

# BÀI 2: MÃ HOÁ VỚI PYTHON

Bài thực hành này cung cấp cho sinh viên các kiến thức cơ bản về mật mã học cùng với các mật mã cổ điển như: Mật mã Caesar, Mật mã Vigenère, Mật mã Railfence, Mật mã Transposition. Nội dung thực hành sẽ tập trung vào việc sử dụng framework Flask trong Python để tạo các API và giao diện trang web cho việc mã hoá, giải mã. Sau đó, sử dụng công cụ Postman để kiểm thử các API phía trên, tạo tiền đề cho việc sử dụng các API này trong các bài thực hành kế tiếp.

## 2.1 MẬT MÃ HỌC

### 2.1.1 Mật mã Caesar

"Mật mã Caesar, còn được gọi là mã hoá dịch chuyển, là một phương pháp đơn giản để mã hoá thông điệp bằng cách dịch chuyển các ký tự trong bản rõ (plaintext) sang các ký tự khác trong bản mã (ciphertext). Phương pháp này được đặt tên theo Julius Caesar, nhà lãnh đạo quân đội La Mã cổ đại, người được cho là đã sử dụng nó để giao tiếp bí mật" [8].

Cách thức hoạt động của mật mã Caesar rất đơn giản. Mỗi ký tự trong bản rõ được dịch chuyển một số lượng cố định các vị trí theo bảng chữ cái. Ví dụ, nếu chúng ta sử dụng một dịch chuyển (shift) của 3, "A" sẽ được mã hoá thành "D", "B" thành "E", và tiếp tục như vậy. Quá trình giải mã được thực hiện bằng cách dịch ngược lại.

Ví dụ minh họa với một thông điệp và một dịch chuyển là 3:

- Bản rõ (Plaintext): HELLO
- Bản mã (Ciphertext): KHOOR

Để giải mã, chúng ta sẽ dịch ngược lại các ký tự:

- Bản mã (Ciphertext): KHOOR
- Bản rõ (Plaintext): HELLO

"Phép mã hóa cũng có thể được biểu diễn thông qua số học mô-đun, bằng cách ánh xạ các ký tự thành các số nguyên theo thứ tự lần lượt, chẳng hạn A → 0, B → 1, ..., Z → 25. Mã hóa một chữ cái x bằng phép dịch chuyển n vị trí có thể mô tả bằng biểu thức toán học sau:  $E_n(x) = (x + n) \bmod 26$ . Giải mã được mô tả tương tự:  $D_n(x) = (x - n) \bmod 26$ " [8].

Mật mã Caesar có nhược điểm là rất dễ bị giải mã bằng cách thử từng phương án dịch chuyển vì trong bảng chữ cái tiếng Anh chỉ có 26 chữ cái.

## 2.1.2 Mật mã Vigenère

"Mật mã Vigenère là một phương pháp mã hóa chữ cái thông điệp ban đầu bằng cách sử dụng một chuỗi khóa (keyword) để thực hiện các phép dịch chuyển trên các ký tự trong thông điệp. Phương pháp này được phát triển bởi Blaise de Vigenère vào thế kỷ 16 và được coi là một cải tiến đáng kể so với mật mã Caesar" [8].

Cách thức hoạt động của mật mã Vigenère khá phức tạp hơn so với mật mã Caesar. Thay vì sử dụng một dịch chuyển cố định cho tất cả các ký tự trong thông điệp, mật mã Vigenère sử dụng một chuỗi khóa (keyword) để xác định các số lượng dịch chuyển khác nhau cho từng ký tự trong thông điệp.

"Quy trình mã hóa thông điệp bằng mật mã Vigenère bắt đầu với việc lặp lại chuỗi khóa đến khi có độ dài bằng hoặc lớn hơn độ dài của thông điệp. Sau đó, mỗi ký tự trong chuỗi khóa được ánh xạ tương ứng với một số trong bảng chữ cái (thông thường từ A đến Z) và sẽ được sử dụng để dịch chuyển ký tự tương ứng trong thông điệp ban đầu" [8]. Ví dụ: Giả sử chúng ta có một thông điệp là "HELLO" và chuỗi khóa là "KEY". Ta thực hiện mã hóa như sau:

- Thông điệp: H E L L O
- Chuỗi khóa: K E Y K E
- Ký tự mã hóa: V I Q P U

Do đó, thông điệp "HELLO" sau khi được mã hóa bằng mật mã Vigenère với chuỗi khóa "KEY" sẽ trở thành "VIQPU".

Để giải mã một thông điệp được mã hoá bằng mật mã Vigenère, người nhận cần biết chuỗi khoá ban đầu để thực hiện các phép dịch ngược từ các ký tự đã mã hoá, từ đó khôi phục lại nội dung thông điệp gốc.

### 2.1.3 Mật mã Rail Fence

Mật mã Rail Fence, hay còn được gọi là mật mã Zig-zag, là một phương pháp đơn giản để mã hoá thông điệp bằng cách viết các ký tự của thông điệp theo một mô hình đặc biệt trên các "rào cờ" hoặc "rào hàng rào".

"Cách thức hoạt động của mật mã Rail Fence rất đơn giản. Đầu tiên, người gửi chọn một số hàng (thường là 2 hoặc nhiều hơn) để tạo ra các "rào". Sau đó, người gửi viết các ký tự của thông điệp theo đường zigzag trên các rào. Khi đã viết hết thông điệp, người gửi sẽ đọc các ký tự theo thứ tự từ trên xuống dưới để tạo ra bản mã" [8]. Ví dụ: Giả sử chúng ta có một thông điệp là "HELLO WORLD" và chọn 3 hàng để tạo ra rào:

H	.	.	.	O	.	.	.	R	.	.	.
.	E	.	L	.	_	.	O	.	L	.	.
.	.	L	.	.	.	W	.	.	.	D	.

Khi viết các ký tự của thông điệp theo mô hình zigzag trên các rào, chúng ta thu được bản mã là "HOREL OLLWD".

Để giải mã thông điệp bằng mật mã Rail Fence, người nhận cũng cần biết số hàng đã được chọn trước đó. Người nhận sẽ tạo ra các rào tương tự và viết các ký tự của thông điệp mã hoá vào các vị trí tương ứng trên các rào. Sau đó, đọc các ký tự theo thứ tự từ trên xuống dưới để khôi phục lại thông điệp ban đầu. Mật mã Rail Fence rất dễ hiểu và triển khai, tuy nhiên, độ bảo mật của nó không cao.

### 2.1.4 Mật mã Playfair

"Mật mã Playfair là một phương pháp mã hóa chữ cái trong một thông điệp bằng cách sử dụng một ma trận 5x5 chứa các ký tự từ bảng chữ cái. Phương pháp này đã được Charles Wheatstone đề xuất, sau đó được Herbert O. Playfair phát triển và đưa ra các quy tắc cụ thể" [6].

"Bảng Playfair thường được tạo ra bằng cách sắp xếp các ký tự từ bảng chữ cái tiếng Anh vào một ma trận  $5 \times 5$ , loại bỏ các ký tự trùng lặp và thường bổ sung một ký tự đặc biệt (thường là "J") nếu cần thiết để đảm bảo rằng ma trận có đủ 25 ký tự. Một khi ma trận đã được tạo, người gửi và người nhận cần thống nhất trước ma trận Playfair này để mã hóa và giải mã thông điệp. Cách thức mã hóa bằng mật mã Playfair như sau:

- Chia thông điệp cần mã hóa thành các cặp ký tự (nguyên tắc này cần một quy tắc xử lý đặc biệt nếu có số lẻ ký tự).
- Dùng các quy tắc sau để mã hóa từng cặp ký tự:
  - Nếu hai ký tự nằm trên cùng một hàng trong ma trận Playfair, thì sẽ lấy ký tự ở bên phải của mỗi ký tự (có thể trở lại đầu hàng nếu nó nằm ở cuối hàng).
  - Nếu hai ký tự nằm trên cùng một cột trong ma trận Playfair, thì sẽ lấy ký tự ở dưới mỗi ký tự (có thể quay trở lại đầu cột nếu nó nằm ở cuối cột).
  - Nếu hai ký tự không nằm trên cùng một hàng hoặc cột, ta sẽ tạo một hình chữ nhật với hai ký tự đó trong ma trận Playfair và chọn ký tự ở góc đường chéo còn lại của hình chữ nhật đó.

Để giải mã, người nhận thực hiện ngược lại quy trình trên ma trận Playfair" [8].

Ví dụ: Giả sử chúng ta có thông điệp cần mã hóa là "HELLOWORLD". Bảng Playfair thông thường bắt đầu với việc tạo ma trận từ khóa, sau đó điền vào các ký tự còn lại từ bảng chữ cái, loại bỏ "J" nếu khóa chứa ký tự "J". Trong ví dụ này, chúng ta sẽ sử dụng khóa "KEYWORD" để tạo ma trận Playfair:

- Tạo ma trận Playfair từ khóa "KEYWORD":

<b>K</b>	<b>E</b>	<b>Y</b>	<b>W</b>	<b>O</b>
<b>R</b>	<b>D</b>	A	B	C
F	G	H	I/J	L
M	N	P	Q	S
T	U	V	X	Z

- Chia thông điệp thành các cặp ký tự (không cần xử lý đặc biệt cho ký tự trùng lặp): HELLOWORLD → HE LL OW OR LD.
- Xử lý đối với các cặp chứa cùng 1 ký tự (thêm pad character "X"): LL → LX. Ta được kết quả: HELLOWORLD → HE LX OW OR LD.
- Thông điệp "HELLOWORLD" được mã hoá như sau: HE → GY, LX → IZ, OW → KO, OR → KC, LD → GC. Kết quả: "GYIZKOKCGC".

Mật mã Playfair được coi là một bước tiến so với các phương pháp mã hóa cổ điển khác vì nó tạo ra các quy tắc phức tạp hơn cho việc mã hoá, làm cho quá trình giải mã trở nên khó khăn hơn so với một số phương pháp khác.

### 2.1.5 Mật mã Transposition

"Mật mã Transposition là phương pháp mã hóa một thông điệp bằng cách thay đổi vị trí của các ký tự trong thông điệp gốc dựa trên quy tắc nhất định. Thay vì thay đổi giá trị của các ký tự, mật mã này tập trung vào sắp xếp lại thứ tự của chúng" [5], [8]. Có nhiều phương pháp khác nhau để thực hiện mật mã Transposition:

- Đảo ngược hoàn toàn bản rõ: Đây là phương pháp viết lại bản rõ theo thứ tự ngược, từ cuối lên đầu. Ví dụ:
  - Bản rõ: "SECUREEMAIL" → Bản mã: "LIAMEERUCES"
- Mã hóa theo hình mẫu hình học: Văn bản gốc được sắp xếp dựa trên một hình mẫu hình học cụ thể, thường là trong một mảng hoặc ma trận hai chiều. Ví dụ, các ký tự trong văn bản có thể được xếp thành hàng và cột trong một ma trận, sau đó đọc theo thứ tự khác để tạo ra bản mã hóa.
  - Bản rõ: "BAOMAT"
  - Chuyển thành ma trận 2x3:

B	A	O
M	A	T

- Nếu lấy các cột theo thứ tự 2, 3, 1, thì bản mã hóa sẽ là: "AAOTBM".

## 66

### BÀI 2: MÃ HOÁ VỚI PYTHON

- Đổi chỗ cột: Sắp xếp lại các ký tự trong bản rõ thành dạng hình chữ nhật theo cột.

Ví dụ:

- Bản rõ: "BAOMATTHUDIENTU"
- Bản rõ được chuyển thành ma trận kích thước  $3 \times 5$  như sau:

B	M	T	D	N
A	A	H	I	T
O	T	U	E	U

- Với 5 cột, chúng ta có thể sắp xếp chúng theo  $5! = 120$  cách khác nhau. Nếu thực hiện phép hoán đổi các cột theo thứ tự 3, 5, 2, 4, 1 và đọc các ký tự theo hàng ngang, bản mã hóa thu được sẽ là: "TNMDBHTAIAUUTEO".
- Hoán vị các kí tự của bản rõ theo chu kỳ cố định d: Nếu hàm f là hoán vị của một khối gồm d kí tự thì khóa mã hóa được biểu diễn bởi  $K(d, f)$ . Ví dụ:
  - Với  $d = 5$ , hoán vị f của dãy 12345 là 35142.
  - Bản rõ: GROUP. Ta có bảng như sau:

VT ban đầu	VT hoán vị	Nội dung mã hóa	Mã hóa
1	3	G	O
2	5	R	P
3	1	O	G
4	4	U	U
5	2	P	R

Theo ví dụ trên từ bản rõ "GROUP" sẽ mã hóa thành "OPGUR".

## 2.2 GIAO DIỆN CHƯƠNG TRÌNH ỨNG DỤNG (API)

API (Application Programming Interface), hay giao diện lập trình ứng dụng, là tập hợp các quy tắc, hàm, giao thức và công cụ mà lập trình viên dùng để kết nối với một phần mềm hay hệ thống khác. Thông qua API, các ứng dụng có thể giao tiếp và truy cập vào chức năng cũng như dữ liệu của nhau theo cách thức chuẩn hóa.

Các thành phần chính của một API:

- Endpoint (Điểm kết nối): Địa chỉ URL cụ thể cho việc truy cập vào dữ liệu hoặc thực hiện một hành động.
- Method (Phương thức): Định nghĩa những phương thức HTTP như GET, POST, PUT, DELETE để thực hiện các thao tác trên dữ liệu.
- Parameters (Tham số): Thông tin được truyền qua URL hoặc body của request để thực hiện các tác vụ cụ thể.
- Response (Phản hồi): Dữ liệu hoặc thông tin trả về từ API sau khi yêu cầu đã được thực hiện.

## 2.3 POSTMAN

Postman là một ứng dụng phần mềm được sử dụng rộng rãi trong việc phát triển và kiểm thử các API. Được phát triển bởi Postman, công cụ này cung cấp một giao diện người dùng đồ họa (GUI) dễ sử dụng để tạo, kiểm thử, gỡ lỗi và tài liệu hóa các API.

Các tính năng chính của Postman bao gồm:

- Tạo và quản lý các yêu cầu API: Postman cho phép người dùng tạo các yêu cầu HTTP GET, POST, PUT, DELETE và các loại yêu cầu khác.
- Kiểm thử và gỡ lỗi API: Postman cung cấp khả năng kiểm tra và gỡ lỗi các yêu cầu API.
- Tạo và chia sẻ bộ sưu tập (Collection): Người dùng có thể tổ chức các yêu cầu API vào các bộ sưu tập, từ đó tạo ra các bộ test case hoặc tài liệu API.
- Tạo tài liệu API (API Documentation): Postman cho phép tạo tài liệu tự động từ các yêu cầu và bộ sưu tập đã được tạo.

- Kiểm thử tự động (Automated Testing): Postman cung cấp khả năng tạo các bộ test case tự động cho API thông qua việc sử dụng các bộ kiểm thử (test scripts) sử dụng JavaScript.

## 2.4 PYTHON FLASK FRAMEWORK

---

Flask là một framework web Python nhẹ, linh hoạt và dễ sử dụng để xây dựng ứng dụng web. Được tạo ra bởi Armin Ronacher, Flask được thiết kế để đơn giản hóa việc phát triển web trong Python bằng cách cung cấp một cấu trúc cơ bản nhưng mạnh mẽ để xây dựng các ứng dụng web từ những dự án nhỏ đến các hệ thống lớn hơn.

Các đặc điểm chính của Flask bao gồm:

- Linh hoạt và dễ sử dụng: Flask là một framework rất linh hoạt và có cấu trúc đơn giản.
- Microframework: Flask được gọi là một "microframework" vì nó không yêu cầu các thư viện hoặc công cụ cụ thể.
- Routing đơn giản: Flask cung cấp một cách tiếp cận dễ dàng cho việc xác định các URL (routing) và xử lý các yêu cầu HTTP đến ứng dụng.
- Template engine Jinja2: Flask sử dụng Jinja2 làm template engine, cho phép chúng ta tạo các template HTML linh hoạt và tái sử dụng được.
- Hỗ trợ mở rộng: Mặc dù Flask rất nhẹ nhàng, nhưng nó cung cấp cơ chế mở rộng mạnh mẽ thông qua việc tích hợp các extensions.

## 2.5 BÀI TẬP THỰC HÀNH

---

### 2.5.1 Bài thực hành 01: Mã hoá, giải mã Caesar

Viết chương trình mã hoá và giải mã sử dụng Mật mã Caesar với Flask Framework.

Hướng dẫn:

- Clone Git repo của Bài 01 về máy tính thực hành (Mở trong CMD):

```
git clone <đường dẫn git repo của sinh viên>
cd .\bmstt-nc-hutech-1511060249\
code .
```

- Tại giao diện VS Code, mở Terminal, pull mới code về:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\PHUOCNTMH\bmmt-nc-hutech-1511060249> git pull origin main
From https://github.com/mrphuoc020597/bmmt-nc-hutech-1511060249
 * branch      main      -> FETCH_HEAD
Already up to date.
PS C:\Users\PHUOCNTMH\bmmt-nc-hutech-1511060249>
```

- Tách nhánh lab02 từ nhánh main.

```
git checkout -b lab02
```

- Tạo folder “lab-02”. Trong folder “lab-02” tạo folder “ex01”.
- Trong folder "ex01" tạo file “requirements.txt”. Nội dung như sau:

```
1 Flask>=2.3.2
```

- Vào Terminal, chạy các lệnh sau:

```
cd lab-02\ex01
pip install -r ./requirements.txt
```

```
PS C:\Users\PHUOCNTMH\bmmt-nc-hutech-1511060249\lab-02\ex01> pip install -r .\requirements.txt
Collecting Flask>=2.3.2 (from -r .\requirements.txt (line 1))
  Obtaining dependency information for Flask>=2.3.2 from https://files.pythonhosted.org/packages/36/42/015c23096649b908c809c69388a805a571a3bea44362fe87e33fc3afa01f/flask-3.0.0-py3-none-any.whl.metadata
Collecting flask>=2.3.2 (from -r .\requirements.txt (line 1))
  Obtaining dependency information for flask>=2.3.2 from https://files.pythonhosted.org/packages/c3/fc/254c3e9b5feb89ff5b9876a23218dafbc99c96ac5941e900b71206e6313b/flask-3.0.0-py3-none-any.whl.metadata
Collecting werkzeug>=3.0.0 (from Flask>=2.3.2->r .\requirements.txt (line 1))
  Obtaining dependency information for werkzeug>=3.0.0 from https://files.pythonhosted.org/packages/fa/2a/7f3714cbc6356a0efec525ce7a0613d581072ed6eb53eb7b9754f33db807/blinker-1.7.0-py3-none-any.whl.metadata
  Downloading werkzeug-3.0.1-py3-none-any.whl.metadata (4.1 kB)
Collecting Jinja2>=3.1.2 (from Flask>=2.3.2->r .\requirements.txt (line 1))
  Using cached Jinja2-3.1.2.1-py3-none-any.whl (132 kB)
Collecting itsdangerous>=2.1.2 (from Flask>=2.3.2->r .\requirements.txt (line 1))
  Using cached itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting click>=8.1.3 (from Flask>=2.3.2->r .\requirements.txt (line 1))
  Obtaining dependency information for click>=8.1.3 from https://files.pythonhosted.org/packages/00/2e/d53fa4befbf2cf713304afffc7ca780ce4fc1fd871052771b58311a3229/click-8.1.7-py3-none-any.whl.metadata
  Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting blinker>=1.6.2 (from Flask>=2.3.2->r .\requirements.txt (line 1))
  Obtaining dependency information for blinker>=1.6.2 from https://files.pythonhosted.org/packages/fa/2a/7f3714cbc6356a0efec525ce7a0613d581072ed6eb53eb7b9754f33db807/blinker-1.7.0-py3-none-any.whl.metadata
  Downloading blinker-1.7.0-py3-none-any.whl.metadata (1.9 kB)
Collecting colorama (from click>=8.1.3->flask>=2.3.2->r .\requirements.txt (line 1))
  Using cached colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Collecting MarkupSafe>=2.0 (from Jinja2>=3.1.2->flask>=2.3.2->r .\requirements.txt (line 1))
  Obtaining dependency information for MarkupSafe>=2.0 from https://files.pythonhosted.org/packages/44/44/dbaf65876e258facd65f586dde158387ab89963e7f2235551af9c2e24c2/MarkupSafe-2.1.3-cp312-cp312-win_amd64.whl.metadata
  Downloading MarkupSafe-2.1.3-cp312-cp312-win_amd64.whl.metadata (3.0 kB)
  Downloading MarkupSafe-2.1.3-cp312-cp312-win_amd64.whl (97.0/99.7 kB) 318.4 kB/s eta 0:00:00
  Downloading MarkupSafe-2.1.3-cp312-cp312-win_amd64.whl (97.0/97.0 kB) 38.7 kB/s eta 0:00:00
  Downloading MarkupSafe-2.1.3-cp312-cp312-win_amd64.whl (16 kB)
```

- Trong folder “ex01” tạo folder “cipher”. Trong folder “cipher” tạo folder “caesar”.
- Trong folder “caesar” tạo file “alphabet.py”.

```
1 from string import ascii_uppercase
2 ALPHABET = list(ascii_uppercase)
```

- Trong folder “caesar” tạo file “caesar\_cipher.py”.

```

1 from cipher.caesar import ALPHABET
2
3 class CaesarCipher:
4     def __init__(self):
5         self.alphabet = ALPHABET
6
7     def encrypt_text(self, text: str, key: int) -> str:
8         alphabet_len = len(self.alphabet)
9         text = text.upper()
10        encrypted_text = []
11        for letter in text:
12            letter_index = self.alphabet.index(letter)
13            output_index = (letter_index + key) % alphabet_len
14            output_letter = self.alphabet[output_index]
15            encrypted_text.append(output_letter)
16        return "".join(encrypted_text)
17
18    def decrypt_text(self, text: str, key: int) -> str:
19        alphabet_len = len(self.alphabet)
20        text = text.upper()
21        decrypted_text = []
22        for letter in text:
23            letter_index = self.alphabet.index(letter)
24            output_index = (letter_index - key) % alphabet_len
25            output_letter = self.alphabet[output_index]
26            decrypted_text.append(output_letter)
27        return "".join(decrypted_text)

```

- Trong folder “caesar” tạo file “\_\_init\_\_.py”.

```

1 from .alphabet import ALPHABET
2 from .caesar_cipher import CaesarCipher

```

- Trong folder “ex01” tạo file “api.py”.

```

1 from flask import Flask, request, jsonify
2 from cipher.caesar import CaesarCipher
3 app = Flask(__name__)
4
5 #CAESAR CIPHER ALGORITHM
6 caesar_cipher = CaesarCipher()
7
8 @app.route("/api/caesar/encrypt", methods=["POST"])
9 def caesar_encrypt():
10     data = request.json
11     plain_text = data['plain_text']
12     key = int(data['key'])
13     encrypted_text = caesar_cipher.encrypt_text(plain_text, key)
14     return jsonify({'encrypted_message': encrypted_text})
15

```

```

16 @app.route("/api/caesar/decrypt", methods=["POST"])
17 def caesar_decrypt():
18     data = request.json
19     cipher_text = data['cipher_text']
20     key = int(data['key'])
21     decrypted_text = caesar_cipher.decrypt_text(cipher_text, key)
22     return jsonify({'decrypted_message': decrypted_text})
23
24 #main function
25 if __name__ == "__main__":
26     app.run(host="0.0.0.0", port=5000, debug=True)

```

- Tại Terminal, quay về folder "ex01". Tiến hành kiểm tra:

```
python .\api.py
```

```

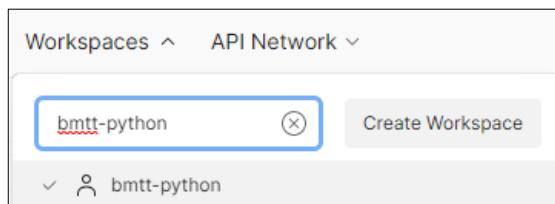
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-02> cd .\ex01\
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-02\ex01> python .\api.py
* Serving Flask app 'api'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.16.6.9:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 929-074-328

```

- Ứng dụng web đang chạy với địa chỉ <http://127.0.0.1:5000>.

- Cài đặt ứng dụng Postman tại địa chỉ: <https://www.postman.com/downloads/>. Chọn phiên bản phù hợp với hệ điều hành đang dùng.
- Tiến hành tải file cài đặt và cài đặt phần mềm vào máy tính. Sau khi cài đặt xong, khởi động Postman. Đăng nhập bằng tài khoản Github của sinh viên.
- Tại menu “Workspaces” chọn “Create Workspace” với tên là “bmtt-python”.



- Trong tab “Environments” ➔ Chọn “Globals”. Tạo mới một giá trị như sau:
  - Variable: base\_url
  - Type: default
  - Initial value/Current value: <http://127.0.0.1:5000/api>

Variable	Type	Initial value	Current value
base_url	default	http://127.0.0.1:5000/api	http://127.0.0.1:5000/api

- Trong tab “Collections” ➔ Chọn “+” ➔ “Blank collection” ➔ Đặt tên là “CAESAR”.
- Chọn nút “New” ➔ “HTTP”. Cấu hình như sau và lưu lại với tên “encrypt” trong collection “CAESAR”.
  - Method: POST (`{{base_url}}`}/caesar/encrypt)
  - Header: Thêm giá trị Key-Value sau: Content-Type application/json
  - Body: Chọn “raw”. Nhập giá trị sau:

The screenshot shows the Postman interface with a POST request named "POST encrypt". The URL is {{base\_url}}/caesar/encrypt. The "Body" tab is selected, showing a JSON payload:

```

1 {
2   "plain_text": "HUTECH",
3   "key": "3"
4 }

```

- Hoặc có thể mở cửa sổ "</> Code snippet" và dùng lệnh cURL sau:

The screenshot shows a "Code snippet" window with the following cURL command:

```

curl --location 'http://127.0.0.1:5000/api/caesar/encrypt' \
--header 'Content-Type: application/json' \
--data '{
  "plain_text": "HUTECH",
  "key": "3"
}'

```

- Bấm nút "Send", thu được kết quả như sau:

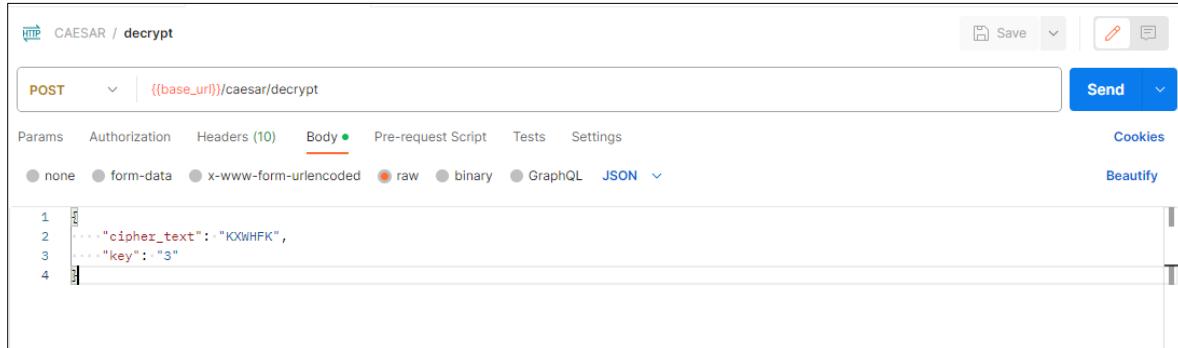
The screenshot shows the Postman interface after sending the request. The "Body" tab is selected, displaying the response:

```

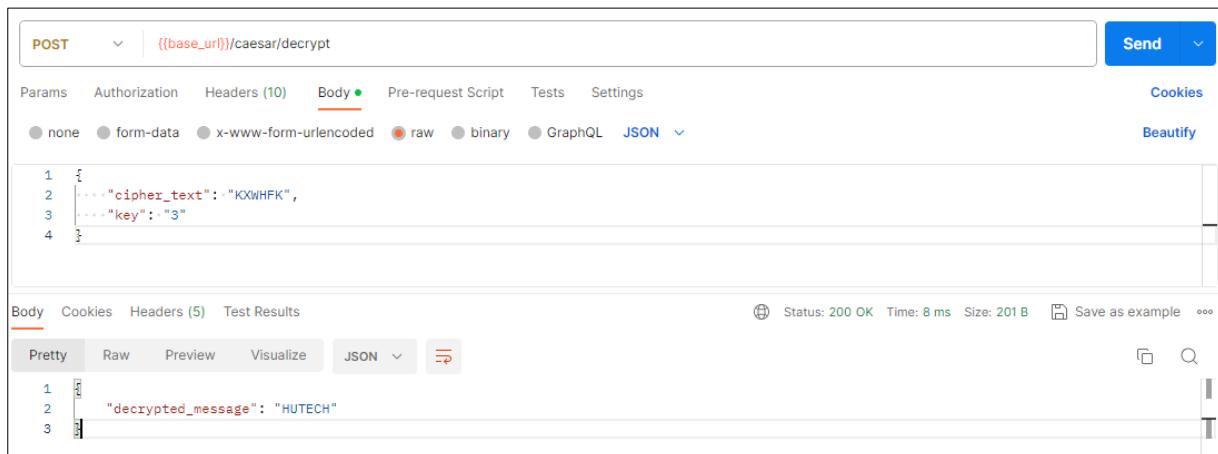
1 {
2   "encrypted_message": "KXWHFK"
3 }

```

- Tiến hành thay đổi nội dung của “plain\_text” và “key”, sau đó thử lại.
- Tạo mới HTTP Request (POST) cho API /decrypt như sau:



- Bấm nút “Send”, ta thu được kết quả như sau:



- Tiến hành thay đổi nội dung của “cipher\_text” và “key”, sau đó thử lại.
- Commit các thay đổi lên Git. Ở Terminal của VS Code, chọn chức năng Split Terminal.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\VMACONTH\bt2-nc-hutech-1511060249\lab-02> cd .\ex01
PS C:\Users\VMACONTH\bt2-nc-hutech-1511060249\lab-02\ex01> python .\api.py
* Serving Flask app "api"
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.16.0.9:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 929-074-328
127.0.0.1 - [27/Dec/2023 12:32:05] "POST /api/caesar/encrypt HTTP/1.1" 200 -
127.0.0.1 - [27/Dec/2023 12:43:36] "POST /api/caesar/encrypt HTTP/1.1" 200 -
127.0.0.1 - [27/Dec/2023 12:46:42] "POST /api/caesar/decrypt HTTP/1.1" 200 -

```

```

git add .
git commit -m "[add] lab-02 ex01 caesar"

```

## 2.5.2 Bài thực hành 02: Mã hoá, giải mã Vigenère

Viết chương trình mã hoá và giải mã sử dụng Mật mã Vigenère với Flask Framework.

## Hướng dẫn:

- Do chúng ta sẽ sử dụng chung folder “cipher” cho các bài tập thực hành nên tôi sẽ di chuyển cấu trúc folder cho phù hợp. Tiến hành di chuyển các folder và file trong folder “ex01” ra bên ngoài (ngang cấp) với “ex01” sau đó xoá folder “ex01”.
  - Trong folder “cipher” tạo folder “vigenere”. Tạo file “vigenere\_cipher.py” trong folder “vigenere”.

```
1 class VigenereCipher:
2     def __init__(self):
3         pass
4
5     def vigenere_encrypt(self, plain_text, key):
6         encrypted_text = ""
7         key_index = 0
8         for char in plain_text:
9             if char.isalpha():
10                 key_shift = ord(key[key_index % len(key)].upper()) - ord('A')
11                 if char.isupper():
12                     encrypted_text += chr((ord(char) - ord('A') + key_shift) % 26 + ord('A'))
13                 else:
14                     encrypted_text += chr((ord(char) - ord('a') + key_shift) % 26 + ord('a'))
15                 key_index += 1
16             else:
17                 encrypted_text += char
18
19
20     def vigenere_decrypt(self, encrypted_text, key):
21         decrypted_text = ""
22         key_index = 0
23         for char in encrypted_text:
24             if char.isalpha():
```

```
25     key_shift = ord(key[key_index % len(key)].upper()) - ord('A')
26     if char.isupper():
27         decrypted_text += chr((ord(char) - ord('A') -
28             key_shift) % 26 + ord('A'))
29     else:
30         decrypted_text += chr((ord(char) - ord('a') -
31             key_shift) % 26 + ord('a'))
32
33     key_index += 1
34
35     else:
36         decrypted_text += char
37
38     return decrypted_text
```

- Tạo file “`__init__.py`” trong folder “`vigenere`”.

```
1 from .vigenere_cipher import VigenereCipher
```

- Cập nhật nội dung của file “api.py”:

```
1 from cipher.vigenere import VigenereCipher      # Thêm vào phần đầu của
file api.py
2
3 # Thêm đoạn sau vào trước hàm main
4 #VIGENERE CIPHER ALGORITHM
5 vigenere_cipher = VigenereCipher()
6
7 @app.route('/api/vigenere/encrypt', methods=['POST'])
8 def vigenere_encrypt():
9     data = request.json
10    plain_text = data['plain_text']
11    key = data['key']
12    encrypted_text = vigenere_cipher.vigenere_encrypt(plain_text, key)
13    return jsonify({'encrypted_text': encrypted_text})
14
```

```
15 @app.route('/api/vigenere/decrypt', methods=['POST'])
16 def vigenere_decrypt():
17     data = request.json
18     cipher_text = data['cipher_text']
19     key = data['key']
20     decrypted_text = vigenere_cipher.vigenere_decrypt(cipher_text, key)
21     return jsonify({'decrypted_text': decrypted_text})
```

- Khởi động lại development server. Nếu development server đang chạy, nhấn tổ hợp phím Ctrl + C để ngắt.

```
python .\api.py
```

```
PS C:\Users\PHUOCNTMH\bmt-nc-hutech-1511060249\lab-02> python .\api.py
 * Serving Flask app 'api'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.16.6.9:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 929-074-328
```

```
PS C:\Users\PHUOCNTMH\bmt-nc-hutech-1511060249> git add .
PS C:\Users\PHUOCNTMH\bmt-nc-hutech-1511060249> git commit -m "[add] lab-02 ex01 caesar"
[lab02 7f6fac] [add] lab-02 ex01 caesar
5 files changed, 58 insertions(+)
create mode 100644 lab-02/ex01/api.py
create mode 100644 lab-02/ex01/cipher/caesar/_init_.py
create mode 100644 lab-02/ex01/cipher/caesar/alphabet.py
create mode 100644 lab-02/ex01/cipher/caesar/caesar_cipher.py
create mode 100644 lab-02/ex01/requirements.txt
```

- Khởi động Postman, tạo mới collection "VIGENERE" có hai HTTP Request là "encrypt" và "decrypt":

```
curl --location 'http://127.0.0.1:5000/api/vigenere/encrypt' \
--header 'Content-Type: application/json' \
--data '{
  "plain_text": "HUTECH",
  "key": "ABC"
}'
```

```
curl --location 'http://127.0.0.1:5000/api/vigenere/decrypt' \
--header 'Content-Type: application/json' \
--data '{
  "cipher_text": "HVVEDJ",
  "key": "ABC"
}'
```

- Tiến hành bấm nút "Send" để kiểm tra các API. Sau đó, thay đổi nội dung và kiểm tra lại hoạt động của API. Kết quả mã hoá:

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 24 ms Size: 198 B Save as example

Pretty	Raw	Preview	Visualize	JSON
1 { 2   "plain_text": "HUTECH", 3   "key": "ABC" 4 }	1 { 2   "plain_text": "HUTECH", 3   "key": "ABC" 4 }  1 { 2   "encrypted_text": "HVVEDJ" 3 } 4			

- Kết quả giải mã:

```

1 {
2   "cipher_text": "HVVEDJ",
3   "key": "ABC"
4 }

Body Cookies Headers (5) Test Results
Pretty Raw Preview Visualize JSON
1 "decrypted_text": "HUTECH"
2
3

```

Status: 200 OK Time: 9 ms Size: 198 B Save as example

- Commit các thay đổi lên Git:

```
git add .
git commit -m "[add] lab-02 vigenere"
```

### 2.5.3 Bài thực hành 03: Mã hoá, giải mã Rail Fence

Viết chương trình mã hoá và giải mã sử dụng Mật mã Rail Fence với Flask Framework.

Hướng dẫn:

- Trong folder “cipher” tạo folder “railfence”. Trong folder “railfence” tạo file “railfence\_cipher.py”.

```

1 class RailFenceCipher:
2     def __init__(self):
3         pass
4
5     def rail_fence_encrypt(self, plain_text, num_rails):
6         rails = [[] for _ in range(num_rails)]
7         rail_index = 0
8         direction = 1 # 1: down, -1: up
9         for char in plain_text:
10             rails[rail_index].append(char)
11             if rail_index == 0:
12                 direction = 1
13             elif rail_index == num_rails - 1:
14                 direction = -1

```

```

15     rail_index += direction
16     cipher_text = ''.join(''.join(rail) for rail in rails)
17     return cipher_text
18

```

```

19     def rail_fence_decrypt(self, cipher_text, num_rails):
20         rail_lengths = [0] * num_rails
21         rail_index = 0
22         direction = 1
23
24         for _ in range(len(cipher_text)):
25             rail_lengths[rail_index] += 1
26             if rail_index == 0:
27                 direction = 1
28             elif rail_index == num_rails - 1:
29                 direction = -1
30             rail_index += direction
31
32         rails = []
33         start = 0
34         for length in rail_lengths:
35             rails.append(cipher_text[start:start + length])
36             start += length
37

```

```

38     plain_text = ""
39     rail_index = 0
40     direction = 1
41
42     for _ in range(len(cipher_text)):
43         plain_text += rails[rail_index][0]
44         rails[rail_index] = rails[rail_index][1:]
45         if rail_index == 0:
46             direction = 1
47         elif rail_index == num_rails - 1:
48             direction = -1
49         rail_index += direction
50
51     return plain_text
52

```

- Tạo file “`__init__.py`” trong folder “`railfence`”.

```

1 from .railfence_cipher import RailFenceCipher

```

- Cập nhật nội dung của file “api.py”:

```

1 from cipher.railfence import RailFenceCipher      # Thêm vào phần đầu của
file api.py
2
3 # Thêm đoạn sau vào trước hàm main
4 #RAILFENCE CIPHER ALGORITHM
5 railfence_cipher = RailFenceCipher()
6
7 @app.route('/api/railfence/encrypt', methods=['POST'])
8 def encrypt():
9     data = request.json
10    plain_text = data['plain_text']
11    key = int(data['key'])
12    encrypted_text = railfence_cipher.rail_fence_encrypt(plain_text, key)
13    return jsonify({'encrypted_text': encrypted_text})
14

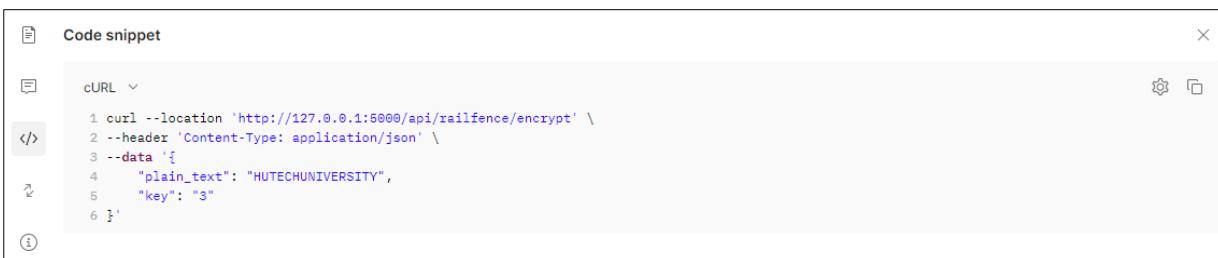
15 @app.route('/api/railfence/decrypt', methods=['POST'])
16 def decrypt():
17     data = request.json
18     cipher_text = data['cipher_text']
19     key = int(data['key'])
20     decrypted_text = railfence_cipher.rail_fence_decrypt(cipher_text, key)
21     return jsonify({'decrypted_text': decrypted_text})

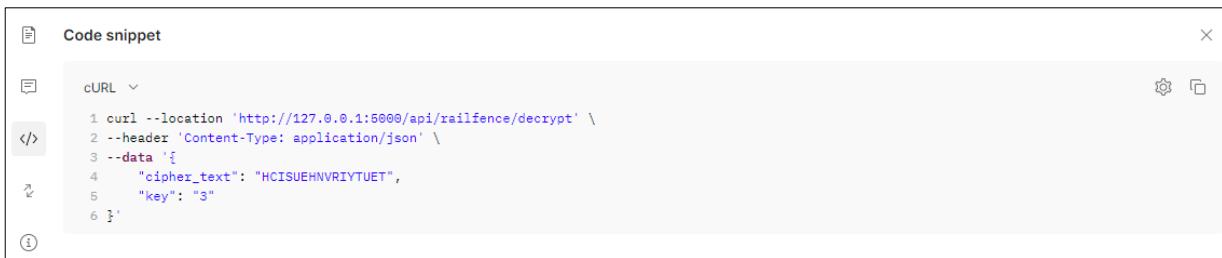
```

- Khởi động lại development server. Nếu development server đang chạy, nhấn tổ hợp phím Ctrl + C để ngắt.

```
python .\api.py
```

- Khởi động Postman, tạo mới collection “RAIL FENCE” có hai HTTP Request là “encrypt” và “decrypt”:



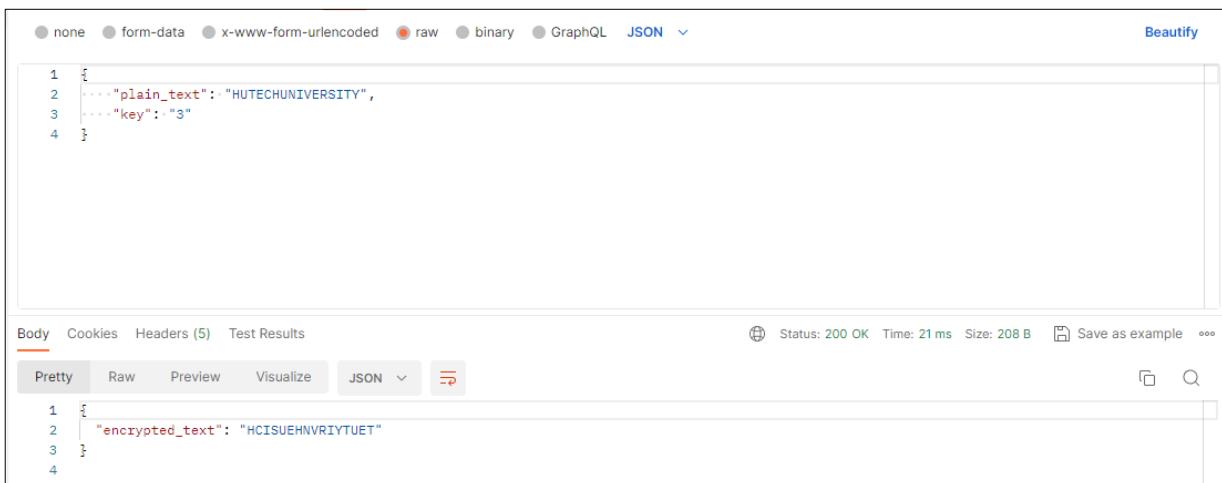


```

Code snippet
curl --location 'http://127.0.0.1:5000/api/railfence/decrypt' \
--header 'Content-Type: application/json' \
--data '{
  "cipher_text": "HCISUEHNVRIFTUET",
  "key": "3"
}'

```

- Tiến hành bấm nút “Send” để kiểm tra các API. Sau đó, thay đổi nội dung và kiểm tra lại hoạt động của API.
- Kết quả mã hóa:

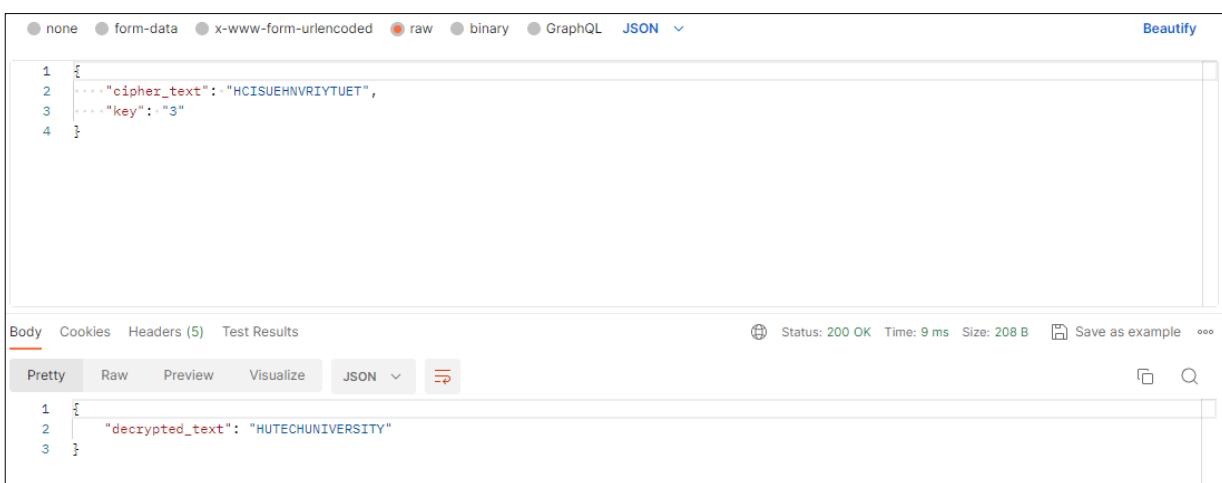


Body Cookies Headers (5) Test Results

Status: 200 OK Time: 21 ms Size: 208 B Save as example

Pretty	Raw	Preview	Visualize	JSON
<pre> 1 { 2   "plain_text": "HUTECHUNIVERSITY", 3   "key": "3" 4 } </pre>				
	<pre> 1 { 2   "encrypted_text": "HCISUEHNVRIFTUET" 3 } </pre>			

- Kết quả giải mã:



Body Cookies Headers (5) Test Results

Status: 200 OK Time: 9 ms Size: 208 B Save as example

Pretty	Raw	Preview	Visualize	JSON
<pre> 1 { 2   "decrypted_text": "HUTECHUNIVERSITY" 3 } </pre>				

- Commit các thay đổi lên Git:

```
git add .
git commit -m "[add] lab-02 railfence"
```

## 2.5.4 Bài thực hành 04: Mã hoá, giải mã Playfair

Viết chương trình mã hoá và giải mã sử dụng Mật mã Playfair với Flask Framework.

Hướng dẫn:

- Trong folder “cipher” tạo folder “playfair”. Trong folder “playfair” tạo file “playfair\_cipher.py”.

```

1 class PlayFairCipher:
2     def __init__(self) -> None:
3         pass
4
5     def __init__(self):
6         pass
7
8     def create_playfair_matrix(self, key):
9         key = key.replace("J", "I") # Chuyển "J" thành "I" trong khóa
10        key = key.upper()
11        key_set = set(key)
12        alphabet = "ABCDEFGHIJKLMNPQRSTUVWXYZ"
13        remaining_letters = [
14            letter for letter in alphabet if letter not in key_set]
15        matrix = list(key)
16
17        for letter in remaining_letters:
18            matrix.append(letter)
19            if len(matrix) == 25:
20                break
21
22        playfair_matrix = [matrix[i:i+5] for i in range(0, len(matrix), 5)]
23        return playfair_matrix
24
25    def find_letter_coords(self, matrix, letter):
26        for row in range(len(matrix)):
27            for col in range(len(matrix[row])):
28                if matrix[row][col] == letter:
29                    return row, col
30

```

```

31     def playfair_encrypt(self, plain_text, matrix):
32         # Chuyển "J" thành "I" trong văn bản đầu vào
33         plain_text = plain_text.replace("J", "I")
34         plain_text = plain_text.upper()
35         encrypted_text = ""
36
37         for i in range(0, len(plain_text), 2):
38             pair = plain_text[i:i+2]
39             if len(pair) == 1: # Xử lý nếu số lượng ký tự lẻ
40                 pair += "X"
41             row1, col1 = self.find_letter_coords(matrix, pair[0])
42             row2, col2 = self.find_letter_coords(matrix, pair[1])
43             if row1 == row2:
44                 encrypted_text += matrix[row1][(col1 + 1) %
45                                         5] + matrix[row2][(col2 + 1) %
46                                         5]
47             elif col1 == col2:
48                 encrypted_text += matrix[(row1 + 1) %
49                                         5][col1] + matrix[(row2 + 1) %
50                                         5][col2]
51             else:
52                 encrypted_text += matrix[row1][col2] + matrix[row2][col1]
53     return encrypted_text
54
55
56
57
58     def playfair_decrypt(self, cipher_text, matrix):
59         cipher_text = cipher_text.upper()
60         decrypted_text = ""
61         decrypted_text1 = ""
62
63         for i in range(0, len(cipher_text), 2):
64             pair = cipher_text[i:i+2]
65             row1, col1 = self.find_letter_coords(matrix, pair[0])
66             row2, col2 = self.find_letter_coords(matrix, pair[1])
67
68             if row1 == row2:
69                 decrypted_text += matrix[row1][(col1 - 1) %
70                                         5] + matrix[row2][(col2 - 1) %
71                                         5]
72             elif col1 == col2:
73                 decrypted_text += matrix[(row1 - 1) %
74                                         5][col1] + matrix[(row2 - 1) %
75                                         5][col2]
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
623
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
15
```

```

69         else:
70             decrypted_text += matrix[row1][col2] + matrix[row2][col1]
71
72     banro = ""
73     # Loại bỏ ký tự 'X' nếu nó là ký tự cuối cùng và là ký tự được thêm
74     # vào
75     for i in range(0, len(decrypted_text)-2, 2):
76         if decrypted_text[i] == decrypted_text[i+2]:
77             banro += decrypted_text[i]
78         else:
79             banro += decrypted_text[i] + "" + decrypted_text[i+1]
80
81         if decrypted_text[-1] == "X":
82             banro += decrypted_text[-2]
83         else:
84             banro += decrypted_text[-2]
85             banro += decrypted_text[-1]
86
87     return banro

```

- Tạo file “\_\_init\_\_.py” trong folder “playfair”.

```
1 from .playfair_cipher import PlayFairCipher
```

- Cập nhật nội dung của file “api.py”:

```

1 from cipher.playfair import PlayFairCipher      # Thêm vào phần đầu của
    file api.py
2
3 # Thêm đoạn sau vào trước hàm main
4 #PLAYFAIR CIPHER ALGORITHM
5 playfair_cipher = PlayFairCipher()
6
7 @app.route('/api/playfair/creatematrix', methods=['POST'])
8 def playfair_creatematrix():
9     data = request.json
10    key = data['key']
11    playfair_matrix = playfair_cipher.create_playfair_matrix(key)
12    return jsonify({"playfair_matrix": playfair_matrix})
13

```

```

14 @app.route('/api/playfair/encrypt', methods=['POST'])
15 def playfair_encrypt():
16     data = request.json
17     plain_text = data['plain_text']
18     key = data['key']
19     playfair_matrix = playfair_cipher.create_playfair_matrix(key)
20     encrypted_text = playfair_cipher.playfair_encrypt(plain_text,
21                                                       playfair_matrix)
22     return jsonify({'encrypted_text': encrypted_text})
23
24 @app.route('/api/playfair/decrypt', methods=['POST'])
25 def playfair_decrypt():
26     data = request.json
27     cipher_text = data['cipher_text']
28     key = data['key']
29     playfair_matrix = playfair_cipher.create_playfair_matrix(key)
30     decrypted_text = playfair_cipher.playfair_decrypt(cipher_text,
31                                                       playfair_matrix)
32     return jsonify({'decrypted_text': decrypted_text})
33

```

- Khởi động lại development server. Nếu development server đang chạy, nhấn tổ hợp phím Ctrl + C để ngắt.

```
python .\api.py
```

- Khởi động Postman, tạo mới collection “PLAY FAIR” có ba HTTP Request là “creatematrix”, “encrypt” và “decrypt”:

```

Code snippet
curl -X POST -H "Content-Type: application/json" -d '{"key": "MONARCHY"}' http://127.0.0.1:5000/api/playfair/creatematrix

```

```

Code snippet
curl -X POST -H "Content-Type: application/json" -d '{"plain_text": "HOMNAYHANHDONG", "key": "MONARCHY"}' http://127.0.0.1:5000/api/playfair/encrypt

```

```

Code snippet
curl --location 'http://127.0.0.1:5000/api/playfair/decrypt' \
--header 'Content-Type: application/json' \
--data '{
  "cipher_text": "FHOANBBOOYHRYQ",
  "key": "MONARCHY"
}'

```

- Tiến hành bấm nút “Send” tại “creatematrix” để tạo ma trận với khoá là “MONARCHY”, ta được kết quả như sau:

Body Headers (5) Test Results

Status: 200 OK Time: 8 ms Size: 531 B Save as example

```

1 {
2   ...
3 }

```

Pretty	Raw	Preview	Visualize	JSON
1 { 2   "playfair_matrix": [ 3     [ 4       "M", 5       "O", 6       "N", 7       "A", 8       "R" 9     ], 10    [ 11      "C", 12      "H", 13      "Y", 14      "B", 15      "D" 16    ],				

- Tiến hành bấm nút “Send” để kiểm tra các API. Sau đó, thay đổi nội dung và kiểm tra lại hoạt động của API.
- Kết quả mã hoá:

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 11 ms Size: 206 B Save as example

```

1 {
2   ...
3 }

```

Pretty	Raw	Preview	Visualize	JSON
	1 2   "encrypted_text": "FHOANBBOOYHRYQ" 3			

- Kết quả giải mã:

```

1 {
2   ...."cipher_text": "FHOANBBOOYHRYQ",
3   ...."key": "MONARCHY"
4 }

```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON  Save as example

```

1 "decrypted_text": "HOMNAYHANHDONG"
2
3

```

- Commit các thay đổi lên Git:

```
git add .
git commit -m "[add] lab-02 playfair"
```

## 2.5.5 Bài thực hành 05: Mã hóa, giải mã Transposition

Viết chương trình mã hóa và giải mã sử dụng Mật mã Transposition với Flask Framework. Hướng dẫn:

- Trong folder “cipher” tạo folder “transposition”. Tạo file “transposition\_cipher.py” trong folder “transposition”.

```

1 class TranspositionCipher:
2     def __init__(self):
3         pass
4
5     def encrypt(self, text, key):
6         encrypted_text = ''
7         for col in range(key):
8             pointer = col
9             while pointer < len(text):
10                 encrypted_text += text[pointer]
11                 pointer += key
12         return encrypted_text
13

```

```

14     def decrypt(self, text, key):
15         decrypted_text = [''] * key
16         row, col = 0, 0
17         for symbol in text:
18             decrypted_text[col] += symbol
19             col += 1
20             if col == key or (col == key - 1 and row >= len(text) % key):
21                 col = 0
22                 row += 1
23         return ''.join(decrypted_text)

```

- Tạo file “\_\_init\_\_.py” trong folder “transposition”.

```
1 from .transposition_cipher import TranspositionCipher
```

- Cập nhật nội dung của file “api.py”:

```

1 from cipher.transposition import TranspositionCipher      # Thêm vào phần đầu
của file api.py
2
3 # Thêm đoạn sau vào trước hàm main
4 #TRANSPOSITION CIPHER ALGORITHM
5 transposition_cipher = TranspositionCipher()
6
7 @app.route('/api/transposition/encrypt', methods=['POST'])
8 def transposition_encrypt():
9     data = request.get_json()
10    plain_text = data.get('plain_text')
11    key = int(data.get('key'))
12    encrypted_text = transposition_cipher.encrypt(plain_text, key)
13    return jsonify({'encrypted_text': encrypted_text})
14
15 @app.route('/api/transposition/decrypt', methods=['POST'])
16 def transposition_decrypt():
17     data = request.get_json()
18     cipher_text = data.get('cipher_text')
19     key = int(data.get('key'))
20     decrypted_text = transposition_cipher.decrypt(cipher_text, key)
21     return jsonify({'decrypted_text': decrypted_text})

```

- Khởi động lại development server. Nếu development server đang chạy, nhấn tổ hợp phím Ctrl + C để ngắt.

```
python .\api.py
```

- Khởi động Postman, tạo mới collection “TRANSPOSITION” có hai HTTP Request là “encrypt” và “decrypt”:

The image contains two separate windows of the Postman application, each titled "Code snippet".  
The top window shows a curl command for the "encrypt" request:

```
curl --location 'http://127.0.0.1:5000/api/transposition/encrypt' \
--header 'Content-Type: application/json' \
--data '{
  "plain_text": "HUTECH",
  "key": "3"
}'
```

The bottom window shows a curl command for the "decrypt" request:

```
curl --location 'http://127.0.0.1:5000/api/transposition/decrypt' \
--header 'Content-Type: application/json' \
--data '{
  "cipher_text": "HEUCTH",
  "key": "3"
}'
```

- Tiến hành bấm nút “Send” để kiểm tra các API. Sau đó, thay đổi nội dung và kiểm tra lại hoạt động của API.
- Kết quả mã hóa:

The screenshot shows the Postman interface after sending the "encrypt" request. The "Body" tab is selected, showing the JSON input:

```
1 {
2   "plain_text": "HUTECH",
3   "key": "3"
4 }
```

Below the body, the "Pretty" tab is selected, showing the resulting JSON output:

```
1 {
2   "encrypted_text": "HEUCTH"
3 }
```

At the top right, the status bar indicates "Status: 200 OK Time: 6 ms Size: 198 B".

- Kết quả giải mã:

```

1 {
2   "cipher_text": "HEUCTH",
3   "key": "3"
4 }

Body Cookies Headers (5) Test Results
Pretty Raw Preview Visualize JSON Status: 200 OK Time: 7 ms Size: 198 B Save as example ...
1 "decrypted_text": "HUTECH"
2
3

```

- Commit các thay đổi lên Git:

```
git add .
git commit -m "[add] lab-02 transposition"
```

## 2.5.6 Bài thực hành 06: Sử dụng Flask tạo giao diện web

Sử dụng Python Flask để tạo ra một ứng dụng web demo cho mã hoá và giải mã bằng Mật mã Caesar. Hướng dẫn:

- Trong folder “lab-02” tạo folder “templates”. Trong folder “templates” tạo file “index.html”.

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <title>Thực hành An toàn thông tin nâng cao</title>
5   <link href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/
bootstrap.min.css" rel="stylesheet"/>
6 </head>
7 <body>
8   <div class="container">
9     
10    <h4 style="font-weight: bold; text-align: center;">BÀI THỰC HÀNH
BẢO MẬT THÔNG TIN NÂNG CAO</h4>
11    <ul style="margin-top: 30px;">
12      <li><a href="/caesar">Ceasar Cipher</a>
13      <li><a href="/rsa">RSA Cipher</a>
14    </ul>
15  </div>
16 </body></html>
```

- Trong folder “templates” tạo file “caesar.html”.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <title>Caesar Cipher</title>
5     <link href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/
6         bootstrap.min.css" rel="stylesheet"/>
7 </head>
8 <body>
9     <div class="container">
10        <table class="table">
11            <tr><td style="text-align: center; font-weight: bold;
12                font-size: 25px;">CAESAR CIPHER</td></tr>
13            <tr><td style="font-weight: bold; color: #blue">ENCRYPTION</
14                td></tr>
15            <tr>
16                <td>
17                    <form method="POST" action="/encrypt">
18                        <div class="mb-3">
19                            <label class="form-label">Plain text:</label>
20                                <input type="text" class="form-control"
21                                    name="inputPlainText" placeholder="Input Plain
22                                    Text" required autofocus/>
23                            </div>
24                            <div class="mb-3">
25                                <label class="form-label">Key:</label>
26                                <input type="number" class="form-control"
27                                    name="inputKeyPlain" placeholder="Input Key"
28                                    required/>
29                            </div>
30                            <button type="submit" class="btn
31                                btn-primary">Encrypt</button>
32                        </form>
33                    </td>
34                </tr>
35
36                <tr><td style="font-weight: bold; color: #blue">DECRYPTION</
37                    td></tr>
38                <tr>
39                    <td>
```

```

31          <form method="POST" action="/decrypt">
32              <div class="mb-3">
33                  <label class="form-label">Cipher text:</label>
34                  <input type="text" class="form-control"
35                      name="inputCipherText" placeholder="Input
36                          Cipher Text" required/>
37              </div>
38              <div class="mb-3">
39                  <label class="form-label">Key:</label>
40                      <input type="number" class="form-control"
41                          name="inputKeyCipher" placeholder="Input Key"
42                          required/>
43                  </div>
44                  <button type="submit" class="btn
45                      btn-success">Decrypt</button>
46          </form>
47      </td>
48  </tr>
49 </table>
50 </div>
51 </body>
52 </html>

```

- Trong folder "lab-02" tạo file "app.py".

```

1 from flask import Flask, render_template, request, json
2 from cipher.caesar import CaesarCipher
3
4 app = Flask(__name__)
5
6 #router routes for home page
7 @app.route("/")
8 def home():
9     return render_template('index.html')
10
11 #router routes for caesar cypher
12 @app.route("/caesar")
13 def caesar():
14     return render_template('caesar.html')
15

```

```

16 @app.route("/encrypt", methods=['POST'])
17 def caesar_encrypt():
18     text = request.form['inputPlainText']
19     key = int(request.form['inputKeyPlain'])
20     Caesar = CaesarCipher()

21     encrypted_text = Caesar.encrypt_text(text, key)
22     return f"text: {text}<br/>key: {key}<br/>encrypted text:
23     {encrypted_text}"

24 @app.route("/decrypt", methods=['POST'])
25 def caesar_decrypt():
26     text = request.form['inputCipherText']
27     key = int(request.form['inputKeyCipher'])
28     Caesar = CaesarCipher()
29     decrypted_text = Caesar.decrypt_text(text, key)
30     return f"text: {text}<br/>key: {key}<br/>decrypted text:
31     {decrypted_text}"

32 #main function
33 if __name__ == "__main__":
34     app.run(host="0.0.0.0", port=5050, debug=True)

```

- Chạy development server.

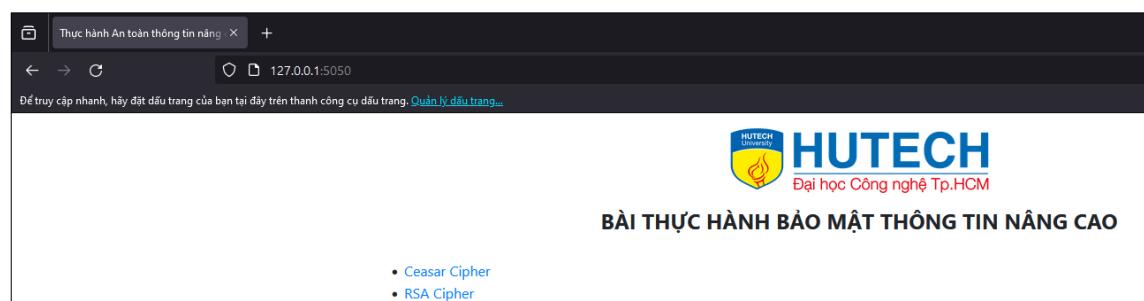


```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-02> python .\app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5050
* Running on http://172.16.6.9:5050
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 929-074-328

```

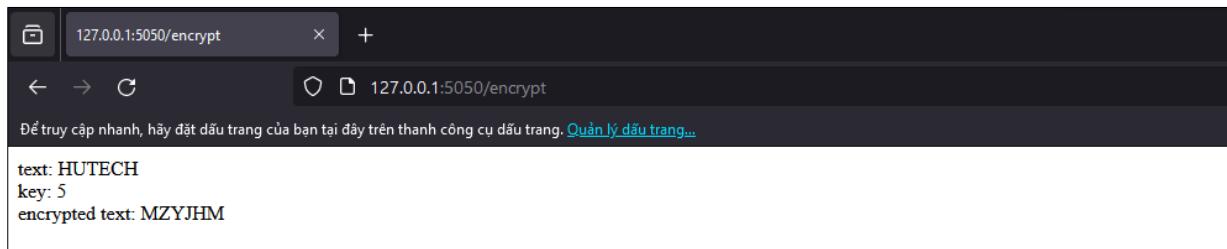
- Sử dụng trình duyệt truy cập địa chỉ <http://127.0.0.1:5050> để kiểm tra, ta thu được kết quả như sau:



## 94

### BÀI 2: MÃ HOÁ VỚI PYTHON

- Chọn vào menu “Caesar Cipher”. Tiến hành kiểm tra các chức năng “Encrypt” và “Decrypt”.
- Kiểm tra mã hoá:



- Kiểm tra giải mã:

### CAESAR CIPHER

**ENCRYPTION**

Plain text:

Key:

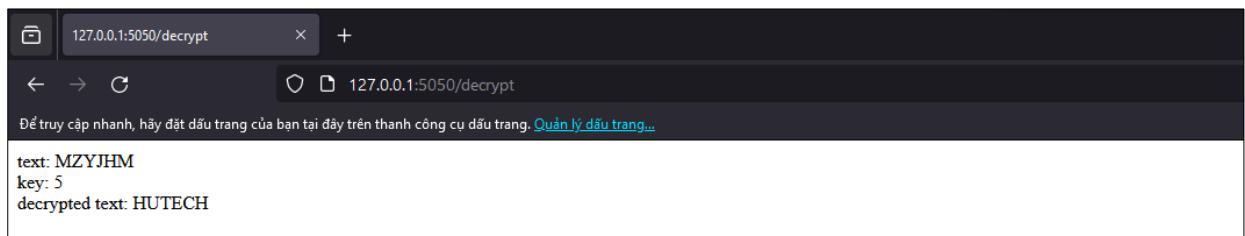
**Encrypt**

**DECRYPTION**

Cipher text:

Key:

**Decrypt**



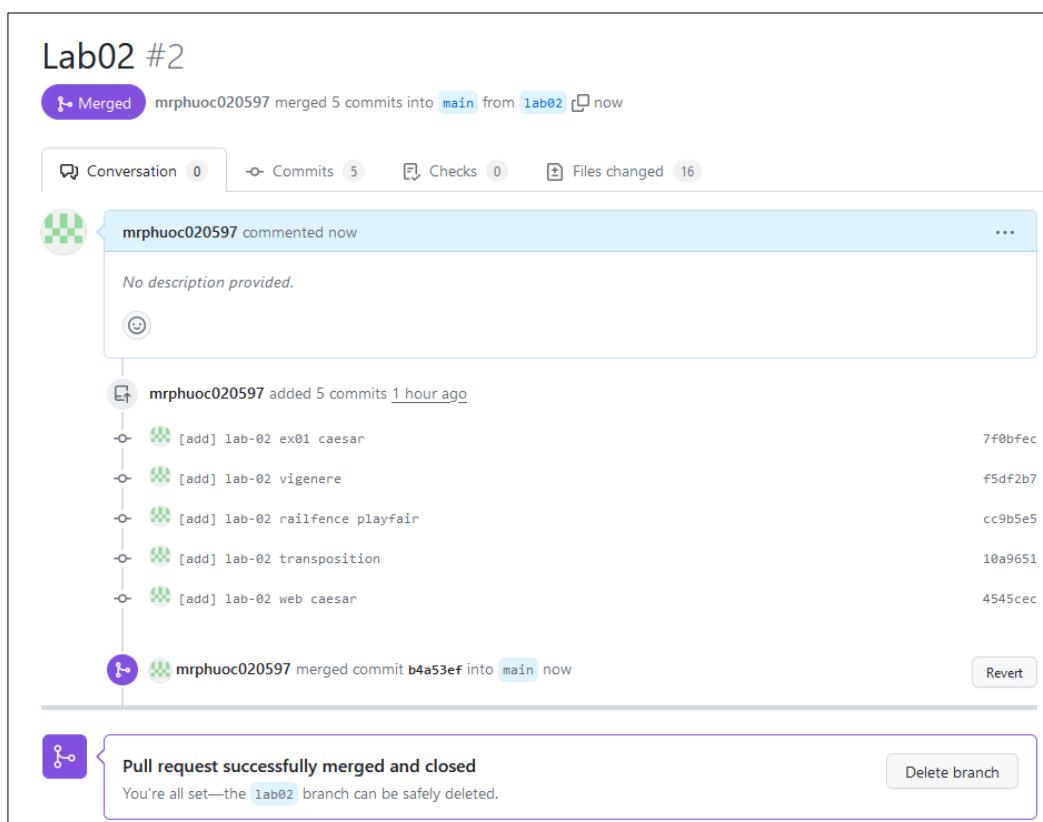
- Commit các thay đổi lên Git:

```
git add .
git commit -m "[add] lab-02 web caesar"
```

- Push các thay đổi lên remote repo:

```
git push origin lab02
```

- Tiến hành tạo PR từ nhánh lab02 về nhánh main. Review, kiểm tra đủ số lượng file, sau đó tiến hành “Merge pull request”.



## 2.6 BÀI TẬP MỞ RỘNG

- **Câu 01:** Sử dụng Python Flask để tạo ra một ứng dụng web demo cho mã hóa và giải mã bằng Mật mã Vigenère, Rail Fence, Playfair, Transposition.
- **Câu 02:** Kết nối các bài tập đã thực hiện ở Câu 01 trên danh mục lựa chọn (menu) của Bài thực hành số 06.