nhưng khó sử dụng trực tiếp trong các ứng dụng thực tế do tính không xác định và các epsilon-chuyển phức tạp.

NFAε có khả năng chuyển đổi giữa các trạng thái mà không cần tiêu thụ ký tự đầu vào. Điều này cho phép xây dựng các trạng thái phức tạp hơn mà không ảnh hưởng đến tính đúng đắn của mô hình.

Với mỗi trạng thái và ký tự đầu vào, có thể có nhiều trạng thái đích, hoặc thậm chí không có trạng thái nào. Tính chất này làm tăng sự linh hoạt trong mô phỏng.

NFAε có khả năng biểu diễn tất cả các ngôn ngữ chính quy và dễ dàng chuyển đổi thành DFA hoặc biểu thức chính quy tương đương.

### Lý do thực hiện đề tài:

NFAε là một mô hình quan trọng trong lý thuyết tự động và ngôn ngữ hình thức, được sử dụng rộng rãi trong nhiều lĩnh vực như xử lý ngôn ngữ tự nhiên, xây dựng trình biên dịch, và phân tích chuỗi ký tự. Việc nghiên cứu và mô phỏng hoạt động của NFAε không chỉ giúp hiểu rõ hơn về cấu trúc và cách hoạt động của các hệ thống tự động mà còn mở ra nhiều ứng dụng thực tiễn trong công nghệ thông tin.

NFAε với khả năng hiển thị chi tiết các bước thực hiện. Công cụ này mang lại nhiều giá trị cho:

* Học thuật: Giúp sinh viên và giảng viên dễ dàng minh họa các khái niệm trừu tượng trong lý thuyết tự động.
* Nghiên cứu: Làm nền tảng để phát triển các thuật toán tối ưu hóa và mở rộng nghiên cứu về NFAε và DFA.

### Tính năng:

**Tự động hóa chuyển đổi**: NFAε có khả năng chuyển đổi giữa các trạng thái mà không cần tiêu thụ ký tự đầu vào. Điều này cho phép xây dựng các trạng thái phức tạp hơn mà không ảnh hưởng đến tính đúng đắn của mô hình.

**Tính Không xác định:** Mỗi trạng thái và ký tự đầu vào, có thể có nhiều trạng thái đích, hoặc thậm chí không có trạng thái nào. Tính chất này làm tăng sự linh hoạt trong mô phỏng.

**Biểu diễn ngôn ngữ chính quy:** NFAε có khả năng biểu diễn tất cả các ngôn ngữ chính quy và dễ dàng chuyển đổi thành DFA hoặc biểu thức chính quy tương đương.

## MỤC TIÊU BÀI TOÁN:

### Hiểu và mô phỏng hoạt động của NFAε

Xây dựng một mô hình trực quan hóa cách thức hoạt động của NFAε để minh họa sự thay đổi trạng thái qua từng bước.

Giúp người học hiểu rõ hơn về cơ chế epsilon-closure và các chuyển đổi trạng thái trong NFAε.

### Tạo công cụ mô phỏng hỗ trợ học tập và nghiên cứu:

**Hỗ trợ học thuật**: Cung cấp một công cụ giúp sinh viên, giảng viên, và nhà nghiên cứu hiểu rõ hơn về nguyên tắc hoạt động của NFAε.

**Trợ giúp nghiên cứu**: Làm nền tảng cho việc phát triển các thuật toán tối ưu hóa DFA hoặc nghiên cứu các biến thể của NFA và DFA.

## PHƯƠNG PHÁP THỰC HIỆN:

### Thiết kế Bài toán:

Cấu trúc của thuật toán:

* **Bước 1: Tính epsilon-closure**
  + Epsilon-closure (ε-closure) của một trạng thái trong NFAε là tập hợp tất cả các trạng thái có thể đạt được từ trạng thái đó thông qua các epsilon-chuyển (ε).
  + Mục tiêu: Xác định epsilon-closure cho từng trạng thái trong NFAε.
  + Thuật toán:
    - Bắt đầu từ trạng thái hiện tại và thêm nó vào tập closure.
    - Sử dụng ngăn xếp (stack) để duyệt qua tất cả các trạng thái có thể đạt được thông qua epsilon-chuyển.
    - Nếu một trạng thái chưa nằm trong tập closure, thêm nó vào và tiếp tục duyệt các trạng thái kế tiếp.
* **Bước 2: Mô phỏng hoạt động của NFAε**
  + • Mục tiêu:
    - Theo dõi trạng thái của NFAε khi xử lý một chuỗi đầu vào.
  + • Quy trình:
    - Khởi tạo:
      * Xác định epsilon-closure của trạng thái bắt đầu (start\_state).
    - Xử lý từng ký tự trong chuỗi đầu vào:
    - Từ các trạng thái hiện tại, duyệt qua tất cả các chuyển đổi dựa trên ký tự đầu vào, tạo thành tập trạng thái mới.
    - Tính epsilon-closure của tập trạng thái vừa đạt được sau khi xử lý ký tự.
    - Lặp lại: Thực hiện lặp qua toàn bộ chuỗi đầu vào.
* **Bước 3: Kiểm tra trạng thái kết thúc**
  + Mục tiêu:
    - Xác định xem chuỗi đầu vào có được chấp nhận bởi NFAε hay không.
  + Quy trình:
    - Sau khi xử lý toàn bộ chuỗi đầu vào, kiểm tra xem tập trạng thái cuối cùng có chứa bất kỳ trạng thái chấp nhận nào hay không
* **Bước 4: Hiển thị thông tin chi tiết**
  + • Mục tiêu:
    - Cung cấp thông tin minh bạch về hoạt động của NFAε qua từng bước.
  + • Thông tin hiển thị:
    - Epsilon-closure của từng trạng thái.
    - Các bước chuyển qua từng ký tự đầu vào.
    - Trạng thái trước và sau khi tính epsilon-closure.
    - Kết quả cuối cùng (chấp nhận hoặc không).
* Công cụ phát triển

#### Ngôn ngữ lập trình:

Ngôn ngữ Python.

#### Cấu trúc chương trình:

**Lớp NFAe:**

Đại diện cho một NFAε, bao gồm các thông tin về tập trạng thái, bảng chữ cái, các chuyển đổi, trạng thái bắt đầu, trạng thái chấp nhận, và epsilon-closure.

**Hàm tính toán epsilon-closure:**

Bao gồm các phương thức để tính epsilon-closure cho từng trạng thái hoặc toàn bộ tập trạng thái.

**Hàm mô phỏng:**

Mô phỏng quá trình xử lý chuỗi đầu vào, hiển thị các bước chuyển đổi qua từng ký tự và tính epsilon-closure cho các trạng thái mới đạt được.

**Hàm đọc dữ liệu từ file:**

Đọc cấu hình NFAε từ file để đảm bảo chương trình linh hoạt khi thay đổi dữ liệu.

**Hàm hiển thị:**

In thông tin chi tiết của NFAε và hiển thị các bước xử lý trong quá trình mô phỏng.

#### Thử nghiệm:

Thử nghiệm với các trường hợp NFAε khác nhau:

* Trường hợp đơn giản: NFAε với ít trạng thái và chuyển đổi.
* Trường hợp phức tạp: NFAε có nhiều trạng thái, epsilon-chuyển chồng chéo.

#### Đánh giá kết quả:

* So sánh DFA đầu ra với DFA lý thuyết.
* Đảm bảo DFA là tương đương và tối ưu.

## CÀI ĐẶT:

NFAe được quản lý bằng một lớp bao gồm các thủ tục khởi tạo, gán giá trị, hiển thị dữ liệu đầu vào, khởi tạo các giá trị epsilon-closure cho từng đỉnh, mô phỏng NFAe

### Định Nghĩa NFAe

#### states: Tập hợp các trạng thái của NFAε, được lưu dưới dạng set để dễ dàng kiểm tra và xử lý.

#### alphabet: Bảng chữ cái đầu vào của NFAε (tập hợp các ký tự mà automaton có thể xử lý).

#### transitions: Bảng tra cứu các chuyển đổi, lưu trữ dưới dạng dict với cặp khóa (state, symbol) và giá trị là danh sách các trạng thái đạt được.

#### start\_state: Trạng thái bắt đầu, chỉ định trạng thái khởi đầu khi xử lý chuỗi.

#### accept\_states: Tập các trạng thái chấp nhận, quyết định đầu ra của automaton.

#### epsilon\_closures: Lưu epsilon-closure của từng trạng thái để tránh tính toán lại

### Các Thủ Tục:

#### compute\_epsilon\_closures():

**Chức năng:**

Tính toán epsilon-closure cho tất cả các trạng thái trong NFAε và lưu kết quả vào thuộc tính epsilon\_closures.

**Cách hoạt động:**

Duyệt qua tất cả các trạng thái trong states.

Gọi phương thức epsilon\_closure(state) để tính epsilon-closure cho từng trạng thái.

Lưu kết quả dưới dạng dict với khóa là trạng thái và giá trị là tập hợp các trạng thái trong epsilon-closure.

#### epsilon\_closure(state):

**Chức năng:**

Tính epsilon-closure cho một trạng thái cụ thể bằng cách duyệt tất cả các chuyển đổi epsilon (ε).

**Cách hoạt động:**

Bắt đầu với tập ban đầu closure chỉ chứa trạng thái đầu vào.

Dùng stack để duyệt các trạng thái trong quá trình tìm epsilon-closure.

Kiểm tra tất cả các trạng thái có thể đạt được từ state qua epsilon.

Nếu trạng thái chưa nằm trong closure, thêm vào và tiếp tục duyệt.

#### simulate(input\_string):

**Chức năng:**

Mô phỏng hoạt động của NFAε trên một chuỗi đầu vào, kiểm tra xem chuỗi có được chấp nhận không và hiển thị các bước xử lý.

**Cách hoạt động:**

Khởi tạo:

Tính toán epsilon-closure của trạng thái bắt đầu để xác định trạng thái hiện tại.

Xử lý từng ký tự:

Với mỗi ký tự trong chuỗi, tìm tập trạng thái mới dựa trên các chuyển đổi trong transitions.

Sau đó, tính epsilon-closure cho các trạng thái mới.

Kiểm tra trạng thái cuối:

Nếu tập trạng thái cuối cùng giao với accept\_states không rỗng, chuỗi được chấp nhận.

#### read\_nfae\_from\_file(filename):

Chức năng:

Đọc dữ liệu cấu hình NFAε từ file và khởi tạo đối tượng NFAe.

Cách hoạt động:

Đọc các dòng trong file.

Tách và lưu trữ các thành phần: states, alphabet, start\_state, accept\_states.

Xử lý các dòng còn lại để tạo bảng tra cứu transitions.

#### print\_nfae\_file(filename):

Chức năng:

Hiển thị nội dung file cấu hình NFAε để kiểm tra dữ liệu đầu vào.

Hoạt động:

Đọc và in toàn bộ nội dung file ra màn hình.

### Cài đặt:

#### Sơ Lược Về Chương Trình:



#### Hàm khởi tạo lớp NFAe

Hàm khởi tạo này thiết lập các thuộc tính cơ bản của một NFAε:

states: Tập hợp các trạng thái của máy (VD: {A, B, C}).

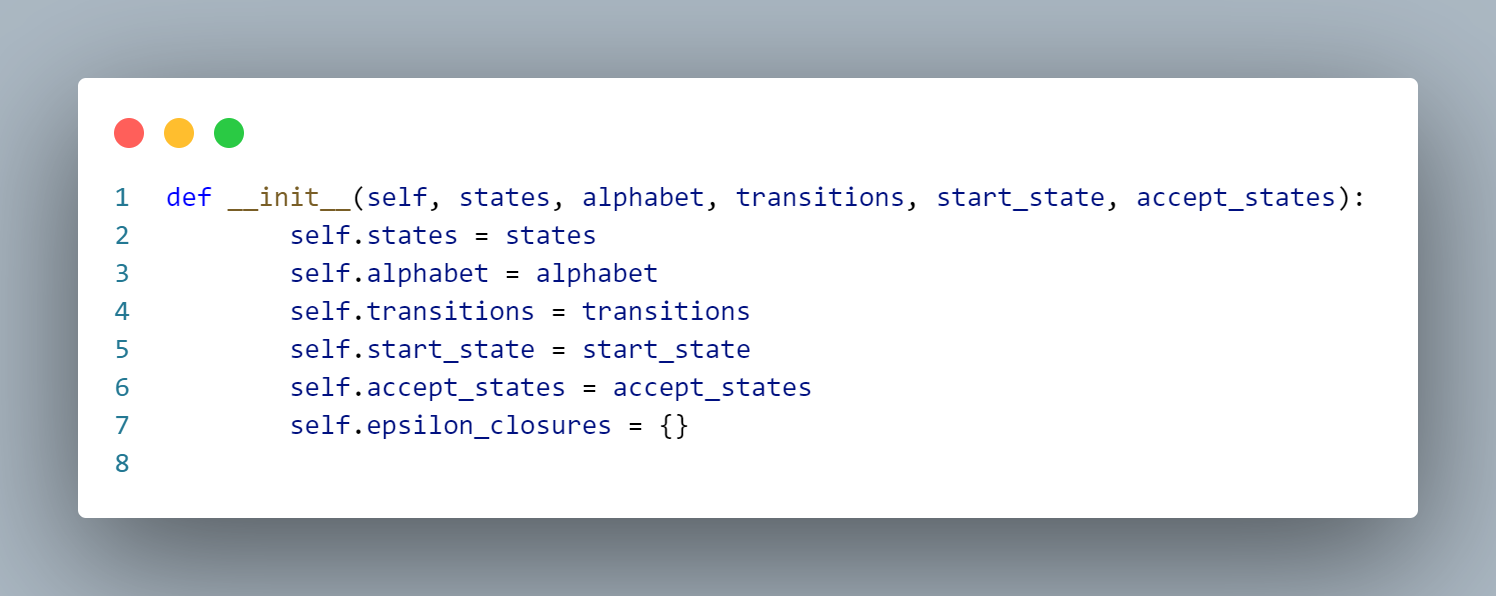
alphabet: Bảng chữ cái đầu vào, không bao gồm epsilon (VD: {'a', 'b'}).

transitions: Các quy tắc chuyển trạng thái (VD: {(A, 'a'): [B], (A, ''): [C]}).

start\_state: Trạng thái bắt đầu (VD: A).

accept\_states: Tập trạng thái chấp nhận (VD: {C}).

epsilon\_closures: Lưu trữ epsilon-closure của từng trạng thái (tính toán một lần, dùng nhiều lần).



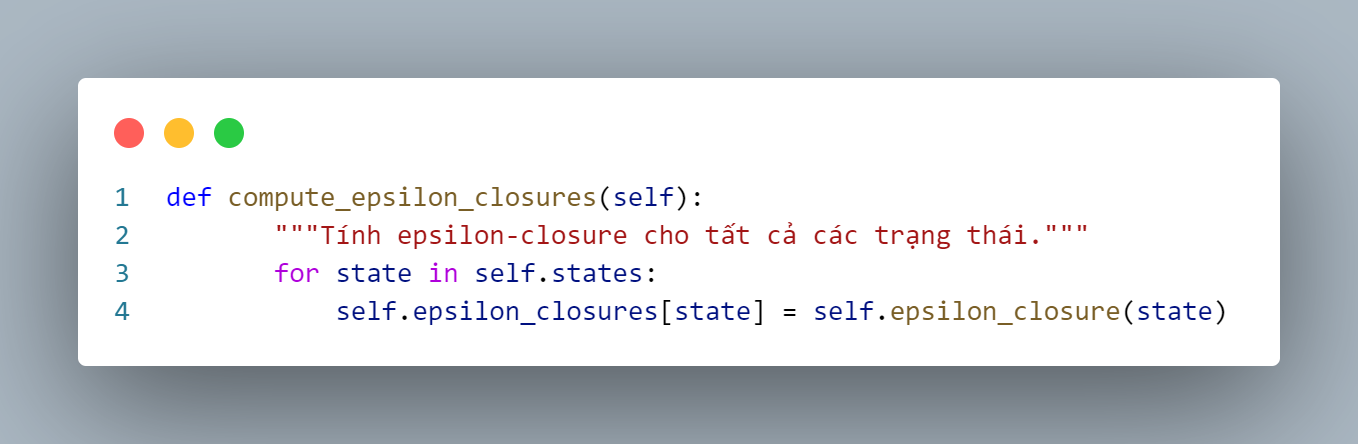
#### Hàm compute\_epsilon\_closures:

Chi tiết hoạt động:

Duyệt qua từng trạng thái trong self.states.

Gọi hàm epsilon\_closure (xem mục tiếp theo) để tính epsilon-closure của trạng thái đó.

Lưu kết quả vào self.epsilon\_closures.



#### Hàm epsilon-closure

**Giải thích thuật toán**

* **Khởi tạo:**
  + closure: Một tập hợp chứa chính trạng thái đầu vào.
  + stack: Một danh sách để duyệt các trạng thái liên kết epsilon.

dùng

* **Lặp:**
  + Lấy một trạng thái từ stack.
  + Lấy tất cả các trạng thái có thể đạt được thông qua epsilon từ trạng thái hiện tại (self.transitions.get((current, ''), [])).
  + Nếu trạng thái mới chưa có trong closure, thêm vào closure và đẩy vào stack để tiếp tục duyệt.
* **Kết thúc:**
  + Trả về closure là tập hợp tất cả các trạng thái có thể đạt được.



#### Hàm simulate:

**Tính toán trạng thái khởi đầu:**

Dùng epsilon\_closures để tìm tất cả các trạng thái có thể đạt được từ trạng thái bắt đầu mà không cần đọc ký tự nào.

**Vòng lặp xử lý chuỗi đầu vào:**

Duyệt từng ký tự trong chuỗi đầu vào.

**Xử lý từng trạng thái:**

Nếu trạng thái hiện tại có một chuyển trạng thái tương ứng với ký tự đầu vào, lấy các trạng thái đích.

Tính epsilon-closure cho các trạng thái đích để tìm tập trạng thái mới.

**Kiểm tra chấp nhận:**

Sau khi xử lý chuỗi, kiểm tra xem có trạng thái nào trong current\_states thuộc accept\_states.

**In kết quả**:

Ghi lại trạng thái qua từng bước và kết quả cuối cùng để hỗ trợ kiểm tra.



## KẾT QUẢ ĐẠT ĐƯỢC:

### Công cụ mô phỏng hoạt động NFAe

#### Tính năng chính:

Kiểm tra xem chuỗi có được chấp nhận bởi NFAe hay không và in cách hoạt động để cho thấy sự chấp nhận đó.

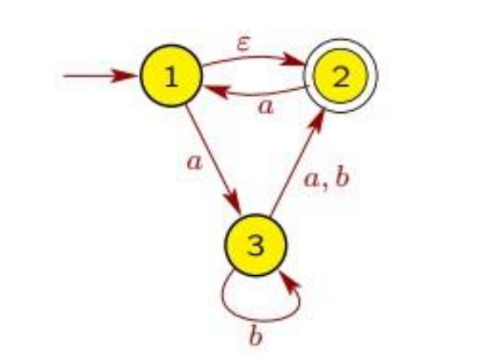
#### Kết quả đầu ra:

Chuỗi có được chấp nhận hay không và các bước hoạt động.

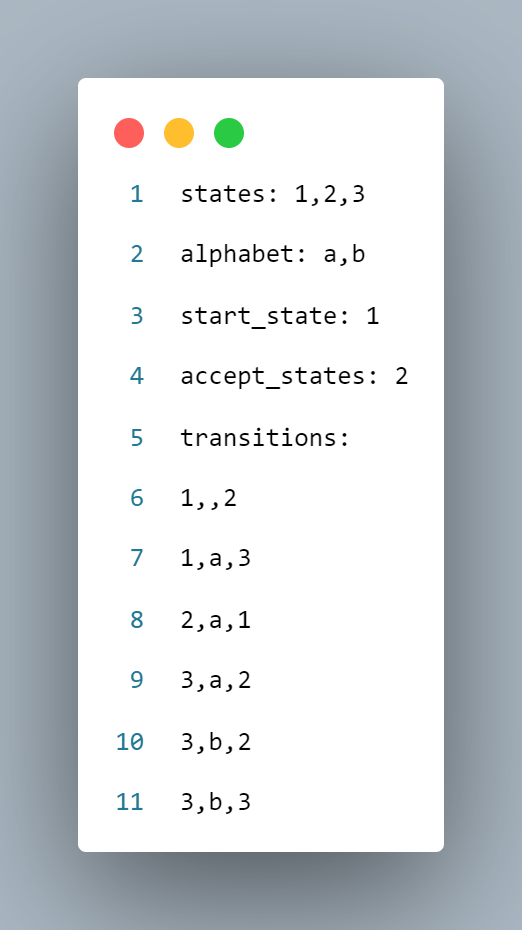
### Giao diện:

#### Hiển thị kết quả:

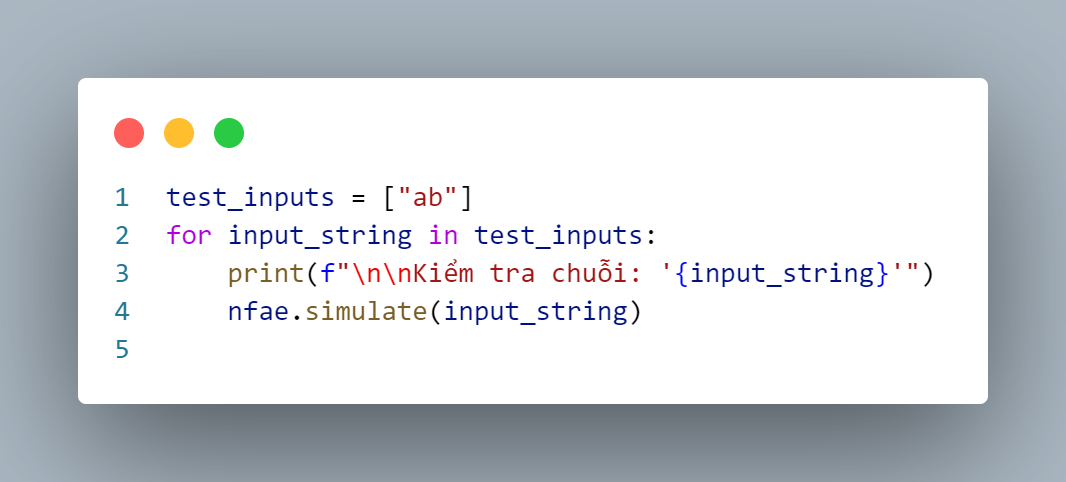
Giả sử NFAe như hình:



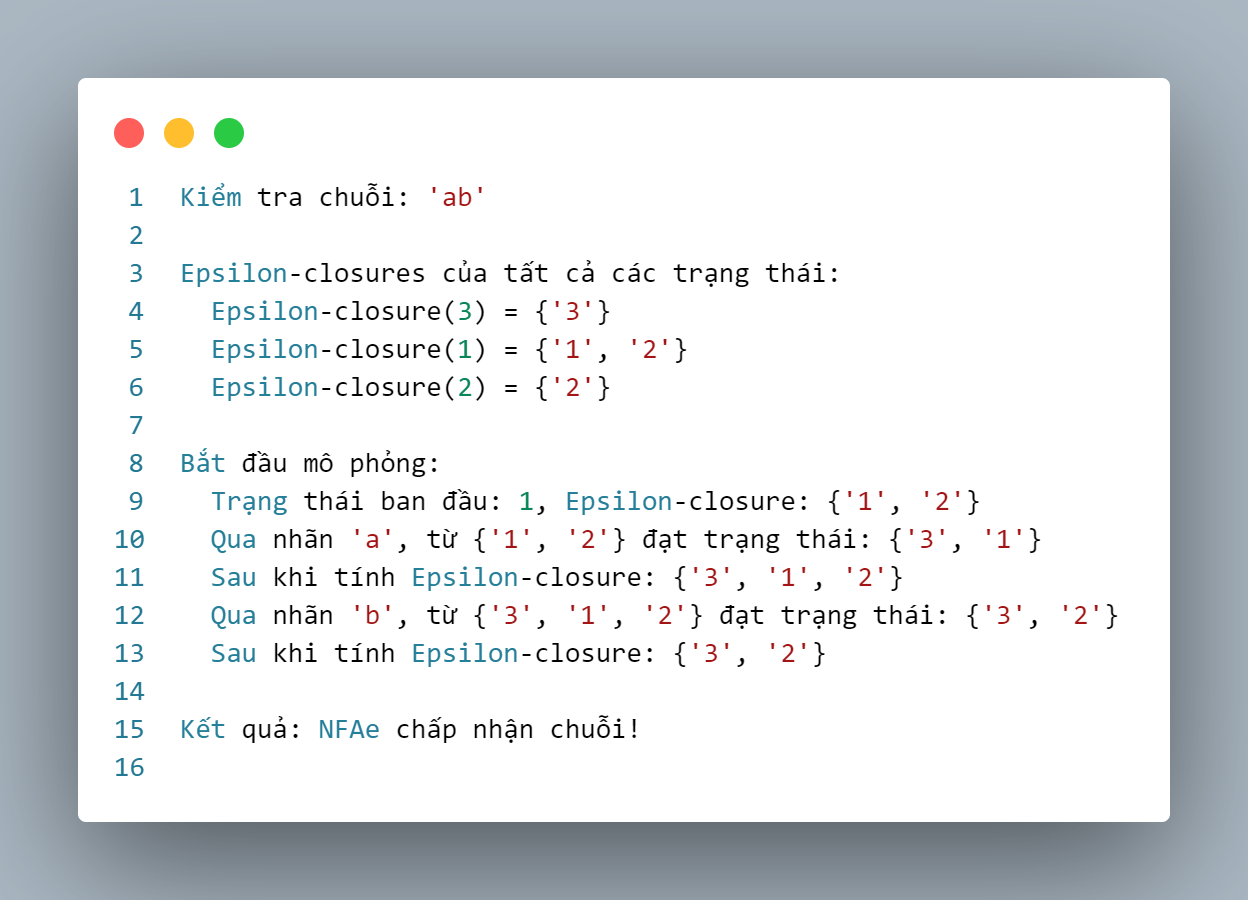
Cấu hình file:



Kiểm tra chuỗi ab có được chấp nhận hay không:



Kết quả đầu ra:



## HƯỚNG PHÁT TRIỂN:

### Mở rộng tính năng của công cụ

Nhập dữ liệu NFAε từ các định dạng chuẩn như:

* File JSON, XML hoặc CSV.
* Đồ thị trạng thái trực quan được tải lên từ các công cụ đồ họa khác.

### Tăng cường trực quan hóa:

#### Biểu đồ tương tác

Phát triển giao diện đồ họa tương tác, cho phép người dùng di chuyển và tùy chỉnh các trạng thái và cạnh trong đồ thị NFAε. Quan sát từng bước trong quá trình tính toán epsilon-closure.

### Ứng dụng trong Giáo dục

#### Công cụ học tập trực tuyến

Xây dựng các bài tập tương tác cho sinh viên có thể tự nhập NFAε và quan sát trực tiếp cách hoạt động của nó.

#### Mở rộng tài liệu hướng dẫn

Cung cấp tài liệu học tập chi tiết và giải thích thuật toán và lý thuyết đằng sau quá trình chuyển đổi và minh họa bằng các ví dụ thực tế và hình ảnh trực quan.

## TÀI LIỆU THAM KHẢO:

Giáo trình: Tin học lý thuyết (Võ Huỳnh Trâm) xuất bản 2009, Đại Học Cần Thơ, Trường Công Nghệ Thông Tin Và Truyền Thông.

Hopcroft, Ullman. Introduction to Automata Theory, Languages, and Computation.