

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**  
**ĐẠI HỌC CÔNG NGHỆ TP.HCM**

**THỰC HÀNH BẢO MẬT  
THÔNG TIN NÂNG CAO**

**Biên Soạn:**

**TS. Văn Thiên Hoàng**

**ThS. Nguyễn Trọng Minh Hồng Phước**

**KS. Đặng Thị Thạch Thảo**

**KS. Đinh Huỳnh Tuệ Tuệ**

**THỰC HÀNH BẢO MẬT THÔNG TIN NÂNG CAO**

**\*1.2024.COS339\***

---

Các ý kiến đóng góp về tài liệu học tập này, xin gửi về e-mail của ban biên tập:  
[tailieuhoctap@hutech.edu.vn](mailto:tailieuhoctap@hutech.edu.vn)

# MỤC LỤC

<b>MỤC LỤC .....</b>	<b>I</b>
<b>HƯỚNG DẪN .....</b>	<b>IV</b>
<b>BÀI 1: LẬP TRÌNH CƠ BẢN VỚI PYTHON .....</b>	<b>1</b>
<b>1.1 NGÔN NGỮ PYTHON .....</b>	<b>1</b>
<b>1.2 THÀNH PHẦN CỦA PYTHON .....</b>	<b>2</b>
1.2.1 Biến, từ khóa .....	2
1.2.2 Kiểu dữ liệu .....	5
1.2.3 Các toán tử số học .....	6
1.2.4 Các toán tử logic .....	8
1.2.5 Nhập, xuất dữ liệu .....	9
1.2.6 Các cấu trúc điều khiển .....	10
1.2.7 Chuỗi .....	12
1.2.8 Hàm (Function) .....	14
<b>1.3 KIỂU DỮ LIỆU CÓ CẤU TRÚC .....</b>	<b>15</b>
1.3.1 Mảng (Array) .....	15
1.3.2 Danh sách (List) .....	17
1.3.3 Kiểu Tuple .....	18
1.3.4 Kiểu Dictionary .....	19
<b>1.4 OOP TRONG PYTHON .....</b>	<b>21</b>
1.4.1 Lớp (Class) và đối tượng (Object) .....	21
1.4.2 Khởi tạo (Constructor) .....	22
1.4.3 Thuộc tính (Attributes) .....	23
1.4.4 Phương thức (Methods) .....	23
1.4.5 Kế thừa (Inheritance) .....	24
1.4.6 Đa hình (Polymorphism) .....	25
1.4.7 Trừu tượng hóa (Abstraction) .....	26
<b>1.5 CÔNG CỤ THỰC HÀNH .....</b>	<b>27</b>
1.5.1 Trình biên dịch Python .....	27
1.5.2 IDE Visual Studio Code .....	28
1.5.3 Hệ thống kiểm soát phiên bản (Git) .....	28
<b>1.6 BÀI TẬP THỰC HÀNH .....</b>	<b>29</b>
1.6.1 Bài thực hành 01: Cài đặt môi trường .....	29
1.6.2 Bài thực hành 02: Lập trình Python cơ bản .....	39
1.6.3 Bài thực hành 03: List, Tuple, Dictionary .....	48
1.6.4 Bài thực hành 04: OOP trong Python .....	53
<b>1.7 BÀI TẬP MỞ RỘNG .....</b>	<b>60</b>
<b>BÀI 2: MÃ HOÁ VỚI PYTHON .....</b>	<b>61</b>

<b>2.1 MẬT MÃ HỌC .....</b>	<b>61</b>
2.1.1 Mật mã Caesar .....	61
2.1.2 Mật mã Vigenère.....	62
2.1.3 Mật mã Rail Fence.....	63
2.1.4 Mật mã Playfair.....	63
2.1.5 Mật mã Transposition .....	65
<b>2.2 GIAO DIỆN CHƯƠNG TRÌNH ỨNG DỤNG (API) .....</b>	<b>67</b>
<b>2.3 POSTMAN.....</b>	<b>67</b>
<b>2.4 PYTHON FLASK FRAMEWORK .....</b>	<b>68</b>
<b>2.5 BÀI TẬP THỰC HÀNH.....</b>	<b>68</b>
2.5.1 Bài thực hành 01: Mã hoá, giải mã Caesar .....	68
2.5.2 Bài thực hành 02: Mã hoá, giải mã Vigenère.....	75
2.5.3 Bài thực hành 03: Mã hoá, giải mã Rail Fence.....	78
2.5.4 Bài thực hành 04: Mã hoá, giải mã Playfair.....	82
2.5.5 Bài thực hành 05: Mã hoá, giải mã Transposition .....	87
2.5.6 Bài thực hành 06: Sử dụng Flask tạo giao diện web.....	90
<b>2.6 BÀI TẬP MỞ RỘNG .....</b>	<b>95</b>
<b>BÀI 3: LẬP TRÌNH GIAO DIỆN ỨNG DỤNG BẢO MẬT.....</b>	<b>96</b>
<b>3.1 PYQT5 .....</b>	<b>96</b>
<b>3.2 QTDESIGNER .....</b>	<b>97</b>
<b>3.3 HỆ MÃ HÓA RSA .....</b>	<b>97</b>
3.3.1 Giới thiệu RSA .....	97
3.3.2 Thư viện RSA trong Python.....	98
<b>3.4 ĐƯỜNG CONG ELIPTIC (ECC) .....</b>	<b>98</b>
3.4.1 Giới thiệu ECC.....	98
3.4.2 Mật mã ECC .....	99
3.4.3 So sánh ECC và RSA.....	100
<b>3.5 BÀI TẬP THỰC HÀNH.....</b>	<b>101</b>
3.5.1 Bài thực hành 01: Tạo Ứng dụng mã hoá Caesar .....	101
3.5.2 Bài thực hành 02: Tạo Ứng dụng mã hoá RSA.....	107
3.5.3 Bài thực hành 03: Tạo Ứng dụng mã hoá ECC.....	120
<b>3.6 BÀI TẬP MỞ RỘNG .....</b>	<b>128</b>
<b>BÀI 4: SOCKET, WEBSOCKET, TRAO ĐỔI KHOÁ VÀ HÀM BĂM .....</b>	<b>129</b>
<b>4.1 SOCKET VÀ MODULE PYTHON SOCKET .....</b>	<b>129</b>
<b>4.2 WEBSOCKET VÀ TORNADO FRAMEWORK .....</b>	<b>130</b>
<b>4.3 GIAO THỨC DIFFIE – HELLMAN .....</b>	<b>130</b>
<b>4.4 TIÊU CHUẨN MÃ HÓA TIỀN TIẾN (AES) .....</b>	<b>131</b>
<b>4.5 HÀM BĂM .....</b>	<b>132</b>
4.5.1 MD5 .....	132
4.5.2 SHA-256 .....	132
4.5.3 SHA-3.....	133

4.5.4 <i>BLAKE2</i> .....	134
<b>4.6 BÀI TẬP THỰC HÀNH.....</b>	<b>134</b>
4.6.1 <i>Bài thực hành 01: Tạo Socket với AES và RSA</i> .....	134
4.6.2 <i>Bài thực hành 02: DH Key Pair</i> .....	140
4.6.3 <i>Bài thực hành 03: Hàm băm</i> .....	143
4.6.4 <i>Bài thực hành 04: WebSocket và Tornado</i> .....	147
<b>4.7 BÀI TẬP MỞ RỘNG .....</b>	<b>151</b>
<b>BÀI 5: ỨNG DỤNG BẢO MẬT .....</b>	<b>152</b>
5.1 BASE64 .....	152
5.2 SSL .....	152
5.3 BLOCKCHAIN .....	154
5.4 STEGANOGRAPHY .....	154
<b>5.5 BÀI TẬP THỰC HÀNH.....</b>	<b>155</b>
5.5.1 <i>Bài thực hành 01: Base64</i> .....	155
5.5.2 <i>Bài thực hành 02: SSL</i> .....	157
5.5.3 <i>Bài thực hành 03: Blockchain</i> .....	162
5.5.4 <i>Bài thực hành 04: Giấu tin trong ảnh</i> .....	167
<b>5.6 BÀI TẬP MỞ RỘNG .....</b>	<b>171</b>
<b>BÀI 6: AN TOÀN MẠNG VÀ GIÁM SÁT HỆ THỐNG .....</b>	<b>172</b>
6.1 WEB SERVER.....	172
6.2 NETWORK SCANNER & NETWORK CAPTURE.....	172
6.3 PORT SCANNER.....	173
6.4 THƯ VIỆN PYTHON SCAPY.....	174
6.5 THƯ VIỆN PYTHON PSUTIL .....	174
<b>6.6 BÀI TẬP THỰC HÀNH.....</b>	<b>175</b>
6.6.1 <i>Bài thực hành 01: Tạo web server</i> .....	175
6.6.2 <i>Bài thực hành 02: Network Scanner</i> .....	178
6.6.3 <i>Bài thực hành 03: Network Capture</i> .....	180
6.6.4 <i>Bài thực hành 04: Port Scanner</i> .....	182
6.6.5 <i>Bài thực hành 05: Lắng nghe và thay đổi gói tin ICMP</i> .....	183
6.6.6 <i>Bài thực hành 06: Giám sát hệ thống và gửi tin qua Telegram</i> .....	186
<b>6.7 BÀI TẬP MỞ RỘNG .....</b>	<b>193</b>
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>194</b>

# HƯỚNG DẪN

## KIẾN THỨC TIỀN ĐỀ

Sinh viên cần có các kiến thức căn bản về bảo mật thông tin, kiến thức về các giải thuật mã hóa, kiến thức căn bản về mạng máy tính và một số ứng dụng nâng cao của bảo mật thông tin.

## YÊU CẦU MÔN HỌC

Sinh viên phải tham dự học đầy đủ các buổi học trên lớp, xem lại bài học và làm các bài tập đầy đủ ở nhà.

## CÁCH TIẾP CẬN NỘI DUNG MÔN HỌC

Để đạt kết quả tốt trong môn học này, sinh viên nên đọc kỹ các nội dung trong tài liệu hướng dẫn trước khi đến lớp; tham gia đầy đủ các buổi học và chủ động đóng góp ý kiến để làm phong phú bài học. Hơn nữa, sinh viên cần nắm vững các khái niệm, tính chất, cũng như các ví dụ được trình bày trong tài liệu. Sau khi hoàn thành bài học, việc ôn lại kiến thức, làm bài tập và giải quyết các câu hỏi tại nhà là rất quan trọng. Ngoài ra, sinh viên cũng nên tìm hiểu thêm các tài liệu khác liên quan và thực hiện thêm các bài tập nâng cao để củng cố kiến thức.

## PHƯƠNG PHÁP ĐÁNH GIÁ MÔN HỌC

- Điểm quá trình: Trọng số 50%. Bao gồm: Điểm kiểm tra thường xuyên trong quá trình học; điểm kiểm tra giữa học phần, điểm làm bài tập trên lớp; hoặc điểm chuyên cần.
- Điểm thi: Trọng số 50%. Bao gồm: Điểm trung bình các bài thực hành của 06 buổi học. Nội dung của các bài tập thực hành bao gồm thực hành cơ bản và thực hành nâng cao, sẽ được trình bày trong tài liệu này.

# BÀI 1: LẬP TRÌNH CƠ BẢN VỚI PYTHON

Bài học này sẽ giới thiệu về ngôn ngữ lập trình Python, cung cấp các kiến thức cơ bản về các kiểu dữ liệu, biến, biểu thức, cấu trúc chương trình và lập trình hướng đối tượng trong Python. Mục tiêu của bài này là giúp sinh viên quen thuộc với các khái niệm Python cơ bản để có thể thực hành và sử dụng trong các bài tiếp theo. Ngoài ra, bài này cũng giới thiệu và hướng dẫn sinh viên làm quen, thực hành với các công cụ hỗ trợ khi lập trình Python, hệ thống kiểm soát phiên bản và sử dụng công cụ kiểm thử cho các bài tập về bảo mật thông tin.

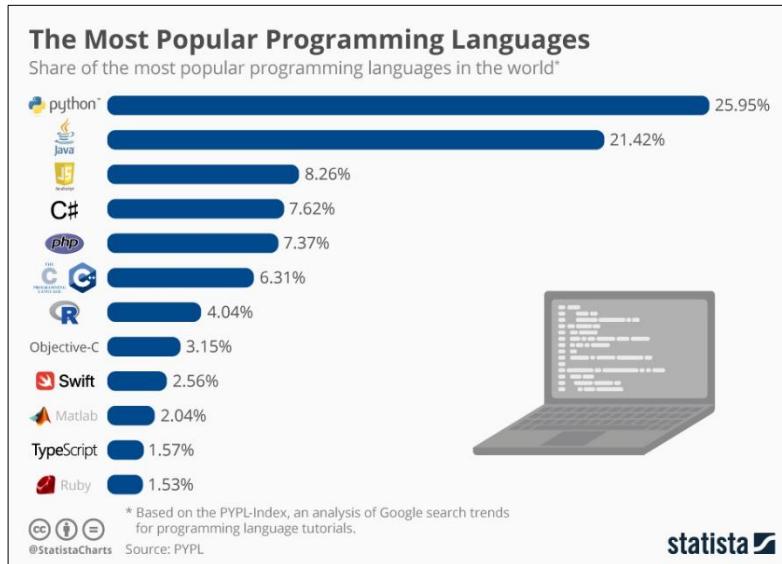
## 1.1 NGÔN NGỮ PYTHON

Python là một ngôn ngữ lập trình bậc cao được thông dịch, nổi bật với cú pháp rõ ràng và dễ hiểu, giúp lập trình viên tập trung vào việc giải quyết vấn đề mà không phải bận tâm quá nhiều về quy tắc ngữ pháp. Được phát triển bởi Guido van Rossum vào đầu những năm 90, Python nhanh chóng trở thành một trong những ngôn ngữ được ưa chuộng nhất toàn cầu.

Một trong những ưu điểm nổi bật của Python là tính linh hoạt, cho phép nó được sử dụng cho nhiều mục đích khác nhau như phát triển ứng dụng web, trí tuệ nhân tạo, khoa học dữ liệu, máy học, và phát triển game. Ngôn ngữ này còn có hàng loạt thư viện phong phú như NumPy, Pandas, TensorFlow, và Django, giúp mở rộng khả năng của nó.

Cộng đồng Python đóng vai trò quan trọng trong sự phát triển và lan tỏa của ngôn ngữ này. Với một cộng đồng lớn mạnh, người dùng không chỉ có thể tiếp cận nhiều tài nguyên phong phú mà còn nhận được sự hỗ trợ và chia sẻ kiến thức thông qua các diễn đàn, trang web học tập, sự kiện và các dự án mã nguồn mở.

Không chỉ linh hoạt và có cộng đồng hỗ trợ mạnh mẽ, Python còn được đánh giá cao về khả năng tích hợp dễ dàng với các ngôn ngữ và hệ thống khác. Nó cung cấp giao diện ứng dụng (API) linh hoạt, hỗ trợ nhiều nền tảng, và có khả năng mở rộng cao, điều này giúp việc phát triển các dự án phức tạp trở nên thuận lợi hơn.



### So sánh độ phổ biến các ngôn ngữ lập trình

(Nguồn: <https://cdn.statcdn.com>)

Với tất cả những ưu điểm trên, Python là một ngôn ngữ thông dịch mạnh mẽ. Sự kết hợp giữa cú pháp dễ đọc, tính linh hoạt, và cộng đồng mạnh mẽ đã đặt Python ở vị thế dẫn đầu trong thế giới lập trình, khẳng định uy tín của mình trong nhiều lĩnh vực công nghệ hiện đại.

## 1.2 THÀNH PHẦN CỦA PYTHON

### 1.2.1 Biến, từ khóa

“Trong Python, biến (variables) được hiểu là một tên gọi dùng để chỉ một giá trị nhất định trong bộ nhớ. Có thể coi biến như một nhãn gán cho một giá trị dữ liệu cụ thể. Việc tạo ra biến rất đơn giản và không yêu cầu khai báo kiểu dữ liệu. Kiểu dữ liệu của biến sẽ được Python xác định tự động dựa trên giá trị mà chúng ta gán cho nó” [1].

Khi xem xét đoạn mã dưới đây, ta thấy rằng Python tự động nhận diện kiểu dữ liệu của biến theo giá trị được gán. Ngoài ra, biến trong Python có khả năng thay đổi giá

trị trong suốt quá trình thực thi chương trình, nghĩa là ta có thể cập nhật giá trị mới cho một biến đã được khởi tạo.

```
1 x = 10          # Biến x lưu trữ giá trị số nguyên 10
2 name = "Alice"  # Biến name lưu trữ chuỗi "Alice"
3 is_valid = True # Biến is_valid lưu trữ giá trị boolean True
```

Từ khóa (keywords) trong Python là các từ đã được định nghĩa, có ý nghĩa đặc biệt trong ngôn ngữ. Các từ khóa này đã được dành riêng thực hiện một số chức năng cụ thể và không được sử dụng làm tên biến, nhãn khác trong chương trình. Ta không thể sử dụng một từ khóa như tên biến:

```
1 if = 5      # Lỗi! "if" là từ khóa, không thể sử dụng làm tên biến
```

Bảng dưới đây là danh sách các từ khóa của Python, các từ này đều có ý nghĩa đặc biệt và đã được dành riêng cho việc thực hiện các chức năng cụ thể:

STT	Từ khoá – Ý nghĩa	STT	Từ khoá – Ý nghĩa
1	<b>False</b> : Đại diện giá trị logic sai (Boolean).	2	<b>None</b> : Đại diện một giá trị không có hoặc thiếu.
3	<b>None</b> : Đại diện một giá trị không có hoặc thiếu.	4	<b>and</b> : Thực hiện phép toán logic "và".
5	<b>as</b> : Được sử dụng để đặt bí danh (alias) cho module hoặc các thành phần khác.	6	<b>assert</b> : Kiểm tra điều kiện, và nếu điều kiện sai, sẽ ném ra một lỗi (exception).
7	<b>async</b> : Được sử dụng khi định nghĩa các hàm bất đồng bộ (asynchronous functions).	8	<b>await</b> : Sử dụng trong các hàm bất đồng bộ để chờ kết quả trả về.
9	<b>break</b> : Kết thúc một vòng lặp (loop).	10	<b>class</b> : Được sử dụng để định nghĩa một lớp (class).

STT	Từ khoá – Ý nghĩa	STT	Từ khoá – Ý nghĩa
11	<b>continue:</b> Bỏ qua phần còn lại của vòng lặp hiện tại và tiếp tục với lần lặp tiếp theo.	12	<b>def:</b> Định nghĩa một hàm (function).
13	<b>del:</b> Xóa một biến, hoặc một phần tử ra khỏi danh sách (list).	14	<b>elif:</b> Viết tắt của "else if", dùng trong câu lệnh rẽ nhánh (conditional statement).
15	<b>else:</b> Dùng trong câu lệnh rẽ nhánh (conditional statement).	16	<b>except:</b> Xử lý các ngoại lệ (exceptions).
17	<b>finally:</b> Xác định khối mã sẽ được thực thi sau khi thực thi các khối try và except.	18	<b>for:</b> Được sử dụng để tạo vòng lặp for.
19	<b>from:</b> Được dùng khi import một phần (module hay package) từ một module.	20	<b>global:</b> Được dùng để xác định rằng biến nằm trong phạm vi toàn cục.
21	<b>if:</b> Sử dụng trong câu lệnh rẽ nhánh (conditional statement).	22	<b>import:</b> Dùng để import các module vào chương trình.
23	<b>in:</b> Kiểm tra một giá trị có tồn tại trong một chuỗi, danh sách, tuple.	24	<b>is:</b> Dùng để so sánh hai đối tượng xem chúng có cùng tham chiếu hay không.
25	<b>lambda:</b> Tạo các hàm nặc danh (anonymous functions).	26	<b>nonlocal:</b> Được dùng để xác định rằng biến nằm ở phạm vi không phải toàn cục cũng không phải là cục bộ.
27	<b>not:</b> Thực hiện phép toán logic "không".	28	<b>or:</b> Thực hiện phép toán logic "hoặc".

STT	Từ khoá – Ý nghĩa	STT	Từ khoá – Ý nghĩa
29	<b>pass</b> : Đánh dấu để bỏ qua đoạn mã mà không làm gì cả.	30	<b>raise</b> : Sử dụng để ném một ngoại lệ.
31	<b>return</b> : Dùng để trả về kết quả từ một hàm.	32	<b>try</b> : Dùng để thực hiện các lệnh có khả năng gây ra những ngoại lệ khác nhau (exceptions).
33	<b>while</b> : Được sử dụng để tạo vòng lặp while.	34	<b>with</b> : Dùng khi làm việc với các nguồn tài nguyên được quản lý tự động.
35	<b>yield</b> : Dùng trong hàm để trả về giá trị, lưu trạng thái của hàm đó.		

## 1.2.2 Kiểu dữ liệu

"Python hỗ trợ nhiều loại dữ liệu khác nhau để lưu trữ và xử lý thông tin" [1], [2]. Dưới đây là một số kiểu dữ liệu cơ bản và thông dụng trong Python:

### - Kiểu dữ liệu số (Numeric Types):

- "**int**": Kiểu dữ liệu số nguyên, lưu trữ số nguyên không có dấu thập phân.
- "**float**": Kiểu dữ liệu số thực, lưu trữ số có dấu thập phân.
- "**complex**": Kiểu dữ liệu số phức, lưu trữ số phức với phần thực, phần ảo.

### - Kiểu dữ liệu chuỗi (String Type):

- "**str**": Kiểu dữ liệu chuỗi, lưu trữ chuỗi ký tự, được bao quanh bởi ' ' hoặc " ".

### - Kiểu dữ liệu Boolean (Boolean Type):

- "**bool**": Kiểu boolean chỉ có hai giá trị là True hoặc False. Được dùng để biểu diễn các trạng thái logic.

### - Kiểu dữ liệu danh sách (List Type):

- "**list**": Kiểu dữ liệu danh sách lưu trữ tập hợp các phần tử có thứ tự và có thể thay đổi.

- **Kiểu dữ liệu tuple (Tuple Type):**
  - “tuple”: Kiểu dữ liệu tuple tương tự như danh sách nhưng không thể thay đổi (immutable).
- **Kiểu dữ liệu từ điển (Dictionary Type):**
  - “dict”: Kiểu dữ liệu lưu trữ các cặp key-value không có thứ tự. Key là duy nhất, không thay đổi; value có thể thay đổi.
- **Kiểu dữ liệu tập hợp (Set Type):**
  - “set”: Đây là kiểu dữ liệu dùng để lưu trữ một tập hợp các phần tử không trùng lặp và không có thứ tự.
- **Kiểu dữ liệu bộ nhớ đóng (Frozen Set Type):**
  - “frozenset”: Tương tự như kiểu dữ liệu set nhưng không thể thay đổi (immutable).
- **Kiểu dữ liệu nhị phân (Binary Types):**
  - “bytes”: Lưu trữ một chuỗi byte.
  - “bytearray”: Tương tự như bytes nhưng có thể thay đổi (mutable).
- **Kiểu dữ liệu None (None Type):**
  - **NoneType**: Chỉ có một giá trị là “None”, được dùng để biểu diễn giá trị rỗng hoặc không tồn tại.

### 1.2.3 Các toán tử số học

“Trong Python, các toán tử số học được sử dụng để thực hiện các phép toán cơ bản trên các số. Các toán tử này cho phép bạn thực hiện các phép cộng, trừ, nhân, chia, các phép tính khác trên các biến hoặc giá trị số” [1], [2]. Sau đây là các toán tử số học cơ bản trong Python:

- **Cộng (+):** Toán tử cộng được dùng để thực hiện phép cộng giữa hai số.

```

1 a = 5
2 b = 3
3 result = a + b # Kết quả: 8
  
```

- **Trừ (-):** Toán tử trừ được dùng để thực hiện phép trừ giữa hai số.

```
1 a = 8
2 b = 4
3 result = a - b # Kết quả: 4
```

- **Nhân (\*):** Toán tử nhân được dùng để thực hiện phép nhân giữa hai số.

```
1 a = 6
2 b = 7
3 result = a * b # Kết quả: 42
```

- **Chia (/):** Toán tử chia được dùng để thực hiện phép chia.

```
1 a = 20
2 b = 5
3 result = a / b # Kết quả: 4.0 (Kết quả luôn là một số thập phân nếu có phần dư)
```

- **Chia lấy phần nguyên (//):** Toán tử chia lấy phần nguyên trả về kết quả là phần nguyên của phép chia.

```
1 a = 20
2 b = 3
3 result = a // b # Kết quả: 6
```

- **Chia lấy dư (%):** Toán tử này sẽ cung cấp phần còn lại sau khi thực hiện phép chia giữa hai số.

```
1 a = 20
2 b = 7
3 remainder = a % b # Kết quả: 6 (Phần dư của 20 chia cho 7)
```

- **Luỹ thừa (\*\*):** Toán tử luỹ thừa được sử dụng để tính toán lũy thừa của một số.

```
1 a = 2
2 b = 3
3 result = a ** b # Kết quả: 8 (2^3 = 8)
```

Các toán tử số học này có thể được dùng trong các biểu thức để thực hiện các phép tính số học cơ bản. Đồng thời, Python cũng hỗ trợ việc sử dụng các dấu ngoặc để xác định ưu tiên của các phép tính, giúp thực hiện các phép toán chính xác theo ý muốn.

## 1.2.4 Các toán tử logic

“Trong Python, các toán tử logic được sử dụng để thực hiện phép toán logic trên các giá trị boolean (True hoặc False) hoặc các biểu thức logic. Các toán tử logic cho phép chúng ta kiểm tra và thực hiện các phép so sánh, kết hợp các điều kiện để quyết định kết quả logic” [1], [2]. Sau đây sẽ trình bày một vài toán tử logic cơ bản trong Python:

- **Phép toán AND:** Toán tử “**and**” trả về “True” nếu cả hai điều kiện đều đúng.

```
1 x = 5
2 y = 3
3 result = (x > 2) and (y < 4) # Kết quả: True
```

- **Phép toán OR:** Toán tử “**or**” trả về “True” nếu ít nhất một trong hai điều kiện đầu vào là đúng.

```
1 x = 5
2 y = 3
3 result = (x > 2) or (y > 4) # Kết quả: True
```

- **Phép toán NOT:** Toán tử “**not**” trả về “True” nếu điều kiện là “False” và ngược lại.

```
1 x = 5
2 result = not (x == 5) # Kết quả: False
```

- **Phép so sánh bằng (==):** Toán tử “**==**” sẽ so sánh hai giá trị có bằng nhau hay không.

```
1 x = 5
2 result = (x == 5) # Kết quả: True
```

- **Phép so sánh không bằng (**`!=`**):** Toán tử "`!=`" sẽ so sánh hai giá trị có khác nhau hay không.

```
1 x = 5
2 result = (x != 3) # Kết quả: True
```

- **Phép so sánh lớn hơn (**`>`**), nhỏ hơn (**`<`**):** Toán tử "`>`" sẽ so sánh xem giá trị bên trái lớn hơn giá trị bên phải hay không và toán tử "`<`" sẽ so sánh xem giá trị bên trái nhỏ hơn giá trị bên phải hay không.

```
1 x = 5
2 result1 = (x > 3) # Kết quả: True
3 result2 = (x < 3) # Kết quả: False
```

- **Phép so sánh lớn hơn hoặc bằng (**`>=`**), nhỏ hơn hoặc bằng (**`<=`**):** Toán tử "`>=`" sẽ so sánh xem giá trị bên trái lớn hơn hoặc bằng giá trị bên phải hay không và toán tử "`<=`" sẽ so sánh xem giá trị bên trái nhỏ hơn hoặc bằng giá trị bên phải hay không.

```
1 x = 5
2 result1 = (x >= 3) # Kết quả: True
3 result2 = (x <= 3) # Kết quả: False
```

## 1.2.5 Nhập, xuất dữ liệu

Trong Python, các hàm nhập và xuất dữ liệu chủ yếu là "**`input()`**" và "**`print()`**".

- **Hàm "`input()`"** được dùng để nhập dữ liệu từ bàn phím. Nó cho phép nhập giá trị và trả về một chuỗi (string) biểu diễn giá trị đã nhập. Ta có thể lưu giá trị này vào biến để sử dụng sau này. Ví dụ:

```
1 name = input("Nhập tên của bạn: ")
2 print("Xin chào, ", name)
```

- **Hàm "`print()`"** được sử dụng để hiển thị dữ liệu ra màn hình hoặc console. Ta có thể truyền biến, chuỗi, hoặc giá trị cần hiển thị vào hàm "**`print()`**". Ví dụ:

```
1 age = 25
2 print("Tuổi của bạn là:", age)
```

Ngoài ra, “**print()**” cũng cho phép định dạng hiển thị qua việc sử dụng các đối số khác như “**sep**” (ký tự phân cách), “**end**” (ký tự kết thúc), và các chuỗi định dạng f-string. Ví dụ:

```
1 print("Python", "là", "ngôn", "ngữ", "lập", "trình", sep="-")
2 # Kết quả: Python-là-ngôn-ngữ-lập-trình
3 print("Xin chào", end=" ")
4 print("các bạn!") # Kết quả: Xin chào các bạn!
```

## 1.2.6 Các cấu trúc điều khiển

Cấu trúc điều khiển là cách thức quản lý luồng của chương trình, quyết định việc thực thi các khối mã khác nhau dựa trên điều kiện hoặc vòng lặp. Các cấu trúc này cung cấp khả năng kiểm soát các điều kiện logic, lặp lại các hoạt động, thực hiện các quyết định trong đoạn mã. Có ba kiểu cấu trúc điều khiển cơ bản trong Python:

- **Câu lệnh điều kiện (Conditional Statements):** Cho phép kiểm tra điều kiện, thực hiện các hành động khác nhau dựa vào kết quả của điều kiện đó. Câu lệnh điều kiện phổ biến nhất trong Python là câu lệnh “**if**”, “**else**”, và “**elif**” (else if). Ví dụ:

```
1 x = 10
2 if x > 5:
3     print("x lớn hơn 5")
4 elif x == 5:
5     print("x bằng 5")
6 else:
7     print("x nhỏ hơn 5")
```

- **Vòng lặp (Loops):** Vòng lặp sẽ cho phép sự lặp lại việc thực hiện một khối mã cho đến khi một điều kiện nào đó được thỏa mãn. Trong Python, hai loại vòng lặp phổ biến nhất là vòng lặp “**for**” và “**while**”.

- Vòng lặp “**for**” được dùng để duyệt qua một chuỗi hoặc một tập hợp các phần tử. Ví dụ:

```

1 fruits = ["apple", "banana", "cherry"]
2 for fruit in fruits:
3     print(fruit)

```

- Vòng lặp “**while**” thực hiện việc lặp lại mã đến khi điều kiện không còn đúng nữa. Ví dụ:

```

1 count = 0
2 while count < 5:
3     print(count)
4     count += 1

```

- **Câu lệnh nhảy (Jump Statements):** Câu lệnh nhảy được dùng để thay đổi luồng điều khiển của chương trình. Các câu lệnh nhảy phổ biến trong Python bao gồm “**break**”, “**continue**” và “**pass**”.
- Sử dụng câu lệnh “**break**” để kết thúc một vòng lặp:

```

1 # Tìm số chia hết cho 5 đầu tiên trong khoảng từ 1 đến 100
2 for i in range(1, 101):
3     if i % 5 == 0:
4         print("Số chia hết cho 5 đầu tiên là:", i)
5         break

```

- Câu lệnh “**continue**” được dùng để bỏ qua phần còn lại của vòng lặp hiện tại và chuyển sang vòng lặp kế tiếp:

```

1 # In các số chẵn từ 1 đến 10 và bỏ qua các số lẻ
2 for i in range(1, 11):
3     if i % 2 != 0:
4         continue
5     print(i)

```

- Sử dụng câu lệnh “**pass**” như là một tuyên bố rỗng:

```

1 # Kiểm tra điều kiện, nếu đúng thực hiện, nếu sai thì
2     không làm gì
3 x = 5
4 if x > 10:
5     print("x lớn hơn 10")
6 else:
7     pass

```

## 1.2.7 Chuỗi

"Trong Python, chuỗi (String) được định nghĩa là một tập hợp các ký tự, được bao bọc bởi cặp dấu nháy đơn (' ') hoặc cặp dấu nháy kép (" "). Chuỗi có thể bao gồm bất kỳ loại ký tự nào, chẳng hạn như chữ cái, số và các ký tự đặc biệt" [1], [2].

- **Khai báo chuỗi trong Python:**

```

1 # Sử dụng dấu ngoặc đơn
2 string_single_quotes = 'Đây là một chuỗi sử dụng dấu ngoặc đơn.'
3 # Sử dụng dấu ngoặc kép
4 string_double_quotes = "Đây là một chuỗi sử dụng dấu ngoặc kép."
5 # Sử dụng dấu ngoặc ba
6 string_triple_quotes = """Đây là một chuỗi
7     sử dụng dấu ngoặc ba,
8     có thể trải dài qua nhiều dòng."""

```

- **Truy cập ký tự trong chuỗi:** Chúng ta có thể truy cập các ký tự trong chuỗi bằng cách sử dụng chỉ số của chúng trong cặp dấu ngoặc vuông []. Chú ý rằng chỉ số trong Python bắt đầu từ 0. Ví dụ:

```

1 my_string = "Hello, World!"
2 print(my_string[0]) # Kết quả: 'H'
3 print(my_string[7]) # Kết quả: 'W'

```

- **Các phép xử lý chuỗi trong Python:**

- **Cắt chuỗi (Slicing):** Cắt chuỗi là quá trình lấy một phần của chuỗi sử dụng chỉ mục hoặc phạm vi chỉ mục.

```

1 my_string = "Hello, World!"
2 print(my_string[7:]) # Lấy từ ký tự thứ 7 đến hết: Kết quả: 'World!'
3 print(my_string[:5]) # Lấy từ đầu đến ký tự thứ 4: Kết quả: 'Hello'
4 print(my_string[3:8])# Lấy từ ký tự thứ 3 đến ký tự thứ 7: Kết quả:
  'lo, W'

```

- *Ghép chuỗi (Concatenation)*: Ghép chuỗi là quá trình nối chuỗi lại với nhau.

```

1 string1 = "Hello"
2 string2 = "World"
3 concatenated_string = string1 + " " + string2 # Kết quả: 'Hello
  World'

```

- *Độ dài chuỗi (Length)*: Hàm “**len()**” được dùng để tính độ dài của chuỗi.

```

1 my_string = "Hello, World!"
2 length = len(my_string) # Kết quả: 13

```

#### - Một số hàm dùng để xử lý các chuỗi trong Python:

- “**upper()**”: Chuyển đổi chuỗi thành chữ hoa.
- “**lower()**”: Chuyển đổi chuỗi thành chữ thường.
- “**strip()**”: Loại bỏ khoảng trắng ở đầu và cuối chuỗi.
- “**split()**”: Phân tách chuỗi thành danh sách các từ hoặc phần tử.
- “**replace()**”: Thay thế một phần của chuỗi bằng một chuỗi khác.

```

1 my_string = "Hello, World! "
2 print(my_string.strip()) # Loại bỏ khoảng trắng: Kết quả: 'Hello,
  World!'
3
4 my_string = "Hello, World!"
5 print(my_string.split(","))
6 # Phân tách chuỗi: Kết quả: ['Hello', ' World!']
7
8 my_string = "Hello, World!"
9 print(my_string.replace("Hello", "Hi"))
10 # Thay thế chuỗi: Kết quả: 'Hi, World!'

```

## 1.2.8 Hàm (Function)

"Hàm trong Python là một đoạn mã có thể được gọi để thực hiện một nhiệm vụ nhất định, cho phép tổ chức mã thành các phần nhỏ hơn, tái sử dụng mã và làm cho chương trình trở nên dễ đọc hơn" [1], [2].

- **Khai báo hàm:** Để định nghĩa một hàm trong Python, ta sử dụng từ khóa "def", tiếp theo là tên hàm và danh sách tham số (nếu có), sau đó là một khối mã được thuộ lề bên trong dấu hai chấm.

```

1 def my_function(parameter1, parameter2):
2     # Khối mã của hàm
3     # Thực hiện các hoạt động dựa trên tham số được truyền vào
4     result = parameter1 + parameter2
5     return result

```

- **Phân loại hàm:**

- *Hàm có giá trị trả về:* Một hàm có khả năng trả lại giá trị thông qua từ khóa "return". Nếu không có lệnh "return" trong hàm, nó sẽ tự động trả về "None" như giá trị mặc định.
- *Hàm không có giá trị nào trả về:* Một số hàm chỉ thực hiện một công việc nhất định mà không cần trả về bất kỳ giá trị nào.

- **Cách sử dụng hàm:**

- *Gọi hàm:* Để sử dụng hàm, chỉ cần gọi tên hàm cùng với các đối số cần thiết (nếu có). Ví dụ:

```

6 result = my_function(10, 20) # Gọi hàm và lưu kết quả vào biến result
7 print(result) # In kết quả của hàm

```

- *Tham số và đối số:* "Tham số" là các biến được định nghĩa trong định nghĩa hàm, còn "đối số" là giá trị được truyền cho tham số khi gọi hàm.

Ví dụ về khai báo và gọi hàm trong Python:

```

1 # Định nghĩa hàm tính tổng
2 def calculate_sum(a, b):
3     result = a + b
4     return result
5 # Gọi hàm và lưu kết quả vào biến
6 sum_result = calculate_sum(10, 20)
7 # In kết quả
8 print("Tổng hai số là:", sum_result)

```

Ví dụ khai báo và gọi hàm không có giá trị trả về:

```

1 # Hàm không trả về giá trị, chỉ in ra thông báo
2 def greet(name):
3     print("Xin chào,", name)
4 # Gọi hàm
5 greet("Alice")

```

## 1.3 KIỂU DỮ LIỆU CÓ CẤU TRÚC

### 1.3.1 Mảng (Array)

"Trong Python, module “**array**” cung cấp một cấu trúc dữ liệu cơ bản gọi là mảng (array) cho phép lưu trữ cùng loại dữ liệu. Mảng trong module array giới hạn các phần tử bên trong mảng phải cùng kiểu dữ liệu" [1], [2].

- Để sử dụng mảng trong Python, chúng ta cần import module “**array**”:

```
1 from array import array
```

- Khai báo mảng trong module “**array**”: Có hai tham số quan trọng khi khai báo một mảng:
  - *Typecode*: Đây là một ký tự xác định kiểu dữ liệu của các phần tử trong mảng.
  - *Initializer*: Đây là danh sách các giá trị ban đầu của mảng.

Ví dụ:

```

1 from array import array
2 # Khai báo một mảng số nguyên
3 int_array = array('i', [1, 2, 3, 4, 5])
4 # Khai báo một mảng số thực
5 float_array = array('f', [3.14, 2.5, 6.7])

```

- Các kiểu dữ liệu (Typecodes) trong module array:

- 'b': signed char (1 byte)
- 'B': unsigned char (1 byte)
- 'i': signed int (2 bytes)
- 'I': unsigned int (2 bytes)
- 'f': float (4 bytes)
- 'd': double (8 bytes)
- và nhiều typecodes khác.

- Phương thức và thuộc tính của mảng:

- Truy cập phần tử trong mảng: Truy cập các phần tử trong mảng bằng cách sử dụng chỉ số của chúng.

```
7 print(int_array[0]) # Truy cập phần tử đầu tiên của mảng số nguyên
8 print(float_array[2]) # Truy cập phần tử thứ ba của mảng số thực
```

- Cập nhật giá trị của phần tử trong mảng:

```
10 int_array[2] = 10
11 # Cập nhật giá trị của phần tử thứ ba trong mảng số nguyên
```

- Sử dụng các phương thức của mảng: Module “**array**” cung cấp một số phương thức để thực hiện các thao tác với mảng:
  - “append()”: Thêm một phần tử vào cuối mảng.
  - “insert()”: Chèn một phần tử vào vị trí chỉ định.
  - “remove()”: Xóa phần tử ở vị trí đầu tiên có giá trị xác định.
  - “pop()”: Xóa phần tử ở vị trí chỉ định và trả về giá trị của phần tử đó.
  - “extend()”: Mở rộng mảng bằng cách thêm các phần tử từ một mảng khác.
  - “index()”: Trả về chỉ số đầu tiên của một phần tử cụ thể.
  - “count()”: Đếm số lần xuất hiện của một phần tử trong mảng.

Ví dụ:

```
13 int_array.append(6) # Thêm phần tử 6 vào cuối mảng số nguyên
14 float_array.remove(6.7) # Xóa phần tử 6.7 khỏi mảng số thực
```

### 1.3.2 Danh sách (List)

"Trong Python, kiểu dữ liệu danh sách (List) thường được sử dụng để lưu trữ và làm việc với các tập hợp dữ liệu có thứ tự. Mặc dù không có một kiểu dữ liệu mảng cố định như trong một số ngôn ngữ khác, nhưng danh sách trong Python có thể được sử dụng tương đương với mảng vì chúng có khả năng lưu trữ nhiều loại dữ liệu và có thể thay đổi kích thước (động)" [1], [2].

- **Khai báo một danh sách (List):**

```
1 # Danh sách số nguyên
2 my_list = [1, 2, 3, 4, 5]
3 # Danh sách chuỗi
4 names = ["Alice", "Bob", "Charlie"]
5 # Danh sách kết hợp kiểu dữ liệu
6 mixed_list = [10, "hello", 3.14, True]
```

- **Cách sử dụng danh sách:**

- Truy cập vào một phần tử ở trong danh sách:

```
8 print(my_list[0]) # Truy cập phần tử đầu tiên: Kết quả: 1
9 print(names[2]) # Truy cập phần tử thứ ba: Kết quả: 'Charlie'
```

- Cập nhật giá trị của một phần tử ở trong danh sách:

```
11 my_list[1] = 20 # Thay đổi giá trị của phần tử thứ hai
12 print(my_list) # Kết quả: [1, 20, 3, 4, 5]
```

- Thêm một phần tử vào danh sách:

```
14 names.append("David") # Thêm phần tử vào cuối danh sách
15 print(names) # Kết quả: ['Alice', 'Bob', 'Charlie', 'David']
```

- Xóa một phần tử khỏi danh sách:

```
17 del my_list[2] # Xóa phần tử thứ ba khỏi danh sách
18 print(my_list) # Kết quả: [1, 20, 4, 5]
```

- Duyệt qua từng phần tử trong danh sách:

```
20 for element in names:
21     print(element) # In từng phần tử trong danh sách
```

- **Lưu ý khi làm việc với danh sách trong Python:**

- Chỉ số trong danh sách bắt đầu từ 0.
- Danh sách có thể chứa nhiều kiểu dữ liệu khác nhau.
- Có thể thay đổi kích thước danh sách bằng cách thêm, xóa, thay đổi phần tử.
- Có nhiều phương thức và hàm sẵn có trong Python để làm việc với danh sách như **append()**, **remove()**, **pop()**, **insert()**, **sort()**, **len()**, và nhiều hơn nữa.

### 1.3.3 Kiểu Tuple

“Tuple là một dạng dữ liệu quan trọng dùng để lưu trữ các tập hợp thông tin không thể thay đổi (immutable). Đặc điểm của nó là sử dụng dấu ngoặc đơn và các phần tử trong Tuple sẽ được phân cách bằng dấu phẩy” [1], [2].

- **Đặc điểm của Tuple:**

- *Không thay đổi (Immutable)*: Sau khi tạo tuple sẽ không thay đổi giá trị của các phần tử trong tuple. Điều này khác với danh sách (list), nơi có thể thay đổi giá trị của các phần tử.
- *Có thứ tự*: Các phần tử trong tuple được xác định bằng vị trí của chúng, giống như danh sách.
- *Có thể chứa nhiều kiểu dữ liệu*: Tuple có thể chứa các phần tử với các kiểu dữ liệu khác nhau.

- **Ưu điểm của Tuple:**

- Sử dụng Tuple khi muốn đảm bảo các dữ liệu không thay đổi và không bị sửa đổi sau khi đã khởi tạo.
- Tốc độ truy cập phần tử trong Tuple thường nhanh hơn so với danh sách.

- **Khai báo Tuple:**

Có thể báo một tuple bằng cách sử dụng dấu ngoặc đơn và phân tách các phần tử bằng dấu phẩy. Ví dụ:

```

1 # Tuple các số nguyên
2 my_tuple = (1, 2, 3, 4, 5)
3 # Tuple các chuỗi
4 names = ("Alice", "Bob", "Charlie")
5 # Tuple kết hợp kiểu dữ liệu
6 mixed_tuple = (10, "hello", 3.14)

```

### - Truy cập vào phần tử trong Tuple:

Tương tự như danh sách, có thể truy cập các phần tử trong Tuple bằng cách sử dụng chỉ số của chúng. Ví dụ:

```

8 print(my_tuple[0]) # Truy cập phần tử đầu tiên: Kết quả: 1
9 print(names[2])   # Truy cập phần tử thứ ba: Kết quả: 'Charlie'

```

### - Các phương thức trong Tuple:

- **count(value)**: Đếm số lần một giá trị cụ thể trong tuple xuất hiện. Ví dụ:

```

1 my_tuple = (1, 2, 3, 1, 4, 1)
2 print(my_tuple.count(1)) # Kết quả: 3 (1 xuất hiện 3 lần trong tuple)

```

- **index(value)**: Trả về chỉ số đầu tiên của một giá trị cụ thể trong tuple. Ví dụ:

```

1 my_tuple = ('a', 'b', 'c', 'd', 'b')
2 print(my_tuple.index('b'))
3 # Kết quả: 1 (chỉ số đầu tiên của 'b' trong tuple là 1)

```

Cần lưu ý rằng vì tính không thay đổi của Tuple, nó không cung cấp các phương thức như thêm mới phần tử (**append()**), xóa phần tử (**remove()**), hoặc sắp xếp (**sort()**) như danh sách (list) trong Python. Điều này làm cho Tuple không linh hoạt nhưng an toàn khi cần bảo vệ các dữ liệu mà không muốn chúng bị thay đổi sau khi đã khởi tạo.

## 1.3.4 Kiểu Dictionary

"Trong Python, "**dictionary**" là một kiểu dữ liệu cấu trúc, mạnh mẽ được sử dụng để lưu trữ các cặp key-value không có thứ tự. Dictionary được biểu diễn bằng dấu ngoặc nhọn "{}" và mỗi cặp key-value được phân tách bởi dấu hai chấm ":". Key trong dictionary phải là duy nhất và không thể thay đổi, trong khi value có thể là bất kỳ kiểu dữ liệu nào" [1], [2].

- **Khai báo dictionary:**

```
1 # Khai báo một dictionary rỗng
2 my_dict = {}
3 # Khai báo một dictionary với các cặp key-value
4 person = {"name": "Alice", "age": 25, "city": "New York"}
```

- **Truy cập vào giá trị trong dictionary:**

Có thể truy cập giá trị trong dictionary bằng cách sử dụng key tương ứng, ví dụ:

```
6 print(person["name"]) # In giá trị của key "name": Kết quả: "Alice"
7 print(person["age"]) # In giá trị của key "age": Kết quả: 25
```

- - **Thêm hoặc cập nhật giá trị trong dictionary:**

```
9 # Thêm một cặp key-value mới
10 person["email"] = "alice@example.com"
11 # Cập nhật giá trị của key đã tồn tại
12 person["age"] = 26
```

- **Xóa một phần tử trong dictionary:**

```
14 # Xóa một cặp key-value từ dictionary
15 del person["city"]
16 # Xóa phần tử và lấy giá trị của key
17 age = person.pop("age")
```

- **Các phương thức và phương thức dành cho dictionary:**

- “keys()”: Trả về tất cả các keys trong dictionary.
- “values()”: Trả về tất cả các values trong dictionary.
- “items()”: Trả về tất cả các cặp key-value trong dictionary dưới dạng tuple.

Ví dụ:

```
19 print(person.keys()) # In ra tất cả các keys trong dictionary
20 print(person.values()) # In ra tất cả các values trong dictionary
21 print(person.items()) # In ra tất cả các cặp key-value trong
dictionary
```

- **Lưu ý:**

- Dictionary trong Python không có thứ tự, nghĩa là không giữ trật tự của các phần tử đã được thêm vào.
- Key trong dictionary phải là duy nhất, không thể thay đổi. Còn Value có thể là bất kỳ một kiểu dữ liệu nào.

## 1.4 OOP TRONG PYTHON

### 1.4.1 Lớp (Class) và đối tượng (Object)

"Python hỗ trợ OOP hoàn toàn và có thể được sử dụng để viết mã sử dụng các khái niệm hướng đối tượng như lớp (class), đối tượng (object), kế thừa (inheritance), đa hình (polymorphism), trừu tượng hóa (abstraction) [1], [2]".

**Lớp (Class):** Là một khuôn mẫu để tạo ra các đối tượng. Nó chứa các thuộc tính (attributes) và phương thức (methods).

**Đối tượng (Object):** Là một thể hiện cụ thể của một lớp, có thể được tạo ra từ lớp bằng cách sử dụng từ khóa class.

Ví dụ sau trình bày về lớp và đối tượng trong Python:

```

1 # Định nghĩa một lớp đơn giản
2 class Car:
3     def __init__(self, brand, model):
4         self.brand = brand
5         self.model = model
6     def get_info(self):
7         return f"{self.brand} {self.model}"
8 # Tạo đối tượng từ lớp Car
9 my_car = Car("Toyota", "Corolla")
10 # Gọi phương thức của đối tượng
11 print(my_car.get_info()) # Kết quả: Toyota Corolla

```

Ở ví dụ trên, "Car" là một lớp đại diện cho các đối tượng xe ô tô. "brand" và "model" là các thuộc tính của lớp, và "get\_info" là một phương thức để trả về thông tin về xe ô tô.

## 1.4.2 Khởi tạo (Constructor)

Trong Python, constructor được định nghĩa bằng phương thức “`__init__()`” trong lớp. Constructor này tự động được gọi khi khởi tạo một đối tượng từ lớp đó. Cú pháp của Constructor trong Python:

```
1 class ClassName:
2     def __init__(self, parameter1, parameter2, ...):
3         self.parameter1 = parameter1
4         self.parameter2 = parameter2
5         # ...
```

Trong đó:

- “`__init__()`” là tên của constructor trong Python.
- “`self`” là tham số đặc biệt đại diện cho chính đối tượng được tạo ra từ lớp.
- “`parameter1`”, “`parameter2`”,... là các tham số bạn muốn truyền vào khi tạo đối tượng.

Ví dụ sau trình bày về Constructor trong Python:

```
1 class Car:
2     def __init__(self, brand, model):
3         self.brand = brand
4         self.model = model
5
6     def get_info(self):
7         return f"{self.brand} {self.model}"
8 # Tạo đối tượng từ lớp Car và truy cập thông tin
9 my_car = Car("Toyota", "Corolla")
10 print(my_car.get_info()) # Kết quả: Toyota Corolla
```

Trong ví dụ trên, “`__init__()`” là constructor của lớp Car. Khi một đối tượng được tạo từ lớp Car, constructor này tự động được gọi. Trong constructor, “`self.brand = brand`” và “`self.model = model`” được sử dụng để gán một giá trị cho các thuộc tính của đối tượng “`my_car`”.

### 1.4.3 Thuộc tính (Attributes)

Attribute (thuộc tính) là các biến được liên kết với đối tượng của một lớp.

- **Phân loại thuộc tính trong Python:**

- **Thuộc tính instance (Instance Attributes):** Là các thuộc tính chỉ thuộc về một đối tượng cụ thể của một lớp.
- **Thuộc tính lớp (Class Attributes):** Là các thuộc tính thuộc về toàn bộ lớp, không chỉ thuộc về một đối tượng cụ thể nào đó. Chúng được chia sẻ giữa tất cả các đối tượng của lớp.

- **Định nghĩa thuộc tính trong Python:**

```

1 class ClassName:
2     def __init__(self, attribute1, attribute2):
3         self.attribute1 = attribute1      # Thuộc tính instance
4         self.attribute2 = attribute2      # Thuộc tính instance
5     class_attribute = "Class Attribute" # Thuộc tính lớp

```

Trong đó:

- “**attribute1**”, “**attribute2**”: Là các thuộc tính instance, được gán giá trị khi một đối tượng được tạo.
- “**class\_attribute**”: Là một thuộc tính lớp, có thể truy cập thông qua tất cả các đối tượng của lớp.

- **Truy cập thuộc tính trong Python:**

```

7 # Tạo đối tượng từ lớp và truy cập các thuộc tính
8 object_name = ClassName(value1, value2)
9 print(object_name.attribute1)      # Truy cập thuộc tính instance
10 print(object_name.class_attribute) # Truy cập thuộc tính lớp

```

### 1.4.4 Phương thức (Methods)

Phương thức (methods) là chỉ các hàm được định nghĩa bên trong một lớp và được gắn liền với các đối tượng của lớp đó.

- **Định nghĩa một phương thức trong Python:**

```

1 class ClassName:
2     def method_name(self, parameter1, parameter2):
3         # Các thao tác hoặc xử lý
4         return something # Trả về kết quả (nếu cần)

```

Trong đó:

- “**method\_name**”: Tên của phương thức.
- “**self**”: Tham chiếu đến chính đối tượng của lớp. self là bắt buộc và được dùng để truy cập các thuộc tính và phương thức khác trong cùng lớp.
- “**parameter1**”, “**parameter2**”: Các tham số mà phương thức có thể nhận (tùy chọn).
- “**return**”: Cho biết phương thức trả về một giá trị nếu cần.

- **Truy cập phương thức trong Python:**

```

6 # Tạo đối tượng từ lớp và gọi phương thức
7 object_name = ClassName()
8 # Gọi phương thức và truyền các tham số
9 object_name.method_name(value1, value2)

```

## 1.4.5 Kế thừa (Inheritance)

Trong Python, kế thừa (inheritance) là khả năng của một lớp con (child class) kế thừa thuộc tính và phương thức từ một lớp cha (parent class). Tuy nhiên, đa kế thừa (multiple inheritance) sẽ cho phép một lớp con kế thừa từ nhiều lớp cha.

- **Kế thừa đơn (Single Inheritance):**

```

1 class ParentClass:
2     # Định nghĩa các thuộc tính và phương thức của lớp cha
3
4 class ChildClass(ParentClass):
5     # Định nghĩa các thuộc tính và phương thức mới hoặc mở rộng từ
       lớp cha

```

- **Đa kế thừa (Multiple Inheritance):**

```

1 class ParentClass1:
2     # Định nghĩa các thuộc tính và phương thức của lớp cha 1
3 class ParentClass2:
4     # Định nghĩa các thuộc tính và phương thức của lớp cha 2
5 class ChildClass(ParentClass1, ParentClass2):
6     # Định nghĩa các thuộc tính và phương thức mới hoặc mở rộng từ
    các lớp cha

```

- **Lợi ích của kế thừa:**

- Tái sử dụng mã: Kế thừa cho phép sử dụng lại mã nguồn có sẵn từ lớp cha.
- Tính mở rộng: Lớp con có thể mở rộng hoặc thay đổi hành vi của lớp cha.
- Tổ chức mã nguồn: Kế thừa giúp tổ chức mã nguồn theo cấu trúc phân cấp, dễ dàng hiểu và bảo trì.

## 1.4.6 Đa hình (Polymorphism)

Đa hình (Polymorphism) là khả năng của các đối tượng có thể hiện thực hành vi khác nhau thông qua cùng một phương thức hoặc tên hàm. Phân loại:

- **Đa hình ở thời điểm biên dịch (Compile-time Polymorphism):**

Được triển khai thông qua kỹ thuật “**overloading**” và “**overriding**”:

- Overloading (Nạp chồng): Là khả năng có nhiều hàm hoặc phương thức cùng tên nhưng có các tham số khác nhau hoặc kiểu dữ liệu khác nhau. Ví dụ:

```

1 class Calculation:
2     def add(self, a, b):
3         return a + b
4     def add(self, a, b, c):
5         return a + b + c

```

- Overriding (Ghi đè): Là khả năng của một lớp con thay đổi triển khai của một phương thức mà đã được định nghĩa trong lớp cha. Ví dụ:

```

1 class Animal:
2     def make_sound(self):
3         return "Generic sound"
4 class Dog(Animal):
5     def make_sound(self):
6         return "Woof!"

```

- **Đa hình ở thời điểm chạy (Runtime Polymorphism):**

Được triển khai thông qua kỹ thuật gọi phương thức của đối tượng dựa trên lớp của đối tượng đó. Ví dụ:

```

1 class Animal:
2     def make_sound(self):
3         return "Generic sound"
4 class Dog(Animal):
5     def make_sound(self):
6         return "Woof!"
7 class Cat(Animal):
8     def make_sound(self):
9         return "Meow!"
10 # Đa hình tại thời điểm chạy
11 def animal_sound(animal):
12     return animal.make_sound()
13 dog = Dog()
14 cat = Cat()
15 print(animal_sound(dog)) # Kết quả: Woof!
16 print(animal_sound(cat)) # Kết quả: Meow!

```

Ở ví dụ trên, hàm “**animal\_sound()**” có thể nhận vào các đối tượng từ các lớp khác nhau kế thừa từ **Animal**. Khi gọi “**make\_sound()**”, nó sẽ thực thi phương thức của đối tượng cụ thể tùy thuộc vào lớp của đối tượng được chuyển vào.

### 1.4.7 Trừu tượng hóa (Abstraction)

Trừu tượng hóa (Abstraction) là quá trình ẩn các chi tiết cài đặt nội tại của một đối tượng và chỉ hiển thị các tính năng quan trọng hoặc cần thiết cho người sử dụng. Các đặc điểm của trừu tượng hóa:

- **Ứng dụng tập trung:** Trừu tượng hóa cho phép người sử dụng tập trung vào việc sử dụng đối tượng mà không cần biết chi tiết cài đặt bên trong của nó.
- **Giảm sự phức tạp:** Nó giúp giảm bớt sự phức tạp của hệ thống bằng cách che giấu các chi tiết phức tạp và chỉ hiển thị những thông tin cần thiết.
- **Tạo ra mô hình trừu tượng:** Trừu tượng hóa cho phép tạo ra mô hình trừu tượng cho các đối tượng trong thế giới thực, giúp trong việc phát triển và bảo trì mã nguồn dễ dàng hơn.

Ví dụ về trừu tượng hóa trong Python:

```

1 from abc import ABC, abstractmethod
2 # Lớp trừu tượng
3 class Animal(ABC):
4     @abstractmethod
5         def make_sound(self):
6             pass
7 # Lớp con thực hiện (định nghĩa phương thức cụ thể)
8 class Dog(Animal):
9     def make_sound(self):
10        return "Woof!"
11 # Lớp con thực hiện (định nghĩa phương thức cụ thể)
12 class Cat(Animal):
13     def make_sound(self):
14         return "Meow!"
15 # Sử dụng các đối tượng trừu tượng
16 dog = Dog()
17 cat = Cat()
18 print(dog.make_sound()) # Kết quả: Woof!
19 print(cat.make_sound()) # Kết quả: Meow!

```

Trong ví dụ trên, lớp “**Animal**” là lớp trừu tượng và có một phương thức trừu tượng “**make\_sound()**”, không có triển khai cụ thể. Hai lớp con “**Dog**” và “**Cat**” kế thừa từ lớp “**Animal**” và triển khai phương thức “**make\_sound()**” một cách cụ thể cho từng loài động vật.

## **1.5 CÔNG CỤ THỰC HÀNH**

### **1.5.1 Trình biên dịch Python**

Để bắt đầu lập trình với Python, chúng ta cần cài đặt Python vào máy tính. Sinh viên thực hiện theo các bước sau:

#### ❖ **Trên Windows:**

- Truy cập vào trang web <https://www.python.org/> và chọn phiên bản Python phù hợp với hệ điều hành Windows (32-bit hoặc 64-bit). Sau đó tải về và cài đặt.
- Chạy trình cài đặt: Đảm bảo đã chọn tùy chọn "Add Python to PATH" để thêm vào biến môi trường PATH.

- Kiểm tra cài đặt: Hãy mở Command Prompt và nhập “**python --version**” hoặc “**python -V**” để kiểm tra phiên bản Python đã cài đặt.
- **Trên macOS:**
  - Cài đặt Python bằng Homebrew: Trong Terminal, nhập lệnh “**brew install python**” và theo hướng dẫn để cài đặt Python.
  - Kiểm tra cài đặt: Mở Terminal và nhập “**python3 --version**” để kiểm tra phiên bản Python đã cài đặt.
- **Trên Linux (phần lớn các bản phân phối):**
  - Mở Terminal và chạy lệnh sau: “**sudo apt-get update && sudo apt-get install python3**”.
  - Kiểm tra cài đặt: Mở Terminal và nhập “**python3 --version**” để kiểm tra.

## 1.5.2 IDE Visual Studio Code

“Visual Studio Code (VS Code) là trình soạn thảo mã nguồn (IDE) phổ biến được sử dụng rộng rãi cho lập trình” [3]. Một số điểm mạnh của VS Code khi sử dụng cho lập trình Python:

- Hỗ trợ ngôn ngữ Python: VS Code cung cấp hỗ trợ tốt cho Python.
- Extension và tích hợp: Có thể cài đặt các extension để cung cấp các tính năng mở rộng như debugging, linting, phân tích code,....
- Debugging Python: VS Code cung cấp công cụ debugging tích hợp cho Python.
- Terminal tích hợp: Có terminal tích hợp giúp thực thi các lệnh Python và quản lý môi trường làm việc một cách thuận tiện.
- Tích hợp Git: Cung cấp tích hợp với Git để quản lý phiên bản và hợp tác với các dự án Python trong quá trình phát triển.
- Mở rộng và tùy chỉnh: Có thể mở rộng VS Code thông qua các extension.

## 1.5.3 Hệ thống kiểm soát phiên bản (Git)

“Các dự án trong thực tế thường có nhiều lập trình viên làm việc song song. Vì vậy, cần có một hệ thống kiểm soát phiên bản như Git là cần thiết để đảm bảo không

có xung đột mã giữa các lập trình viên. Ngoài ra, những yêu cầu trong dự án thay đổi thường xuyên. Vì vậy, cần có một hệ thống có thể cho phép nhà phát triển quay lại phiên bản cũ hơn của mã” [4].

Một số khái niệm trong Git:

- Repository (Kho hoặc repo): Là nơi lưu trữ tất cả các file, thư mục và lịch sử thay đổi của dự án.
- Commit: Là hành động lưu trữ các thay đổi trong mã nguồn vào repo. Dùng lệnh “**git commit**” để tạo ra một commit trong Git.
- Branch (Nhánh): Là một phiên bản song song của mã nguồn trong repo, giúp phát triển các tính năng, sửa lỗi mà không làm ảnh hưởng đến nhánh chính.
- Merge và Pull Request (PR): Merge là quá trình hợp nhất các thay đổi từ một nhánh vào nhánh khác. Pull Request (PR) là một yêu cầu được tạo ra để hợp nhất các thay đổi từ một nhánh vào nhánh chính.
- Remote: Là các repo trên máy chủ từ xa như GitHub, GitLab, nơi có thể chia sẻ và đồng bộ hóa code.
- Staging Area (Index): Là nơi lưu trữ các thay đổi đã được chuẩn bị để commit. Trước khi commit, các thay đổi cần được thêm vào staging area.
- Conflict (Xung đột): Xảy ra khi có sự mâu thuẫn giữa các thay đổi trên cùng một dòng trong mã nguồn từ các nhánh khác nhau.

## 1.6 BÀI TẬP THỰC HÀNH

### 1.6.1 Bài thực hành 01: Cài đặt môi trường

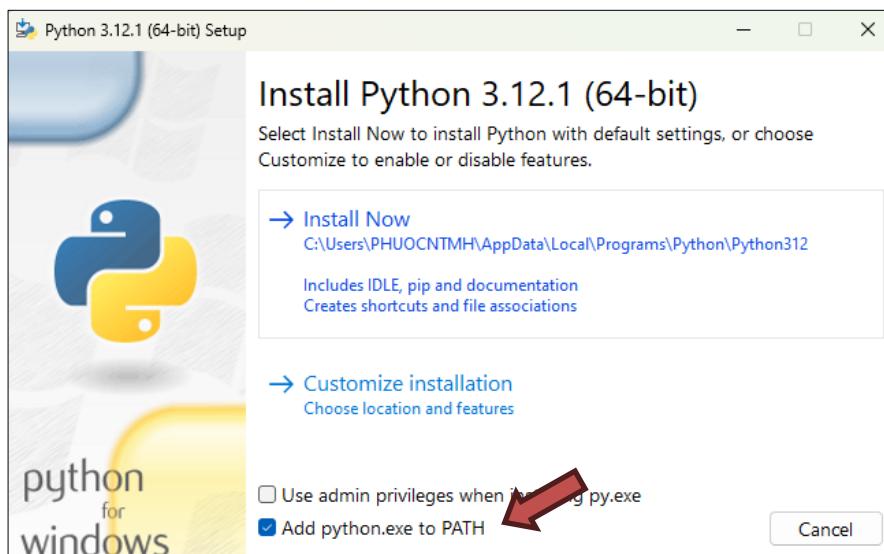
Sinh viên tiến hành cài đặt môi trường lập trình cho ngôn ngữ Python và tạo các tài khoản cần thiết, bao gồm các phần sau:

#### 1. Cài đặt Python:

- Truy cập trang web <https://www.python.org/> ➔ Download và chọn phiên bản phù hợp với hệ điều hành.



- Chạy trình cài đặt: Đảm bảo rằng đã tích chọn tùy chọn "Add Python to PATH" để thêm Python vào trong biến môi trường PATH.



- Kiểm tra cài đặt: Hãy mở Command Prompt và nhập vào "python -version" hoặc "python -V".

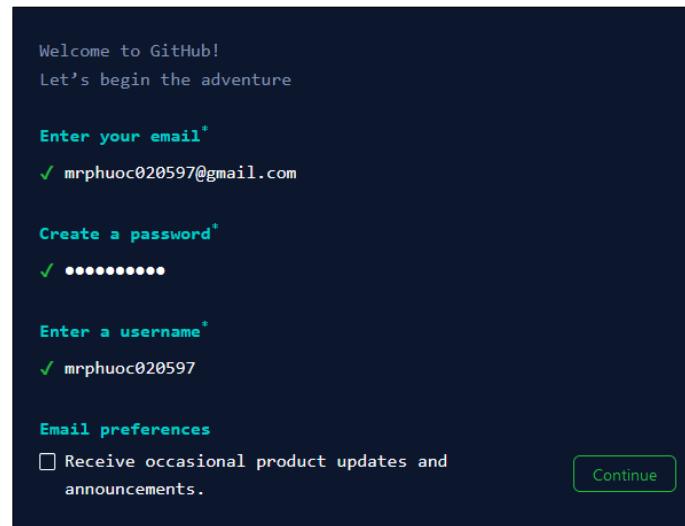
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

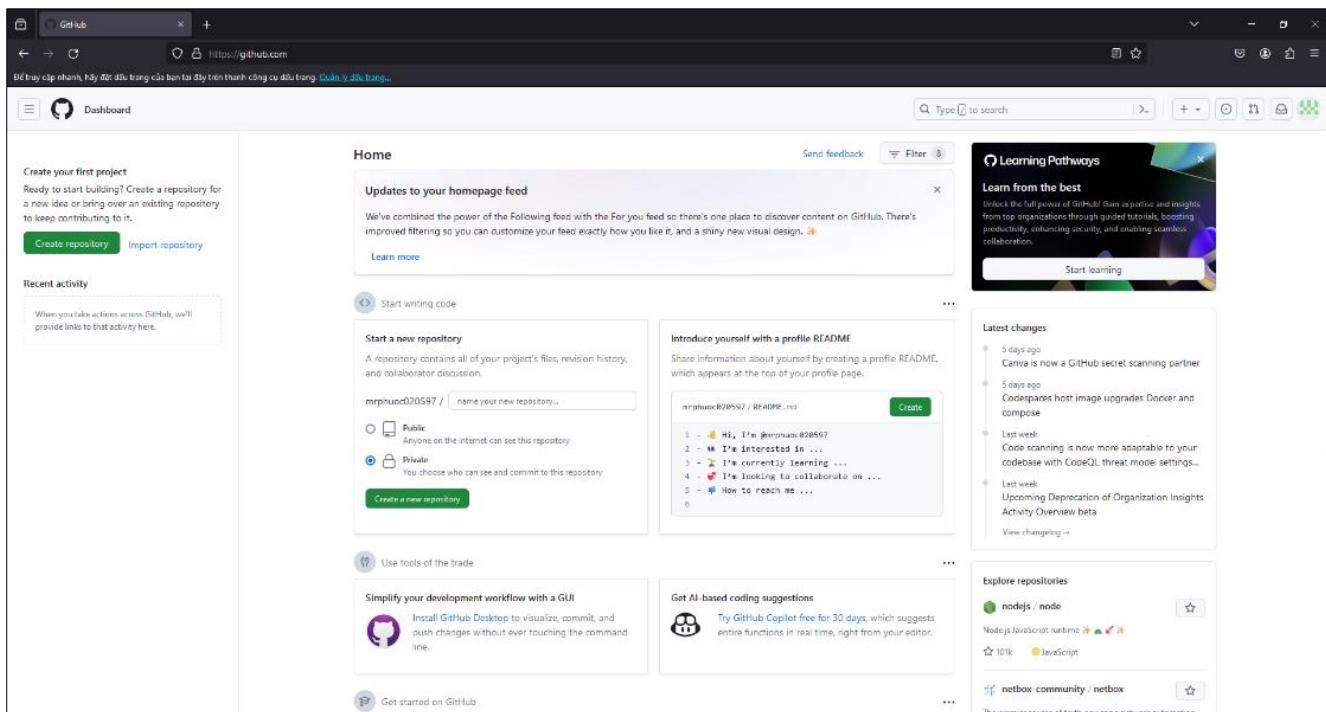
PS C:\Users\PHUOCNTMH> python --version
Python 3.12.3
PS C:\Users\PHUOCNTMH> |
```

## 2. Tạo tài khoản Github, cài đặt Git, tạo và kéo repo về local:

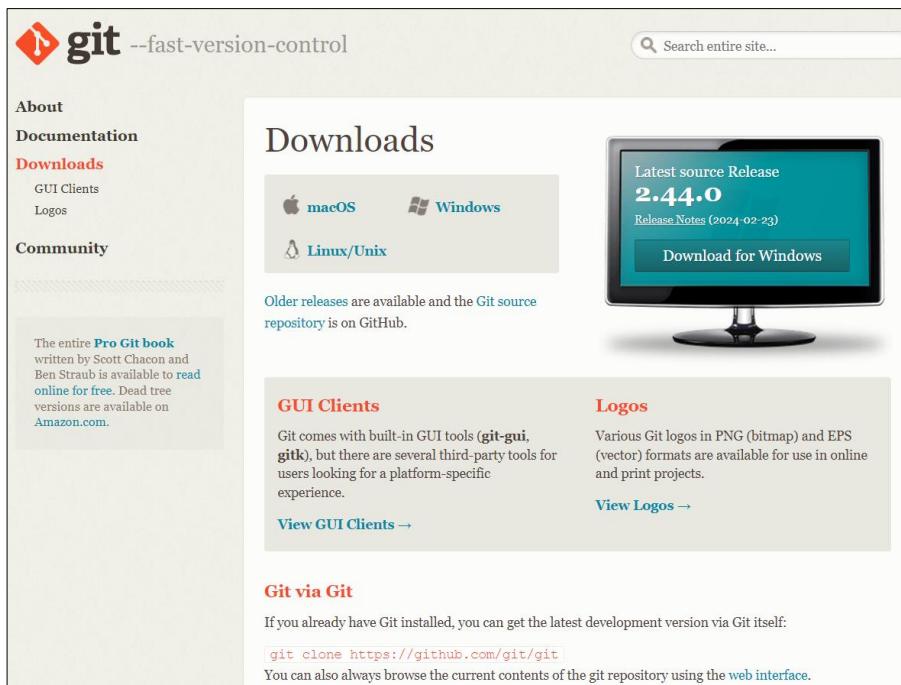
- Truy cập <https://github.com/> → Chọn “Sign up” để đăng ký tài khoản.



- Sau khi xác thực thông tin thành công, tiến hành đăng nhập, chúng ta được giao diện như sau:



- Truy cập <https://git-scm.com/downloads> và chọn phiên bản phù hợp với hệ điều hành.



- Chạy file vừa tải về, tiến hành cài đặt phần mềm vào máy tính.
- Kiểm tra cài đặt: Hãy mở Command Prompt và nhập "git --version".

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\PHUOCNTMH> git --version
git version 2.44.0.windows.1
PS C:\Users\PHUOCNTMH> |
```

- Vào địa chỉ <https://github.com/> ➔ Chọn "+" ➔ Chọn "New Repository".
  - Repository name: Nhập "bmttnc-hutech-<MSV>".
  - Description: Nhập "< họ\_tên\_SV >\_<MSV>".
  - Chọn "Private".
  - Chọn "Add a README file".
  - Chọn "Create repository" để tạo mới repo.

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

Owner *	Repository name *
 mrphuoc020597	/ bmtt-nc-hutech-151106024
 bmtt-nc-hutech-1511060249 is available.	

Great repository names are short and memorable. Need inspiration? How about [expert-meme](#)?

Description (optional)  
NguyenTrongMinhHongPhuoc\_1511060249

 **Public**  
Anyone on the internet can see this repository. You choose who can commit.

 **Private**  
You choose who can see and commit to this repository.

Initialize this repository with  
 **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore  
.gitignore template: None

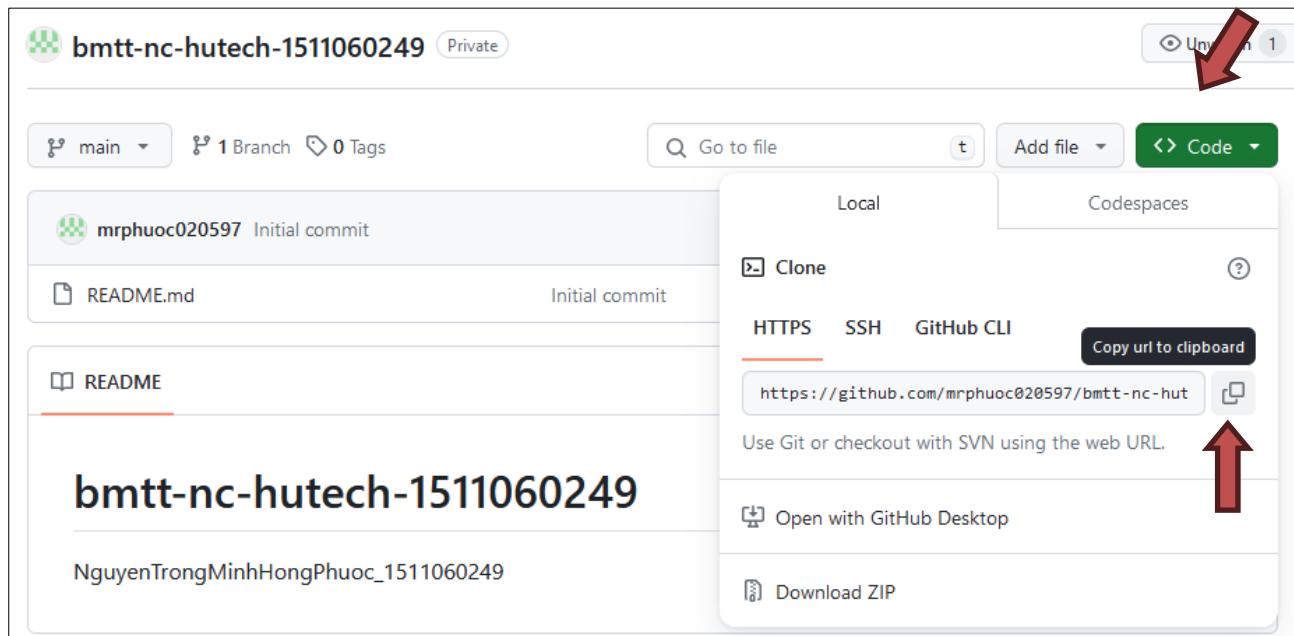
Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license  
License: None

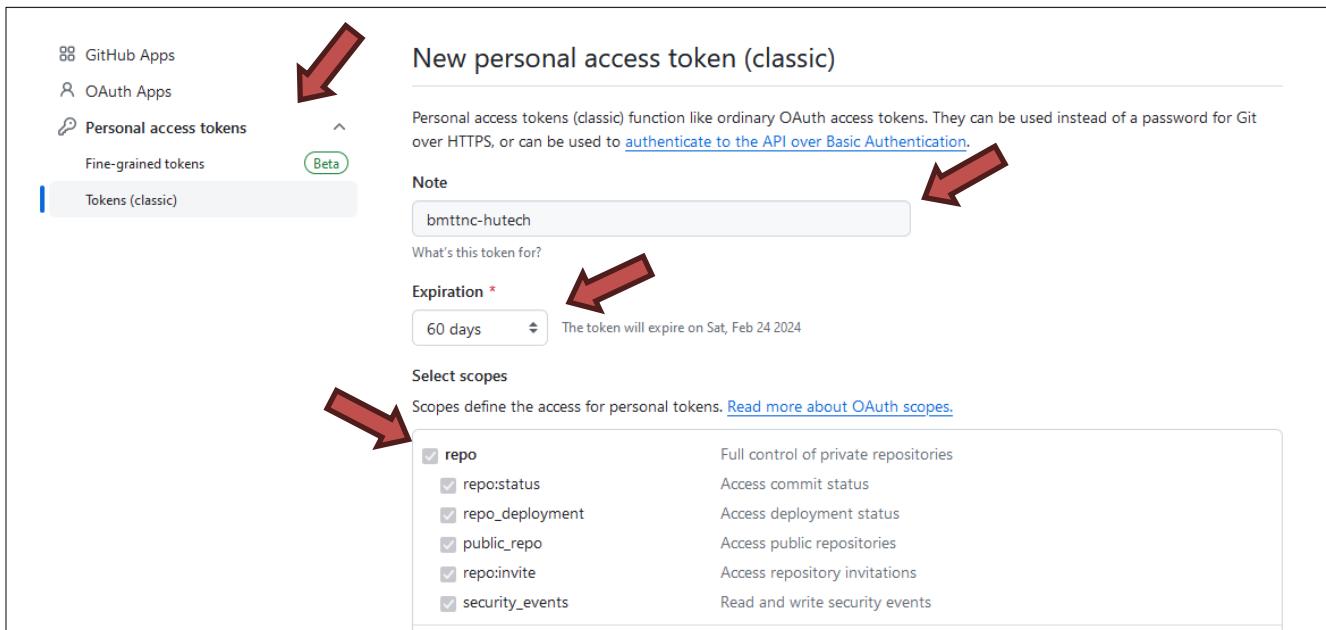
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  main as the default branch. Change the default name in your [settings](#).

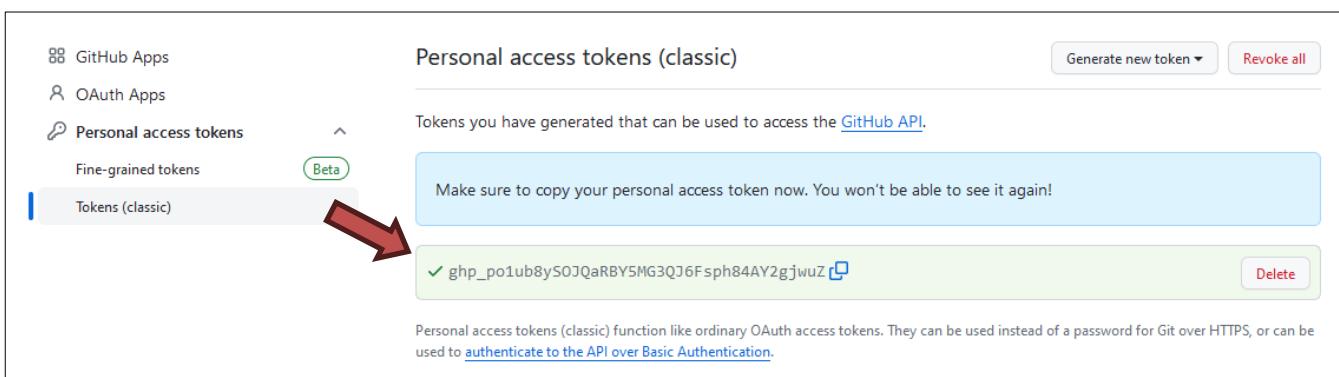
- Chọn "<> Code" → HTTPS → Nhấn copy đường dẫn tới repo.



- Vào <https://github.com/settings/profile> → Chọn “Developer Settings” → Chọn “Personal access tokens” → “Tokens (classic)” → Chọn “Generate a personal access token”.
- Nhập Note là “bmtt-hutech”, ở Expiration chọn “60 days”. Chọn các mục trong “Select scopes” → Chọn “Generate token” để tạo mới token.



- Lưu trữ “Personal access tokens (classic)” tránh thất lạc cho các lần dùng tiếp theo.



- Tại máy tính thực hành, tiến hành clone repo về local. Mở Command Prompt (lưu ý chuyển ở đĩa hiện thời sang ổ đĩa thực hành):

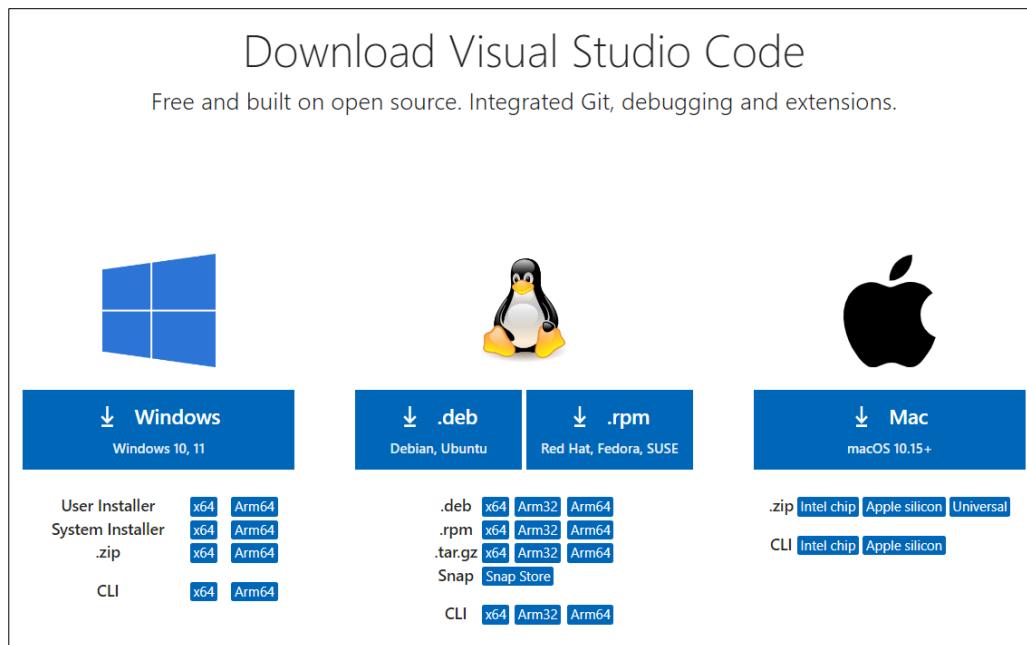
```
C:\Users\PHUOCNTMH>D:  
D:\>git clone <đường dẫn tới repo github của sinh viên>  
Cloning into 'bmtt-nc-hutech-1511060249'...
```

- Cửa sổ “Connect to Github” hiện ra → Chọn “Token” → Nhập “Personal access token” vừa được tạo ở bước trên vào. Repo được clone về thành công.

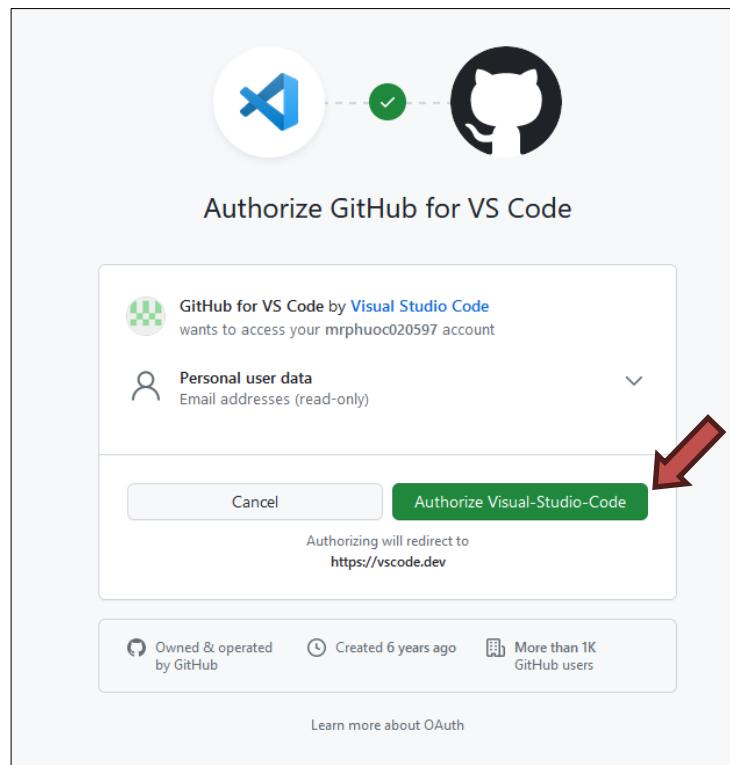
```
D:\>git clone <đường dẫn tới repo github của sinh viên>
Cloning into 'bmtt-nc-hutech-1511060249'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

### 3. Cài đặt Visual Studio Code và cấu hình:

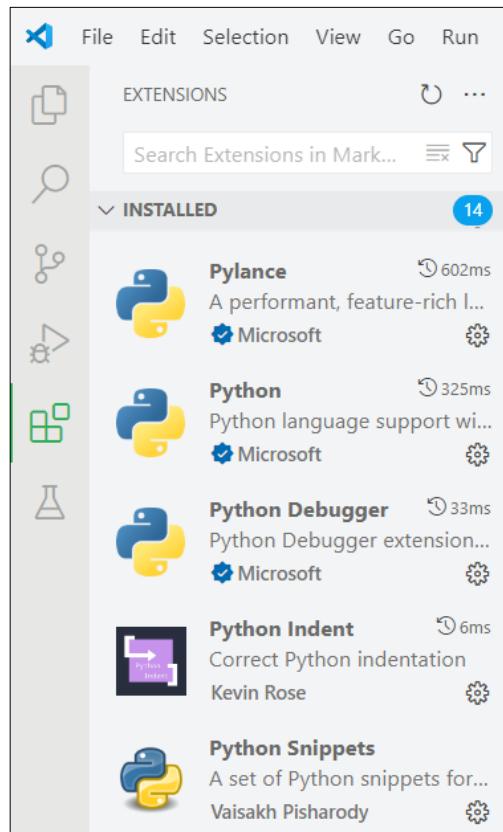
- Vào trang <https://code.visualstudio.com/download> và tải phiên bản phù hợp với hệ điều hành.



- Sau khi tải file cài đặt về, tiến hành cài đặt phần này mềm vào máy tính.
- Mở VS Code, chọn “Accounts” → Chọn “Sign in to Sync Settings”. Chọn “Authorize Visual-Studio-Code”. Hệ thống sẽ đưa trở về giao diện VS Code. Kiểm tra tại mục “Accounts” xem đã có tài khoản được kết nối hay chưa.

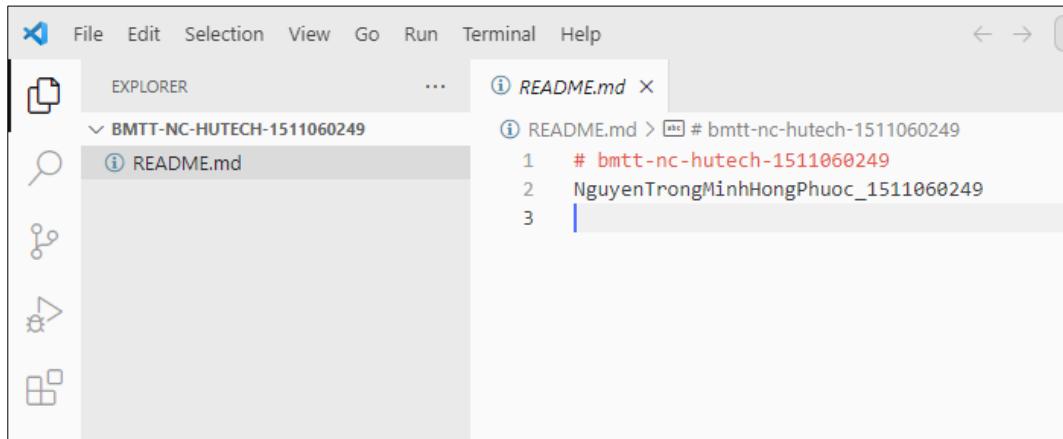


- Vào thẻ “Extensions”, tiến hành cài đặt các extension sau: Python, Pylance, Python Indent, Python Snippets.

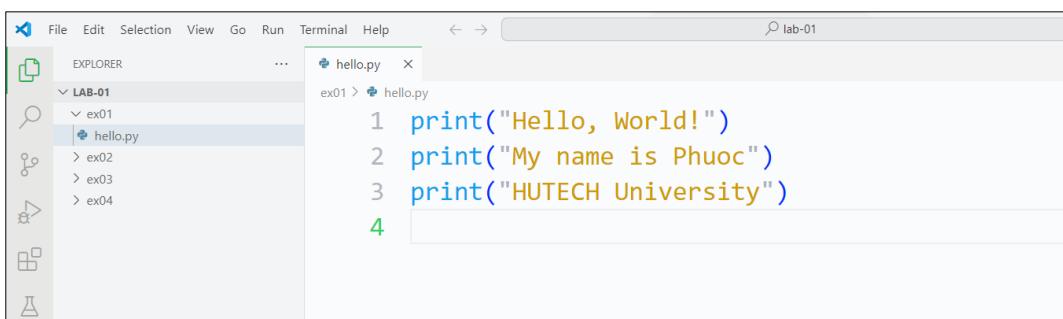


#### 4. Ví dụ cơ bản với Python:

- Mở VS Code → Chọn “File” → “Open Folder...”. Chọn đường dẫn đến project được clone về từ Github ở bước trên → Chọn “Yes, I trust the authors”.



- Tạo mới thư mục “lab-01”. Trong thư mục “lab-01” tạo file “hello.py”.



- Lưu file. Vào menu “Terminal” → “New Terminal” hoặc nhấn tổ hợp phím Ctrl + Shift + `. Kiểm tra chương trình vừa tạo.

```

PS D:\bmtt-nc-hutech-1511060249> cd .\lab-01\
PS D:\bmtt-nc-hutech-1511060249\lab-01> python .\hello.py
Hello, World!
My name is Phuoc
HUTECH University
PS D:\bmtt-nc-hutech-1511060249\lab-01>

```

- Cài đặt thông tin “Git account's default identity”:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\bmmtt-nc-hutech-1511060249\lab-01> git config --global user.email "mrphuoc020597@gmail.com"
PS D:\bmmtt-nc-hutech-1511060249\lab-01> git config --global user.name "mrphuoc020597"
PS D:\bmmtt-nc-hutech-1511060249\lab-01> git config --global user.email
mrphuoc020597@gmail.com
PS D:\bmmtt-nc-hutech-1511060249\lab-01> git config --global user.name
mrphuoc020597
```

- Đưa các thay đổi lên remote repo:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\bmmtt-nc-hutech-1511060249\lab-01> git add .
PS D:\bmmtt-nc-hutech-1511060249\lab-01> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   hello.py

PS D:\bmmtt-nc-hutech-1511060249\lab-01> git commit -m "lab01 hello world"
[main b9d827e] lab01 hello world
 1 file changed, 3 insertions(+)
 create mode 100644 lab-01/hello.py
PS D:\bmmtt-nc-hutech-1511060249\lab-01> git push origin
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 385 bytes | 385.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/mrphuoc020597/bmmtt-nc-hutech-1511060249.git
 8a5d785..b9d827e main -> main
PS D:\bmmtt-nc-hutech-1511060249\lab-01>
```

- Kiểm tra trên website <https://github.com>:

**bmmtt-nc-hutech-1511060249** (Private)

Unwatch 1

main 1 Branch 0 Tags

Go to file Add file Code

	mrphuoc020597 lab01 hello world	b9d827e · 1 minute ago 2 Commits
	lab-01	lab01 hello world 1 minute ago
	README.md	Initial commit 51 minutes ago

## 1.6.2 Bài thực hành 02: Lập trình Python cơ bản

- Câu 01:** Viết chương trình yêu cầu người dùng nhập vào họ tên và tuổi của họ, sau đó in ra màn hình thông điệp chào mừng với thông tin vừa nhập.

Hướng dẫn: Tạo file “ex02\_01.py” trong “lab\_01”.

```
1 # Nhập tên và tuổi từ người dùng
2 ten = input("Nhập tên của bạn: ")
3 tuoi = input("Nhập tuổi của bạn: ")
4 # In thông điệp chào mừng với thông tin vừa nhập
5 print("Chào mừng,", ten, "! Bạn", tuoi, "tuổi.")
```

Kết quả:

```
ex02_01.py x
ex02 > ex02_01.py > ...
1 # Nhập tên và tuổi từ người dùng
2 ten = input("Nhập tên của bạn: ")
3 tuoi = input("Nhập tuổi của bạn: ")
4 # In thông điệp chào mừng với thông tin vừa nhập
5 print("Chào mừng,", ten, "! Bạn", tuoi, "tuổi.")
6

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\lab-01> cd ex02
PS D:\lab-01\ex02> python ex02_01.py
Nhập tên của bạn: Phuoc Nguyen
Nhập tuổi của bạn: 26
Chào mừng, Phuoc Nguyen ! Bạn 26 tuổi.
PS D:\lab-01\ex02>
```

- Câu 02:** Viết chương trình thực hiện tính diện tích của hình tròn với bán kính được nhập từ người dùng. Sử dụng giá trị Pi là 3.14.

Hướng dẫn: Tạo file “ex02\_02.py” trong “lab\_01”.

```
1 # Nhập bán kính từ người dùng
2 ban_kinh = float(input("Nhập bán kính của hình tròn: "))
3 # Tính diện tích của hình tròn
4 dien_tich = 3.14 * (ban_kinh ** 2)
5 # In diện tích của hình tròn ra màn hình
6 print("Diện tích của hình tròn là:", dien_tich)
```

- Kết quả:

```

1 # Nhập bán kính từ người dùng
2 ban_kinh = float(input("Nhập bán kính của hình tròn: "))
3 # Tính diện tích của hình tròn
4 dien_tich = 3.14 * (ban_kinh ** 2)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\lab-01> cd ex02
PS D:\lab-01\ex02> python ex02_02.py
Nhập bán kính của hình tròn: 5.7
Diện tích của hình tròn là: 102.0186
PS D:\lab-01\ex02>

```

- Câu 03:** Viết chương trình kiểm tra một số nhập vào từ người dùng có phải là số chẵn hay không.

Hướng dẫn: Tạo file "ex02\_03.py" trong "lab\_01".

```

1 # Nhập số từ người dùng
2 so = int(input("Nhập một số nguyên: "))
3 # Kiểm tra xem số đó có phải số chẵn hay không
4 if so % 2 == 0:
5     print(so, "là số chẵn.")
6 else:
7     print(so, "không phải là số chẵn.")


```

Kết quả:

```

1 # Nhập số từ người dùng
2 so = int(input("Nhập một số nguyên: "))
3 # Kiểm tra xem số đó có phải số chẵn hay không
4 if so % 2 == 0:
5     print(so, "là số chẵn.")
6 else:
7     print(so, "không phải là số chẵn.")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\lab-01> cd ex02
PS D:\lab-01\ex02> python ex02_03.py
Nhập một số nguyên: 10
10 là số chẵn.
PS D:\lab-01\ex02> python ex02_03.py
Nhập một số nguyên: 7
7 không phải là số chẵn.
PS D:\lab-01\ex02>

```

- Câu 04:** Xây dựng một chương trình nhằm tìm tất cả các số nằm trong khoảng từ 2000 đến 3200 (bao gồm cả 2000 và 3200) mà chia hết cho 7 và không phải là bội số của 5. Các số tìm được sẽ được in ra trên một dòng, ngăn cách bởi dấu phẩy.

Hướng dẫn: Tạo file "ex02\_04.py" trong "lab\_01".

```

1 # Tạo một danh sách rỗng để lưu kết quả
2 j=[]
3 # Duyệt qua tất cả các số trong đoạn từ 2000 đến 3200, kiểm tra xem
   số i có chia hết cho 7 và không phải là bội số của 5 không
4 for i in range(2000, 3201):
5     if (i % 7 == 0) and (i % 5 != 0):
6         j.append(str(i))
7 print (','.join(j))

```

Kết quả:

The screenshot shows a code editor window with the file `ex02_04.py` open. The code is identical to the one above. Below the code editor is a terminal window showing the execution of the script and its output. The terminal output is as follows:

```

PS D:\lab-01> cd ex02
PS D:\lab-01\ex02> python ex02_04.py
2002,2009,2016,2023,2037,2044,2051,2058,2072,2079,2086,2093,2107,2114,2121,2128,2142,2149,2156,2163,2177,2184,2191,2198,2212,2219,2226,2233,2247,225
4,2261,2268,2282,2289,2296,2303,2317,2324,2331,2338,2352,2359,2366,2373,2387,2394,2401,2408,2422,2429,2436,2443,2457,2464,2471,2478,2492,2499,2506,2
513,2527,2534,2541,2548,2562,2569,2576,2583,2597,2604,2611,2618,2632,2639,2646,2653,2667,2674,2681,2688,2702,2709,2716,2723,2737,2744,2751,2758,2772
,2779,2786,2793,2807,2814,2821,2828,2842,2849,2856,2863,2877,2884,2891,2898,2912,2919,2926,2933,2947,2954,2961,2968,2982,2989,2996,3003,3017,3024,30
31,3038,3052,3059,3066,3073,3087,3094,3101,3108,3122,3129,3136,3143,3157,3164,3171,3178,3192,3199
PS D:\lab-01\ex02>

```

- Câu 05:** Xây dựng một chương trình nhằm nhập số giờ làm việc hàng tuần của nhân viên và mức lương theo giờ tiêu chuẩn. Từ đó, thực hiện tính toán số tiền thực nhận của nhân viên. Cần lưu ý rằng số giờ tiêu chuẩn mỗi tuần là 44 giờ, và mỗi giờ làm thêm sẽ được trả 150% so với mức lương theo giờ tiêu chuẩn.

Hướng dẫn: Tạo file “ex02\_05.py” trong “lab\_01”.

```

1 so_gio_lam = float(input("Nhập số giờ làm mỗi tuần: "))
2 luong_gio = float(input("Nhập thù lao trên mỗi giờ làm tiêu chuẩn: "))
3 gio_tieu_chuan = 44 # Số giờ làm chuẩn mỗi tuần
4 gio_vuot_chuan = max(0, so_gio_lam - gio_tieu_chuan) # Số giờ làm
   vượt chuẩn mỗi tuần
5 thuc_linh = gio_tieu_chuan * luong_gio + gio_vuot_chuan * luong_gio *
   1.5 # Tính tổng thu nhập
6 print(f"Số tiền thực lĩnh của nhân viên: {thuc_linh}")

```

Kết quả:

```

ex02_05.py >
ex02 > ex02_05.py > ...
1 so_gio_lam = float(input("Nhập số giờ làm mỗi tuần: "))
2 luong_gio = float(input("Nhập thù lao trên mỗi giờ làm tiêu chuẩn: "))
3 gio_tieu_chuan = 44 # Số giờ làm chuẩn mỗi tuần
4 gio_vuot_chuan = max(0, so_gio_lam - gio_tieu_chuan) # Số giờ làm
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\lab-01> cd ex02
PS D:\lab-01\ex02> python ex02_05.py
Nhập số giờ làm mỗi tuần: 76.5
Nhập thù lao trên mỗi giờ làm tiêu chuẩn: 150000
Số tiền thực lĩnh của nhân viên: 13912500.0
PS D:\lab-01\ex02>

```

- **Câu 06:** Tạo một chương trình để nhập vào hai số X và Y; từ đó, xây dựng một mảng hai chiều. Giá trị của mỗi phần tử tại hàng i và cột j của mảng sẽ là  $i*j$ , với i chạy từ 0 đến X-1 và j từ 0 đến Y-1. Chẳng hạn, nếu X và Y được nhập là 3 và 5, thì kết quả sẽ là: [[0, 0, 0, 0, 0], [0, 1, 2, 3, 4], [0, 2, 4, 6, 8]].

Hướng dẫn: Tạo file "ex02\_06.py" trong "lab\_01".

```

1 input_str = input("Nhập X, Y: ")
2 dimensions=[int(x) for x in input_str.split(',')]
3 rowNum=dimensions[0]
4 colNum=dimensions[1]
5 multilist = [[0 for col in range(colNum)] for row in range(rowNum)]
6 for row in range(rowNum):
7     for col in range(colNum):
8         multilist[row][col]= row*col
9 print (multilist)

```

Kết quả:

```

ex02_06.py >
ex02 > ex02_06.py > ...
1 input_str = input("Nhập X, Y: ")
2 dimensions=[int(x) for x in input_str.split(',')]
3 rowNum=dimensions[0]
4 colNum=dimensions[1]
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\lab-01> cd ex02
PS D:\lab-01\ex02> python ex02_06.py
Nhập X, Y: 3, 5
[[0, 0, 0, 0, 0], [0, 1, 2, 3, 4], [0, 2, 4, 6, 8]]
PS D:\lab-01\ex02>

```

- **Câu 07:** Xây dựng một chương trình để nhận các chuỗi đầu vào là những dòng được nhập, sau đó chuyển đổi các dòng này thành chữ hoa và hiển thị kết quả lên màn hình.

Hướng dẫn: Tạo file "ex02\_07.py" trong "lab\_01".

```

1 # Nhập các dòng từ người dùng
2 print("Nhập các dòng văn bản (Nhập 'done' để kết thúc):")
3 lines = []
4 while True:
5     line = input()
6     if line.lower() == 'done':
7         break
8     lines.append(line)
9 # Chuyển các dòng thành chữ in hoa và in ra màn hình
10 print("\nCác dòng đã nhập sau khi chuyển thành chữ in hoa:")
11 for line in lines:
12     print(line.upper())

```

Kết quả:



```

ex02_07.py x
ex02 > ex02_07.py > ...
1 # Nhập các dòng từ người dùng
2 print("Nhập các dòng văn bản (Nhập 'done' để kết thúc):")
3 lines = []
4 while True:
5     line = input()
6     if line.lower() == 'done':
7         break
8     lines.append(line)
9 # Chuyển các dòng thành chữ in hoa và in ra màn hình
10 print("\nCác dòng đã nhập sau khi chuyển thành chữ in hoa:")
11 for line in lines:
12     print(line.upper())
13

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ ... ^ x
Nhập các dòng văn bản (Nhập 'done' để kết thúc):
hutech university
Thuc hanh Bao mat thong tin nang cao
Lap trinh bang ngon ngu Python
done

Các dòng đã nhập sau khi chuyển thành chữ in hoa:
HUTECH UNIVERSITY
THUC HANH BAO MAT THONG TIN NANG CAO
LAP TRINH BANG NGON NGU PYTHON
PS D:\lab-01\ex02>

```

- Câu 08:** Hãy phát triển một chương trình để nhập vào một chuỗi các số nhị phân với 4 chữ số, phân cách bằng dấu phẩy. Chương trình sẽ kiểm tra từng số để xác định xem chúng có chia hết cho 5 hay không, sau đó in ra những số thỏa mãn điều kiện này, cũng được phân tách bằng dấu phẩy. Ví dụ, nếu đầu vào là: '0100', '0011', '1010', '1001', thì kết quả đầu ra sẽ là: '1010'.

Hướng dẫn: Tạo file "ex02\_08.py" trong "lab\_01".

```

1 # Hàm kiểm tra số nhị phân có chia hết cho 5 không
2 def chia_het_cho_5(so_nhi_phan):
3     # Chuyển số nhị phân sang số thập phân
4     so_thap_phan = int(so_nhi_phan, 2)
5     # Kiểm tra xem số thập phân có chia hết cho 5 không
6     if so_thap_phan % 5 == 0:
7         return True
8     else:
9         return False
10 # Nhập chuỗi số nhị phân từ người dùng
11 chuoisogniphinan = input("Nhập chuỗi số nhị phân (phân tách bởi dấu
12 phẩy): ")
13
14 # Tách chuỗi thành các số nhị phân và kiểm tra số chia hết cho 5
15 so_nhi_phan_list = chuoisogniphinan.split(',')
16 so_chia_het_cho_5 = [so for so in so_nhi_phan_list if chia_het_cho_5
17 (so)]
18
19 # In ra các số nhị phân chia hết cho 5
20 if len(so_chia_het_cho_5) > 0:
21     ket_qua = ','.join(so_chia_het_cho_5)
22     print("Các số nhị phân chia hết cho 5 là:", ket_qua)
23 else:
24     print("Không có số nhị phân nào chia hết cho 5 trong chuỗi đã
25 nhập.")

```

Kết quả:

```

ex02_08.py x
ex02 > ex02_08.py > ...
1 # Hàm kiểm tra số nhị phân có chia hết cho 5 không
2 def chia_het_cho_5(so_nhi_phan):
3     # Chuyển số nhị phân sang số thập phân
4     so_thap_phan = int(so_nhi_phan, 2)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\lab-01> cd ex02
PS D:\lab-01\ex02> python ex02_08.py
Nhập chuỗi số nhị phân (phân tách bởi dấu phẩy): 0100, 0011, 1010, 1001
Các số nhị phân chia hết cho 5 là: 1010
PS D:\lab-01\ex02>

```

- **Câu 09:** Viết hàm kiểm tra xem một số được nhập vào có phải là số nguyên tố hay không.

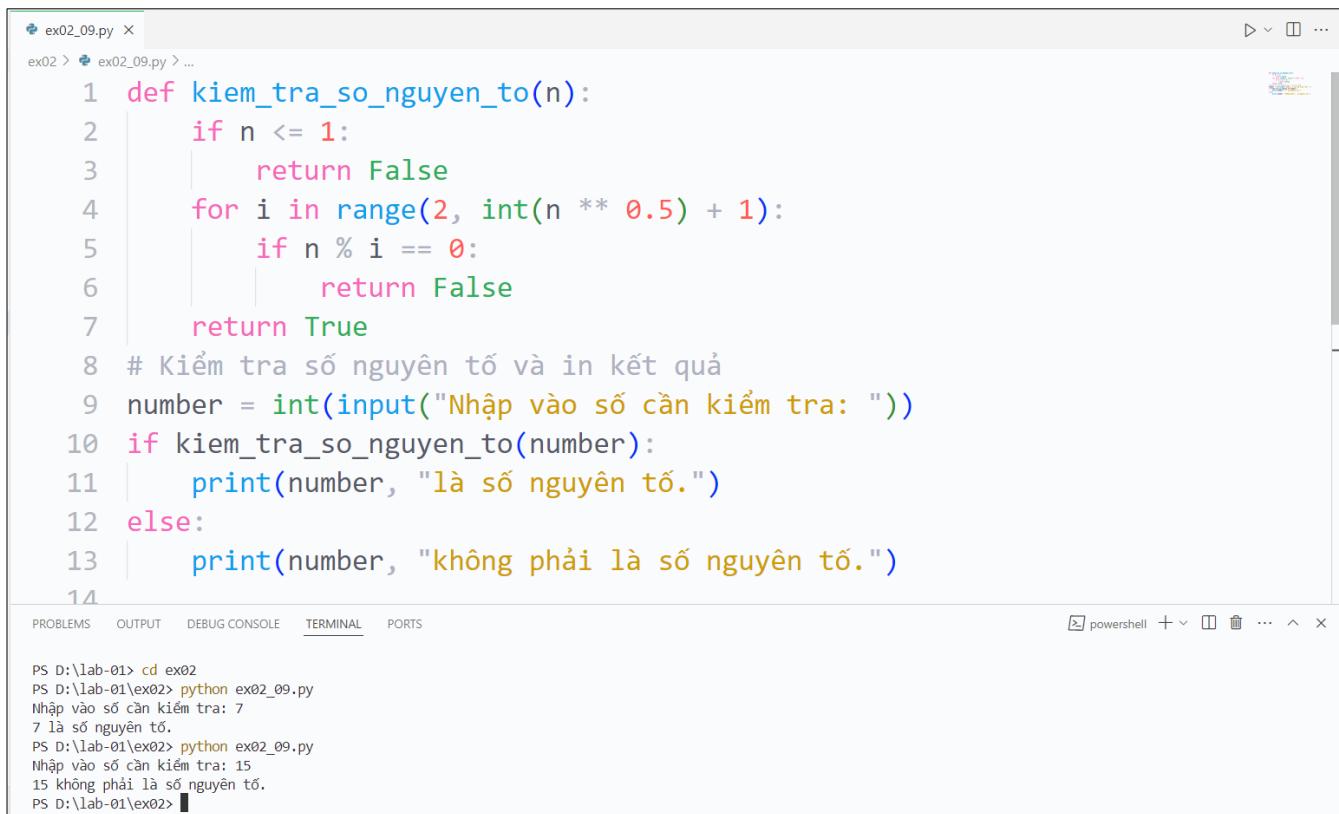
Hướng dẫn: Tạo file “ex02\_09.py” trong “lab\_01”.

```

1 def kiem_tra_so_nguyen_to(n):
2     if n <= 1:
3         return False
4     for i in range(2, int(n ** 0.5) + 1):
5         if n % i == 0:
6             return False
7     return True
8 # Kiểm tra số nguyên tố và in kết quả
9 number = int(input("Nhập vào số cần kiểm tra: "))
10 if kiem_tra_so_nguyen_to(number):
11     print(number, "là số nguyên tố.")
12 else:
13     print(number, "không phải là số nguyên tố.")

```

Kết quả:



```

ex02_09.py ×
ex02 > ex02_09.py > ...
1 def kiem_tra_so_nguyen_to(n):
2     if n <= 1:
3         return False
4     for i in range(2, int(n ** 0.5) + 1):
5         if n % i == 0:
6             return False
7     return True
8 # Kiểm tra số nguyên tố và in kết quả
9 number = int(input("Nhập vào số cần kiểm tra: "))
10 if kiem_tra_so_nguyen_to(number):
11     print(number, "là số nguyên tố.")
12 else:
13     print(number, "không phải là số nguyên tố.")
14

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS D:\lab-01> cd ex02
PS D:\lab-01\ex02> python ex02_09.py
Nhập vào số cần kiểm tra: 7
7 là số nguyên tố.
PS D:\lab-01\ex02> python ex02_09.py
Nhập vào số cần kiểm tra: 15
15 không phải là số nguyên tố.
PS D:\lab-01\ex02>

```

- **Câu 10:** Viết một hàm nhận vào một chuỗi và trả về chuỗi đảo ngược của nó.

Hướng dẫn: Tạo file “ex02\_10.py” trong “lab\_01”.

```
1 def dao_nguoc_chuoi(chuoi):
2     return chuoi[::-1]
3 # Sử dụng hàm và in kết quả
4 input_string = input("Mời nhập chuỗi cần đảo ngược: ")
5 print("Chuỗi đảo ngược là:", dao_nguoc_chuoi(input_string))
```

Kết quả:

The screenshot shows the VS Code interface. The code editor contains the following Python script:

```
1 def dao_nguoc_chuoi(chuoi):
2     return chuoi[::-1]
3 # Sử dụng hàm và in kết quả
4 input_string = input("Mời nhập chuỗi cần đảo ngược: ")
5 print("Chuỗi đảo ngược là:", dao_nguoc_chuoi(input_string))
```

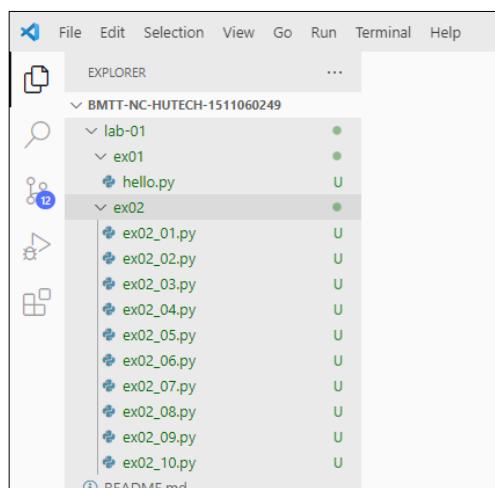
Below the code editor is the terminal window, which shows the command being run and its output:

```
PS D:\lab-01> cd ex02
PS D:\Lab-01\ex02> python ex02_10.py
Mời nhập chuỗi cần đảo ngược: hutech university
Chuỗi đảo ngược là: ytisrevinu hcetuh
PS D:\lab-01\ex02>
```

Cập nhật thư mục theo các bài tập (nhập lệnh vào cửa sổ Terminal của VS Code):

```
mkdir ex01
move .\hello.py .\ex01\
mkdir ex02
Move-Item .\ex02_*.py .\ex02\
```

Kết quả:



Đưa các thay đổi lên Git (tách nhánh lab01 từ nhánh master, đưa các thay đổi hiện tại vào stash):

```
git add .
git stash
git checkout -b lab01
git stash apply
git add .
git commit -m "[add] lab-01 ex01 ex02"
git push origin lab01
```

The screenshot shows a terminal window with the following command history:

```
PS D:\bmtt-nc-hutech-1511060249> git add .
PS D:\bmtt-nc-hutech-1511060249> git stash
Saved working directory and index state WIP on main: b9d827e lab01 hello world
PS D:\bmtt-nc-hutech-1511060249> git checkout -b lab-01
Switched to a new branch 'lab-01'
PS D:\bmtt-nc-hutech-1511060249> git stash apply
On branch lab-01
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:  lab-01/ex01/hello.py
    new file:  lab-01/ex02/ex02_01.py
    new file:  lab-01/ex02/ex02_02.py
    new file:  lab-01/ex02/ex02_03.py
    new file:  lab-01/ex02/ex02_04.py
    new file:  lab-01/ex02/ex02_05.py
    new file:  lab-01/ex02/ex02_06.py
    new file:  lab-01/ex02/ex02_07.py
    new file:  lab-01/ex02/ex02_08.py
    new file:  lab-01/ex02/ex02_09.py
    new file:  lab-01/ex02/ex02_10.py

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    deleted:   lab-01/hello.py

PS D:\bmtt-nc-hutech-1511060249> git commit -m "[add] lab-01 ex01 ex02"
[lab-01 8a81644] [add] lab-01 ex01 ex02
11 files changed, 90 insertions(+)
 rename lab-01/{ => ex01}/hello.py (100%)
  create mode 100644 lab-01/ex02/ex02_01.py
  create mode 100644 lab-01/ex02/ex02_02.py
  create mode 100644 lab-01/ex02/ex02_03.py
  create mode 100644 lab-01/ex02/ex02_04.py
  create mode 100644 lab-01/ex02/ex02_05.py
  create mode 100644 lab-01/ex02/ex02_06.py
  create mode 100644 lab-01/ex02/ex02_07.py
  create mode 100644 lab-01/ex02/ex02_08.py
  create mode 100644 lab-01/ex02/ex02_09.py
  create mode 100644 lab-01/ex02/ex02_10.py
PS D:\bmtt-nc-hutech-1511060249> git push origin lab-01
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 8 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (14/14), 2.96 KiB | 2.96 MiB/s, done.
Total 14 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'lab-01' on GitHub by visiting:
remote:   https://github.com/mrphuoc020597/bmtt-nc-hutech-1511060249/pull/new/lab-01
remote:
To https://github.com/mrphuoc020597/bmtt-nc-hutech-1511060249.git
 * [new branch]  lab-01 -> lab-01
PS D:\bmtt-nc-hutech-1511060249>
```

### 1.6.3 Bài thực hành 03: List, Tuple, Dictionary

- **Câu 01:** Viết một chương trình để tính tổng của tất cả các số chẵn trong một List.

Hướng dẫn: Tại folder “lab-01”, tạo folder “ex03”. Tạo file “ex03\_01.py” trong “ex03”.

```

1 def tinh_tong_so_chan(lst):
2     tong = 0
3     for num in lst:
4         if num % 2 == 0:
5             tong += num
6     return tong
7
8 # Nhập danh sách từ người dùng và xử lý chuỗi
9 input_list = input("Nhập danh sách các số, cách nhau bằng dấu phẩy: ")
10 numbers = list(map(int, input_list.split(',')))
11
12 # Sử dụng hàm và in kết quả
13 tong_chan = tinh_tong_so_chan(numbers)
14 print("Tổng các số chẵn trong List là:", tong_chan)

```

Kết quả:

```

ex03_01.py x
ex03 > ex03_01.py > ...
1 def tinh_tong_so_chan(lst):
2     tong = 0
3     for num in lst:
4         if num % 2 == 0:
5             tong += num
6     return tong
7
8 # Nhập danh sách từ người dùng và xử lý chuỗi
9 input_list = input("Nhập danh sách các số, cách nhau bằng dấu phẩy: ")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\lab-01> cd ex03
PS D:\lab-01\ex03> python ex03\_01.py
Nhập danh sách các số, cách nhau bằng dấu phẩy: 1,-2,3,4,5,-6,7,8,-9
Tổng các số chẵn trong List là: 4
PS D:\lab-01\ex03>

- **Câu 02:** Viết chương trình để đảo ngược vị trí của các phần tử trong một danh sách.

Hướng dẫn: Tạo file “ex03\_02.py” trong “ex03”.

```

1 def dao_nguoc_list(lst):
2     return lst[::-1]
3
4 # Nhập danh sách từ người dùng và xử lý chuỗi
5 input_list = input("Nhập danh sách các số, cách nhau bằng dấu phẩy: ")
6 numbers = list(map(int, input_list.split(',')))
7
8 # Sử dụng hàm và in kết quả
9 list_dao_nguoc = dao_nguoc_list(numbers)
10 print("List sau khi đảo ngược:", list_dao_nguoc)

```

Kết quả:

The screenshot shows a code editor window with a tab labeled "ex03\_02.py". The code defines a function `dao\_nguoc\_list` that returns a reversed list. It then reads a comma-separated string from input, converts it to a list of integers, and prints the reversed list. Below the code editor is a terminal window showing the command `python ex03\_02.py` being run, followed by the input "Nhập danh sách các số, cách nhau bằng dấu phẩy: 1,-2,3,4,5,-6,7,8,-9" and the output "List sau khi đảo ngược: [-9, 8, 7, -6, 5, 4, 3, -2, 1]".

- **Câu 03:** Viết chương trình để tạo một Tuple từ một List nhập vào từ bàn phím.

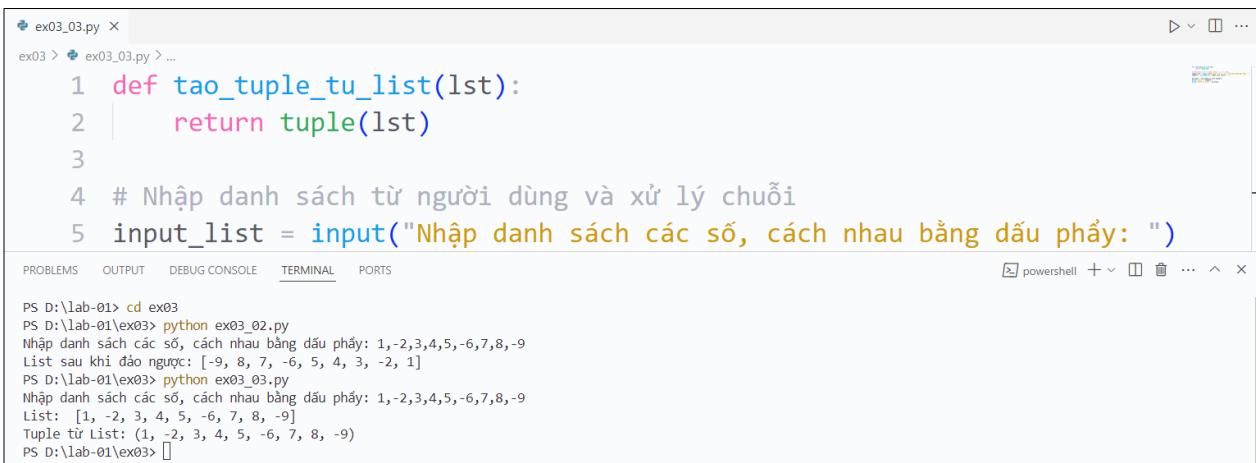
Hướng dẫn: Tạo file "ex03\_03.py" trong "ex03".

```

1 def tao_tuple_tu_list(lst):
2     return tuple(lst)
3
4 # Nhập danh sách từ người dùng và xử lý chuỗi
5 input_list = input("Nhập danh sách các số, cách nhau bằng dấu phẩy: ")
6 numbers = list(map(int, input_list.split(',')))
7
8 my_tuple = tao_tuple_tu_list(numbers)
9 print("List: ", numbers)
10 print("Tuple từ List:", my_tuple)

```

Kết quả:



```
def tao_tuple_tu_list(lst):
    return tuple(lst)

# Nhập danh sách từ người dùng và xử lý chuỗi
input_list = input("Nhập danh sách các số, cách nhau bằng dấu phẩy: ")

PS D:\lab-01> cd ex03
PS D:\lab-01\ex03> python ex03_02.py
Nhập danh sách các số, cách nhau bằng dấu phẩy: 1,-2,3,4,5,-6,7,8,-9
List sau khi đảo ngược: [-9, 8, 7, -6, 5, 4, 3, -2, 1]
PS D:\lab-01\ex03> python ex03_03.py
Nhập danh sách các số, cách nhau bằng dấu phẩy: 1,-2,3,4,5,-6,7,8,-9
List: [1, -2, 3, 4, 5, -6, 7, 8, -9]
Tuple từ List: (1, -2, 3, 4, 5, -6, 7, 8, -9)
PS D:\lab-01\ex03>
```

- **Câu 04:** Viết chương trình để truy cập các phần tử đầu tiên và cuối cùng trong một Tuple.

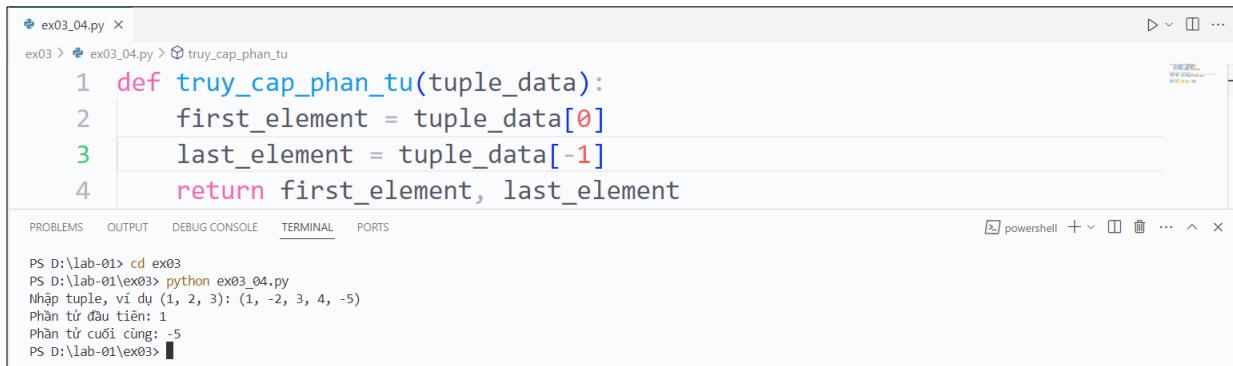
Hướng dẫn: Tạo file “ex03\_04.py” trong “ex03”.

```
def truy_cap_phan_tu(tuple_data):
    first_element = tuple_data[0]
    last_element = tuple_data[-1]
    return first_element, last_element

# Nhập tuple từ người dùng
input_tuple = eval(input("Nhập tuple, ví dụ (1, 2, 3): "))
first, last = truy_cap_phan_tu(input_tuple)

# In kết quả
print("Phần tử đầu tiên:", first)
print("Phần tử cuối cùng:", last)
```

Kết quả:



```
def truy_cap_phan_tu(tuple_data):
    first_element = tuple_data[0]
    last_element = tuple_data[-1]
    return first_element, last_element

tuple, ví dụ (1, 2, 3): (1, -2, 3, 4, -5)
Phần tử đầu tiên: 1
Phần tử cuối cùng: -5
PS D:\lab-01\ex03>
```

- **Câu 05:** Viết chương trình để đếm số lần xuất hiện của mỗi phần tử trong một List và lưu kết quả vào một Dictionary.

Hướng dẫn: Tạo file “ex03\_05.py” trong “ex03”.

```

1 def dem_so_lan_xuat_hien(lst):
2     count_dict = {}
3     for item in lst:
4         if item in count_dict:
5             count_dict[item] += 1
6         else:
7             count_dict[item] = 1
8     return count_dict
9
10 # Nhập danh sách từ người dùng
11 input_string = input("Nhập danh sách các từ, cách nhau bằng dấu cách:")
12 word_list = input_string.split()
13
14 # Sử dụng hàm và in kết quả
15 so_lan_xuat_hien = dem_so_lan_xuat_hien(word_list)
16 print("Số lần xuất hiện của các phần tử:", so_lan_xuat_hien)

```

Kết quả:

The screenshot shows the VS Code interface with the code for `ex03_05.py`. The code defines a function `dem_so_lan_xuat_hien` that counts the occurrences of words in a list and returns a dictionary. It then demonstrates the function with the input "hutech, khoa, cong, nghe, thong, tin, bao, mat, thong, tin". The output shows the resulting dictionary where each word is a key and its count is the value.

```

PS D:\lab-01> cd ex03
PS D:\lab-01\ex03> python ex03_05.py
Nhập danh sách các từ, cách nhau bằng dấu cách: hutech, khoa, cong, nghe, thong, tin, bao, mat, thong, tin,
Số lần xuất hiện của các phần tử: {'hutech': 1, 'khoa': 1, 'cong': 1, 'nghe': 1, 'thong': 2, 'tin': 2, 'bao': 1, 'mat': 1}
PS D:\lab-01\ex03>

```

- **Câu 06:** Viết chương trình để xóa một phần tử từ Dictionary theo key đã cho.

Hướng dẫn: Tạo file “ex03\_06.py” trong “ex03”.

```

1 def xoa_phan_tu(dictionary, key):
2     if key in dictionary:
3         del dictionary[key]
4         return True
5     else:
6         return False
7
8 # Sử dụng hàm và in kết quả
9 my_dict = {'a': 1, 'b': 2, 'c': 3, 'd': 4}
10 key_to_delete = 'b'
11 result = xoa_phan_tu(my_dict, key_to_delete)
12 if result:
13     print("Phần tử đã được xóa từ Dictionary:", my_dict)
14 else:
15     print("Không tìm thấy phần tử cần xóa trong Dictionary.")

```

Kết quả:

The screenshot shows a code editor window with a Python file named 'ex03\_06.py'. The code defines a function 'xoa\_phan\_tu' that removes a key from a dictionary and returns True if successful, or False if the key was not found. The code then creates a dictionary 'my\_dict' and attempts to delete the key 'b'. The terminal below shows the command 'python ex03\_06.py' being run, followed by the output: 'Phần tử đã được xóa từ Dictionary: {"a": 1, "c": 3, "d": 4}'.

```

ex03_06.py x
ex03 > ex03_06.py > ...
1 def xoa_phan_tu(dictionary, key):
2     if key in dictionary:
3         del dictionary[key]
4         return True
5     else:
6         return False
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\lab-01> cd ex03
PS D:\lab-01\ex03> python ex03_06.py
Phần tử đã được xóa từ Dictionary: {"a": 1, "c": 3, "d": 4}
PS D:\lab-01\ex03>

```

Đưa các thay đổi lên Git repo:

```

git add .
git commit -m "[add] lab-01 ex03"
git push origin lab01

```

```

PS D:\bmmt-nc-hutech-1511060249\lab-01\ex03> git push origin lab-01
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 8 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 2.02 KiB | 2.02 MiB/s, done.
Total 10 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/mrphuoc020597/bmmt-nc-hutech-1511060249.git
  8a81644..1b91322 lab-01 -> lab-01
PS D:\bmmt-nc-hutech-1511060249\lab-01\ex03>

```

## 1.6.4 Bài thực hành 04: OOP trong Python

Hãy xây dựng một chương trình Python quản lý thông tin sinh viên. Mỗi sinh viên sẽ có các thuộc tính: mã sinh viên (id), tên, giới tính, chuyên ngành, và điểm trung bình (hệ 10). Trong đó, mã sinh viên sẽ tự động tăng cho mỗi đối tượng mới.

Học lực được tính như sau:

- Loại Giỏi: nếu điểm trung bình từ 8 trở lên.
- Loại Khá: nếu điểm trung bình từ 6,5 đến dưới 8.
- Loại Trung bình: nếu điểm trung bình từ 5 đến dưới 6,5.
- Loại Yếu: nếu điểm trung bình dưới 5.

Yêu cầu: Tạo ra một menu với các chức năng sau:

- Thêm sinh viên.
- Cập nhật thông tin sinh viên theo ID.
- Xóa sinh viên theo ID.
- Tìm kiếm sinh viên qua tên.
- Sắp xếp danh sách sinh viên theo điểm trung bình.
- Sắp xếp danh sách sinh viên theo tên chuyên ngành.
- Hiển thị danh sách sinh viên.

Hướng dẫn:

- Tại folder "lab-01", tạo folder "ex04". Tạo file "SinhVien.py" trong "ex04".

```
1 class SinhVien:  
2     def __init__(self, id, name, sex, major, diemTB):  
3         self._id = id  
4         self._name = name  
5         self._sex = sex  
6         self._major = major  
7         self._diemTB = diemTB  
8         self._hocLuc = ""
```

- Tạo file “QuanLySinhVien.py” trong “ex04”.

```
1 from SinhVien import SinhVien
2
3 class QuanLySinhVien:
4     listSinhVien = []
5
6     def generateID(self):
7         maxId = 1
8         if (self.soLuongSinhVien() > 0):
9             maxId = self.listSinhVien[0]._id
10            for sv in self.listSinhVien:
11                if (maxId < sv._id):
12                    maxId = sv._id
13            maxId = maxId + 1
14        return maxId
15
16    def soLuongSinhVien(self):
17        return self.listSinhVien.__len__()
18
19    def nhapSinhVien(self):
20        svId = self.generateID()
21        name = input("Nhập tên sinh viên: ")
22        sex = input("Nhập giới tính sinh viên: ")
23        major = input("Nhập chuyên ngành của sinh viên: ")
24        diemTB = float(input("Nhập điểm của sinh viên: "))
25        sv = SinhVien(svId, name, sex, major, diemTB)
26        self.xepLoaiHocLuc(sv)
27        self.listSinhVien.append(sv)
28
29    def updateSinhVien(self, ID):
30        sv:SinhVien = self.findByID(ID)
31        if (sv != None):
32            name = input("Nhập tên sinh viên: ")
33            sex = input("Nhập giới tính sinh viên: ")
34            major = int(input("Nhập chuyên ngành của sinh viên: "))
35            diemTB = float(input("Nhập điểm của sinh viên: "))
36            sv._name = name
37            sv._sex = sex
38            sv._major = major
39            sv._diemTB = diemTB
40            self.xepLoaiHocLuc(sv)
```

```
41         else:
42             print("Sinh viên có ID = {} không tồn tại.".format(ID))
43
44     def sortByID(self):
45         self.listSinhVien.sort(key=lambda x: x._id, reverse=False)
46
47     def sortByName(self):
48         self.listSinhVien.sort(key=lambda x: x._name, reverse=False)
49
50     def sortByDiemTB(self):
51         self.listSinhVien.sort(key=lambda x: x._diemTB, reverse=False)
52
53     def findById(self, ID):
54         searchResult = None
55         if (self.soluongSinhVien() > 0):
56             for sv in self.listSinhVien:
57                 if (sv._id == ID):
58                     searchResult = sv
59         return searchResult
60
61     def findByName(self, keyword):
62         listSV = []
63         if (self.soluongSinhVien() > 0):
64             for sv in self.listSinhVien:
65                 if (keyword.upper() in sv._name.upper()):
66                     listSV.append(sv)
67         return listSV
68
69     def deleteById(self, ID):
70         isDeleted = False
71         sv = self.findById(ID)
72         if (sv != None):
73             self.listSinhVien.remove(sv)
74             isDeleted = True
75         return isDeleted
76
77     def xepLoaiHocLuc(self, sv:SinhVien):
78         if (sv._diemTB >= 8):
79             sv._hocLuc = "Gioi"
80         elif (sv._diemTB >= 6.5):
81             sv._hocLuc = "Kha"
```

```

81         sv._hocLuc = "Kha"
82     elif (sv._diemTB >= 5):
83         sv._hocLuc = "Trung binh"
84     else:
85         sv._hocLuc = "Yeu"
86
87     def showSinhVien(self, listSV):
88         print("{:<8} {:<18} {:<8} {:<8}{:<8} {:<8}"
89             .format("ID", "Name", "Sex", "Major", "Diem TB", "Hoc
90             Luc"))
91         if (listSV.__len__() > 0):
92             for sv in listSV:
93                 print("{:<8} {:<18} {:<8} {:<8}{:<8} {:<8} "
94                     .format(sv._id, sv._name, sv._sex, sv._major,
95                     sv._diemTB, sv._hocLuc))
96
97     def getListSinhVien(self):
98         return self.listSinhVien

```

- Tạo file “Main.py” trong “ex04”.

```

1 from QuanLySinhVien import QuanLySinhVien
2
3 qlsv = QuanLySinhVien()
4 while (1 == 1):
5     print("\nCHUONG TRINH QUAN LY SINH VIEN")
6     print("*****MENU*****")
7     print("** 1. Them sinh vien.          **")
8     print("** 2. Cap nhat thong tin sinh vien boi ID.      **")
9     print("** 3. Xoa sinh vien boi ID.          **")
10    print("** 4. Tim kiem sinh vien theo ten.      **")
11    print("** 5. Sap xep sinh vien theo diem trung binh.  **")
12    print("** 6. Sap xep sinh vien theo ten chuyen nganh. **")
13    print("** 7. Hien thi danh sach sinh vien.        **")
14    print("** 0. Thoat          **")
15    print("*****")
16
17    key = int(input("Nhap tuy chon: "))
18    if (key == 1):
19        print("\n1. Them sinh vien.")
20        qlsv.nhapSinhVien()
21        print("\nThem sinh vien thanh cong!")

```

```
22     elif (key == 2):
23         if (qlsv.soLuongSinhVien() > 0):
24             print("\n2. Cap nhat thong tin sinh vien. ")
25             print("\nNhap ID: ")
26             ID = int(input())
27             qlsv.updateSinhVien(ID)
28         else:
29             print("\nSanh sach sinh vien trong!")
30     elif (key == 3):
31         if (qlsv.soLuongSinhVien() > 0):
32             print("\n3. Xoa sinh vien.")
33             print("\nNhap ID: ")
34             ID = int(input())
35             if (qlsv.deleteById(ID)):
36                 print("\nSinh vien co id = ", ID, " da bi xoa.")
37             else:
38                 print("\nSinh vien co id = ", ID, " khong ton tai.")
39         else:
40             print("\nSanh sach sinh vien trong!")
41     elif (key == 4):
42         if (qlsv.soLuongSinhVien() > 0):
43             print("\n4. Tim kiem sinh vien theo ten.")
44             print("\nNhap ten de tim kiem: ")
45             name = input()
46             searchResult = qlsv.findByName(name)
47             qlsv.showSinhVien(searchResult)
48         else:
49             print("\nSanh sach sinh vien trong!")
50     elif (key == 5):
51         if (qlsv.soLuongSinhVien() > 0):
52             print("\n5. Sap xep sinh vien theo diem trung binh (GPA). ")
53             qlsv.sortByDiemTB()
54             qlsv.showSinhVien(qlsv.getListSinhVien())
55         else:
56             print("\nSanh sach sinh vien trong!")
57     elif (key == 6):
```

```

58     if (qlsv.soluongSinhVien() > 0):
59         print("\n6. Sap xep sinh vien theo ten.")
60         qlsv.sortByName()
61         qlsv.showSinhVien(qlsv.getListSinhVien())
62     else:
63         print("\nSanh sach sinh vien trong!")
64     elif (key == 7):
65         if (qlsv.soluongSinhVien() > 0):
66             print("\n7. Hien thi danh sach sinh vien.")
67             qlsv.showSinhVien(qlsv.getListSinhVien())
68         else:
69             print("\nSanh sach sinh vien trong!")
70     elif (key == 0):
71         print("\nBan da chon thoat chuong trinh!")
72         break
73     else:
74         print("\nKhong co chuc nang nay!")
75         print("\nHay chon chuc nang trong hop menu.")

```

Kết quả: Chạy file “Main.py” và kiểm tra chương trình.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

*****MENU*****
PS D:\bmtt-nc-hutech-1511060249\lab-01\ex04> python .\Main.py

CHUONG TRINH QUAN LY SINH VIEN
*****MENU*****
** 1. Them sinh vien.      **
** 2. Cap nhap thong tin sinh vien boi ID.      **
** 3. Xoa sinh vien boi ID.      **
** 4. Tim kiem sinh vien theo ten.      **
** 5. Sap xep sinh vien theo diem trung binh.      **
** 6. Sap xep sinh vien theo ten chuyen nganh.      **
** 7. Hien thi danh sach sinh vien.      **
** 0. Thoat.      **

Nhap tuy chon: 1

1. Them sinh vien.
Nhap ten sinh vien: phuoc nguyen
Nhap gioi tinh sinh vien: nam
Nhap chuyen nganh cua sinh vien: cnnt
Nhap diem cua sinh vien: 8.5

Them sinh vien thanh cong!

CHUONG TRINH QUAN LY SINH VIEN
*****MENU*****
** 1. Them sinh vien.      **
** 2. Cap nhap thong tin sinh vien boi ID.      **
** 3. Xoa sinh vien boi ID.      **
** 4. Tim kiem sinh vien theo ten.      **
** 5. Sap xep sinh vien theo diem trung binh.      **
** 6. Sap xep sinh vien theo ten chuyen nganh.      **
** 7. Hien thi danh sach sinh vien.      **
** 0. Thoat.      **

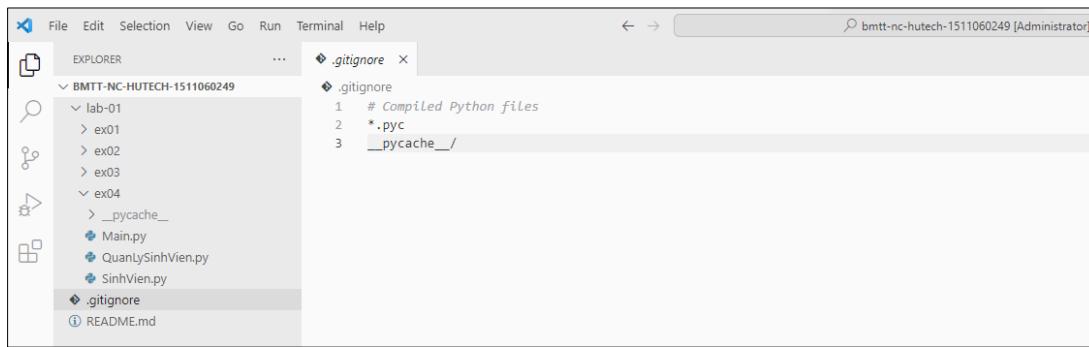
Nhap tuy chon: 7

7. Hien thi danh sach sinh vien.
ID      Name      Sex      Major      Diem TB      Hoc Luc
1      phuoc nguyen      nam      cnnt      8.5      Gioi

```

Thêm file “.gitignore” tại thư mục gốc để không đưa các file cache trong thư mục “\_\_pycache\_\_” lên Git repo.

```
# Compiled Python files
*.pyc
__pycache__/
```



Đưa các thay đổi lên Git repo:

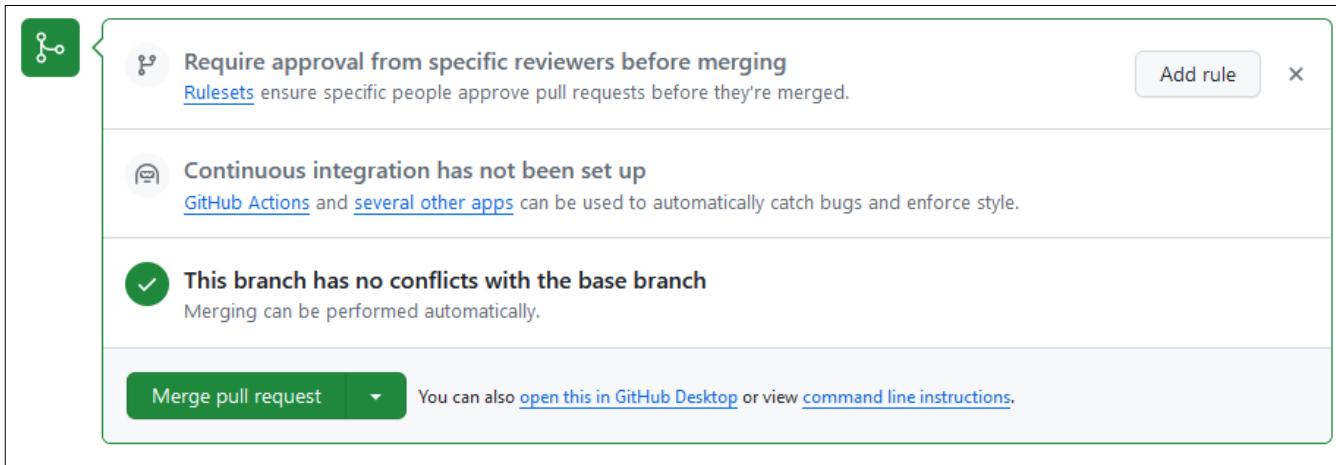
```
git add .
git commit -m "[add] lab-01 ex04 and .gitignore"
git push origin lab01
```

Tạo PR từ branch lab01 về branch main trên Git repo:

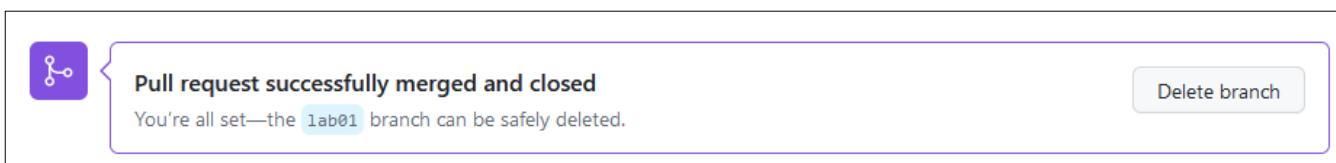
- Vào địa chỉ của Git repo. Chọn menu “Pull requests”. Chọn “New pull request”.
- Chọn compare branch là “lab01”. Chọn base branch là “main”. Kiểm tra các thay đổi và nhấn “Create pull request”.

Commit	Author	Date	Actions
[add] lab-01 ex01 ex02	mrphuoc020597	committed 1 hour ago	
[add] lab-01 ex03	mrphuoc020597	committed 1 hour ago	
[add] lab-01 ex04	mrphuoc020597	committed 32 minutes ago	
[add] .gitignore	mrphuoc020597	committed 30 minutes ago	
[add] lab-01 ex05	mrphuoc020597	committed 30 minutes ago	
[add] lab-01 ex06	mrphuoc020597	committed 28 minutes ago	
[add] lab-01 ex07	mrphuoc020597	committed 27 minutes ago	
[add] lab-01 ex08	mrphuoc020597	committed 26 minutes ago	
[add] lab-01 ex09	mrphuoc020597	committed 25 minutes ago	
[add] lab-01 ex10	mrphuoc020597	committed 24 minutes ago	

- Một lần nữa, kiểm tra lại các thay đổi đã phù hợp hay chưa. Sau đó chọn “Merge pull request” để hoàn tất.



Kết quả:



## 1.7 BÀI TẬP MỞ RỘNG

- **Câu 01:** Hãy phát triển một chương trình để liệt kê tất cả các hoán vị của danh sách [1, 2, 3]. Bạn có thể tham khảo việc sử dụng “**itertools.permutations()**” để thu thập tất cả các hoán vị có thể của danh sách này.
- **Câu 02:** Hãy xây dựng một chương trình để tính tổng của tất cả các số nguyên dương và âm có trong một chuỗi được nhập vào. Ví dụ:

Chuỗi ban đầu là “-100#^sdfkj8902w3ir021@swf-20”.

Kết quả: Giá trị dương: 9046. Giá trị âm: -120.

# 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:  $E_n(x) = (x + n) \text{ mod } 26$ . Giải mã được mô tả tương tự:  $D_n(x) = (x - n) \text{ mod } 26$ " [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":

K	E	Y	W	O
R	D	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 hóa 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 hóa, 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".

- 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 ký tự của bản rõ theo chu kỳ cố định d: Nếu hàm f là hoán vị của một khối gồm d ký 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 .\bmtt-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\bmtt-nc-hutech-1511060249> git pull origin main
From https://github.com/mrphuoc020597/bmtt-nc-hutech-1511060249
 * branch      main      -> FETCH_HEAD
Already up to date.
PS C:\Users\PHUOCNTMH\bmtt-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\bmtt-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
  Downloading flask-3.0.0-py3-none-any.whl.metadata (3.6 kB)
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/c3/fc/254c3e9b5feb89ff5b9876a23218dfa099c96ac5941e900b71206e6313b/werkzeug-3.0.1-py3-none-any.whl.metadata
  Downloading werkzeug-3.0.0-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-py3-none-any.whl (133 kB)
Collecting itsdangerous>=2.1.1 (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/d53fa4befbf2cfa713304affc7ca780ce4fc1fd8710527771b58311a3229(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/7f3714cbc6356a0fefc525ce7a0613d58102ed6eb53eb7b9754f33db807/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/dbaf65876e258facd65f586dde158387ab89963e7f233551afc9c2e4c2/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 flask-3.0.0-py3-none-any.whl (99 kB)
  99.7/99.7 kB 318.4 kB/s eta 0:00:00
Downloaded blinker-1.7.0-py3-none-any.whl (13 kB)
Downloaded click-8.1.7-py3-none-any.whl (97 kB)
  97.9/97.9 kB 38.7 kB/s eta 0:00:00
Downloaded werkzeug-3.0.1-py3-none-any.whl (226 kB)
  226.7/226.7 kB 657.8 kB/s eta 0:00:00
Downloaded 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
```

```

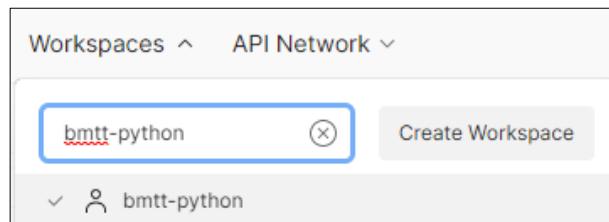
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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
Add new variable			

- 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 application interface. At the top, it says "POST encrypt". Below that, the URL is "HTTP CAESAR / encrypt". The method is set to "POST". In the "Body" tab, the content type is "application/json". The JSON payload is:

```

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 title "Code snippet". It contains a "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:

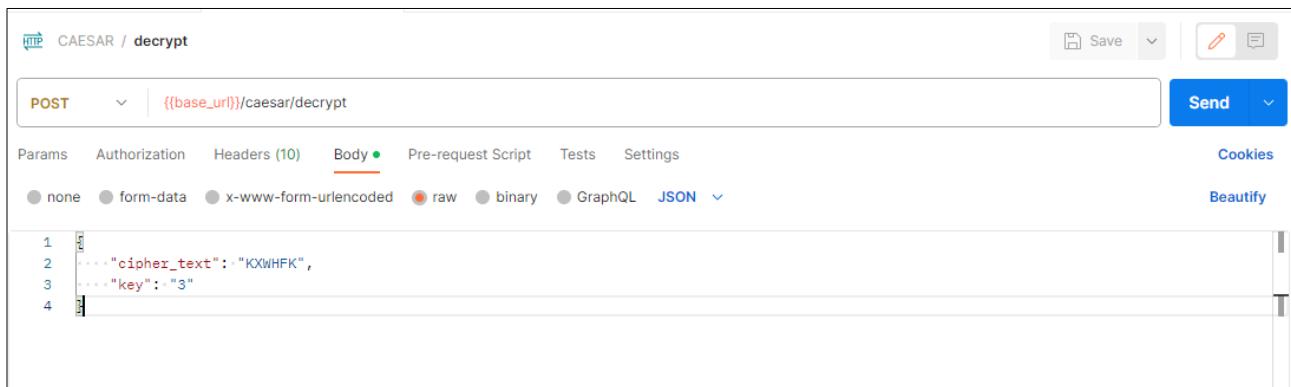
The screenshot shows the Postman interface after sending the request. The "Body" tab is selected, showing the raw JSON response:

```

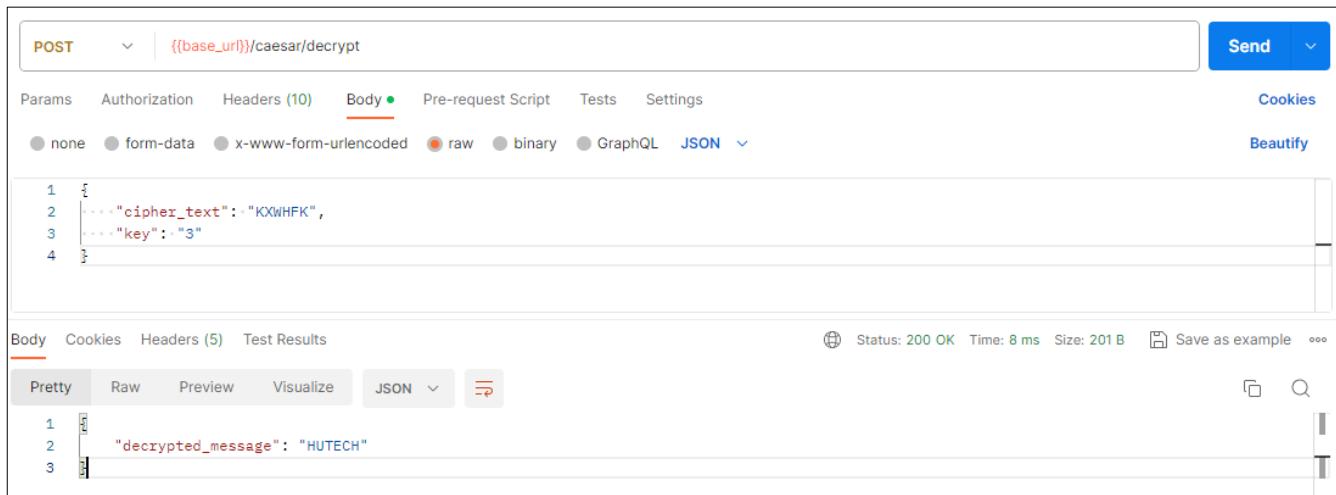
1 {
2   "encrypted_message": "KXWHFK"
3 }

```

- 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\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-02> cd .\ex01
PS C:\Users\PHUOCNTMH\bmtt-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
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         return encrypted_text
20
21     def vigenere_decrypt(self, encrypted_text, key):
22         decrypted_text = ""
23         key_index = 0
24         for char in encrypted_text:
25             if char.isalpha():

```

```

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

```

- 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
```

The terminal window shows the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PHUOCNTMH\bmtt-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://172.16.9:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 929-074-328

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249> git add .
[lab-02 7f0fec] [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
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249> []
```

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

Code snippet for the 'encrypt' API endpoint:

```
cURL
1 curl --location 'http://127.0.0.1:5000/api/vigenere/encrypt' \
2 --header 'Content-Type: application/json' \
3 --data '{
4     "plain_text": "HUTECH",
5     "key": "ABC"
6 }'
```

Code snippet for the 'decrypt' API endpoint:

```
cURL
1 curl --location 'http://127.0.0.1:5000/api/vigenere/decrypt' \
2 --header 'Content-Type: application/json' \
3 --data '{
4     "cipher_text": "HVVEDJ",
5     "key": "ABC"
6 }'
```

- 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:

The Postman interface shows the following response for the 'encrypt' API:

Body: `{ "plain_text": "HUTECH", "key": "ABC" }`

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

Pretty JSON:

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

Raw JSON:

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

- Kết quả giải mã:

```

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

```

Body Cookies Headers (5) Test Results

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

Pretty Raw Preview Visualize JSON

```

1 "decrypted_text": "HUTECH"
2
3

```

- 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”:



The screenshot shows a code snippet editor window with the title "Code snippet". It contains a "curl" command to send a POST request to the "/api/railfence/encrypt" endpoint of a local development server. The command includes headers for Content-Type and application/json, and a JSON payload with "plain\_text" set to "HUTECHUNIVERSITY" and "key" set to "3".

```

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

```



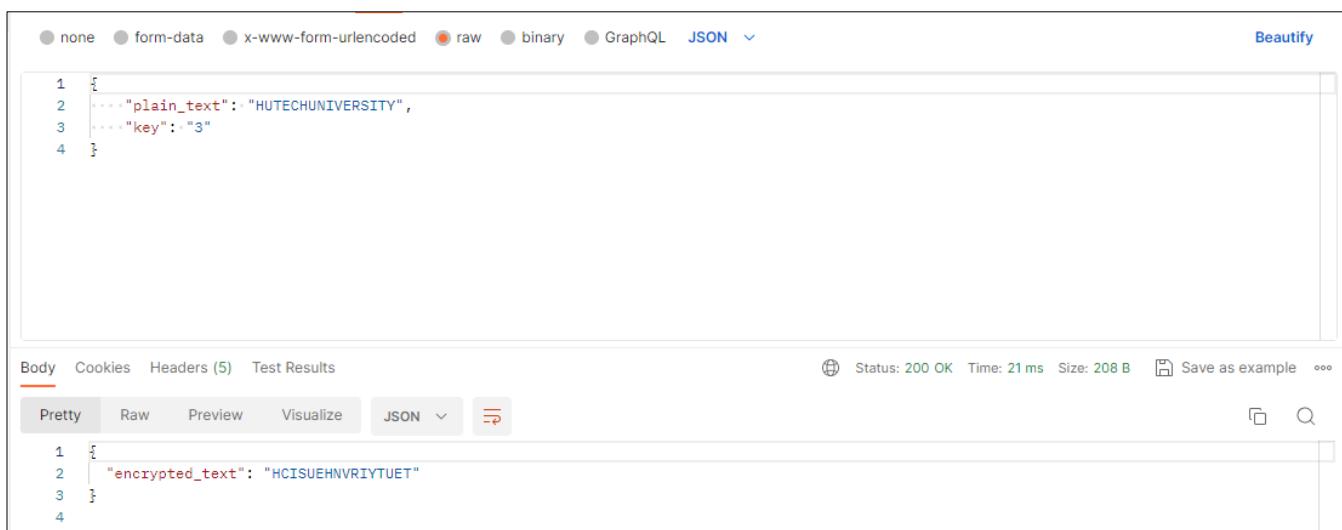
```

Code snippet

curl --location 'http://127.0.0.1:5000/api/railfence/decrypt' \
--header 'Content-Type: application/json' \
--data '{
  "cipher_text": "HCISUEHNVRIYTUET",
  "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 hoá:



Body

```

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

```

Body

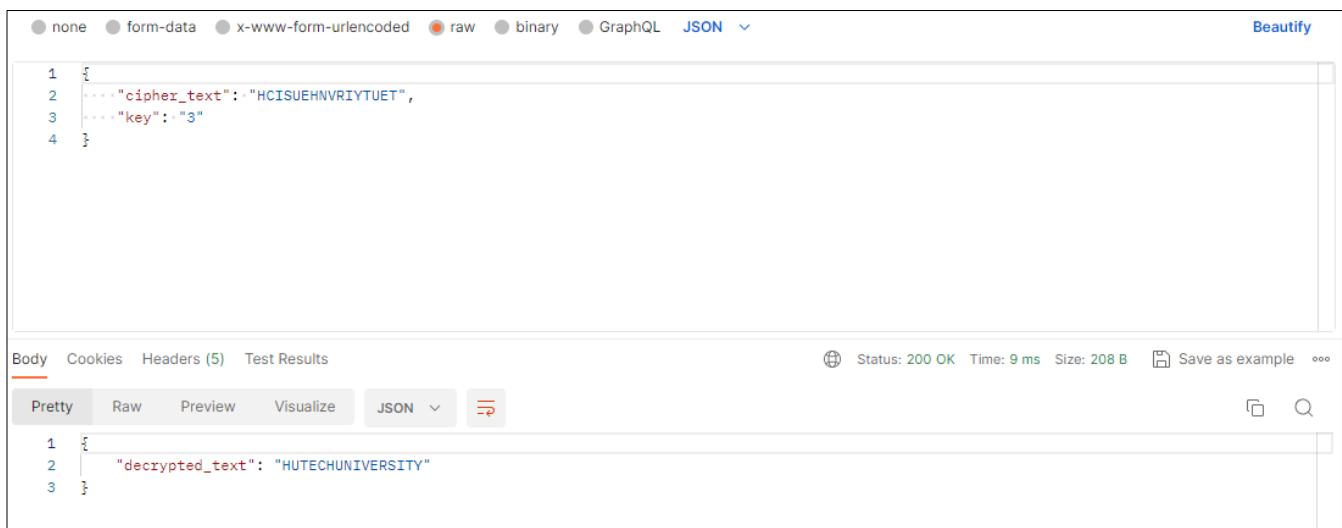
```

1 {
2   "encrypted_text": "HCISUEHNVRIYTUET"
3 }
4

```

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

- Kết quả giải mã:



Body

```

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

```

Body

```

1 {
2   "decrypted_text": "HUTECHUNIVERSITY"
3 }
4

```

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

- 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     def playfair_decrypt(self, cipher_text, matrix):
56         cipher_text = cipher_text.upper()
57         decrypted_text = ""
58         decrypted_text1 = ""
59
60         for i in range(0, len(cipher_text), 2):
61             pair = cipher_text[i:i+2]
62             row1, col1 = self.find_letter_coords(matrix, pair[0])
63             row2, col2 = self.find_letter_coords(matrix, pair[1])
64
65             if row1 == row2:
66                 decrypted_text += matrix[row1][(col1 - 1) %
67                                         5] + matrix[row2][(col2 - 1) %
68                                         5]
69             elif col1 == col2:
70                 decrypted_text1 += matrix[(row1 - 1) %
71                                         5][col1] + matrix[(row2 - 1) %
72                                         5][col2]
73
74         decrypted_text += decrypted_text1
75
76     return decrypted_text
77
```

```

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})

```

- 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 --location 'http://127.0.0.1:5000/api/playfair/creatematrix' \
--header 'Content-Type: application/json' \
--data '{
  "key": "MONARCHY"
}'

```

```

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

```

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:

```

1 {
2   "key": "MONARCHY"
3 }

```

```

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:

```

1 {
2   "plain_text": "HOMNAYHANHDONG",
3   "key": "MONARCHY"
4 }

```

```

1 {
2   "encrypted_text": "FHOANBBOOYHRYQ"
3 }

```

- Kết quả giải mã:

```

1 {
2   "cipher_text": "FHOANBB00YHRYQ",
3   "key": "MONARCHY"
4 }

Body Cookies Headers (5) Test Results
Pretty Raw Visualize JSON Status: 200 OK Time: 8 ms Size: 206 B Save as example ...
1
2   "decrypted_text": "HOMNAYHANHDONG"
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. Both windows have a title bar 'Code snippet' and a 'curl' dropdown menu.

**Top Window (Encrypt Request):**

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

**Bottom Window (Decrypt Request):**

```
curl --location 'http://127.0.0.1:5000/api/transposition/decrypt' \
--header 'Content-Type: application/json' \
--data '{
  "ciphertext": "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 hoá:

The screenshot shows the Postman interface with the 'Body' tab selected. The 'Pretty' view displays the JSON input:

```
{
  "plain_text": "HUTECH",
  "key": "3"
}
```

The 'Raw' view shows the raw JSON output:

```
{"encrypted_text": "HEUCTH"}
```

At the top, there are tabs for 'none', 'form-data', 'x-www-form-urlencoded', 'raw', 'binary', 'GraphQL', and 'JSON'. On the right, there are status indicators: 'Status: 200 OK', 'Time: 6 ms', 'Size: 198 B', and 'Save as example'.

- Kết quả giải mã:

The screenshot shows a POST request in Postman. The request body contains the following JSON:

```

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

```

The response status is 200 OK, and the decrypted text is "HUTECH".

- 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
21                            <input type="text" class="form-control"
22                                name="inputPlainText" placeholder="Input Plain
23                                Text" required autofocus/>
24
25                            </div>
26                            <div class="mb-3">
27                                <label class="form-label">Key:</label>
28                                <input type="number" class="form-control"
29                                    name="inputKeyPlain" placeholder="Input Key"
30                                    required/>
31
32                            </div>
33                            <button type="submit" class="btn
34                                btn-primary">Encrypt</button>
35
36                    </form>
37                </td>
38
39            </tr>
40
41            <tr><td style="font-weight: bold; color: blue">DECRYPTION</
42                td></tr>
43            <tr>
44                <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.

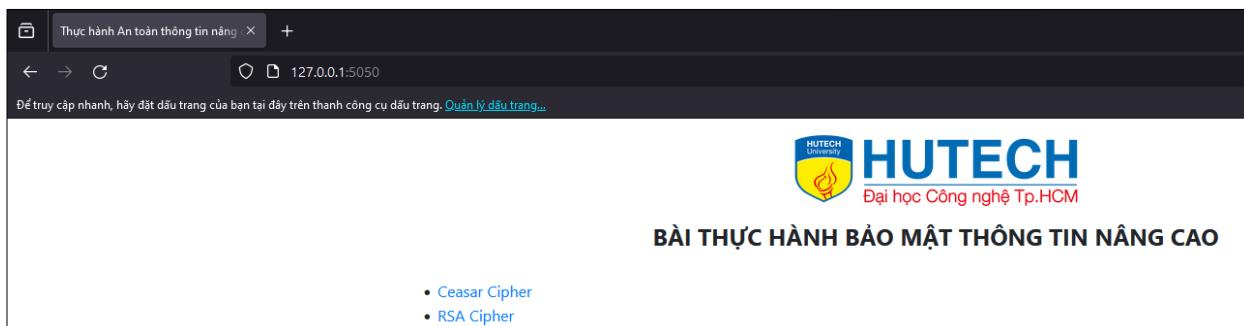
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

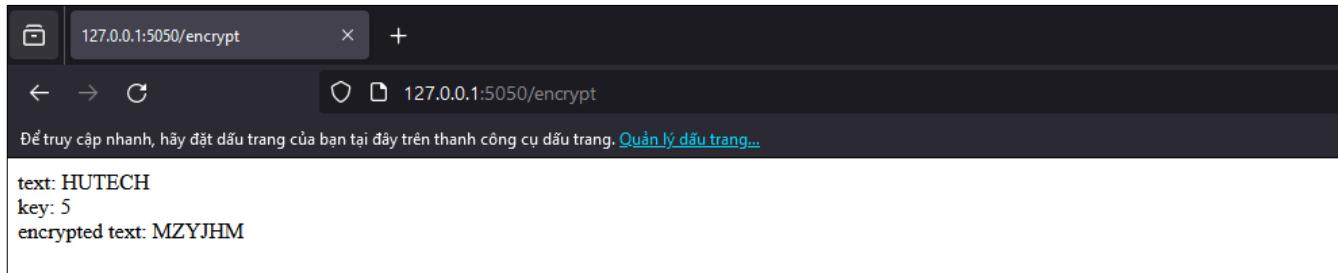
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-02> python .\app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5050
 * 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:



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



127.0.0.1:5050/encrypt

Để truy cập nhanh, hãy đặt dấu trang của bạn tại đây trên thanh công cụ dấu trang. [Quản lý dấu trang...](#)

text: HUTECH  
key: 5  
encrypted text: MZYJHM

- Kiểm tra giải mã:



## CAESAR CIPHER

**ENCRYPTION**

Plain text:

Key:

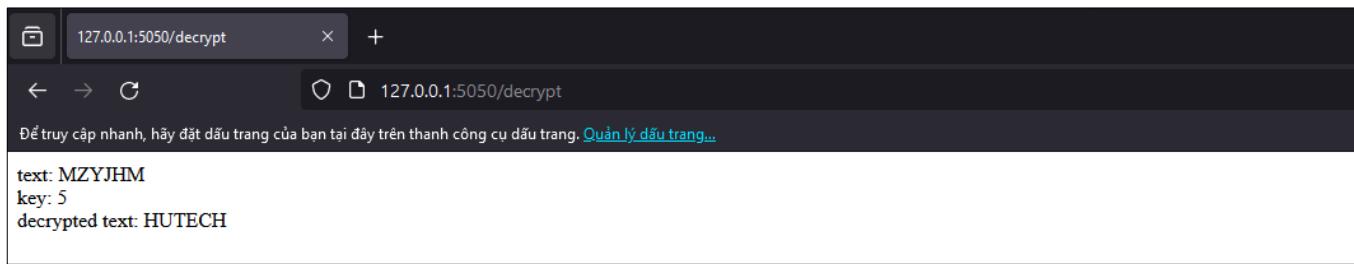
**Encrypt**

**DECRYPTION**

Cipher text:

Key:

**Decrypt**



127.0.0.1:5050/decrypt

Để truy cập nhanh, hãy đặt dấu trang của bạn tại đây trên thanh công cụ dấu trang. [Quản lý dấu trang...](#)

text: MZYJHM  
key: 5  
decrypted text: HUTECH

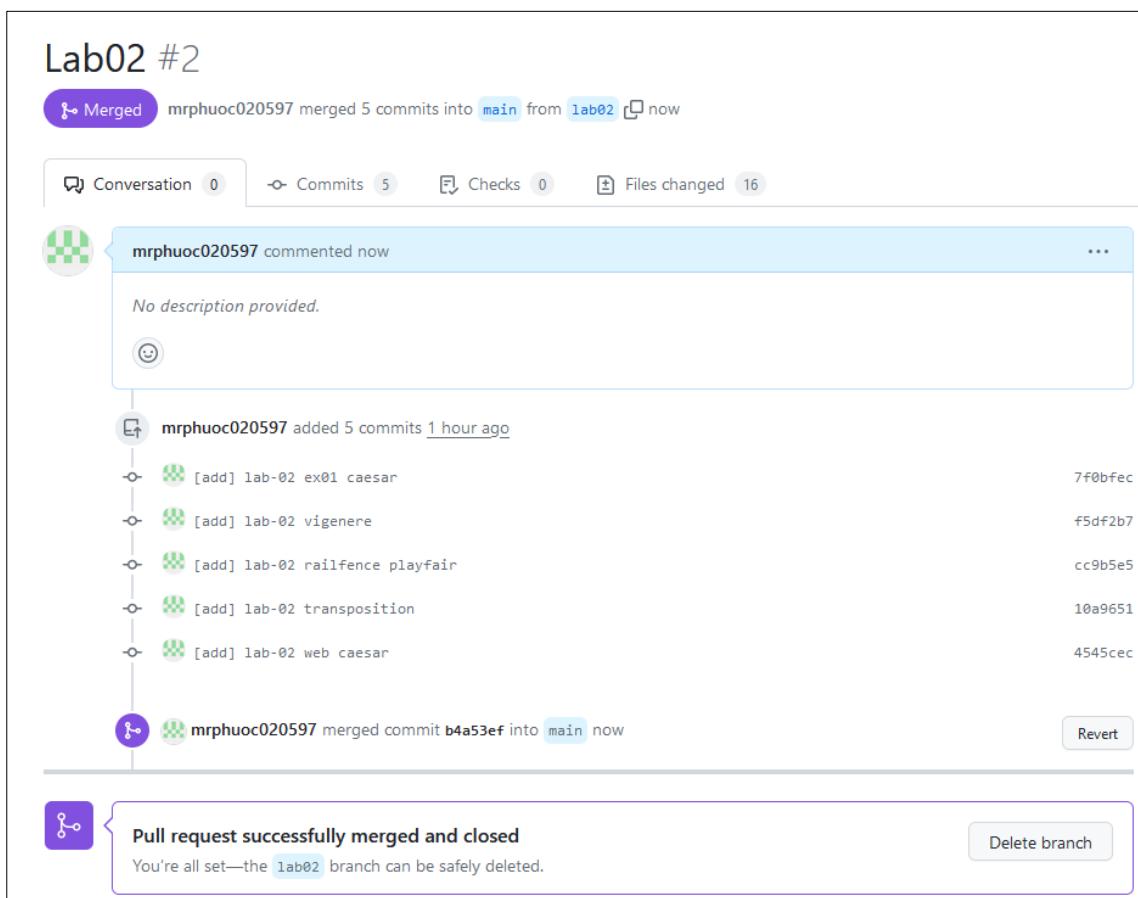
- Commit các thay đổi lên Git:

```
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 hoá 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.

# BÀI 3: LẬP TRÌNH GIAO DIỆN ỨNG DỤNG BẢO MẬT

Nếu chúng ta đã quen thuộc với các công nghệ tạo ứng dụng Desktop truyền thống như Winform, WPF (C#) hoặc Swing, JavaFx (Java), thì trong bài thực hành này, chúng ta sẽ khám phá cách sử dụng các công cụ hỗ trợ phát triển giao diện desktop cho ứng dụng bảo mật bằng Python, bao gồm PyQt5 và QtDesigner. Đồng thời, chúng ta cũng sẽ tìm hiểu về hệ mã hóa RSA, đường cong Elliptic và cách ứng dụng chúng trong việc phát triển các phần mềm bảo mật.

## 3.1 PYQT5

---

PyQt là một thư viện giúp chúng ta truy cập Qt GUI, một framework nổi tiếng được phát triển bằng C++. Trong số nhiều phiên bản của PyQt, phiên bản mới nhất và phổ biến nhất hiện nay là PyQt5. Thư viện này hỗ trợ người dùng xây dựng các ứng dụng desktop có giao diện đồ họa mạnh mẽ bằng Python, kết hợp khả năng lập trình linh hoạt của Python với các công cụ và tính năng phong phú từ Qt5. Các đặc điểm chính của PyQt5 bao gồm:

- Widget và Layout: PyQt5 cung cấp rất nhiều loại widget (đối tượng giao diện) để xây dựng các thành phần giao diện người dùng như nút, ô nhập liệu, menu,....
- Signals và Slots: PyQt5 sử dụng hệ thống "signals and slots" để xử lý sự kiện và tương tác giữa các thành phần.
- Tích hợp với Python: PyQt5 được viết bằng ngôn ngữ C++ và Python, nhưng API được thiết kế sao cho người dùng có thể dễ dàng sử dụng từ Python.
- Đa nền tảng: Ứng dụng được tạo bằng PyQt5 có thể chạy trên nhiều nền tảng khác nhau.

- Phong phú với Qt Designer: PyQt5 đi kèm với Qt Designer, một công cụ đồ họa cho phép thiết kế giao diện người dùng bằng cách kéo và thả các widget.

## 3.2 QTDESIGNER

---

Qt Designer là một công cụ được cung cấp bởi Qt Toolkit, được sử dụng để thiết kế giao diện người dùng đồ họa (GUI) cho ứng dụng sử dụng Qt. Một số điểm quan trọng về Qt Designer bao gồm:

- Giao diện đồ họa dễ sử dụng: Qt Designer cho phép người dùng thiết kế giao diện người dùng một cách trực quan thông qua giao diện đồ họa.
- Tích hợp tốt với PyQt và Qt: Qt Designer có thể được sử dụng để thiết kế giao diện cho các ứng dụng sử dụng PyQt framework hoặc bất kỳ ứng dụng Qt.
- Công cụ kéo và thả (Drag-and-Drop): Điểm mạnh của Qt Designer là khả năng sử dụng công cụ kéo và thả.
- Tạo mã nguồn tự động: Qt Designer có khả năng tạo mã nguồn tự động dựa trên các thiết kế giao diện người dùng.

## 3.3 HỆ MÃ HÓA RSA

---

### 3.3.1 Giới thiệu RSA

"RSA là một phương pháp mã hóa và giải mã dựa trên cặp khóa công khai (cryptography public-key), được sáng tạo bởi ba nhà khoa học Ronald Rivest, Adi Shamir và Leonard Adleman vào những năm 1970. Tên gọi của thuật toán này được đặt theo tên của ba nhà phát minh. Mã hóa RSA có khả năng được sử dụng trong nhiều hệ thống khác nhau nhằm bảo vệ thông tin khi truyền tải qua mạng, cũng như trong việc ký và xác thực chữ ký số" [8]. Nguyên lý hoạt động của RSA dựa trên việc áp dụng cặp khóa, bao gồm khóa công khai (public key) và khóa riêng tư (private key):

- **Khóa công khai (Public key):** Đây là khóa được chia sẻ với mọi người và dùng để mã hóa dữ liệu.
- **Khóa riêng tư (Private key):** Đây là khóa bí mật chỉ được chủ sở hữu biết và dùng để giải mã dữ liệu.

Thực hiện mã hoá và giải mã trong RSA được diễn ra như sau:

- Tạo khóa:
  - Bước 1: Chọn hai số nguyên tố lớn: Chọn hai số nguyên tố lớn và ngẫu nhiên p và q.
  - Bước 2: Tính n và  $\phi(n)$ : Tính  $n = p * q$ , và tính  $\phi(n) = (p - 1) * (q - 1)$ , trong đó  $\phi(n)$  là hàm số Euler của n.
  - Bước 3: Chọn số nguyên e: Chọn một số nguyên e sao cho  $1 < e < \phi(n)$  và e là số nguyên tố cùng nhau với  $\phi(n)$  (không có ước chung nào ngoài 1).
  - Bước 4: Tìm khóa công khai và khóa riêng tư: Khóa công khai là cặp  $(e, n)$ , và khóa riêng tư là cặp  $(d, n)$  sao cho  $d * e \equiv 1 \pmod{\phi(n)}$ , nghĩa là d là phần dư của  $e^{-1}$  khi chia cho  $\phi(n)$ .
- Mã hoá và giải mã:
  - Mã hoá (Encryption): Người gửi sử dụng khóa công khai  $(e, n)$  để mã hoá thông điệp M thành ciphertext C, theo công thức:  $C = M^e \pmod{n}$ .
  - Giải mã (Decryption): Người nhận sử dụng khóa riêng tư  $(d, n)$  để giải mã ciphertext C thành thông điệp M, theo công thức:  $M = C^d \pmod{n}$ .

### 3.3.2 Thư viện RSA trong Python

Trong Python, có nhiều thư viện hỗ trợ việc thực hiện mã hóa RSA và các thao tác liên quan đến nó. Một số thư viện phổ biến trong Python để thực hiện RSA bao gồm:

- “**cryptography**”: Đây là thư viện Python mạnh mẽ hỗ trợ nhiều thuật toán mã hóa và bảo mật, trong đó có cả RSA.
- “**PyCryptodome**”: Đây là thư viện mã nguồn mở của Python cung cấp nhiều thuật toán mã hóa và bảo mật, bao gồm RSA.

## 3.4 ĐƯỜNG CONG ELIPTIC (ECC)

### 3.4.1 Giới thiệu ECC

Mã hóa đường cong elliptic (Elliptic Curve Cryptography - ECC) là một phương pháp mã hóa ứng dụng các đường cong elliptic trong toán học để xây dựng hệ thống

mã hóa công khai. ECC đóng vai trò quan trọng trong lĩnh vực bảo mật, đặc biệt thích hợp cho các ứng dụng yêu cầu khóa nhỏ và hiệu suất cao.

Ưu điểm của ECC:

- Kích thước khóa nhỏ: ECC có thể đạt được mức độ bảo mật tương đương với RSA với các khóa nhỏ hơn nhiều lần.
- Hiệu suất cao: So với RSA, ECC yêu cầu ít tài nguyên tính toán hơn, giúp tăng hiệu suất trong việc mã hóa và giải mã.

### 3.4.2 Mật mã ECC

Mật mã ECC là một kỹ thuật mã hóa khóa công khai dựa trên lý thuyết đường cong elliptic có thể được sử dụng để tạo các khóa mật mã nhanh hơn, nhỏ hơn và hiệu quả hơn. ECC tạo khóa thông qua các thuộc tính của phương trình đường cong elliptic thay vì phương pháp tạo truyền thống như là tích của các số nguyên tố rất lớn. Quá trình hoạt động của ECC bao gồm việc tạo ra cặp khóa công khai và khóa riêng tư, mã hóa thông điệp và giải mã như sau:

- Tạo khóa:
  - Bước 1: Chọn đường cong elliptic: Chọn một đường cong elliptic, thường được biểu diễn trong hệ tọa độ affine hoặc các hệ tọa độ khác.
  - Bước 2: Tạo cặp khóa:
    - Khóa riêng tư (Private Key): Chọn một số nguyên ngẫu nhiên làm khóa riêng tư.
    - Khóa công khai (Public Key): Tính điểm trên đường cong elliptic bằng cách nhân khóa riêng tư với điểm gốc trên đường cong, tạo ra khóa công khai.
- Mã hóa:
  - Bước 1: Lựa chọn thông điệp: Người gửi chọn thông điệp cần mã hóa.
  - Bước 2: Lựa chọn điểm ngẫu nhiên: Người gửi chọn một điểm ngẫu nhiên trên đường cong elliptic.
  - Bước 3: Tính toán:

- Người gửi thực hiện phép nhân điểm ngẫu nhiên với khóa công khai của người nhận.
  - Tính toán phép cộng điểm giữa điểm ngẫu nhiên và điểm đã được nhân với khóa công khai.
  - Mã hóa thông điệp bằng cách kết hợp thông điệp với các toạ độ  $x$  của điểm ngẫu nhiên và điểm cộng đã tính được.
- Giải mã:
- Bước 1: Nhận thông điệp: Người nhận nhận được thông điệp đã được mã hóa.
  - Bước 2: Tính toán:
    - Người nhận sử dụng khóa riêng tư của mình để giải mã các thành phần trong thông điệp mã hóa, bằng cách sử dụng các toạ độ  $x$  được trích xuất từ thông điệp.
    - Tính toán phép cộng điểm giữa điểm đã giải mã và điểm ngẫu nhiên ban đầu đã được gửi đi.
    - Giải mã thông điệp từ thông điệp được trích xuất từ phép cộng điểm này.

### 3.4.3 So sánh ECC và RSA

So sánh độ an toàn giữa ECC và RSA thường liên quan đến độ dài của các khóa cần thiết để đảm bảo mức độ an toàn tương đương.

- Độ dài khóa:
  - ECC: Để đạt được mức độ an toàn tương đương, ECC yêu cầu khóa ngắn hơn so với RSA.
  - RSA: Đối với RSA, độ dài khóa cần phải lớn hơn để đạt được cùng mức độ an toàn so với ECC.
- Hiệu suất:
  - ECC: ECC yêu cầu ít tài nguyên tính toán hơn so với RSA khi sử dụng cùng mức độ an toàn.

- RSA: RSA cần nhiều tài nguyên tính toán hơn đối với cùng mức độ an toàn so với ECC, do đó có thể có hiệu suất kém hơn trong một số trường hợp.
- Bảo mật:
  - ECC: ECC dựa trên độ khó của việc giải quyết bài toán điểm trên đường cong elliptic, một bài toán rất khó khiến cho việc tìm ra khóa riêng tư từ khóa công khai trở nên khó khăn.
  - RSA: RSA dựa trên độ khó của việc phân tích thành phần nguyên tố của một số nguyên lớn hợp thành từ hai số nguyên tố ngẫu nhiên. Đối với RSA, việc tìm ra khóa riêng tư từ khóa công khai cũng là một bài toán khó.

## **3.5 BÀI TẬP THỰC HÀNH**

### **3.5.1 Bài thực hành 01: Tạo ứng dụng mã hóa Caesar**

Tạo ứng dụng desktop demo cho Mật mã Caesar.

Hướng dẫn:

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

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

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

```
git pull origin main
```

```
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 lab03 từ nhánh main.

```
git checkout -b lab03
```

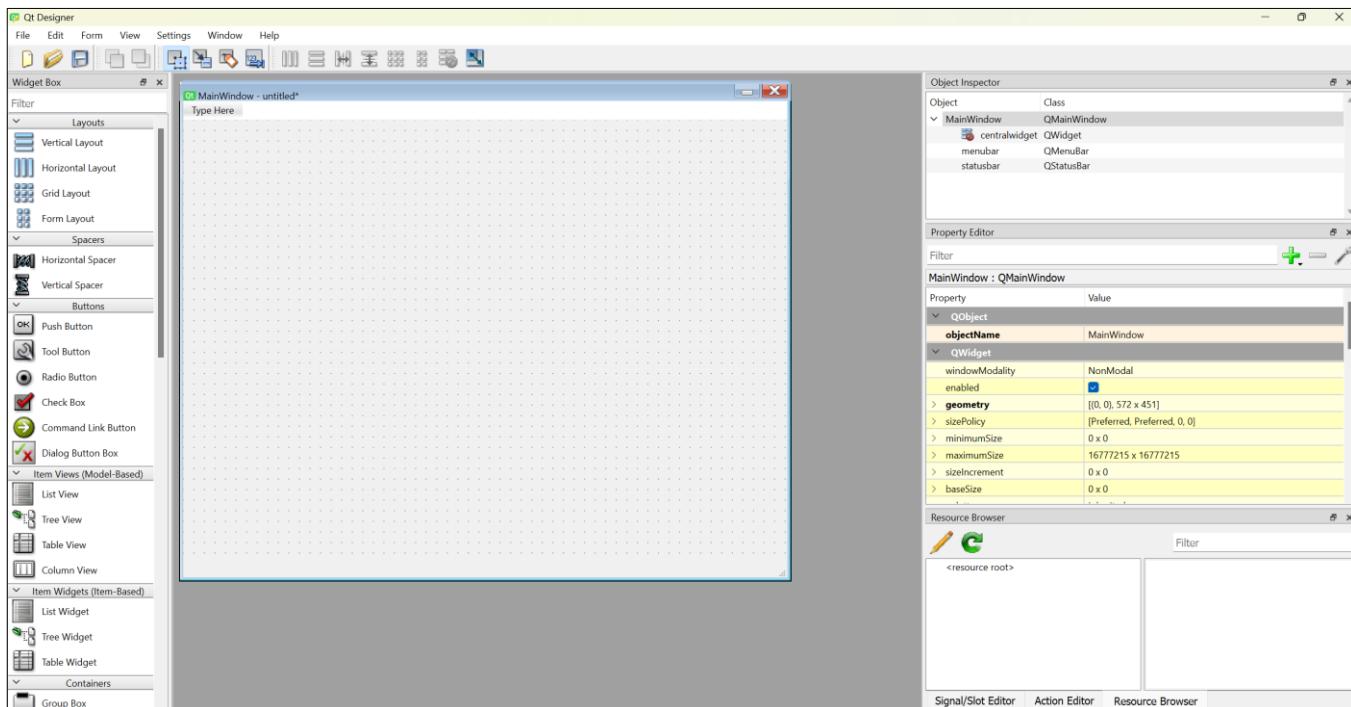
- Tạo folder “lab-03”. Trong folder “lab-03” tạo file “requirements.txt”.

```
requirements.txt
lab-03 > requirements.txt
1 PyQt5
2 requests
```

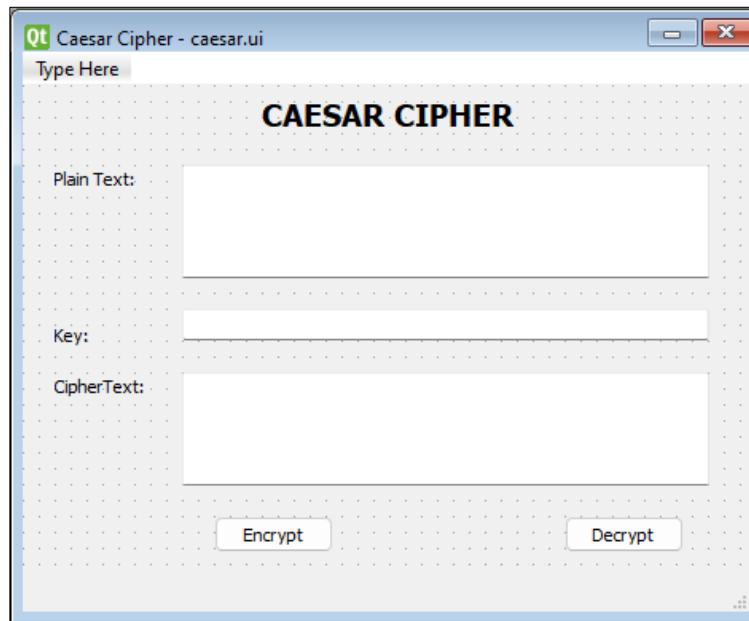
- Vào Terminal, chạy các lệnh sau:

```
cd lab-03
pip install -r .\requirements.txt
```

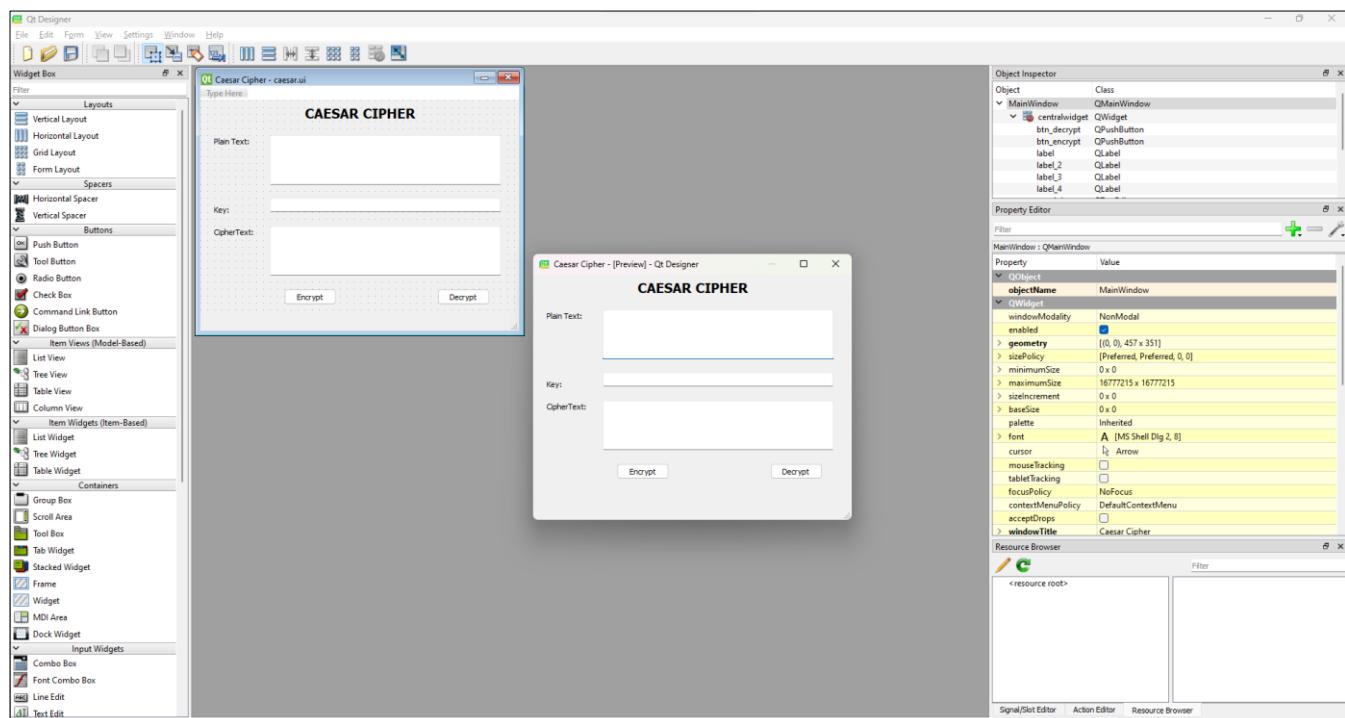
- Trong folder “lab-03” tạo folder “ui”, “platforms”.
- Tải về và cài đặt công cụ QtDesigner tại địa chỉ: <https://build-system.fman.io/qt-designer-download>. Sau khi cài đặt, mở phần mềm QtDesigner.
- Ở phần chọn template, chúng ta chọn “Main Window” ➔ Nhấn “Create”.
- Giao diện của QtDesigner xuất hiện. Bên trái là các Widget, ở giữa là giao diện xem trước và bên phải là tùy chỉnh các thuộc tính Property.



- Tiến hành sử dụng các Widget và thiết kế giao diện cho ứng dụng như sau:



- Sau khi thiết kế xong, nhấn tổ hợp phím Ctrl + R để xem lại giao diện khi thực thi.

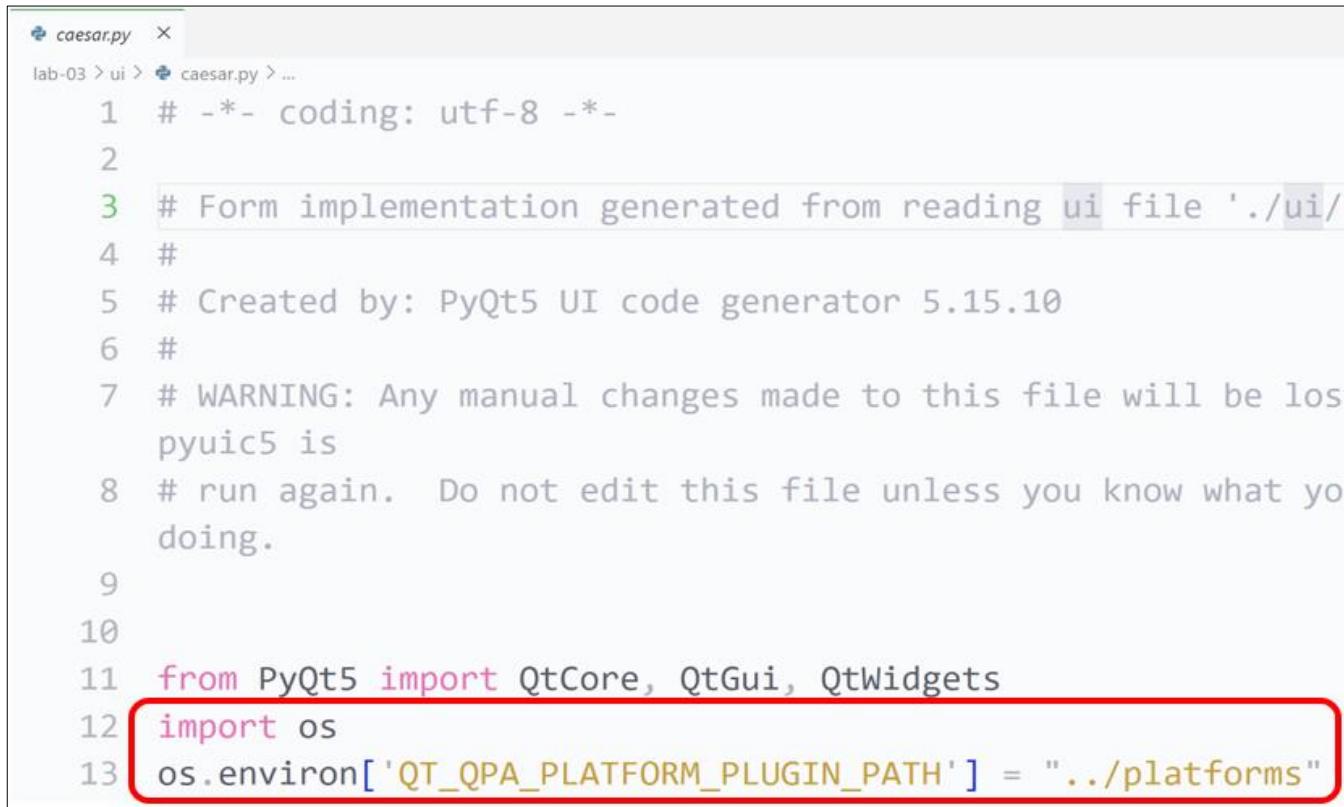


- Tiến hành lưu file thiết kế .ui vào folder "ui" trong "lab-03" với tên "caesar.ui".
- Chuyển giao diện sang code PyQt5 bằng cách chạy lệnh sau trong Terminal của VS Code (thực thi tại thư mục "lab-03"):

```
pyuic5 -x ./ui/caesar.ui -o ./ui/caesar.py
```

- Copy các file từ thư mục sau (<đường\_dẫn\_thư\_mục\_cài\_Python>\Lib\site-packages\PyQt5\Qt5\plugins\platforms) sang thư mục “platforms” của project.
- Thêm hai dòng sau vào đầu file “caesar.py”:

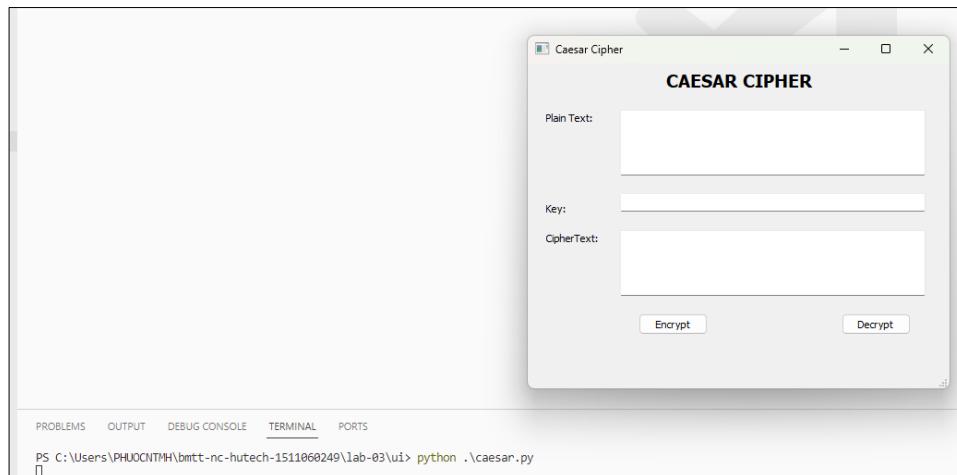
```
import os
os.environ['QT_QPA_PLATFORM_PLUGIN_PATH'] = "../platforms"
```



```
caesar.py ×
lab-03 > ui > caesar.py > ...
1 # -*- coding: utf-8 -*-
2
3 # Form implementation generated from reading ui file './ui/
4 #
5 # Created by: PyQt5 UI code generator 5.15.10
6 #
7 # WARNING: Any manual changes made to this file will be los-
8 # pyuic5 is
9 # run again. Do not edit this file unless you know what yo
10 # doing.
11 from PyQt5 import QtCore, QtGui, QtWidgets
12 import os
13 os.environ['QT_QPA_PLATFORM_PLUGIN_PATH'] = "../platforms"
```

- Kiểm tra thực thi của giao diện:

```
python .\caesar.py
```



- Vừa rồi, chúng ta đã làm xong front-end, tiếp theo, chúng ta sẽ làm phần back-end.
- Trong folder “lab-03” tạo file “caesar\_cipher.py” (Lưu ý: Kiểm tra tên của các button cho đúng).

```

caesar_cipher.py ×

lab-03 > caesar_cipher.py > MyApp > call_api_decrypt
1 import sys
2 from PyQt5.QtWidgets import QApplication, QMainWindow, QMessageBox
3 from ui.caesar import Ui_MainWindow
4 import requests
5
6 class MyApp(QMainWindow):
7     def __init__(self):
8         super().__init__()
9         self.ui = Ui_MainWindow()
10        self.ui.setupUi(self)
11        self.ui.btn_encrypt.clicked.connect(self.call_api_encrypt)
12        self.ui.btn_decrypt.clicked.connect(self.call_api_decrypt)
13
14    def call_api_encrypt(self):
15        url = "http://127.0.0.1:5000/api/caesar/encrypt"
16        payload = {
17            "plain_text": self.ui.txt_plain_text.toPlainText(),
18            "key": self.ui.txt_key.text()
19        }
20        try:
21            response = requests.post(url, json=payload)
22            if response.status_code == 200:
23                data = response.json()
24                self.ui.txt_cipher_text.setText(data["encrypted_message"])
25
26                msg = QMessageBox()
27                msg.setIcon(QMessageBox.Information)
28                msg.setText("Encrypted Successfully")
29                msg.exec_()
30            else:
31                print("Error while calling API")
32        except requests.exceptions.RequestException as e:
33            print("Error: %s" % e.message)
34
35    def call_api_decrypt(self):
36        url = "http://127.0.0.1:5000/api/caesar/decrypt"
37        payload = {
38            "cipher_text": self.ui.txt_cipher_text.toPlainText(),
39            "key": self.ui.txt_key.text()
40        }

```

```

41     try:
42         response = requests.post(url, json=payload)
43         if response.status_code == 200:
44             data = response.json()
45             self.ui.txt_plain_text.setText(data["decrypted_message"])
46
47             msg = QMessageBox()
48             msg.setIcon(QMessageBox.Information)
49             msg.setText("Decrypted Successfully")
50             msg.exec_()
51         else:
52             print("Error while calling API")
53     except requests.exceptions.RequestException as e:
54         print("Error: %s" % e.message)
55
56 if __name__ == "__main__":
57     app = QApplication(sys.argv)
58     window = MyApp()
59     window.show()
60     sys.exit(app.exec_())

```

- Tiến hành kiểm tra ứng dụng:

- Bước 1: Split Terminal của VS Code, di chuyển vào folder “lab-02” và chạy file “api.py”.

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

- Bước 2: Tại Terminal của “lab-03”. Tiến hành chạy file “caesar\_cipher.py”.

```
python .\caesar_cipher.py
```

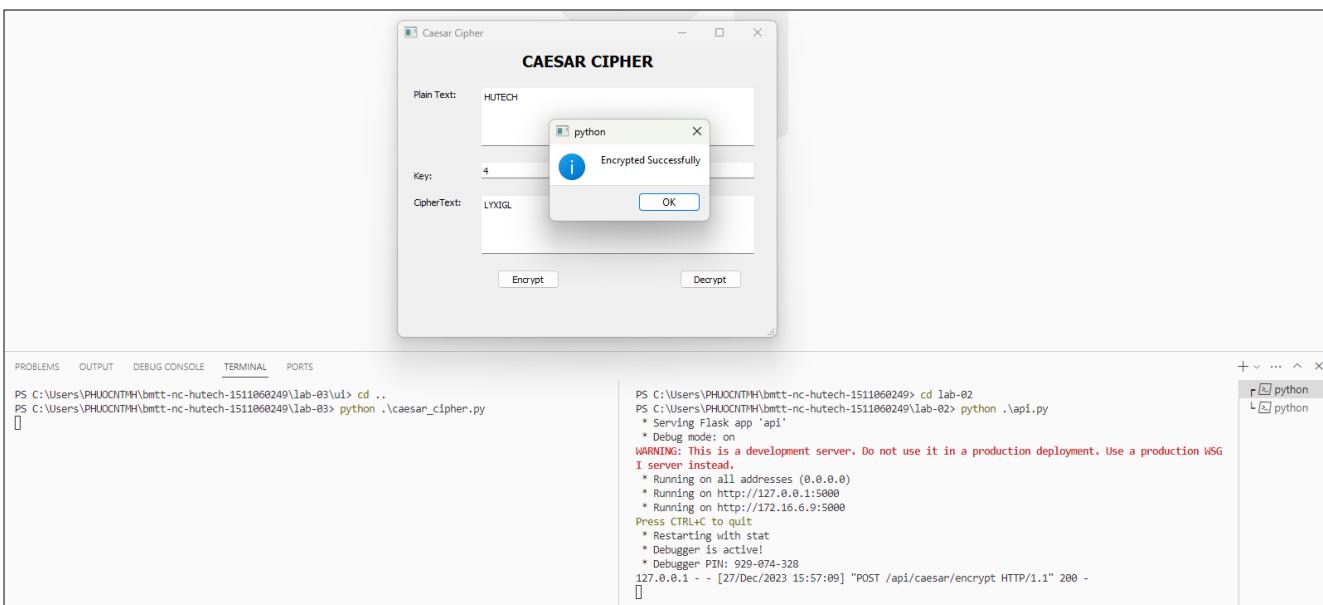
- Bước 3: Giao diện ứng dụng xuất hiện, tiến hành kiểm tra các chức năng mã hoá, giải mã.
- Lưu ý: Cần phải chạy lại project của “lab-02” vì chương trình sẽ kết nối và sử dụng API encrypt và decrypt của Mã hoá Caesar đã làm ở tuần trước. Khi click button sẽ gửi một HTTP Request (hai project đóng vai trò client-server).

```

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249> cd lab-02
PS C:\Users\PHUOCNTMH\bmtt-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
127.0.0.1 - - [27/Dec/2023 15:57:09] "POST /api/caesar/encrypt HTTP/1.1" 200 -
127.0.0.1 - - [27/Dec/2023 15:57:35] "POST /api/caesar/decrypt HTTP/1.1" 200 -

```

- Kết quả kiểm tra:



- Commit code:

```

git add .
git commit -m "[add] lab-03 ui caesar"

```

### 3.5.2 Bài thực hành 02: Tạo ứng dụng mã hóa RSA

Tạo ứng dụng desktop demo cho Mật mã RSA. Hướng dẫn:

- Trong folder "lab-03" tạo folder "cipher". Trong folder "cipher" tạo folder "rsa".
- Trong folder "rsa" tạo folder "keys" để chứa private key và public key.

- Cập nhật file “requirements.txt”, thêm nội dung sau vào:

```
Flask>=2.3.2
rsa>=4.9
```

- Cài đặt các gói cần thiết:

```
pip install -r .\requirements.txt
```

- Tạo file “rsa\_cipher.py” trong folder “rsa”.

```
rsa_cipher.py ×

lab-03 > rsa_cipher.py > MyApp > call_api_encrypt
1 import sys
2 from PyQt5.QtWidgets import QApplication, QMainWindow, QMessageBox
3 from ui.rsa import Ui_MainWindow
4 import requests
5
6 class MyApp(QMainWindow):
7     def __init__(self):
8         super().__init__()
9         self.ui = Ui_MainWindow()
10        self.ui.setupUi(self)
11        self.ui.btn_gen_keys.clicked.connect(self.call_api_gen_keys)
12        self.ui.btn_encrypt.clicked.connect(self.call_api_encrypt)
13        self.ui.btn_decrypt.clicked.connect(self.call_api_decrypt)
14        self.ui.btn_sign.clicked.connect(self.call_api_sign)
15        self.ui.btn_verify.clicked.connect(self.call_api_verify)
16
17    def call_api_gen_keys(self):
18        url = "http://127.0.0.1:5000/api/rsa/generate_keys"
19        try:
20            response = requests.get(url)
21            if response.status_code == 200:
22                data = response.json()
23                msg = QMessageBox()
24                msg.setIcon(QMessageBox.Information)
25                msg.setText(data["message"])
26                msg.exec_()
27            else:
28                print("Error while calling API")
29        except requests.exceptions.RequestException as e:
30            print("Error: %s" % e.message)
31
```

```
32     def call_api_encrypt(self):
33         url = "http://127.0.0.1:5000/api/rsa/encrypt"
34         payload = {
35             "message": self.ui.txt_plain_text.toPlainText(),
36             "key_type": "public"
37         }
38         try:
39             response = requests.post(url, json=payload)
40             if response.status_code == 200:
41                 data = response.json()
42                 self.ui.txt_cipher_text.setText(data["encrypted_message"])
43
44                 msg = QMessageBox()
45                 msg.setIcon(QMessageBox.Information)
46                 msg.setText("Encrypted Successfully")
47                 msg.exec_()
48         else:
49             print("Error while calling API")
50     except requests.exceptions.RequestException as e:
51         print("Error: %s" % e.message)
52 
```

```
53     def call_api_decrypt(self):
54         url = "http://127.0.0.1:5000/api/rsa/decrypt"
55         payload = {
56             "ciphertext": self.ui.txt_cipher_text.toPlainText(),
57             "key_type": "private"
58         }
59         try:
60             response = requests.post(url, json=payload)
61             if response.status_code == 200:
62                 data = response.json()
63                 self.ui.txt_plain_text.setText(data["decrypted_message"])
64
65                 msg = QMessageBox()
66                 msg.setIcon(QMessageBox.Information)
67                 msg.setText("Decrypted Successfully")
68                 msg.exec_()
69         else:
70             print("Error while calling API")
71     except requests.exceptions.RequestException as e:
72         print("Error: %s" % e.message)
73 
```

```
74     def call_api_sign(self):
75         url = "http://127.0.0.1:5000/api/rsa/sign"
76         payload = {
77             "message": self.ui.txt_info.toPlainText(),
78         } 
```

```

79     try:
80         response = requests.post(url, json=payload)
81         if response.status_code == 200:
82             data = response.json()
83             self.ui.txt_sign.setText(data["signature"])
84
85             msg = QMessageBox()
86             msg.setIcon(QMessageBox.Information)
87             msg.setText("Signed Successfully")
88             msg.exec_()
89         else:
90             print("Error while calling API")
91     except requests.exceptions.RequestException as e:
92         print("Error: %s" % e.message)
93

```

```

94     def call_api_verify(self):
95         url = "http://127.0.0.1:5000/api/rsa/verify"
96         payload = {
97             "message": self.ui.txt_info.toPlainText(),
98             "signature": self.ui.txt_sign.toPlainText()
99         }
100        try:
101            response = requests.post(url, json=payload)
102            if response.status_code == 200:
103                data = response.json()
104                if (data["is_verified"]):
105                    msg = QMessageBox()
106                    msg.setIcon(QMessageBox.Information)
107                    msg.setText("Verified Successfully")
108                    msg.exec_()
109                else:
110                    msg = QMessageBox()
111                    msg.setIcon(QMessageBox.Information)
112                    msg.setText("Verified Fail")
113                    msg.exec_()
114            else:
115                print("Error while calling API")
116        except requests.exceptions.RequestException as e:
117            print("Error: %s" % e.message)
118
119    if __name__ == "__main__":
120        app = QApplication(sys.argv)
121        window = MyApp()
122        window.show()
123        sys.exit(app.exec_())

```

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

```
✚ __init__.py ✘
lab-03 > cipher > rsa > ✚ __init__.py
1   from .rsa_cipher import RSACipher
```

- Tạo file “api.py” trong thư mục “lab-03”.

```
✚ api.py 1 ✘
lab-03 > ✚ api.py > rsa_decrypt
1   from flask import Flask, request, jsonify
2   from cipher.rsa import RSACipher
3
4
5   app = Flask(__name__)
6
7   #RSA CIPHER ALGORITHM
8   rsa_cipher = RSACipher()
9
10  @app.route('/api/rsa/generate_keys', methods=['GET'])
11  def rsa_generate_keys():
12      rsa_cipher.generate_keys()
13      return jsonify({'message': 'Keys generated successfully'})
14
15  @app.route("/api/rsa/encrypt", methods=["POST"])
16  def rsa_encrypt():
17      data = request.json
18      message = data['message']
19      key_type = data['key_type']
20      private_key, public_key = rsa_cipher.load_keys()
21      if key_type == 'public':
22          key = public_key
23      elif key_type == 'private':
24          key = private_key
25      else:
26          return jsonify({'error': 'Invalid key type'})
27      encrypted_message = rsa_cipher.encrypt(message, key)
28      encrypted_hex = encrypted_message.hex()
29      return jsonify({'encrypted_message': encrypted_hex})
30
```

```
31  @app.route("/api/rsa/decrypt", methods=["POST"])
32  def rsa_decrypt():
33      data = request.json
34      ciphertext_hex = data['ciphertext']
35      key_type = data['key_type']
36      private_key, public_key = rsa_cipher.load_keys()
37      if key_type == 'public':
```

```

38     key = public_key
39 elif key_type == 'private':
40     key = private_key
41 else:
42     return jsonify({'error': 'Invalid key type'})
43 ciphertext = bytes.fromhex(ciphertext_hex)
44 decrypted_message = rsa_cipher.decrypt(ciphertext, key)
45 return jsonify({'decrypted_message': decrypted_message})
46
47 @app.route('/api/rsa/sign', methods=['POST'])
48 def rsa_sign_message():
49     data = request.json
50     message = data['message']
51     private_key, _ = rsa_cipher.load_keys()
52     signature = rsa_cipher.sign(message, private_key)
53     signature_hex = signature.hex()
54     return jsonify({'signature': signature_hex})
55

```

```

56 @app.route('/api/rsa/verify', methods=['POST'])
57 def rsa_verify_signature():
58     data = request.json
59     message = data['message']
60     signature_hex = data['signature']
61     public_key, _ = rsa_cipher.load_keys()
62     signature = bytes.fromhex(signature_hex)
63     is_verified = rsa_cipher.verify(message, signature, public_key)
64     return jsonify({'is_verified': is_verified})
65 #main function
66 if __name__ == "__main__":
67     app.run(host="0.0.0.0", port=5000, debug=True)

```

- Khởi động development server.

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

- Khởi động Postman, tạo mới collection “RSA” có 04 HTTP Request là “generate\_keys”, “encrypt”, “decrypt”, “sign”, “verify”:

```
1 curl --location 'http://127.0.0.1:5000/api/rsa/generate_keys'
```

```

1 curl --location 'http://127.0.0.1:5000/api/rsa/encrypt' \
2 --header 'Content-Type: application/json' \
3 --data '{
4     "message": "",
5     "key_type": "public"
6 }'

```

```
1 curl --location 'http://127.0.0.1:5000/api/rsa/decrypt' \
2 --header 'Content-Type: application/json' \
3 --data '{
4     "ciphertext": "",
5     "key_type": "private"
6 }'
```

```
1 curl --location 'http://127.0.0.1:5000/api/rsa/sign' \
2 --header 'Content-Type: application/json' \
3 --data '{
4     "message": "HUTECH University"
5 }'
```

```
1 curl --location 'http://127.0.0.1:5000/api/rsa/verify' \
2 --header 'Content-Type: application/json' \
3 --data '{
4     "message": "HUTECH University",
5     "signature": ""
6 }'
```

- Tiến hành bấm nút “Send” để kiểm tra các API.
- Kết quả tạo key thành công (kiểm tra folder “key” thấy có 2 key là publicKey.pem và privateKey.pem):

The screenshot shows the Postman interface with the following details:

- Request URL:** {{base\_url}}/rsa/generate\_keys
- Method:** GET
- Body:** None (This request does not have a body)
- Response Status:** 200 OK (291 ms, 212 B)
- Code snippet:**

```
curl --location 'http://127.0.0.1:5000/api/rsa/generate_keys'
```
- Response Body (Pretty JSON):**

```
1 {
2     "message": "Keys generated successfully"
3 }
```

- Kết quả mã hoá:

The screenshot shows a POST request to `http://127.0.0.1:5000/api/rsa/encrypt`. The Body is set to raw JSON:

```

1 {
2   ... "message": "Nguyen Trong Minh Hong Phuoc",
3   ... "key_type": "public"
4 }

```

The response status is 200 OK, and the JSON body contains the encrypted message:

```

1 {
2   "encrypted_message": "53d900858ada3c12a06b63ebdb19d5b87b5ba9c66d175c056503c01409cb8dc5764f37b7b85650c21982824722fc6d2e1e669c2f5684efc07f9997055fe4d1fa18098b105f749ae65034fa32f471ec5b737b4755a639cfe1cbb7d15a5bb5eeff05c53950c09d0ce21f0cf3a013e3cb58703793fb8482679fcfc19cd121b15297"
3 }

```

### - Kết quả giải mã:

The screenshot shows a POST request to `http://127.0.0.1:5000/api/rsa/decrypt`. The Body is set to raw JSON:

```

1 {
2   ... "ciphertext": "b6f50e13b2a4e5e9c1dcb904da2d4b8259acf87e613b59bbcd9b1058c702a1fc2f59d8d69ac98e90c4986ab1d31b2b1fe93302ae0e7cddecfea262839f56d2faacf0203f642b5c97f8288600771df4ac23e4a6e2e2fc4c4649742b696da65c02328335f3de91156d9023abde74489f036fb059360dd49e900ddb6df7fafdf6e696",
3   ... "key_type": "private"
4 }

```

The response status is 200 OK, and the JSON body contains the decrypted message:

```

1 {
2   "decrypted_message": "Nguyen Trong Minh Hong Phuoc"
3 }

```

### - Kết quả tạo chữ ký:

The screenshot shows a POST request to `http://127.0.0.1:5000/api/rsa/sign`. The Body is set to raw JSON:

```

1 {
2   ... "message": "HUTECH University"
3 }

```

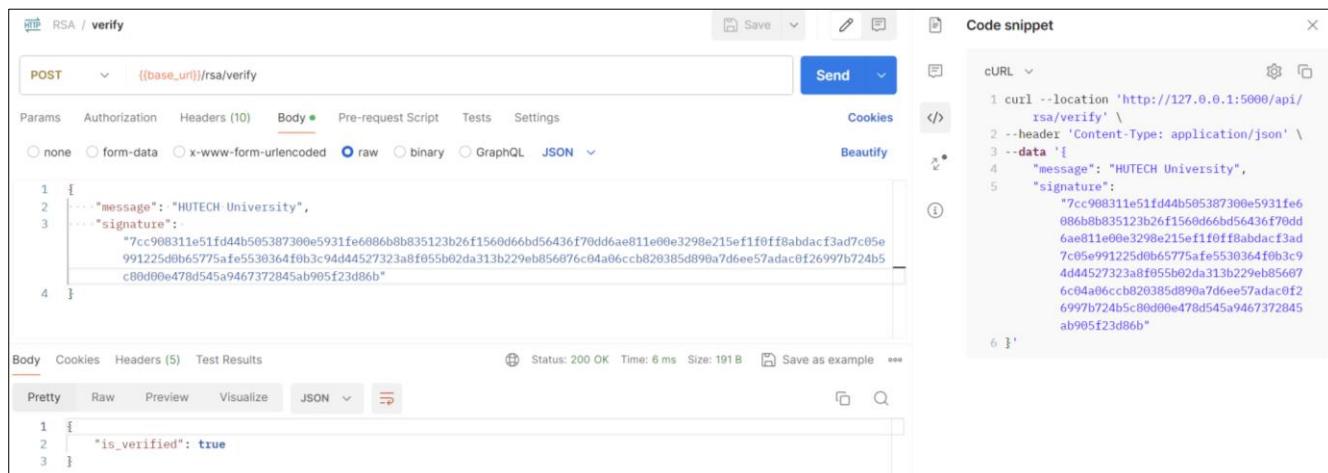
The response status is 200 OK, and the JSON body contains the signature:

```

1 {
2   "signature": "7cc908311e51fd44b505387300e5931fe6086b8b835123b26f1560d66bd56436f70dd6ae811e00e3298e215ef1ffbabdacf3ad7c05e99125d0b575safe5530364fb3c94d44527323a8f055b02da313b229eb856076c04a06ccb820385d890a7d6ee57adac0f26997b724b5c80d00e478d545a9467372845ab905f23d06b"
3 }

```

- Kết quả xác thực chữ ký:



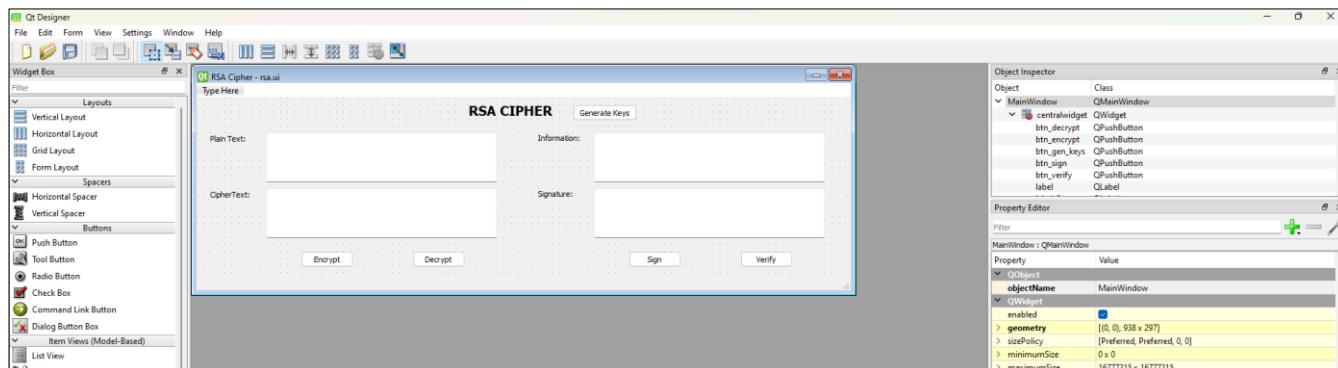
The screenshot shows the Postman interface with a POST request to `http://{{base_url}}/rsa/verify`. The request body contains a JSON object with "message": "HUTECH University" and "signature": a long hex string. The response status is 200 OK, and the JSON body is {"is\_verified": true}.

```

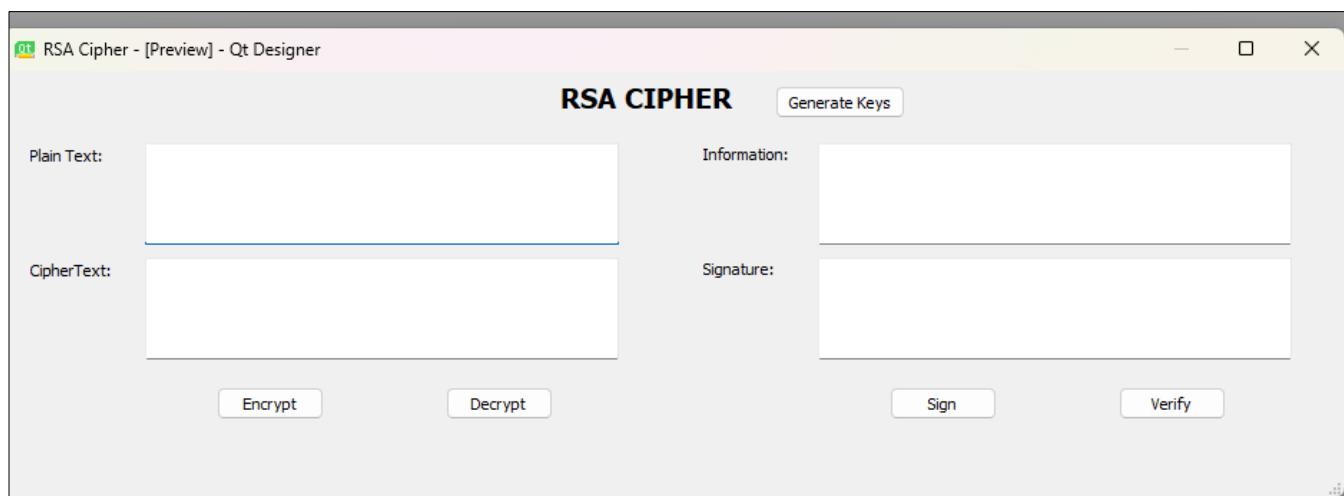
POST {{base_url}}/rsa/verify
{
  "message": "HUTECH University",
  "signature": "7cc908311e51fd44b505387300e5931fe6086b8b835123b26f1560d66bd56436f70ddae811e00e3298e215ef1f0ff8abdacf3ad7c05e91225d0b65775afe5530364fb3c94d44527323a8f055b02da313b229eb856076c04a06ccb820385d890a7d6ee57adac0f26997b724b5c80d00e478d545a9467372845ab905f23d86b"
}
  
```

Body: {"is\_verified": true}

- Mở QtDesigner và thiết kế giao diện cho ứng dụng:



- Sau khi thiết kế xong, nhấn tổ hợp phím Ctrl + R để xem lại giao diện khi thực thi.



- Tiến hành lưu file thiết kế .ui vào folder "ui" trong "lab-03" với tên "rsa.ui".

- Chuyển giao diện sang code PyQt5 bằng cách chạy lệnh sau trong Terminal của VS Code (thực thi tại thư mục “lab-03”):

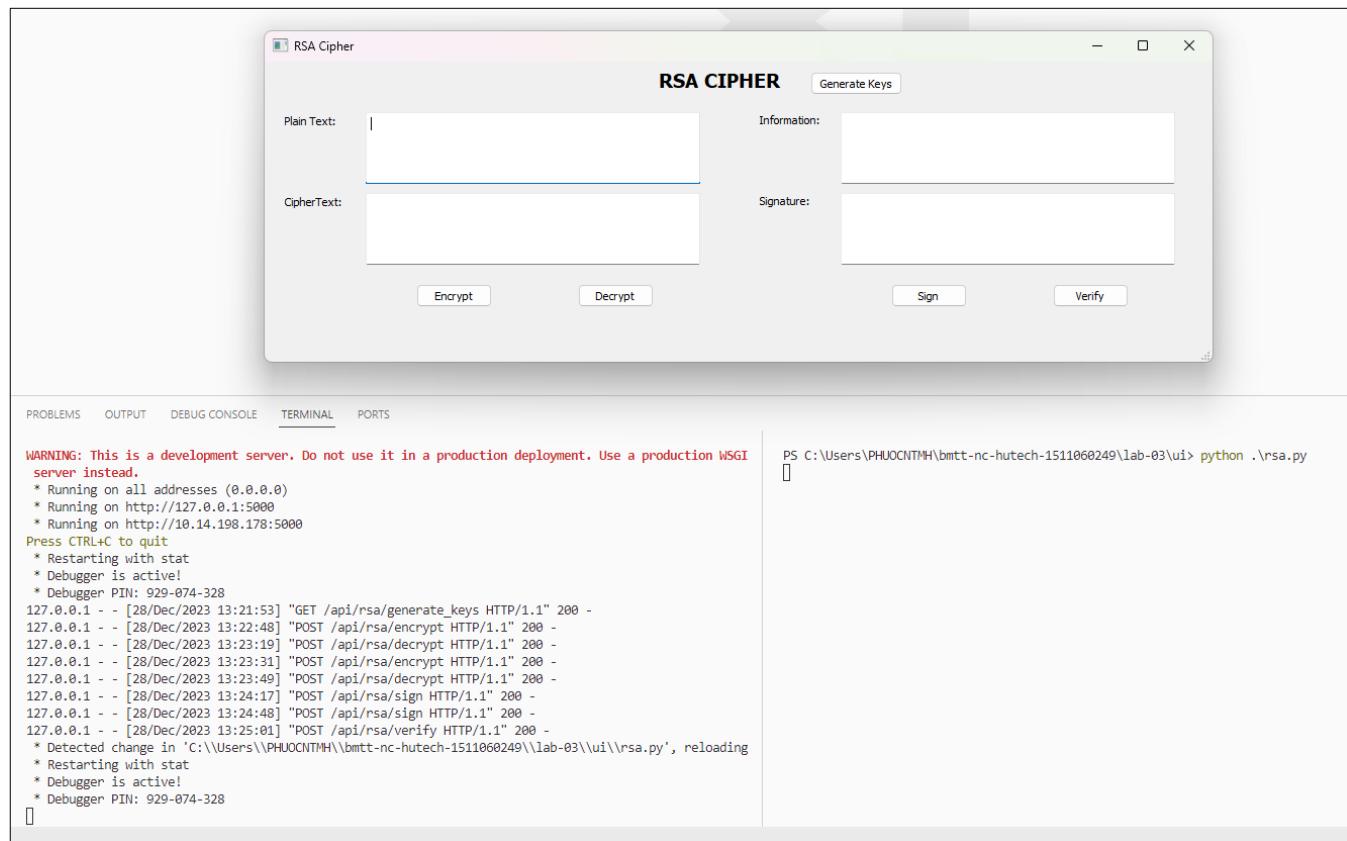
```
pyuic5 -x ./ui/rsa.ui -o ./ui/rsa.py
```

- Thêm hai dòng sau vào đầu file “rsa.py”:

```
import os
os.environ['QT_QPA_PLATFORM_PLUGIN_PATH'] = "../platforms"
```

- Kiểm tra thực thi của giao diện:

```
python .\rsa.py
```



- Vừa rồi, chúng ta đã làm xong front-end, tiếp theo, chúng ta sẽ làm phần back-end.
- Trong folder “lab-03” tạo file “rsa\_cipher.py” (Lưu ý: Kiểm tra tên của các button cho đúng).

```
rsa_cipher.py ×  
lab-03 > rsa_cipher.py > MyApp > call_api_encrypt  
1 import sys  
2 from PyQt5.QtWidgets import QApplication, QMainWindow, QMessageBox  
3 from ui.rsa import Ui_MainWindow  
4 import requests  
5  
6 class MyApp(QMainWindow):  
7     def __init__(self):  
8         super().__init__()  
9         self.ui = Ui_MainWindow()  
10        self.ui.setupUi(self)  
11        self.ui.btn_gen_keys.clicked.connect(self.call_api_gen_keys)  
12        self.ui.btn_encrypt.clicked.connect(self.call_api_encrypt)  
13        self.ui.btn_decrypt.clicked.connect(self.call_api_decrypt)  
14        self.ui.btn_sign.clicked.connect(self.call_api_sign)  
15        self.ui.btn_verify.clicked.connect(self.call_api_verify)  
16  
17    def call_api_gen_keys(self):  
18        url = "http://127.0.0.1:5000/api/rsa/generate_keys"  
19        try:  
20            response = requests.get(url)  
21            if response.status_code == 200:  
22                data = response.json()  
23                msg = QMessageBox()  
24                msg.setIcon(QMessageBox.Information)  
25                msg.setText(data["message"])  
26                msg.exec_()  
27            else:  
28                print("Error while calling API")  
29        except requests.exceptions.RequestException as e:  
30            print("Error: %s" % e.message)  
31
```

```
32    def call_api_encrypt(self):  
33        url = "http://127.0.0.1:5000/api/rsa/encrypt"  
34        payload = {  
35            "message": self.ui.txt_plain_text.toPlainText(),  
36            "key_type": "public"  
37        }  
38        try:  
39            response = requests.post(url, json=payload)  
40            if response.status_code == 200:  
41                data = response.json()  
42                self.ui.txt_cipher_text.setText(data["encrypted_message"])  
43
```

```

44             msg = QMessageBox()
45             msg.setIcon(QMessageBox.Information)
46             msg.setText("Encrypted Successfully")
47             msg.exec_()
48         else:
49             print("Error while calling API")
50     except requests.exceptions.RequestException as e:
51         print("Error: %s" % e.message)
52

```

```

53     def call_api_decrypt(self):
54         url = "http://127.0.0.1:5000/api/rsa/decrypt"
55         payload = {
56             "ciphertext": self.ui.txt_cipher_text.toPlainText(),
57             "key_type": "private"
58         }
59         try:
60             response = requests.post(url, json=payload)
61             if response.status_code == 200:
62                 data = response.json()
63                 self.ui.txt_plain_text.setText(data["decrypted_message"])
64
65                 msg = QMessageBox()
66                 msg.setIcon(QMessageBox.Information)
67                 msg.setText("Decrypted Successfully")
68                 msg.exec_()
69             else:
70                 print("Error while calling API")
71         except requests.exceptions.RequestException as e:
72             print("Error: %s" % e.message)
73

```

```

74     def call_api_sign(self):
75         url = "http://127.0.0.1:5000/api/rsa/sign"
76         payload = {
77             "message": self.ui.txt_info.toPlainText(),
78         }
79         try:
80             response = requests.post(url, json=payload)
81             if response.status_code == 200:
82                 data = response.json()
83                 self.ui.txt_sign.setText(data["signature"])
84
85                 msg = QMessageBox()
86                 msg.setIcon(QMessageBox.Information)
87                 msg.setText("Signed Successfully")
88                 msg.exec_()
89             else:

```

```

90         print("Error while calling API")
91     except requests.exceptions.RequestException as e:
92         print("Error: %s" % e.message)

93
94     def call_api_verify(self):
95         url = "http://127.0.0.1:5000/api/rsa/verify"
96         payload = {
97             "message": self.ui.txt_info.toPlainText(),
98             "signature": self.ui.txt_sign.toPlainText()
99         }
100        try:
101            response = requests.post(url, json=payload)
102            if response.status_code == 200:
103                data = response.json()
104                if (data["is_verified"]):
105                    msg = QMessageBox()
106                    msg.setIcon(QMessageBox.Information)
107                    msg.setText("Verified Successfully")
108                    msg.exec_()
109                else:
110                    msg = QMessageBox()
111                    msg.setIcon(QMessageBox.Information)
112                    msg.setText("Verified Fail")
113                    msg.exec_()
114            else:
115                print("Error while calling API")
116            except requests.exceptions.RequestException as e:
117                print("Error: %s" % e.message)

118
119    if __name__ == "__main__":
120        app = QApplication(sys.argv)
121        window = MyApp()
122        window.show()
123        sys.exit(app.exec_())

```

- Tiến hành kiểm tra ứng dụng:

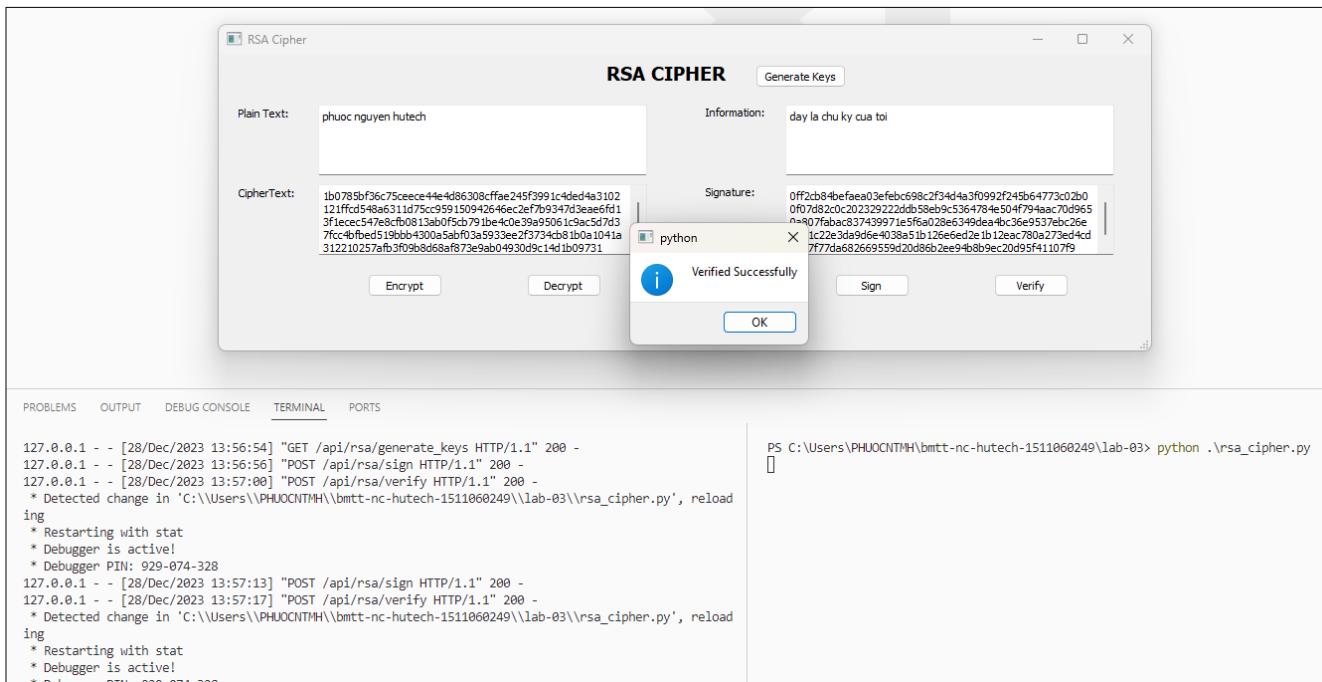
- Bước 1: Split Terminal của VS Code, di chuyển vào folder “lab-03” và chạy file “api.py”.

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

- Bước 2: Tại Terminal của “lab-03”. Tiến hành chạy file “rsa\_cipher.py”.

```
python .\rsa_cipher.py
```

- Bước 3: Giao diện ứng dụng xuất hiện, tiến hành kiểm tra các chức năng tạo khoá, mã hoá, giải mã, ký, xác thực chữ ký.
- Kết quả kiểm tra:



- Commit code:

```
git add .
git commit -m "[add] lab-03 ui rsa"
```

### 3.5.3 Bài thực hành 03: Tạo ứng dụng mã hoá ECC

Tạo ứng dụng desktop demo cho Mật mã ECC.

Hướng dẫn:

- Trong folder "cipher" tạo folder "ecc".
- Trong folder "ecc" tạo folder "keys" để chứa private key và public key.
- Cập nhật file "requirements.txt", thêm nội dung sau vào:

```
ecdsa
```

- Cài đặt các gói cần thiết:

```
pip install -r .\requirements.txt
```

- Tạo file “ecc\_cipher.py” trong folder “ecc”.

```

萍 ecc_cipher.py ×

lab-03 > cipher > ecc >萍 ecc_cipher.py > ...
1 import ecdsa, os
2
3 if not os.path.exists('cipher/ecc/keys'):
4     os.makedirs('cipher/ecc/keys')
5
6 class ECCCipher:
7     def __init__(self):
8         pass
9
10    def generate_keys(self):
11        sk = ecdsa.SigningKey.generate()      # Tạo khóa riêng tư
12        vk = sk.get_verifying_key()          # Lấy khóa công khai từ khóa riêng
13        tự
14
15        with open('cipher/ecc/keys/privateKey.pem', 'wb') as p:
16            p.write(sk.to_pem())
17
18        with open('cipher/ecc/keys/publicKey.pem', 'wb') as p:
19            p.write(vk.to_pem())
20
21    def load_keys(self):
22        with open('cipher/ecc/keys/privateKey.pem', 'rb') as p:
23            sk = ecdsa.SigningKey.from_pem(p.read())
24
25        with open('cipher/ecc/keys/publicKey.pem', 'rb') as p:
26            vk = ecdsa.VerifyingKey.from_pem(p.read())
27
28        return sk, vk
29
30    def sign(self, message, key):
31        # Ký dữ liệu bằng khóa riêng tư
32        return key.sign(message.encode('ascii'))
33
34    def verify(self, message, signature, key):
35        _, vk = self.load_keys()
36        try:
37            return vk.verify(signature, message.encode('ascii'))
38        except ecdsa.BadSignatureError:
39            return False

```

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

```
✚ __init__.py ✘
lab-03 > cipher > ecc > ✚ __init__.py
1   from .ecc_cipher import ECCCipher
```

- Cập nhật thêm code vào file “api.py” trong thư mục “lab-03”.

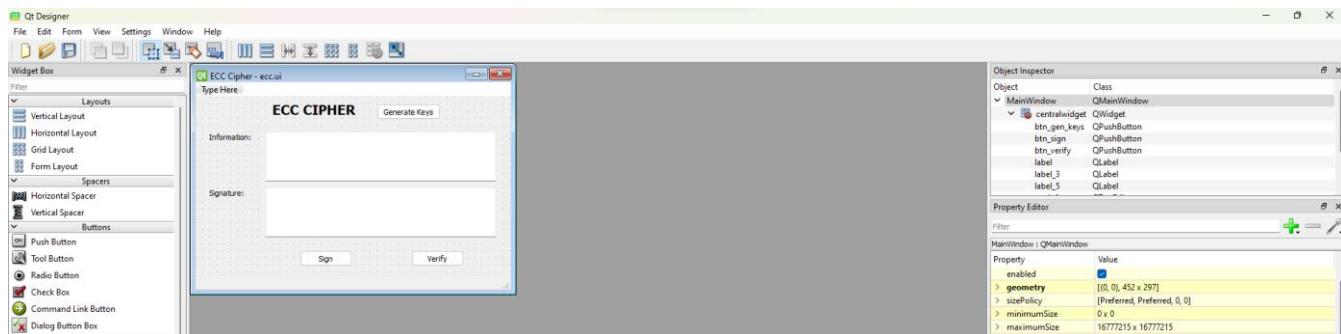
```
✚ api.py ✘
lab-03 > ✚ api.py > ✎ rsa_verify_signature
1   from flask import Flask, request, jsonify
2   from cipher.rsa import RSACipher
3   from cipher.ecc import ECCCipher # Thêm vào đầu file
```

```
66  # Thêm đoạn này trước hàm main
67  #ECC CIPHER ALGORITHM
68  ecc_cipher = ECCCipher()
69
70  @app.route('/api/ecc/generate_keys', methods=['GET'])
71  def ecc_generate_keys():
72      ecc_cipher.generate_keys()
73      return jsonify({'message': 'Keys generated successfully'})
74
75  @app.route('/api/ecc/sign', methods=['POST'])
76  def ecc_sign_message():
77      data = request.json
78      message = data['message']
79      private_key, _ = ecc_cipher.load_keys()
80      signature = ecc_cipher.sign(message, private_key)
81      signature_hex = signature.hex()
82      return jsonify({'signature': signature_hex})
83
84  @app.route('/api/ecc/verify', methods=['POST'])
85  def ecc_verify_signature():
86      data = request.json
87      message = data['message']
88      signature_hex = data['signature']
89      public_key, _ = ecc_cipher.load_keys()
90      signature = bytes.fromhex(signature_hex)
91      is_verified = ecc_cipher.verify(message, signature, public_key)
92      return jsonify({'is_verified': is_verified})
93
```

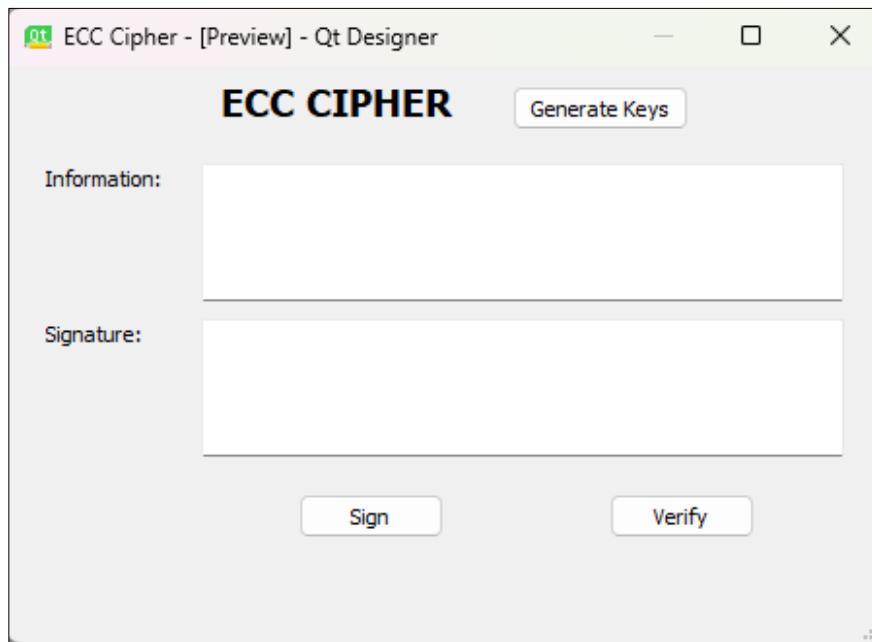
- Khởi động lại development server. Nếu development server đang chạy, nhấn tổ hợp phím Ctrl + C để ngắt.

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

- Sinh viên có thể dùng Postman để kiểm tra các API trước khi tiến hành bước tiếp theo.
- Mở QtDesigner và thiết kế giao diện cho ứng dụng:



- Sau khi thiết kế xong, nhấn tổ hợp phím Ctrl + R để xem lại giao diện khi thực thi.



- Tiến hành lưu file thiết kế .ui vào folder "ui" trong "lab-03" với tên "ecc.ui".
- Chuyển giao diện sang code PyQt5 bằng cách chạy lệnh sau trong Terminal của VS Code (thực thi tại thư mục "lab-03"):

```
pyuic5 -x ./ui/ecc.ui -o ./ui/ecc.py
```

- Thêm hai dòng sau vào đầu file "ecc.py":

```

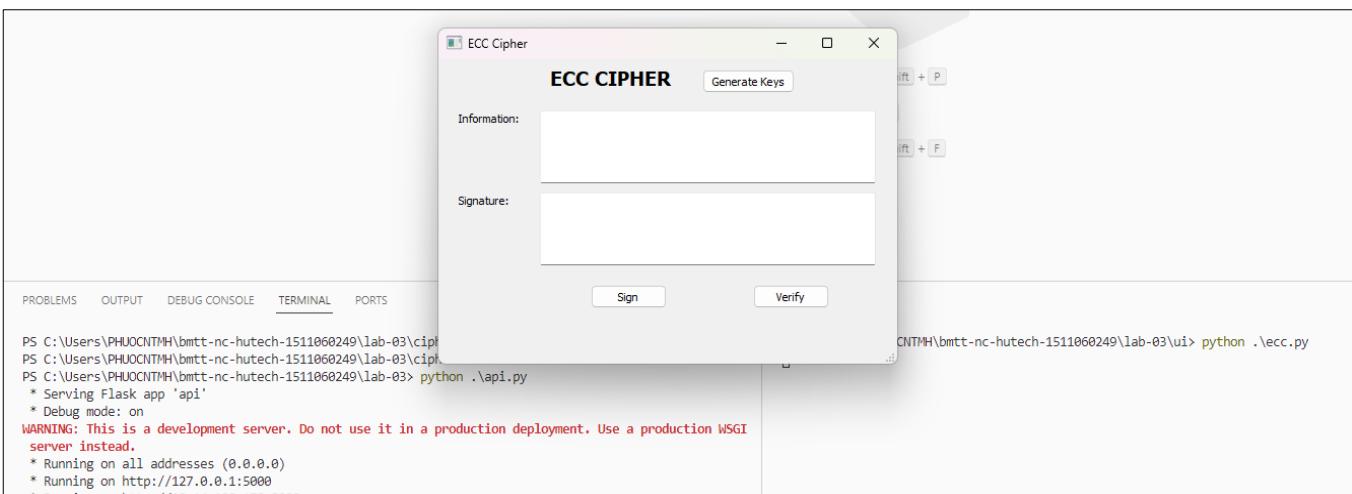
ecc.py  X

lab-03 > ui > ecc.py > Ui_MainWindow > setupUi
1  # -*- coding: utf-8 -*-
2
3  # Form implementation generated from reading ui file './ui/ecc.ui'
4  #
5  # Created by: PyQt5 UI code generator 5.15.10
6  #
7  # WARNING: Any manual changes made to this file will be lost when pyuic5 is
8  # run again. Do not edit this file unless you know what you are doing.
9
10
11  from PyQt5 import QtCore, QtGui, QtWidgets
12  import os
13  os.environ['QT_QPA_PLATFORM_PLUGIN_PATH'] = "../platforms"

```

- Kiểm tra thực thi của giao diện:

```
python .\ecc.py
```



- Vừa rồi, chúng ta đã làm xong front-end, tiếp theo, chúng ta sẽ làm phần back-end.
- Trong folder "lab-03" tạo file "ecc\_cipher.py" (Lưu ý: Kiểm tra tên của các button cho đúng).

```

ecc_cipher.py  X

lab-03 > ecc_cipher.py > ...
1  import sys
2  from PyQt5.QtWidgets import QApplication, QMainWindow, QMessageBox
3  from ui.ecc import Ui_MainWindow
4  import requests
5

```

```
6  class MyApp(QMainWindow):
7      def __init__(self):
8          super().__init__()
9          self.ui = Ui_MainWindow()
10         self.ui.setupUi(self)
11         self.ui.btn_gen_keys.clicked.connect(self.call_api_gen_keys)
12         self.ui.btn_sign.clicked.connect(self.call_api_sign)
13         self.ui.btn_verify.clicked.connect(self.call_api_verify)
14
15     def call_api_gen_keys(self):
16         url = "http://127.0.0.1:5000/api/ecc/generate_keys"
17         try:
18             response = requests.get(url)
19             if response.status_code == 200:
20                 data = response.json()
21                 msg = QMessageBox()
22                 msg.setIcon(QMessageBox.Information)
23                 msg.setText(data["message"])
24                 msg.exec_()
25             else:
26                 print("Error while calling API")
27         except requests.exceptions.RequestException as e:
28             print("Error: %s" % e.message)
29
```

```
30     def call_api_sign(self):
31         url = "http://127.0.0.1:5000/api/ecc/sign"
32         payload = {
33             "message": self.ui.txt_info.toPlainText(),
34         }
35         try:
36             response = requests.post(url, json=payload)
37             if response.status_code == 200:
38                 data = response.json()
39                 self.ui.txt_sign.setText(data["signature"])
40
41                 msg = QMessageBox()
42                 msg.setIcon(QMessageBox.Information)
43                 msg.setText("Signed Successfully")
44                 msg.exec_()
45             else:
46                 print("Error while calling API")
47         except requests.exceptions.RequestException as e:
48             print("Error: %s" % e.message)
49
```

```

50     def call_api_verify(self):
51         url = "http://127.0.0.1:5000/api/ecc/verify"
52         payload = {
53             "message": self.ui.txt_info.toPlainText(),
54             "signature": self.ui.txt_sign.toPlainText()
55         }
56         try:
57             response = requests.post(url, json=payload)
58             if response.status_code == 200:
59                 data = response.json()
60                 if (data["is_verified"]):
61                     msg = QMessageBox()
62                     msg.setIcon(QMessageBox.Information)
63                     msg.setText("Verified Successfully")
64                     msg.exec_()
65                 else:
66                     msg = QMessageBox()
67                     msg.setIcon(QMessageBox.Information)
68                     msg.setText("Verified Fail")
69                     msg.exec_()
70             else:
71                 print("Error while calling API")
72         except requests.exceptions.RequestException as e:
73             print("Error: %s" % e.message)
74
75     if __name__ == "__main__":
76         app = QApplication(sys.argv)
77         window = MyApp()
78         window.show()
79         sys.exit(app.exec_())

```

- Tiến hành kiểm tra ứng dụng:

- Bước 1: Split Terminal của VS Code, di chuyển vào folder “lab-03” và chạy file “api.py”.

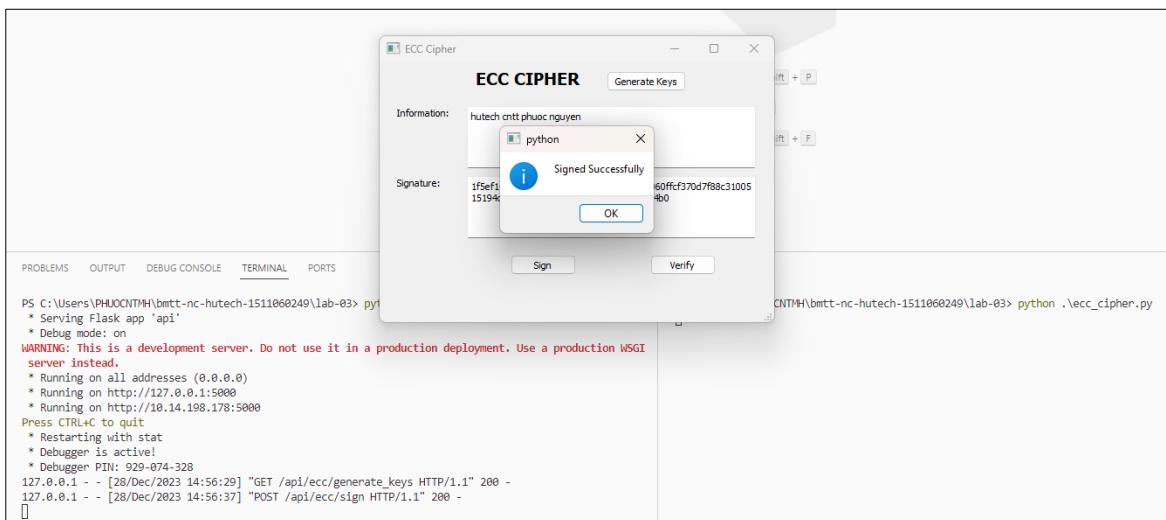
```
python .\api.py
```

- Bước 2: Tại Terminal của “lab-03”. Tiến hành chạy file “ecc\_cipher.py”.

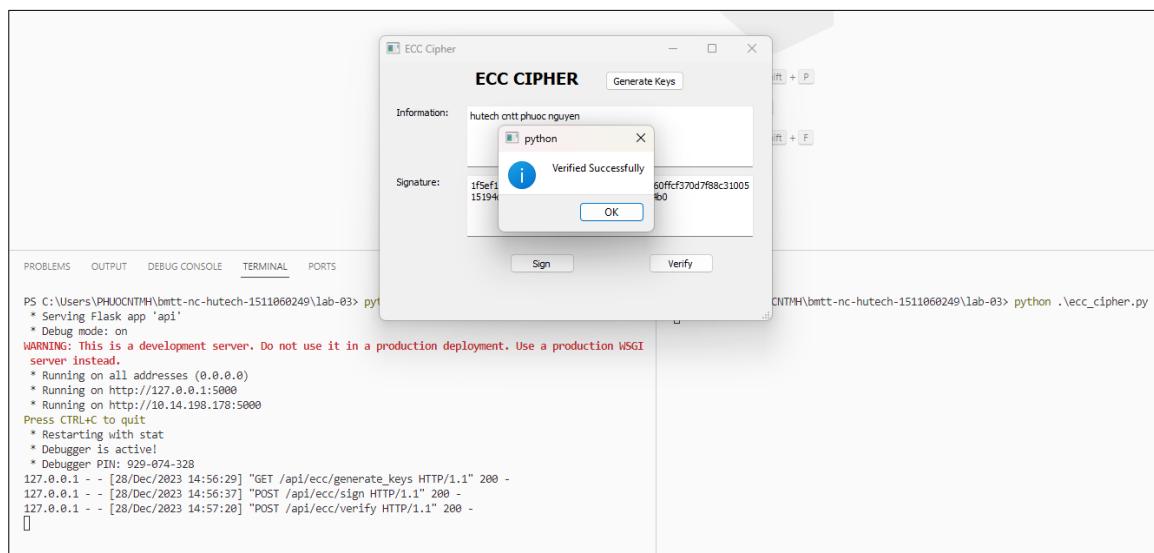
```
python .\ecc_cipher.py
```

- Bước 3: Giao diện ứng dụng xuất hiện, tiến hành kiểm tra các chức năng tạo khoá, ký, xác thực chữ ký.

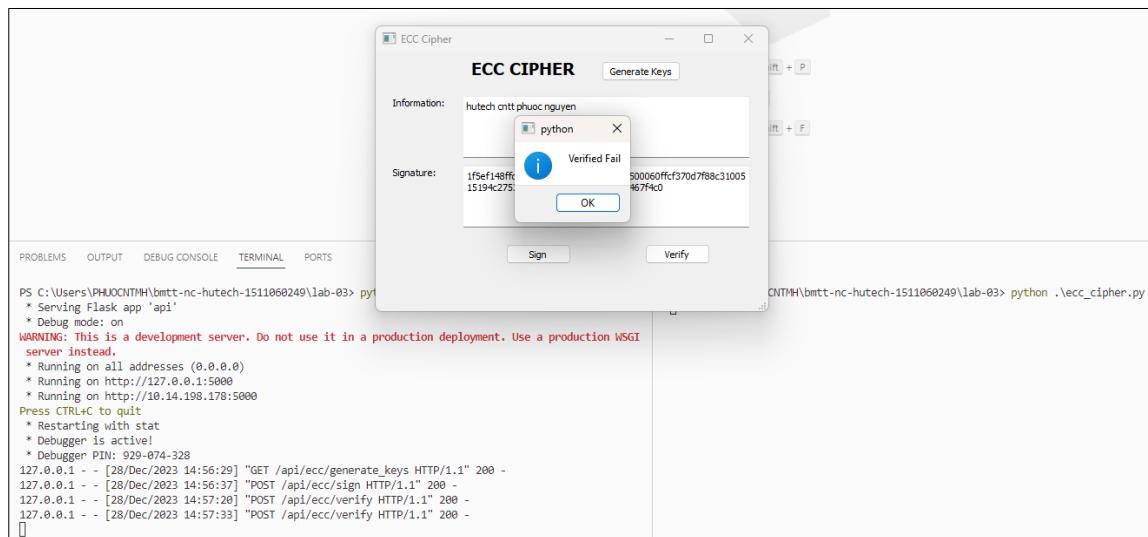
- Kết quả kiểm tra tạo chữ ký:



- Kết quả kiểm tra xác minh chữ ký:



- Thay đổi chữ ký, kiểm tra lại:



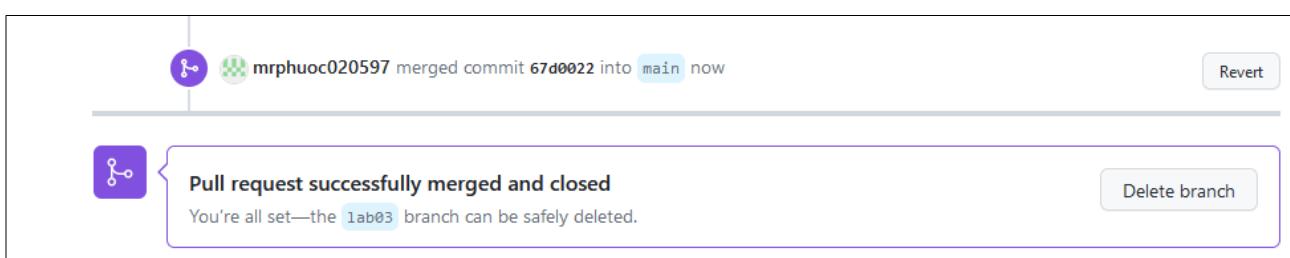
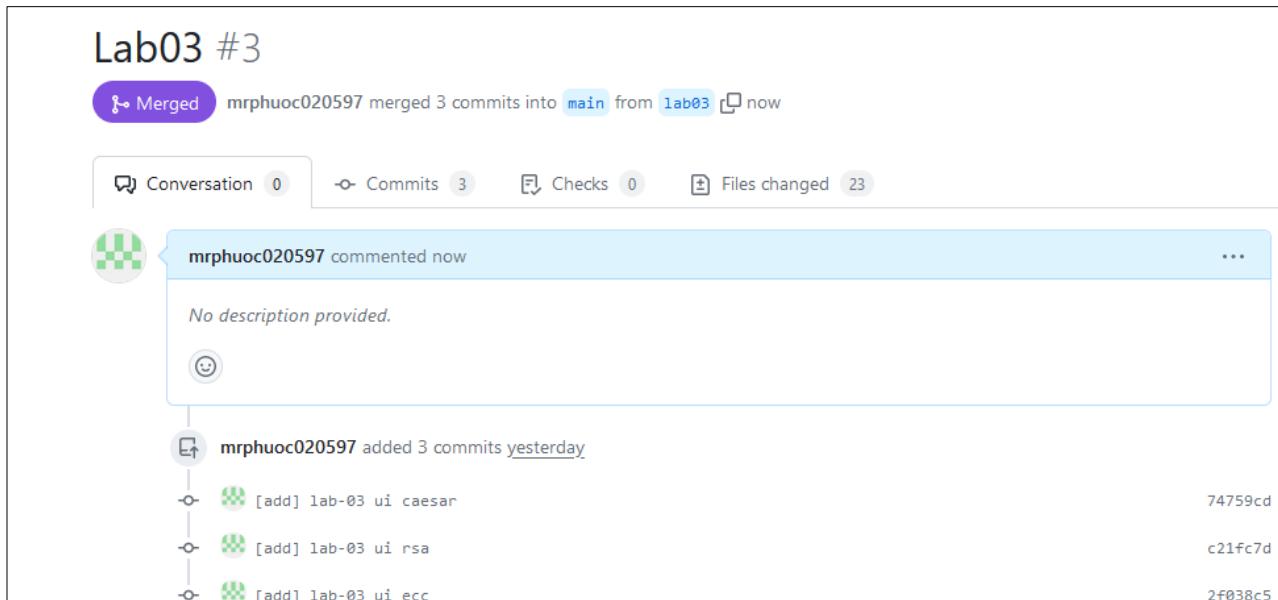
- Commit code:

```
git add .
git commit -m "[add] lab-03 ui ecc"
```

- Push các thay đổi lên remote repo:

```
git push origin lab03
```

- Tiến hành tạo PR từ nhánh lab03 về nhánh main. Review, kiểm tra đủ số lượng file, sau đó tiến hành “Merge pull request”.



## 3.6 BÀI TẬP MỞ RỘNG

- **Câu 01:** Thực hiện tạo giao diện trên desktop demo cho các mật mã: Vigenère, Rail Fence, Playfair, Transposition.
- **Câu 02:** Thực hiện tạo giao diện tạo chữ ký số và xác minh chữ ký của hai mật mã RSA và ECC trên cùng một màn hình.

# BÀI 4: SOCKET, WEBSOCKET, TRAO ĐỔI KHOÁ VÀ HÀM BĂM

Trong bài thực hành này, chúng ta sẽ tìm hiểu về Socket và WebSocket, cũng như cách triển khai chúng trong Python bằng module Python Socket và framework Tornado. Ngoài ra, chúng ta sẽ khám phá phương pháp Trao đổi khóa bằng thuật toán Diffie-Hellman và các hàm băm như MD5, SHA-256, SHA-3, và Blake2. Bài học sẽ đi sâu vào cách thức hoạt động, đặc điểm và ứng dụng của từng thuật toán trong việc đảm bảo tính toàn vẹn của dữ liệu.

## 4.1 SOCKET VÀ MODULE PYTHON SOCKET

Socket (hay còn gọi là ổ cắm) là một giao diện chuẩn giúp các ứng dụng trên các thiết bị khác nhau trong mạng có thể giao tiếp và trao đổi dữ liệu. Socket có vai trò quan trọng trong việc thiết lập kết nối và truyền dữ liệu giữa các thiết bị trong mạng máy tính, làm cho quá trình trao đổi thông tin giữa các ứng dụng trở nên linh hoạt. Nhờ vậy, việc giao tiếp có thể diễn ra trên nhiều loại thiết bị, chẳng hạn như máy tính cá nhân, máy chủ, điện thoại di động và các thiết bị thuộc hệ sinh thái IoT (Internet of Things).

Socket được xác định bởi một địa chỉ IP và một số cổng. Địa chỉ IP giúp nhận diện máy tính hoặc thiết bị trong mạng, còn số cổng xác định ứng dụng cụ thể đang hoạt động trên thiết bị đó. Khi một ứng dụng cần kết nối với một máy chủ hoặc thiết bị khác, nó sẽ sử dụng Socket để gửi yêu cầu kết nối qua mạng. Có hai loại chính của Socket:

- **TCP Socket:** Được sử dụng trong giao thức TCP để thiết lập kết nối đáng tin cậy giữa các thiết bị.
- **UDP Socket:** Sử dụng giao thức UDP, là một giao thức giao tiếp không đảm bảo tính đáng tin cậy cao như TCP, nhưng nó có tốc độ truyền dữ liệu nhanh hơn và ít tốn kém hơn về tài nguyên mạng.

Trong Python, module “**socket**” cung cấp giao diện cho việc làm việc với các Socket, cho phép ứng dụng Python tạo, kết nối và tương tác với các Socket TCP/IP và UDP. Module “**socket**” trong Python cung cấp một cách mạnh mẽ và linh hoạt để thực hiện các hoạt động mạng cơ bản và là một công cụ quan trọng cho việc phát triển ứng dụng liên quan đến mạng.

## **4.2 WEBSOCKET VÀ TORNADO FRAMEWORK**

---

**WebSocket** là một công nghệ truyền thông cho phép thiết lập một kết nối liên tục và hai chiều giữa trình duyệt và máy chủ thông qua một kênh giao tiếp duy nhất. Công nghệ này giúp truyền tải dữ liệu theo thời gian thực giữa các thiết bị khác nhau một cách hiệu quả, giảm độ trễ so với các phương thức truyền thống. Điểm khác biệt chính giữa WebSocket và các phương pháp như HTTP là khả năng duy trì kết nối luôn mở.

**Tornado** là một framework web mạnh mẽ viết bằng Python, ban đầu được phát triển bởi công ty FriendFeed (sau đó được Facebook mua lại). Framework này được thiết kế để hỗ trợ các ứng dụng web thời gian thực, với khả năng quản lý hàng nghìn kết nối đồng thời một cách hiệu quả. Tornado thường được dùng để xây dựng các ứng dụng cần thời gian thực như dịch vụ streaming, ứng dụng chat và các hệ thống mạng khác. Nó cung cấp các công cụ hiệu suất cao để xử lý nhiều tác vụ cùng lúc, giúp giảm thiểu tác động đến tài nguyên của hệ thống.

## **4.3 GIAO THỨC DIFFIE – HELLMAN**

---

“Giao thức Diffie-Hellman là một phương pháp mật mã học được sử dụng để tạo ra một khoá bí mật chung giữa hai hoặc nhiều bên thông qua một kênh không an toàn. Phương pháp này đã định hình cơ sở cho việc thiết lập kết nối an toàn trong các hệ thống mạng và bảo mật thông tin trực tuyến.

Ý tưởng chính của giao thức này là cho phép các bên giao tiếp tạo ra một khoá bí mật chung mà không cần truyền bất kỳ thông tin về khoá bí mật qua kênh không an toàn” [6], [8]. Cách thức hoạt động của Diffie-Hellman như sau:

- Lựa chọn các tham số chia sẻ công khai: Hai bên tham gia giao tiếp (ví dụ là Alice và Bob) đồng ý về các tham số chung mà bất kỳ ai cũng có thể biết được. Tham số

này gồm số nguyên tố lớn  $p$  và một số nguyên  $g$  (thường là một số nguyên tố gốc).

- Tạo khoá bí mật cục bộ: Mỗi bên tạo ra một số nguyên bí mật của mình (đôi khi được gọi là số private key):  $a$  cho Alice và  $b$  cho Bob. Những số này chỉ được bên tạo ra biết.
- Tính toán khoá công khai: Mỗi bên tính toán khoá công khai của mình bằng cách sử dụng tham số chia sẻ công khai và số bí mật của mình. Alice tính  $A = g^a \text{ mod } p$ , trong khi Bob tính  $B = g^b \text{ mod } p$ .
- Trao đổi và tính toán khoá chung: Alice và Bob trao đổi khoá công khai của mình (tức là  $A$  và  $B$ ). Sau đó, mỗi bên sử dụng khoá công khai của đối phương và số bí mật của mình để tính toán khoá bí mật chung. Alice tính  $s = B^a \text{ mod } p$ , trong khi Bob tính  $s = A^b \text{ mod } p$ . Khoá bí mật  $s$  này sẽ giống nhau ở cả hai bên.

## 4.4 TIÊU CHUẨN MÃ HÓA TIỀN TIẾN (AES)

"AES (Advanced Encryption Standard) là một thuật toán mã hóa khối (block cipher) phổ biến, thường được sử dụng trong các ứng dụng mã hóa và bảo mật dữ liệu. Được chính phủ Hoa Kỳ chọn làm tiêu chuẩn mã hóa chính thức, AES hiện đã trở thành một trong những thuật toán mã hóa được sử dụng rộng rãi nhất trên toàn cầu" [7], [8]. Cách hoạt động của AES như sau:

- **Khởi tạo:** Bắt đầu với một khóa được chọn và chuẩn bị dữ liệu cần mã hóa thành các khối cố định kích thước 128-bit.
- **SubBytes:** Mỗi byte trong khối dữ liệu được thay thế bằng một byte khác thông qua một bảng thay thế (S-box).
- **ShiftRows:** Các hàng trong khối dữ liệu được dịch chuyển sang trái theo một số lượng byte xác định.
- **MixColumns:** Các cột trong khối dữ liệu được kết hợp thông qua các phép nhân ma trận.
- **AddRoundKey:** Mỗi khối dữ liệu được kết hợp với một khóa con được tạo từ khóa chính.

## 4.5 HÀM BĂM

---

### 4.5.1 MD5

"MD5 (Message Digest Algorithm 5) là một trong những thuật toán băm (hash) được sử dụng rộng rãi để tạo ra một giá trị băm duy nhất (đôi khi được gọi là checksum hoặc hash value) từ một đầu vào dữ liệu. Nó được thiết kế để tạo ra một giá trị băm 128-bit, thường được biểu diễn bằng một chuỗi 32 ký tự hexa" [5], [8].

Tuy nhiên, cần lưu ý rằng MD5 đã bị các vấn đề về bảo mật phát hiện từ năm 1996 và các tấn công ngày càng phát triển dẫn đến việc MD5 không còn an toàn cho các ứng dụng bảo mật cao. Cách thức hoạt động của MD5:

- **Băm dữ liệu:** Đầu vào (dữ liệu cần băm) được chia thành các khối 512-bit và được xử lý theo từng khối.
- **Thêm bit padding:** Để đảm bảo rằng đầu vào có độ dài phù hợp, bit padding được thêm vào cuối dữ liệu.
- **Giai đoạn nén:** Mỗi khối dữ liệu được xử lý qua một loạt các vòng nén sử dụng hàm luân phiên (permutation function), hàm non-linear (non-linear function) và các phép lấy modulo.
- **Tạo giá trị băm:** Cuối cùng, sau khi tất cả các khối đã được xử lý, giá trị băm 128-bit được tạo ra từ các giá trị trung gian được tính toán trong quá trình nén.

### 4.5.2 SHA-256

Ban đầu, MD5 được sử dụng rộng rãi trong việc bảo mật mật khẩu và xác thực, cũng như trong việc kiểm tra tính toàn vẹn của các file. Tuy nhiên, do các vấn đề bảo mật đã được phát hiện, nên các thuật toán khác như SHA-256, SHA-3 đã được khuyến nghị thay thế MD5 trong các ứng dụng yêu cầu mức độ bảo mật cao hơn.

SHA-256 là một trong các thuật toán băm thuộc họ SHA-2 (Secure Hash Algorithm 2), được thiết kế để tạo ra một giá trị băm có độ dài 256-bit từ một đầu vào dữ liệu bất kỳ, đảm bảo tính duy nhất và khó khăn để trở lại dữ liệu ban đầu.

SHA-256, được phát triển bởi Cơ quan An ninh Quốc gia Hoa Kỳ (NSA), đã trở thành một trong những thuật toán băm được sử dụng rộng rãi nhất trong các ứng dụng bảo mật thông tin. Cách thức hoạt động của SHA-256 như sau:

- **Chuẩn bị đầu vào:** Đầu vào dữ liệu được chia thành các khối 512-bit.
- **Thêm bit padding:** Một số bit được thêm vào cuối dữ liệu để đảm bảo rằng kích thước cuối cùng của dữ liệu là 512-bit, theo định dạng của thuật toán.
- **Giai đoạn nén:** Các khối dữ liệu được xử lý qua một loạt các vòng nén sử dụng hàm luân phiên (permutation function), hàm non-linear (non-linear function), và các phép lấy modulo, tương tự như cách hoạt động của các thuật toán băm khác.
- **Tạo giá trị băm:** Cuối cùng, sau khi tất cả các khối đã được xử lý, giá trị băm 256-bit được tạo ra từ các giá trị trung gian được tính toán trong quá trình nén.

### 4.5.3 SHA-3

SHA-3 (Secure Hash Algorithm 3) là một thuật toán băm được thiết kế để tạo ra các giá trị băm từ các đầu vào dữ liệu bất kỳ. Nó là một trong những tiêu chuẩn bảo mật được công nhận quốc tế, được phát triển bởi hội đồng tiêu chuẩn hóa Quốc tế (NIST - National Institute of Standards and Technology) và được công bố vào năm 2015.

Thuật toán SHA-3 không chỉ đơn thuần là một phiên bản nâng cấp của SHA-2, mà là một thuật toán băm hoàn toàn mới được xây dựng từ các cơ sở thiết kế hoàn toàn khác biệt:

- **Kiến trúc hoàn toàn khác biệt:** SHA-3 không chia sẻ cùng kiến trúc hoặc cơ sở với SHA-2.
- **Khả năng chống tấn công:** SHA-3 được thiết kế để chống lại các phương pháp tấn công phổ biến hiện đại như collision attacks (tấn công va chạm).
- **Độ dài băm linh hoạt:** SHA-3 có thể tạo ra các giá trị băm với độ dài khác nhau, không giống như SHA-2 chỉ tạo ra các giá trị băm với độ dài cố định.
- **Tính hiệu quả và hiệu suất:** SHA-3 có khả năng chạy trên nhiều nền tảng khác nhau với tốc độ cao và hiệu suất tốt.

## 4.5.4 BLAKE2

Blake2 là một thuật toán băm hiện đại và linh hoạt, được phát triển nhằm tạo ra giá trị băm từ bất kỳ đầu vào dữ liệu nào. Đây là phiên bản nâng cấp của thuật toán Blake, do Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn và Christian Winnerlein phát triển vào năm 2013. Blake2 có những đặc điểm nổi bật sau:

- **Hiệu suất cao:** Blake2 được thiết kế để có hiệu suất cao trên nhiều nền tảng khác nhau, từ thiết bị nhúng đến máy tính cá nhân và các hệ thống có khả năng tính toán cao.
- **Đa năng và linh hoạt:** Blake2 hỗ trợ việc tạo ra các giá trị băm với độ dài khác nhau, từ 1 đến 64 byte.
- **Bảo mật cao:** Blake2 được thiết kế để đáp ứng các tiêu chí an ninh mật mã hiện đại, đặc biệt là tránh các vấn đề như collision attacks.
- **Hiệu quả và tiết kiệm tài nguyên:** Blake2 được thiết kế để tiết kiệm tài nguyên tính toán và bộ nhớ so với một số thuật toán băm khác.
- **Hỗ trợ nhiều dạng đầu vào:** Blake2 có khả năng xử lý cả dữ liệu có kích thước từ nhỏ đến lớn, từ tin nhắn văn bản đến file lớn.

Thuật toán Blake2 có hai biến thể chính:

- **Blake2b:** Được thiết kế cho các ứng dụng đòi hỏi độ dài băm lớn, với giá trị băm có độ dài từ 1 đến 64 byte.
- **Blake2s:** Thiết kế cho các ứng dụng yêu cầu giá trị băm có độ dài ngắn hơn, với giá trị băm từ 1 đến 32 byte.

## 4.6 BÀI TẬP THỰC HÀNH

---

### 4.6.1 Bài thực hành 01: Tạo Socket với AES và RSA

Tạo ứng dụng client-server giao tiếp thông qua cơ chế Socket và được mã hoá bằng mật mã AES và RSA.

Hướng dẫn:

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

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

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

```
git pull origin main
```

The screenshot shows the VS Code interface with the 'TERMINAL' tab selected. The terminal window displays the following text:  
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 lab04 từ nhánh main.

```
git checkout -b lab04
```

- Tạo folder “lab-04”. Trong folder “lab-04” tạo folder “aes\_rsa\_socket”.
- Trong folder “aes\_rsa\_socket” tạo file “requirements.txt”.

The screenshot shows a file explorer window with the following path:  
lab-04 > aes\_rsa\_socket > requirements.txt  
1 pycryptodome

- Vào Terminal, chạy các lệnh sau:

```
cd .\lab-04\aes_rsa_socket\
pip install -r .\requirements.txt
```

- Trong folder “aes\_rsa\_socket” tạo file “server.py”.

```

server.py  X

lab-04 > aes_rsa_socket > server.py > handle_client
1  from Crypto.Cipher import AES, PKCS1_OAEP
2  from Crypto.PublicKey import RSA
3  from Crypto.Random import get_random_bytes
4  from Crypto.Util.Padding import pad, unpad
5  import socket
6  import threading
7  import hashlib
8
9  # Initialize server socket
10 server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
11 server_socket.bind(('localhost', 12345))
12 server_socket.listen(5)
13
14 # Generate RSA key pair
15 server_key = RSA.generate(2048)
16
17 # List of connected clients
18 clients = []
19
20 # Function to encrypt message
21 def encrypt_message(key, message):
22     cipher = AES.new(key, AES.MODE_CBC)
23     ciphertext = cipher.encrypt(pad(message.encode(), AES.block_size))
24     return cipher.iv + ciphertext
25
26 # Function to decrypt message
27 def decrypt_message(key, encrypted_message):
28     iv = encrypted_message[:AES.block_size]
29     ciphertext = encrypted_message[AES.block_size:]
30     cipher = AES.new(key, AES.MODE_CBC, iv)
31     decrypted_message = unpad(cipher.decrypt(ciphertext), AES.block_size)
32     return decrypted_message.decode()
33

34 # Function to handle client connection
35 def handle_client(client_socket, client_address):
36     print(f"Connected with {client_address}")
37
38     # Send server's public key to client
39     client_socket.send(server_key.publickey().export_key(format='PEM'))
40
41     # Receive client's public key
42     client_received_key = RSA.import_key(client_socket.recv(2048))
43

```

```

44     # Generate AES key for message encryption
45     aes_key = get_random_bytes(16)
46
47     # Encrypt the AES key using the client's public key
48     cipher_rsa = PKCS1_OAEP.new(client_received_key)
49     encrypted_aes_key = cipher_rsa.encrypt(aes_key)
50     client_socket.send(encrypted_aes_key)
51
52     # Add client to the list
53     clients.append((client_socket, aes_key))
54
55     while True:
56         encrypted_message = client_socket.recv(1024)
57         decrypted_message = decrypt_message(aes_key, encrypted_message)
58         print(f"Received from {client_address}: {decrypted_message}")
59
60         # Send received message to all other clients
61         for client, key in clients:
62             if client != client_socket:
63                 encrypted = encrypt_message(key, decrypted_message)
64                 client.send(encrypted)
65

```

```

66             if decrypted_message == "exit":
67                 break
68
69             clients.remove((client_socket, aes_key))
70             client_socket.close()
71             print(f"Connection with {client_address} closed")
72
73     # Accept and handle client connections
74     while True:
75         client_socket, client_address = server_socket.accept()
76         client_thread = threading.Thread(target=handle_client, args=(client_socket,
77         client_address))
78         client_thread.start()

```

- Trong folder “aes\_rsa\_socket” tạo file “client.py”.

```

client.py  ×
lab-04 > aes_rsa_socket > client.py > encrypt_message
1  from Crypto.Cipher import AES, PKCS1_OAEP
2  from Crypto.PublicKey import RSA
3  from Crypto.Random import get_random_bytes
4  from Crypto.Util.Padding import pad, unpad
5  import socket
6  import threading
7  import hashlib
8

```

```
9 # Initialize client socket
10 client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
11 client_socket.connect(('localhost', 12345))
12
13 # Generate RSA key pair
14 client_key = RSA.generate(2048)
15
16 # Receive server's public key
17 server_public_key = RSA.import_key(client_socket.recv(2048))
18
19 # Send client's public key to the server
20 client_socket.send(client_key.publickey().export_key(format='PEM'))
21
22 # Receive encrypted AES key from the server
23 encrypted_aes_key = client_socket.recv(2048)
24
25 # Decrypt the AES key using client's private key
26 cipher_rsa = PKCS1_OAEP.new(client_key)
27 aes_key = cipher_rsa.decrypt(encrypted_aes_key)
28
```

```
29 # Function to encrypt message
30 def encrypt_message(key, message):
31     cipher = AES.new(key, AES.MODE_CBC)
32     ciphertext = cipher.encrypt(pad(message.encode(), AES.block_size))
33     return cipher.iv + ciphertext
34
35 # Function to decrypt message
36 def decrypt_message(key, encrypted_message):
37     iv = encrypted_message[:AES.block_size]
38     ciphertext = encrypted_message[AES.block_size:]
39     cipher = AES.new(key, AES.MODE_CBC, iv)
40     decrypted_message = unpad(cipher.decrypt(ciphertext), AES.block_size)
41     return decrypted_message.decode()
42
43 # Function to receive messages from server
44 def receive_messages():
45     while True:
46         encrypted_message = client_socket.recv(1024)
47         decrypted_message = decrypt_message(aes_key, encrypted_message)
48         print("Received:", decrypted_message)
49
```

```

50 # Start the receiving thread
51 receive_thread = threading.Thread(target=receive_messages)
52 receive_thread.start()
53
54 # Send messages from the client
55 while True:
56     message = input("Enter message ('exit' to quit): ")
57     encrypted_message = encrypt_message(aes_key, message)
58     client_socket.send(encrypted_message)
59     if message == "exit":
60         break
61
62 # Close the connection when done
63 client_socket.close()

```

- Kiểm tra ứng dụng:

- Bước 1: Split Terminal thành 3 cửa sổ. Tại cửa sổ 1 chạy file “server.py”.

```
python .\server.py
```



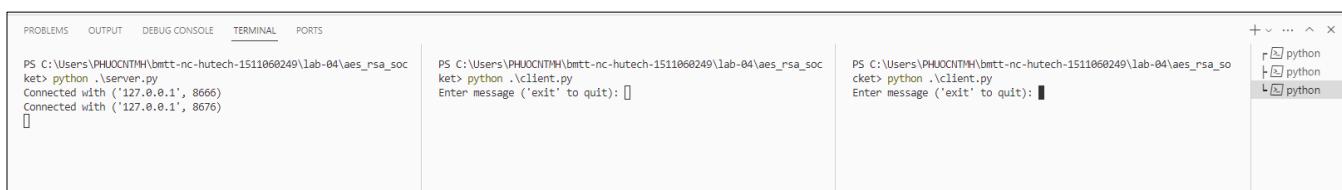
- Bước 2: Tại cửa sổ Terminal thứ 2, chạy file “client.py”. Ở server hiển thị kết nối.

```
python .\client.py
```



- Bước 3: Tại cửa sổ Terminal thứ 3, chạy file “client.py”. Ở server hiển thị kết nối.

```
python .\client.py
```



- Bước 4: Nhập nội dung chat tại hai client và kiểm tra.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\aes_rsa_soc
ket> python .\server.py
Connected with ('127.0.0.1', 8666)
Connected with ('127.0.0.1', 8676)
Received from ('127.0.0.1', 8666): xin chao
Received from ('127.0.0.1', 8676): toi la Phuoc
Received from ('127.0.0.1', 8666): chuc ban ngay moi vui ve
Received from ('127.0.0.1', 8676): ban cung vay :D
[]

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\aes_rsa_soc
ket> python .\client.py
Enter message ('exit' to quit): xin chao
Enter message ('exit' to quit): Received: xin chao
toi la Phuoc
Enter message ('exit' to quit): Received: chuc ban ngay moi vui ve
ban cung vay :D
Enter message ('exit' to quit):
```

- Commit code:

```
git add .
git commit -m "[add] lab-04 aes_rsa_socket"
```

## 4.6.2 Bài thực hành 02: DH Key Pair

Tạo ứng dụng client-server sử dụng Giao thức Diffie-Hellman để tạo và chia sẻ khoá.

Hướng dẫn:

- Trong folder “lab-04” tạo folder “dh\_key\_pair”. Trong folder “dh\_key\_pair” tạo file “requirements.txt”.

```
requirements.txt X
lab-04 > dh_key_pair > requirements.txt
1 cryptography
```

- Vào Terminal, chạy các lệnh sau:

```
cd .\lab-04\dh_key_pair\
pip install -r ./requirements.txt
```

- Trong folder “dh\_key\_pair” tạo file “server.py”.

```
server.py X
lab-04 > dh_key_pair > server.py ...
1 from cryptography.hazmat.primitives.asymmetric import dh
2 from cryptography.hazmat.primitives import serialization
3
4 def generate_dh_parameters():
5     parameters = dh.generate_parameters(generator=2, key_size=2048)
6     return parameters
7
```

```

8 def generate_server_key_pair(parameters):
9     private_key = parameters.generate_private_key()
10    public_key = private_key.public_key()
11    return private_key, public_key
12
13 def main():
14     parameters = generate_dh_parameters()
15     private_key, public_key = generate_server_key_pair(parameters)
16
17     with open("server_public_key.pem", "wb") as f:
18         f.write(public_key.public_bytes(
19             encoding=serialization.Encoding.PEM,
20             format=serialization.PublicFormat.SubjectPublicKeyInfo
21         ))
22
23 if __name__ == "__main__":
24     main()

```

- Trong folder “dh\_key\_pair” tạo file “client.py”.

```

client.py  X
lab-04 > dh_key_pair > client.py > ...
1 from cryptography.hazmat.primitives.asymmetric import dh
2 from cryptography.hazmat.primitives import serialization
3 from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
4 from cryptography.hazmat.primitives import hashes
5
6 def generate_client_key_pair(parameters):
7     private_key = parameters.generate_private_key()
8     public_key = private_key.public_key()
9     return private_key, public_key
10
11 def derive_shared_secret(private_key, server_public_key):
12     shared_key = private_key.exchange(server_public_key)
13     return shared_key
14
15 def main():
16     # Load server's public key
17     with open("server_public_key.pem", "rb") as f:
18         server_public_key = serialization.load_pem_public_key(f.read())
19
20     parameters = server_public_key.parameters()
21     private_key, public_key = generate_client_key_pair(parameters)
22
23     shared_secret = derive_shared_secret(private_key, server_public_key)
24
25     print("Shared Secret:", shared_secret.hex())
26
27 if __name__ == "__main__":
28     main()

```

- Kiểm tra ứng dụng:

- Bước 1: Chạy file “server.py”. Chờ server tạo server\_public\_key và được lưu thành file server\_public\_key.pem.

```
python .\server.py
```

```
File Edit Selection View Go Run Terminal Help
EXPLORER ... 🔍 server_public_key.pem U
lab-04 > dh_key_pair > 🔑 server_public_key.pem
1 -----BEGIN PUBLIC KEY-----
2 MIICJTCARCCGCSqGSIb3DQEDEATCAQgCggEBAJuw8X1YVrxQWQKhZkv5vBkcPH9s
3 qH/pVV0E8amd2NUKtCe+rmjaxZEIMtbrf1vh5zWLG9tBdiRe+oWBkIzaOGD/5eku
4 k4nJ3jqAFzek1YjaPfxZ9qh1xJrJyyalGmL9eCeFz9ClqlcYy9XGf4rt+VL2t0Qy0
5 4y+MiNP6JNWQ8+j1RgWUEB/rNKYL36TjQ+n/+vxoR+RrISM82ZTdShX1P1JnTjTQ
6 Wy15xnqEMhLyYEsgD7ocHKAJsq1AFXRIVk9r6bKc84KEGLPs95n4IS7f+v/0EIhd
7 qGdD/qKnuvcl1q0NzzqcjCrQ74/7W7JChy+f+SBHCKPK77ekumSwcR18CAQID
8 ggEGAACQAEmcmjFlQV5oyNxwcKUbxAOhZV711b30ZyJv3YsiAGC3rqIeMDMKc
9 np4dfxeu28mySAxsMsM1MEpBGAv0w7TFj3zx4mQ1KU1p84ytFTNhnn4pcC2uLS
10 HzEkS9x/qpm354epPmeli4NyXNDL+I1odlhdLro/fY6ze8tuI6G2y5+NAL4DLvGm
11 WwNXiNMYQ1ef0yUhtMbpr60LvXinStav3mo2H8CX1iT+3fFoSx7f6G07DvXkkOR
12 Fzrc5rk1x6ETS/rToDzw+u0efJ5qlp7XiVo1hbpbjZrooxe5sBa493CqqmV1L4JK
13 ZQ0DDG9BLEzhDxrI0UDQ1E+U1+6adnBcw==
14 -----END PUBLIC KEY-----
15
```

- Bước 2: Chạy file “client.py”. Client sẽ đọc và hiển thị nội dung thông điệp bí mật được chia sẻ.

```
python .\client.py
```

```
Terminal Help
server_public_key.pem U
lab-04 > dh_key_pair > 🔑 server_public_key.pem
1 -----BEGIN PUBLIC KEY-----
2 MIICJTCARCCGCSqGSIb3DQEDEATCAQgCggEBAJuw8X1YVrxQWQKhZkv5vBkcPH9s
3 qH/pVV0E8amd2NUKtCe+rmjaxZEIMtbrf1vh5zWLG9tBdiRe+oWBkIzaOGD/5eku
4 k4nJ3jqAFzek1YjaPfxZ9qh1xJrJyyalGmL9eCeFz9ClqlcYy9XGf4rt+VL2t0Qy0
5 4y+MiNP6JNWQ8+j1RgWUEB/rNKYL36TjQ+n/+vxoR+RrISM82ZTdShX1P1JnTjTQ
6 Wy15xnqEMhLyYEsgD7ocHKAJsq1AFXRIVk9r6bKc84KEGLPs95n4IS7f+v/0EIhd
7 qGdD/qKnuvcl1q0NzzqcjCrQ74/7W7JChy+f+SBHCKPK77ekumSwcR18CAQID
8 ggEGAACQAEmcmjFlQV5oyNxwcKUbxAOhZV711b30ZyJv3YsiAGC3rqIeMDMKc
9 np4dfxeu28mySAxsMsM1MEpBGAv0w7TFj3zx4mQ1KU1p84ytFTNhnn4pcC2uLS
10 HzEkS9x/qpm354epPmeli4NyXNDL+I1odlhdLro/fY6ze8tuI6G2y5+NAL4DLvGm
11 WwNXiNMYQ1ef0yUhtMbpr60LvXinStav3mo2H8CX1iT+3fFoSx7f6G07DvXkkOR
12 Fzrc5rk1x6ETS/rToDzw+u0efJ5qlp7XiVo1hbpbjZrooxe5sBa493CqqmV1L4JK
13 ZQ0DDG9BLEzhDxrI0UDQ1E+U1+6adnBcw==
14 -----END PUBLIC KEY-----
15
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\dh_key_pair> python .\server.py
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\dh_key_pair>
```

```
Terminal Help
server_public_key.pem U
lab-04 > dh_key_pair > 🔑 server_public_key.pem
1 -----BEGIN PUBLIC KEY-----
2 MIICJTCARCCGCSqGSIb3DQEDEATCAQgCggEBAJuw8X1YVrxQWQKhZkv5vBkcPH9s
3 qH/pVV0E8amd2NUKtCe+rmjaxZEIMtbrf1vh5zWLG9tBdiRe+oWBkIzaOGD/5eku
4 k4nJ3jqAFzek1YjaPfxZ9qh1xJrJyyalGmL9eCeFz9ClqlcYy9XGf4rt+VL2t0Qy0
5 4y+MiNP6JNWQ8+j1RgWUEB/rNKYL36TjQ+n/+vxoR+RrISM82ZTdShX1P1JnTjTQ
6 Wy15xnqEMhLyYEsgD7ocHKAJsq1AFXRIVk9r6bKc84KEGLPs95n4IS7f+v/0EIhd
7 qGdD/qKnuvcl1q0NzzqcjCrQ74/7W7JChy+f+SBHCKPK77ekumSwcR18CAQID
8 ggEGAACQAEmcmjFlQV5oyNxwcKUbxAOhZV711b30ZyJv3YsiAGC3rqIeMDMKc
9 np4dfxeu28mySAxsMsM1MEpBGAv0w7TFj3zx4mQ1KU1p84ytFTNhnn4pcC2uLS
10 HzEkS9x/qpm354epPmeli4NyXNDL+I1odlhdLro/fY6ze8tuI6G2y5+NAL4DLvGm
11 WwNXiNMYQ1ef0yUhtMbpr60LvXinStav3mo2H8CX1iT+3fFoSx7f6G07DvXkkOR
12 Fzrc5rk1x6ETS/rToDzw+u0efJ5qlp7XiVo1hbpbjZrooxe5sBa493CqqmV1L4JK
13 ZQ0DDG9BLEzhDxrI0UDQ1E+U1+6adnBcw==
14 -----END PUBLIC KEY-----
15
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\dh_key_pair> cd ..\lab-04\dh_key_pair>
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\dh_key_pair> python .\client.py
Shared Secret: 3b0f052a85800bc07a03c0e98f26af6f2de9e0374605cd819f4834ab01d62e1edf3318eced083e4c9c
26fed6b06e5448551ef9453e92ee6c20b2cb47e0673d607876b96d0fd9289debae21462c323b8f024fed10511858ed2c
3ce532cd075a1394ea3eb4a1fc5ccbfb08d4b4aa600e64c1b2717c9372ec96536a8adbf91d087515487a2ddc5cd2389c56
50a5335397ef7f5fbfa227feee94ebbbdf152c341b334fff499b8a0d13b5c5535a8bc5834a68f264cef2144e46d0d17866
baef3a27c0bc2d9e1fe53a847fc9e47335f1f5b9c980dca8257891e333e70ef3af65233974c72c42ac9fb755d597f5aae
32696e10a27f47ca614b4
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\dh_key_pair>
```

- Commit code:

```
git add .
git commit -m "[add] lab-04 dh_key_pair"
```

### 4.6.3 Bài thực hành 03: Hàm băm

Tạo ứng dụng dòng lệnh để thực hiện các hàm băm: MD5, SHA-256, SHA-3, Blake2.

Hướng dẫn:

- Trong folder “lab-04” tạo folder “hash”. Trong folder “hash” tạo file “md5\_hash.py” để thực hiện hàm băm MD5 không dùng thư viện.

```
md5_hash.py ×
lab-04 > hash > md5_hash.py > md5
1 def left_rotate(value, shift):
2     return ((value << shift) | (value >> (32 - shift))) & 0xFFFFFFFF
```

```
4 def md5(message):
5     # Khởi tạo các biến ban đầu
6     a = 0x67452301
7     b = 0xEFCDAB89
8     c = 0x98BADCFE
9     d = 0x10325476
10
11    # Tiền xử lý chuỗi văn bản
12    original_length = len(message)
13    message += b'\x80'
14    while len(message) % 64 != 56:
15        message += b'\x00'
16    message += original_length.to_bytes(8, 'little')
17
18    # Chia chuỗi thành các block 512-bit
19    for i in range(0, len(message), 64):
20        block = message[i:i+64]
21
22        words = [int.from_bytes(block[j:j+4], 'little') for j in range(0, 64, 4)]
23
24        a0, b0, c0, d0 = a, b, c, d
25
26        # Vòng lặp chính của thuật toán MD5
27        for j in range(64):
28            if j < 16:
29                f = (b & c) | ((~b) & d)
30                g = j
31            elif j < 32:
32                f = (d & b) | ((~d) & c)
33                g = (5*j + 1) % 16
```

```

34     elif j < 48:
35         f = b ^ c ^ d
36         g = (3*j + 5) % 16
37     else:
38         f = c ^ (b | (~d))
39         g = (7*j) % 16
40
41     temp = d
42     d = c
43     c = b
44     b = b + left_rotate((a + f + 0x5A827999 + words[g]) & 0xFFFFFFFF, 3)
45     a = temp
46
47     a = (a + a0) & 0xFFFFFFFF
48     b = (b + b0) & 0xFFFFFFFF
49     c = (c + c0) & 0xFFFFFFFF
50     d = (d + d0) & 0xFFFFFFFF
51
52     return '{:08x}{:08x}{:08x}{:08x}'.format(a, b, c, d)
53
54 input_string = input("Nhập chuỗi cần băm: ")
55 md5_hash = md5(input_string.encode('utf-8'))
56
57 print("Mã băm MD5 của chuỗi '{}' là: {}".format(input_string, md5_hash))

```

- Kiểm tra bằng cách chạy file "md5\_hash.py".

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\hash> python .\md5_hash.py
Nhập chuỗi cần băm: hutech university
Mã băm MD5 của chuỗi 'hutech university' là: d78961379bf78cd5b3f89000ca12f01a
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\hash> python .\md5_hash.py
Nhập chuỗi cần băm: nguyen trong minh hong phuoc
Mã băm MD5 của chuỗi 'nguyen trong minh hong phuoc' là: a61ef8b56d9fe0dd786df7aded16ce13
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\hash>

```

- Trong folder "hash" tạo file "md5\_library.py" để thực hiện hàm băm MD5 sử dụng thư viện có sẵn.

```

md5_library.py ×
lab-04 > hash > md5_library.py > ...
1 import hashlib
2
3 def calculate_md5(input_string):
4     md5_hash = hashlib.md5()
5     md5_hash.update(input_string.encode('utf-8'))
6     return md5_hash.hexdigest()
7
8 input_string = input("Nhập chuỗi cần băm: ")
9 md5_hash = calculate_md5(input_string)
10
11 print("Mã băm MD5 của chuỗi '{}' là: {}".format(input_string, md5_hash))

```

- Kiểm tra bằng cách chạy file "md5\_library.py".

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\hash> python .\md5_library.py
Nhập chuỗi cần băm: cong nghe thong tin hutech
Mã băm MD5 của chuỗi 'cong nghe thong tin hutech' là: fc727295ede53a235489a0b62a6be8cd
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\hash> python .\md5_library.py
Nhập chuỗi cần băm: xin chao cac ban toi la sinh vien hutech
Mã băm MD5 của chuỗi 'xin chao cac ban toi la sinh vien hutech' là: c8dd44c9e891eb4439c1e4e19873cf86
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\hash>

```

- Trong folder "hash" tạo file "sha-256.py" để thực hiện hàm băm SHA-256 sử dụng thư viện có sẵn.

```

sha-256.py ×

lab-04 > hash > sha-256.py > ...
1 import hashlib
2
3 def calculate_sha256_hash(data):
4     sha256_hash = hashlib.sha256()
5     sha256_hash.update(data.encode('utf-8')) # Chuyển đổi dữ liệu thành bytes
6     và cập nhật vào đối tượng hash
7     return sha256_hash.hexdigest() # Trả về biểu diễn hex chuỗi hash
8
9 data_to_hash = input("Nhập dữ liệu để hash bằng SHA-256: ")
10 hash_value = calculate_sha256_hash(data_to_hash)
11 print("Giá trị hash SHA-256:", hash_value)

```

- Kiểm tra bằng cách chạy file "sha-256.py".

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\hash> python .\sha-256.py
Nhập dữ liệu để hash bằng SHA-256: hutech university
Giá trị hash SHA-256: f36e7b18473931d91715be3b1dd3a5bdd2fc1bc53a8ed867440d4b29c1905506
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\hash> python .\sha-256.py
Nhập dữ liệu để hash bằng SHA-256: nguyen trong minh hong phuoc
Giá trị hash SHA-256: 1b446456b7f78174c76e763437854d5d0f3ac7054c0023546d00f1a867758564
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\hash>

```

- Trong folder "hash" tạo file "sha-3.py" để thực hiện hàm băm SHA-3 sử dụng thư viện có sẵn.

```
sha-3.py  X
lab-04 > hash > sha-3.py > ...
1  from Crypto.Hash import SHA3_256
2
3  def sha3(message):
4      sha3_hash = SHA3_256.new()
5      sha3_hash.update(message)
6      return sha3_hash.digest()
7
8  def main():
9      text = input("Nhập chuỗi văn bản: ").encode('utf-8')
10     hashed_text = sha3(text)
11
12     print("Chuỗi văn bản đã nhập:", text.decode('utf-8'))
13     print("SHA-3 Hash:", hashed_text.hex())
14
15 if __name__ == "__main__":
16     main()
```

- Kiểm tra bằng cách chạy file “sha-3.py”.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\PHUOCNTMH\bmmtt-nc-hutech-1511060249\lab-04\hash> python .\sha-3.py
Nhập chuỗi văn bản: hutech university
Chuỗi văn bản đã nhập: hutech university
SHA-3 Hash: f94e5d66bef9f7be64c5a78c781f6b8c055715c0d8628999790869a283866077
PS C:\Users\PHUOCNTMH\bmmtt-nc-hutech-1511060249\lab-04\hash> python .\sha-3.py
Nhập chuỗi văn bản: nguyen trong minh hong phuoc
Chuỗi văn bản đã nhập: nguyen trong minh hong phuoc
SHA-3 Hash: f6da21fad86c5bae1bdc05c0197dd1f7e756d819896af338fd38e4243a1dd9af
PS C:\Users\PHUOCNTMH\bmmtt-nc-hutech-1511060249\lab-04\hash>
```

- Trong folder “hash” tạo file “blake2.py” để thực hiện hàm băm Blake2 sử dụng thư viện có sẵn.

```
blake2.py  X
lab-04 > hash > blake2.py > ...
1  import hashlib
2
3  def blake2(message):
4      blake2_hash = hashlib.blake2b(digest_size=64)
5      blake2_hash.update(message)
6      return blake2_hash.digest()
7
8  def main():
9      text = input("Nhập chuỗi văn bản: ").encode('utf-8')
10     hashed_text = blake2(text)
11
12     print("Chuỗi văn bản đã nhập:", text.decode('utf-8'))
13     print("BLAKE2 Hash:", hashed_text.hex())
14
15 if __name__ == "__main__":
16     main()
```

- Kiểm tra bằng cách chạy file “blake2.py”.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\hash> python .\blake2.py
Nhập chuỗi văn bản: hutech university
Chuỗi văn bản đã nhập: hutech university
BLAKE2 Hash: 12d7347881b1e8da4bd4bca2832b25f3551e0f2571201febbd1572c41169e0d08a0df343a2950acdf1884089a69f8b4c3eedf526f3a926cd8d465a2594925e2d
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\hash> python .\blake2.py
Nhập chuỗi văn bản: nguyen trong minh hong phuoc
Chuỗi văn bản đã nhập: nguyen trong minh hong phuoc
BLAKE2 Hash: 9545f211b44f21e17630e9a4ef642bc626d3997541fcde4c98ec4f8bc69b1e69b244d5274b31de6b166dd8604faee75299f76b3632392817b25ae46cc2ee9d40
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\hash>
```

- Commit code:

```
git add .
git commit -m "[add] lab-04 hash"
```

#### 4.6.4 Bài thực hành 04: WebSocket và Tornado

Sử dụng Tornado để tạo một WebSocket cho chương trình giao tiếp client-server (streaming data), cứ mỗi 3 giây server sẽ gửi thông tin là tên một loại trái cây, các client sẽ nhận thông tin này và in ra màn hình.

Hướng dẫn:

- Trong folder “lab-04” tạo folder “websocket”. Trong folder “websocket” tạo file “requirements.txt”.

```
☰ requirements.txt ×
lab-04 > websocket > ☰ requirements.txt
  1  tornado
```

- Vào Terminal, chạy các lệnh sau:

```
cd .\lab-04\websocket\
pip install -r .\requirements.txt
```

- Trong folder “websocket” tạo file “server.py”.

```
server.py  X
lab-04 > websocket > server.py > main
1 import random
2 import tornado.ioloop
3 import tornado.web
4 import tornado.websocket
5
6 class WebSocketServer(tornado.websocket.WebSocketHandler):
7     clients = set()
8
9     def open(self):
10         WebSocketServer.clients.add(self)
11
12     def on_close(self):
13         WebSocketServer.clients.remove(self)
14
15     @classmethod
16     def send_message(cls, message: str):
17         print(f"Sending message {message} to {len(cls.clients)} client(s).")
18         for client in cls.clients:
19             client.write_message(message)
20
21 class RandomWordSelector:
22     def __init__(self, word_list):
23         self.word_list = word_list
24
25     def sample(self):
26         return random.choice(self.word_list)
27
```

```
28 def main():
29     app = tornado.web.Application(
30         [(r"/websocket/", WebSocketServer)],
31         websocket_ping_interval=10,
32         websocket_ping_timeout=30,
33     )
34     app.listen(8888)
35
36     io_loop = tornado.ioloop.IOLoop.current()
37
38     word_selector = RandomWordSelector(['apple', 'banana', 'orange', 'grape',
39                                         'melon'])
40
41     periodic_callback = tornado.ioloop.PeriodicCallback(
42         lambda: WebSocketServer.send_message(word_selector.sample()), 3000
43     )
44     periodic_callback.start()
45
46     io_loop.start()
47
48 if __name__ == "__main__":
49     main()
```

- Trong folder “websocket” tạo file “client.py”.

```

client.py  X

lab-04 > websocket > client.py > WebSocketClient > on_message
1 import tornado.ioloop
2 import tornado.websocket
3
4 class WebSocketClient:
5     def __init__(self, io_loop):
6         self.connection = None
7         self.io_loop = io_loop
8
9     def start(self):
10        self.connect_and_read()
11
12    def stop(self):
13        self.io_loop.stop()
14
15    def connect_and_read(self):
16        print("Reading...")
17        tornado.websocket.websocket_connect(
18            url=f"ws://localhost:8888/websocket/",
19            callback=self.maybe_retry_connection,
20            on_message_callback=self.on_message,
21            ping_interval=10,
22            ping_timeout=30,
23        )
24

25    def maybe_retry_connection(self, future) -> None:
26        try:
27            self.connection = future.result()
28        except:
29            print("Could not reconnect, retrying in 3 seconds...")
30            self.io_loop.call_later(3, self.connect_and_read)
31

32    def on_message(self, message):
33        if message is None:
34            print("Disconnected, reconnecting...")
35            self.connect_and_read()
36            return
37
38        print(f"Received word from server: {message}")
39
40        self.connection.read_message(callback=self.on_message)
41
42    def main():
43        io_loop = tornado.ioloop.IOLoop.current()
44
45        client = WebSocketClient(io_loop)
46        io_loop.add_callback(client.start)
47
48        io_loop.start()
49
50    if __name__ == "__main__":
51        main()

```

- Kiểm tra ứng dụng:

- Bước 1: Split Terminal, chạy file “server.py”.

```
python .\server.py
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\websocket> python .\server.py
Sending message grape to 0 client(s).
Sending message grape to 0 client(s).
Sending message melon to 0 client(s).

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\websocket>
+ ⌂ ... ⌄ x
r [ ] python
l [ ] powershell
```

- Bước 2: Chạy file “client.py”.

```
python .\client.py
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\websocket> python .\server.py
Sending message grape to 0 client(s).
Sending message grape to 0 client(s).
Sending message melon to 0 client(s).
Sending message orange to 0 client(s).
Sending message apple to 0 client(s).
Sending message grape to 0 client(s).
Sending message grape to 0 client(s).
Sending message banana to 0 client(s).
Sending message banana to 0 client(s).
Sending message banana to 0 client(s).
Sending message grape to 0 client(s).
Sending message grape to 0 client(s).
Sending message grape to 0 client(s).
Sending message orange to 0 client(s).
Sending message apple to 0 client(s).
Sending message apple to 0 client(s).
Sending message banana to 1 client(s).
Sending message orange to 1 client(s).
Sending message grape to 1 client(s).
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\websocket> python .\client.py
Reading...
Received word from server: banana
Received word from server: orange
Received word from server: grape

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\websocket>
+ ⌂ ... ⌄ x
r [ ] python
l [ ] python
```

### Bước 3: Chạy thêm một “client.py”.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\websocket> python .\client.py
Reading...
Received word from server: banana
Received word from server: orange
Received word from server: grape
Received word from server: melon
Received word from server: banana
Received word from server: melon
Received word from server: banana
Received word from server: orange
Received word from server: melon
Received word from server: orange
Received word from server: grape
Received word from server: orange
Received word from server: melon
Received word from server: banana
Received word from server: grape
Received word from server: orange
Received word from server: grape
Received word from server: orange
Received word from server: banana
Received word from server: grape
Received word from server: orange
Received word from server: grape
Received word from server: orange
Received word from server: melon
Received word from server: orange
Received word from server: grape
Received word from server: orange
Received word from server: grape
Received word from server: orange
Received word from server: banana
Received word from server: grape
Received word from server: apple
Received word from server: apple

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249> cd .\lab-04\websocket>
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\websocket> python .\client.py
Reading...
Received word from server: orange
Received word from server: banana
Received word from server: grape
Received word from server: apple
Received word from server: apple

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-04\websocket>
+ ⌂ ... ⌄ x
r [ ] python
l [ ] python
u [ ] python
```

- Commit code:

```
git add .
git commit -m "[add] lab-04 tornado websocket"
```

- Push các thay đổi lên remote repo:

```
git push origin lab04
```

- Tiến hành tạo PR từ nhánh lab04 về nhánh main. Review, kiểm tra đủ số lượng file, sau đó tiến hành “Merge pull request”.

The screenshot shows a GitHub pull request merge history. At the top, it says "Merged" by "mrphuoc020597" merging 4 commits from "lab04" into "main" now. Below this, there are tabs for Conversation (0), Commits (4), Checks (0), and Files changed (15). The commit list shows:

- mrphuoc020597 commented now: "No description provided."
- mrphuoc020597 added 4 commits 2 hours ago:
  - [add] lab-04 aes\_rsa\_socket 01c300b
  - [add] lab-04 dh\_key\_pair ad74081
  - [add] lab-04 hash 97a88bd
  - [add] lab-04 tornado websocket 99271e1
- mrphuoc020597 merged commit 46ea034 into main now

At the bottom, a message says "Pull request successfully merged and closed" and "You're all set—the lab04 branch can be safely deleted." There is also a "Delete branch" button.

## 4.7 BÀI TẬP MỞ RỘNG

- **Câu 01:** Xây dựng một giao diện UI desktop cho Bài thực hành 01.
- **Câu 02:** Xây dựng một giao diện UI desktop cho Bài thực hành 02.
- **Câu 03:** Xây dựng một giao diện UI desktop cho Bài thực hành 03.
- **Câu 04:** Sử dụng Tornado để tạo một WebSocket cho chương trình giao tiếp client-server. Client sẽ gửi thông điệp đến server. Server sẽ mã hoá thông điệp đó bằng AES và gửi kết quả về client.

# BÀI 5: ỨNG DỤNG BẢO MẬT

Trong bài thực hành này, chúng ta sẽ tập trung vào những khái niệm cơ bản liên quan đến Base64 - một phương pháp mã hóa dùng để chuyển đổi dữ liệu nhị phân thành chuỗi văn bản, SSL (Secure Sockets Layer) - giao thức bảo mật giúp thiết lập kết nối mạng an toàn, Blockchain - công nghệ cho phép lưu trữ và xác minh thông tin một cách an toàn trong mạng phân tán, và Steganography - kỹ thuật giấu tin trong các phương tiện truyền thông như hình ảnh, video, văn bản, hoặc âm thanh mà khó bị phát hiện. Chúng ta sẽ tìm hiểu cách thức hoạt động, đặc điểm của từng phương pháp và ứng dụng của chúng trong việc bảo vệ thông tin và dữ liệu.

## 5.1 BASE64

---

Base64 là một chuẩn mã hóa cơ bản giúp biểu diễn dữ liệu nhị phân dưới dạng văn bản ASCII. Phương pháp này được sử dụng để chuyển đổi các dữ liệu nhị phân không thể đọc thành dạng văn bản có thể hiểu được và ngược lại, từ đó giúp việc truyền tải và lưu trữ thông tin trong các hệ thống không hỗ trợ dữ liệu nhị phân trở nên dễ dàng hơn.

Quá trình mã hóa Base64 chia dữ liệu nhị phân thành các nhóm 6 bit, sau đó mã hóa mỗi nhóm thành một ký tự ASCII tương ứng. Base64 thường được áp dụng trong các trường hợp như: đính kèm email, mã hóa URL và lưu trữ dữ liệu nhị phân trong các định dạng văn bản.

Mặc dù mã hóa Base64 không phải là một phương pháp bảo mật, vì nó không mã hóa dữ liệu mà chỉ chuyển đổi dữ liệu từ định dạng này sang định dạng khác, nhưng nó rất hữu ích trong việc bảo vệ dữ liệu khỏi việc mất mát trong quá trình truyền tải, cũng như trong việc lưu trữ dữ liệu nhị phân trong các hệ thống không hỗ trợ trực tiếp.

## 5.2 SSL

---

SSL (Secure Sockets Layer) là công nghệ mã hóa dữ liệu nhằm đảm bảo an toàn cho thông tin khi được truyền tải qua internet. Công nghệ này được phát triển để

cung cấp lớp bảo mật bổ sung cho các kết nối mạng, đặc biệt là trong việc bảo vệ liên kết giữa máy khách và máy chủ trên các ứng dụng web.

SSL hoạt động bằng cách mã hóa dữ liệu trong quá trình truyền qua internet, thường sử dụng mã hóa đối xứng như RSA để mã hóa dữ liệu trước khi gửi và giải mã khi dữ liệu đến đích. Quá trình này đóng vai trò quan trọng trong việc ngăn chặn các cuộc tấn công, giúp bảo vệ thông tin khỏi bị đọc hoặc can thiệp khi đang truyền tải.

SSL đã được nâng cấp thành TLS (Transport Layer Security), một phiên bản tiên tiến hơn của SSL. TLS hiện nay được áp dụng phổ biến trong ngành công nghiệp công nghệ thông tin. Khi người dùng truy cập vào một trang web có giao thức "https://" ("s" → "secure"), điều này chứng tỏ rằng trang web đó đang sử dụng SSL hoặc TLS để đảm bảo an toàn cho giao tiếp giữa máy khách (client) và máy chủ (server).

Cơ chế hoạt động của SSL (và TLS) bao gồm các bước chính sau:

- **Bắt đầu phiên kết nối (Handshake):** Khi một máy khách kết nối đến một máy chủ thông qua giao thức bảo mật SSL, quá trình bắt đầu bằng một bước gọi là "handshake".
- **Xác thực (Authentication):** Máy chủ gửi chứng chỉ số công khai của mình đến máy khách để xác minh danh tính của mình. Máy khách kiểm tra chứng chỉ để đảm bảo rằng máy chủ được xác thực và có thể tin cậy.
- **Trao đổi khóa (Key Exchange):** Máy khách và máy chủ sử dụng một loạt các thuật toán để thống nhất và trao đổi thông tin về khóa mã hóa.
- **Mã hóa và Giải mã (Encryption and Decryption):** Sau khi các thông tin khóa được trao đổi, máy khách và máy chủ sử dụng thông tin này để mã hóa và giải mã dữ liệu được truyền qua kết nối.
- **Kiểm tra Tính toàn vẹn (Integrity Check):** SSL cũng cung cấp kiểm tra tính toàn vẹn dữ liệu. Mỗi gói tin được gửi đi đều được đính kèm mã xác thực để ngăn chặn việc dữ liệu bị sửa đổi khi truyền qua mạng.

Quá trình này cung cấp một kết nối bảo mật giữa máy khách và máy chủ, đảm bảo rằng dữ liệu được truyền qua internet không bị lộ thông tin và không bị thay đổi trên đường truyền.

## 5.3 BLOCKCHAIN

Blockchain là một công nghệ lưu trữ và truyền tải thông tin phi tập trung, được thiết kế để ghi lại các giao dịch và dữ liệu một cách an toàn, không thể thay đổi và minh bạch. Công nghệ này không chỉ được áp dụng trong lĩnh vực tài chính mà còn có thể được sử dụng trong nhiều lĩnh vực khác như y tế, chuỗi cung ứng, quản lý tài sản, bầu cử điện tử và nhiều lĩnh vực khác. Blockchain mang lại sự minh bạch và an toàn trong quá trình giao dịch cũng như lưu trữ dữ liệu. Các đặc điểm quan trọng của blockchain bao gồm:

- Phi tập trung (Decentralization): Thay vì có một trung tâm kiểm soát duy nhất, dữ liệu trên blockchain được phân tán trên nhiều nút (nodes) khác nhau trên mạng.
- Mã hoá và bảo mật: Dữ liệu trên blockchain được bảo vệ bằng mã hoá, giúp đảm bảo tính toàn vẹn và an toàn của thông tin.
- Giao dịch phi tập trung và minh bạch: Blockchain cho phép các giao dịch diễn ra một cách minh bạch và không cần đến sự can thiệp của bên thứ ba.
- Smart contracts (Hợp đồng thông minh): Blockchain cũng có khả năng thực hiện các hợp đồng thông minh tự động dựa trên các điều khoản đã được định sẵn, giúp tiết kiệm thời gian và chi phí trong việc thực hiện các giao dịch.

## 5.4 STEGANOGRAPHY

Steganography là một phương pháp ẩn tin và che giấu thông tin bên trong dữ liệu khác một cách không dễ dàng để phát hiện. Khác với mã hóa thông tin, Steganography không chỉ mã hóa dữ liệu mà còn ẩn nó vào các phương tiện khác một cách khá tinh vi, chẳng hạn như ẩn tin nhắn trong hình ảnh, video, file âm thanh, hoặc bất kỳ loại tệp tin nào khác mà không làm thay đổi quá nhiều nội dung gốc. Các phương pháp Steganography bao gồm:

- Ẩn tin trong hình ảnh: Đây là phương pháp phổ biến, trong đó thông tin được ẩn dưới dạng bit hoặc đổi màu sắc, không gây thay đổi rõ ràng đối với hình ảnh.
- Ẩn tin trong file âm thanh hoặc video: Tương tự như hình ảnh, dữ liệu có thể được ẩn trong các file âm thanh hoặc video bằng cách thêm thông tin không nghe thấy hoặc không nhìn thấy.

- Ẩn tin trong văn bản hoặc tập tin khác: Steganography cũng có thể được thực hiện bằng cách chèn thông tin vào các tập tin văn bản hoặc các loại tệp tin khác thông qua các kỹ thuật như việc thay đổi khoảng trắng, thêm ký tự ẩn, hoặc sử dụng kỹ thuật mã hóa dữ liệu để che giấu thông tin.

## 5.5 BÀI TẬP THỰC HÀNH

### 5.5.1 Bài thực hành 01: Base64

Viết chương trình mã hoá và giải mã thông tin sử dụng Base64.

Hướng dẫn:

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

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

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

```
git pull origin main
```

The screenshot shows the VS Code interface with the 'TERMINAL' tab selected. The terminal window displays the following command and its execution:

```
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 lab05 từ nhánh main.

```
git checkout -b lab05
```

- Tạo folder "lab-05". Trong folder "lab-05" tạo folder "base64".
- Trong folder "base64" tạo file "encrypt.py".

```

1 import base64
2
3 def main():
4     input_string = input("Nhập thông tin cần mã hóa: ")
5
6     encoded_bytes = base64.b64encode(input_string.encode("utf-8"))
7     encoded_string = encoded_bytes.decode("utf-8")
8
9     with open("data.txt", "w") as file:
10         file.write(encoded_string)
11
12     print("Đã mã hóa và ghi vào tệp data.txt")
13
14 if __name__ == "__main__":
15     main()

```

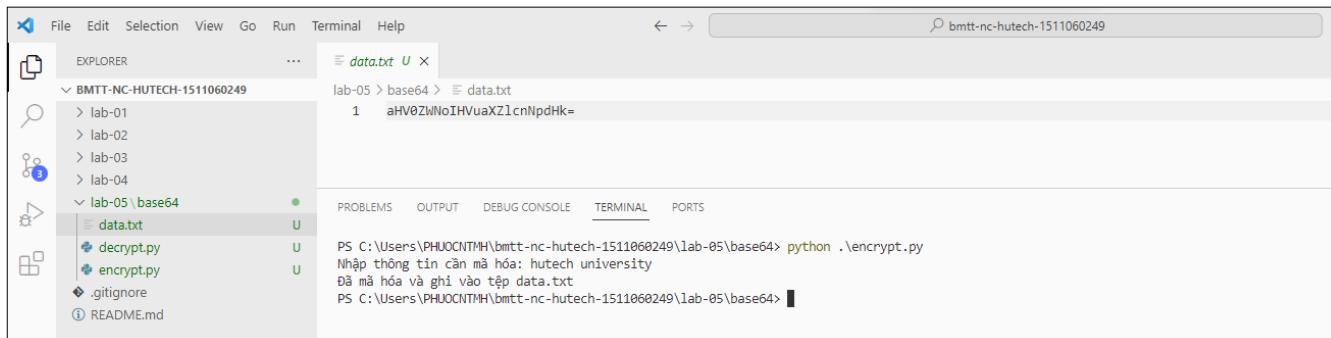
- Trong folder “base64” tạo file “decrypt.py”.

```

1 import base64
2
3 def main():
4     try:
5         with open("data.txt", "r") as file:
6             encoded_string = file.read().strip()
7
8             decoded_bytes = base64.b64decode(encoded_string)
9             decoded_string = decoded_bytes.decode("utf-8")
10
11         print("Chuỗi sau khi giải mã:", decoded_string)
12     except Exception as e:
13         print("Lỗi:", e)
14
15 if __name__ == "__main__":
16     main()

```

- Kiểm tra chương trình, chạy file “encrypt.py”.



- Chạy file "decrypt.py".

```

File Edit Selection View Go Run Terminal Help
EXPLORER ... data.txt U
BMTT-NC-HUTECH-1511060249 lab-05 > base64 > data.txt
1 aHV0ZWN0IHZlcnNpdHk=
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\base64> python .\decrypt.py
Nhập thông tin cần mã hóa: hutech university
Đã mã hóa và ghi vào tập data.txt
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\base64> python .\decrypt.py
Chuỗi sau khi giải mã: hutech university
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\base64>

```

- Commit code:

```

git add .
git commit -m "[add] lab-05 base64"

```

## 5.5.2 Bài thực hành 02: SSL

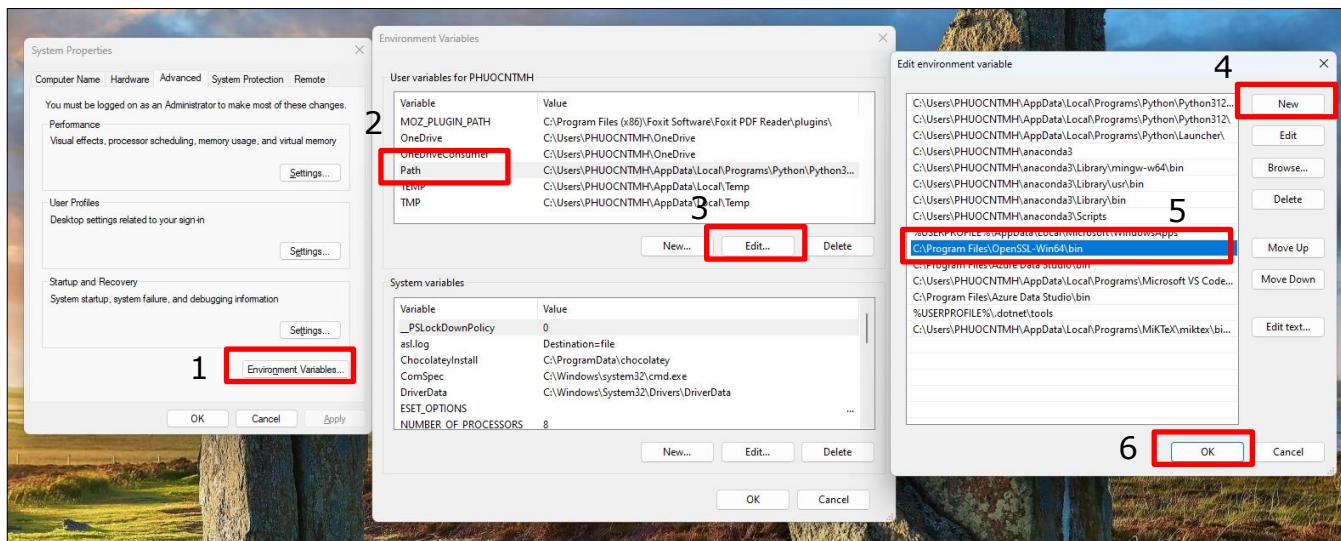
Viết chương trình demo cho việc kết nối và gửi dữ liệu từ client lên server thông qua SSL (được tạo từ OpenSSL).

- Tải và cài đặt OpenSSL tại liên kết sau:

["https://slproweb.com/products/Win32OpenSSL.html"](https://slproweb.com/products/Win32OpenSSL.html)

Download Win32/Win64 OpenSSL		
Download Win32/Win64 OpenSSL today using the links below!		
File	Type	Description
Win64 OpenSSL v3.2.0 Light <a href="#">EXE</a>   <a href="#">MSI</a>	5MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.2.0 (Recommended for users by the creators of <a href="#">OpenSSL</a> ) this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement.
Win64 OpenSSL v3.2.0 <a href="#">EXE</a>   <a href="#">MSI</a>	200MB Installer	Installs Win64 OpenSSL v3.2.0 (Recommended for software developers by the creators of <a href="#">OpenSSL</a> ). Only installs on 64-bit OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v3.2.0 Light <a href="#">EXE</a>   <a href="#">MSI</a>	4MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v3.2.0 (Only install this if you need 32-bit OpenSSL for Windows). More information can be found in the legal agreement of the installation.
Win32 OpenSSL v3.2.0 <a href="#">EXE</a>   <a href="#">MSI</a>	162MB Installer	Installs Win32 OpenSSL v3.2.0 (Only install this if you need 32-bit OpenSSL for Windows). Note that this is a default build of legal agreement of the installation.
Win64 OpenSSL v3.2.0 Light for ARM (EXPERIMENTAL) <a href="#">EXE</a>   <a href="#">MSI</a>	6MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.2.0 for ARM64 devices (Only install this VERY EXPERIMENTAL that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.2.0 for ARM (EXPERIMENTAL) <a href="#">EXE</a>   <a href="#">MSI</a>	158MB Installer	Installs Win64 OpenSSL v3.2.0 for ARM64 devices (Only install this VERY EXPERIMENTAL build if you want to try 64-bit OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.

- Thêm OpenSSL vào Path của Windows:



- Kiểm tra OpenSSL bằng cách truy cập vào CMD và gõ lệnh:

```
Microsoft Windows [Version 10.0.22621.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PHUOCNTMH>openssl
OpenSSL> |
```

- Trong folder "lab-05" tạo folder "ssl". Trong folder "ssl" tạo folder "certificates".
- Trong folder "certificates" tạo file "server-cert.cnf".

```
1 [req]
2 default_bits = 2048
3 prompt = no
4 default_md = sha256
5 distinguished_name = dn
6
7 [dn]
8 C=US
9 ST=State
10 L=Location
11 O=Organization
12 OU=Organizational Unit
13 CN=localhost
```

- Trong folder "certificates" tạo file "make-cert.bat".

```
1 openssl req -new -x509 -newkey rsa:2048 -nodes -keyout server-key.key -out
server-cert.crt -days 365 -config server-cert.cnf
```

- Chạy file "make-cert.bat", ta thu được hai file "server-cert.crt" và "server-key.key"

The screenshot shows the VS Code interface with the terminal tab active. The command `.\make-cert.bat` is being run in the terminal window. The output shows the OpenSSL command being executed and its long output, followed by the command `PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\ssl\certificates>`.

- Trong folder "ssl" tạo file "server.py".

```
1 import socket
2 import ssl
3 import threading
4
5 # Thông tin server
6 server_address = ('localhost', 12345)
7
8 # Danh sách các client đã kết nối
9 clients = []
10
11 def handle_client(client_socket):
12     # Thêm client vào danh sách
13     clients.append(client_socket)
14
15     print("Đã kết nối với:", client_socket.getpeername())
16
17     try:
18         # Nhận và gửi dữ liệu
19         while True:
20             data = client_socket.recv(1024)
21             if not data:
22                 break
23             print("Nhận:", data.decode('utf-8'))
24
25             # Gửi dữ liệu đến tất cả các client khác
26             for client in clients:
27                 if client != client_socket:
28                     try:
29                         client.send(data)
```

```
30     except:  
31         clients.remove(client)  
32     except:  
33         clients.remove(client_socket)  
34     finally:  
35         print("Đã ngắt kết nối:", client_socket.getpeername())  
36         clients.remove(client_socket)  
37         client_socket.close()  
38  
39 # Tạo socket server  
40 server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
41 server_socket.bind(server_address)  
42 server_socket.listen(5)  
43  
44 print("Server đang chờ kết nối...")  
45
```

```
46 # Lắng nghe các kết nối  
47 while True:  
48     client_socket, client_address = server_socket.accept()  
49  
50     # Tạo SSL context  
51     context = ssl.SSLContext(ssl.PROTOCOL_TLS)
```

```
52     context.load_cert_chain(certfile="./certificates/server-cert.crt",  
53                             keyfile="./certificates/server-key.key")  
54  
55     # Thiết lập kết nối SSL  
56     ssl_socket = context.wrap_socket(client_socket, server_side=True)  
57  
58     # Bắt đầu một luồng xử lý cho mỗi client  
59     client_thread = threading.Thread(target=handle_client, args=  
59                                     (ssl_socket,))  
59     client_thread.start()
```

- Trong folder “ssl” tạo file “client.py”.

```
1 import socket
2 import ssl
3 import threading
4
5 # Thông tin server
6 server_address = ('localhost', 12345)
7
8 def receive_data(ssl_socket):
9     try:
10         while True:
11             data = ssl_socket.recv(1024)
12             if not data:
13                 break
14             print("Nhận:", data.decode('utf-8'))
15     except:
16         pass
17     finally:
18         ssl_socket.close()
19         print("Kết nối đã đóng.")
20
```

```
21 # Tạo socket client
22 client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
23
24 # Tạo SSL context
25 context = ssl.SSLContext(ssl.PROTOCOL_TLS)
26 context.verify_mode = ssl.CERT_NONE # Change this according to your needs
27 context.check_hostname = False # Change this according to your needs
28
```

```
29 # Thiết lập kết nối SSL
30 ssl_socket = context.wrap_socket(client_socket, server_hostname='localhost')
31
32 ssl_socket.connect(server_address)
33
34 # Bắt đầu một luồng để nhận dữ liệu từ server
35 receive_thread = threading.Thread(target=receive_data, args=(ssl_socket,))
36 receive_thread.start()
37
```

```

38 # Gửi dữ liệu lên server
39 try:
40     while True:
41         message = input("Nhập tin nhắn: ")
42         ssl_socket.send(message.encode('utf-8'))
43 except KeyboardInterrupt:
44     pass
45 finally:
46     ssl_socket.close()

```

- Kiểm tra ứng dụng:

- Bước 1: Split Terminal, chạy file "server.py".

```
python .\server.py
```

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249> cd .\lab-05\ssl\
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\ssl> python .\server.py
Server đang chờ kết nối...

```

- Bước 2: Chạy file "client.py". Nhập tin nhắn và kiểm tra.

```
python .\client.py
```

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249> cd .\lab-05\ssl\
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\ssl> python .\server.py
Server đang chờ kết nối...
C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\ssl> python .\client.py
server.py:51: DeprecationWarning: ssl.PROTOCOL_TLS is deprecated
  context = ssl.SSLContext(ssl.PROTOCOL_TLS)
Đã kết nối với: ('127.0.0.1', 11538)
Nhận: hutech
Nhận: xin chào các bạn
Nhận: xin chào các bạn

```

```

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249> cd .\lab-05\ssl\
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\ssl> python .\client.py
C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\ssl> client.py:25: DeprecationWarning: ssl.PROTOCOL_TLS is deprecated
  context = ssl.SSLContext(ssl.PROTOCOL_TLS)
Nhập tin nhắn: hutech
Nhập tin nhắn: xin chào các bạn
Nhập tin nhắn:

```

- Commit code:

```

git add .
git commit -m "[add] lab-05 ssl"

```

### 5.5.3 Bài thực hành 03: Blockchain

Viết chương trình mô phỏng blockchain bằng Python, giữ được các khái niệm cơ bản của blockchain, gồm các khối (blocks), giao dịch (transactions), chứng minh công việc (proof of work).

Hướng dẫn:

- Trong folder “lab-05” tạo folder “blockchain”. Trong folder “blockchain” tạo file “block.py”.

```
1 import hashlib
2 import time
3
4 class Block:
5     def __init__(self, index, previous_hash, timestamp, transactions, proof):
6         :
7         self.index = index
8         self.previous_hash = previous_hash
9         self.timestamp = timestamp
10        self.transactions = transactions
11        self.proof = proof
12        self.hash = self.calculate_hash()
13
14    def calculate_hash(self):
15        data = str(self.index) + str(self.previous_hash) + str(self.
16            timestamp) + str(self.transactions) + str(self.proof)
17        return hashlib.sha256(data.encode()).hexdigest()
```

- Trong folder “blockchain” tạo file “blockchain.py”.

```
1 from block import Block
2 import hashlib
3 import time
4
5 class Blockchain:
6     def __init__(self):
7         self.chain = []
8
9         self.current_transactions = []
10        self.create_block(proof=1, previous_hash='0')
11
12    def create_block(self, proof, previous_hash):
13        block = Block(len(self.chain) + 1, previous_hash, time.time(), self.
14            current_transactions, proof)
15        self.current_transactions = []
16        self.chain.append(block)
17        return block
18
19    def get_previous_block(self):
20        return self.chain[-1]
```

```
20     def proof_of_work(self, previous_proof):
21         new_proof = 1
22         check_proof = False
23         while not check_proof:
24             hash_operation = hashlib.sha256(str(new_proof**2) -
25                                             previous_proof**2).encode()).hexdigest()
26             if hash_operation[:4] == '0000':
27                 check_proof = True
28             else:
29                 new_proof += 1
30         return new_proof
31
32     def add_transaction(self, sender, receiver, amount):
33         self.current_transactions.append({'sender': sender, 'receiver': receiver,
34                                         'amount': amount})
35         return self.get_previous_block().index + 1
36
37
38
39
40
41
42
43
44
```

```
35     def is_chain_valid(self, chain):
36         previous_block = chain[0]
37         block_index = 1
38         while block_index < len(chain):
39             block = chain[block_index]
40             if block.previous_hash != previous_block.hash:
41                 return False
42             previous_proof = previous_block.proof
43             proof = block.proof
44             hash_operation = hashlib.sha256(str(proof**2) -
45                                             previous_proof**2).encode()).hexdigest()
```

```
45             if hash_operation[:4] != '0000':
46                 return False
47             previous_block = block
48             block_index += 1
49         return True
50
```

- Trong folder “blockchain” tạo file “test\_blockchain.py”.

```
1 from blockchain import Blockchain
2
3 # Testing the blockchain
4 my_blockchain = Blockchain()
5
6 # Adding transactions
7 my_blockchain.add_transaction('Alice', 'Bob', 10)
8 my_blockchain.add_transaction('Bob', 'Charlie', 5)
9 my_blockchain.add_transaction('Charlie', 'Alice', 3)
10
11 # Mining a new block
12 previous_block = my_blockchain.get_previous_block()
13 previous_proof = previous_block.proof
14 new_proof = my_blockchain.proof_of_work(previous_proof)
15 previous_hash = previous_block.hash
16 my_blockchain.add_transaction('Genesis', 'Miner', 1)
17 new_block = my_blockchain.create_block(new_proof, previous_hash)
18
```

```
19 # Displaying the blockchain
20 for block in my_blockchain.chain:
21     print(f"Block #{block.index}")
22     print("Timestamp:", block.timestamp)
23     print("Transactions:", block.transactions)
24     print("Proof:", block.proof)
25     print("Previous Hash:", block.previous_hash)
26     print("Hash:", block.hash)
27     print("-----")
28
29 # Check if the blockchain is valid
30 print("Is Blockchain Valid:", my_blockchain.is_chain_valid(my_blockchain.
chain))
```

- Kiểm tra chương trình: Chạy file “test\_blockchain.py”.

```
python .\test_blockchain.py
```

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\blockchain> python .\test_blockchain.py
Block #1
Timestamp: 1703922903.5387828
Transactions: []
Proof: 1
Previous Hash: 0
Hash: f53393bf2d18487c5220187b46dfae646a2f996a3b755d4a72d7988c1b24ab6a
-----
Block #2
Timestamp: 1703922903.54079
Transactions: [{"sender": "Alice", "receiver": "Bob", "amount": 10}, {"sender": "Bob", "receiver": "Charlie", "amount": 5}, {"sender": "Miner", "receiver": null, "amount": 1}]
Proof: 533
Previous Hash: f53393bf2d18487c5220187b46dfae646a2f996a3b755d4a72d7988c1b24ab6a
Hash: 26e600507cc02dc06e91ba4046ef516a01483eb079c15a3a3ad1eecc21f60fe
-----
Is Blockchain Valid: True
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\blockchain>

```

- Giải thích:

- Kết quả xuất ra thông qua việc chạy "**test\_blockchain.py**" trình bày chi tiết của hai khối trên blockchain và kết quả của hàm kiểm tra tính hợp lệ của chuỗi blockchain.

1. Block #1:

- Index: 1    Timestamp: 1703922903.5387828
- Transactions: Rỗng ([]), tức là không có giao dịch nào trong khối này.
- Proof: 1    Previous Hash: Giá trị hash của khối trước đó.
- Hash: Giá trị hash của khối này.

2. Block #2:

- Index: 2    Timestamp: 1703922903.54079
- Transactions: Danh sách giao dịch bao gồm 4 giao dịch, từ 'Alice' đến 'Bob' với số tiền 10, từ 'Bob' đến 'Charlie' với số tiền 5, từ 'Charlie' đến 'Alice' với số tiền 3 và một giao dịch khác từ 'Genesis' đến 'Miner' với số tiền 1.
- Proof: 533    Previous Hash: Giá trị hash của khối trước đó.
- Hash: Giá trị hash của khối này.
- + Cuối cùng, dòng cuối cùng trong output chứa thông điệp "Is Blockchain Valid: True" biểu thị rằng hàm kiểm tra tính hợp lệ của chuỗi blockchain đã trả về kết quả là True.

- Commit code:

```
git add .
git commit -m "[add] lab-05 blockchain"
```

#### 5.5.4 Bài thực hành 04: Giấu tin trong ảnh

Viết chương trình Python để thực hiện việc giấu thông tin trong ảnh. Sau đó, giải mã thông điệp đã được ẩn trong ảnh sau khi hoàn thành quá trình giấu tin.

Hướng dẫn:

- Trong folder "lab-05" tạo folder "img-hidden". Trong folder "img-hidden" tạo file "requirements.txt".

```
1 Pillow
2 cryptography
```

- Vào Terminal, chạy các lệnh sau:

```
cd .\lab-05\img-hidden\
pip install -r .\requirements.txt
```

- Trong folder "img-hidden" tạo file "encrypt.py".

```
1 import sys
2 from PIL import Image
3
4 def encode_image(image_path, message):
5     img = Image.open(image_path)
6     width, height = img.size
7     pixel_index = 0
8     binary_message = ''.join(format(ord(char), '08b') for char in message)
9     binary_message += '11111111111110' # Đánh dấu kết thúc thông điệp
10
```

```

11     data_index = 0
12     for row in range(height):
13         for col in range(width):
14             pixel = list(img.getpixel((col, row)))
15
16             for color_channel in range(3):
17                 if data_index < len(binary_message):
18                     pixel[color_channel] = int(format(pixel[color_channel],
19                         '08b')[::-1] + binary_message[data_index], 2)
20                     data_index += 1
21
22             img.putpixel((col, row), tuple(pixel))
23
24             if data_index >= len(binary_message):
25                 break
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40

```

```

26     encoded_image_path = 'encoded_image.png'
27     img.save(encoded_image_path)
28     print("Steganography complete. Encoded image saved as",
29           encoded_image_path)
30
31     def main():
32         if len(sys.argv) != 3:
33             print("Usage: python encrypt.py <image_path> <message>")
34             return
35
36         image_path = sys.argv[1]
37         message = sys.argv[2]
38         encode_image(image_path, message)
39
40     if __name__ == "__main__":
41         main()

```

- Trong folder "img-hidden" tạo file "decrypt.py".

```

1 import sys
2 from PIL import Image
3
4 def decode_image(encoded_image_path):
5     img = Image.open(encoded_image_path)
6     width, height = img.size
7     binary_message = ""
8

```

```

9   for row in range(height):
10      for col in range(width):
11          pixel = img.getpixel((col, row))
12
13          for color_channel in range(3):
14              binary_message += format(pixel[color_channel], '08b')[-1]
15

```

```

16     message = ""
17     for i in range(0, len(binary_message), 8):
18         char = chr(int(binary_message[i:i+8], 2))
19         if char == '\0': # Kết thúc thông điệp khi gặp dấu '\0'
20             break
21         message += char
22
23     return message
24

```

```

25 def main():
26     if len(sys.argv) != 2:
27         print("Usage: python decrypt.py <encoded_image_path>")
28         return
29
30     encoded_image_path = sys.argv[1]
31     decoded_message = decode_image(encoded_image_path)
32     print("Decoded message:", decoded_message)
33
34 if __name__ == "__main__":
35     main()

```

- Kiểm tra chương trình:

- Bước 1: Chuẩn bị một ảnh, lưu vào thư mục “img-hidden” với tên “image.jpg”.



- Bước 2: Thực thi file “encrypt.py”. Sau khi thực thi, file “encoded\_image.jpg” được tạo.

```
python .\encrypt.py .\image.jpg thongdiepcuaban
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\img-hidden> python .\encrypt.py .\image.jpg phuocnguyen
Steganography complete. Encoded image saved as encoded_image.png
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\img-hidden>
```

- Bước 3: Thực hiện giải mã thông điệp bằng cách chạy file “decrypt.py”.

```
python .\decrypt.py .\encoded_image.png
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\img-hidden> python .\encrypt.py .\image.jpg phuocnguyen
Steganography complete. Encoded image saved as encoded_image.png
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\img-hidden> python .\decrypt.py .\encoded_image.png
Decoded message: phuocnguyenýþÙø$Ùm
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\img-hidden>
```

- Bước 4: Thực hiện giải mã bằng hình gốc ban đầu, ta không thu được kết quả như mong muốn.

```
python .\decrypt.py .\image.jpg
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\img-hidden> python .\encrypt.py .\image.jpg phuocnguyen
Steganography complete. Encoded image saved as encoded_image.png
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\img-hidden> python .\decrypt.py .\encoded_image.png
Decoded message: phuocnguyenýþÙø$Ùm
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\img-hidden> python .\decrypt.py .\image.jpg
Decoded message: J8åØåd"å
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-05\img-hidden>
```

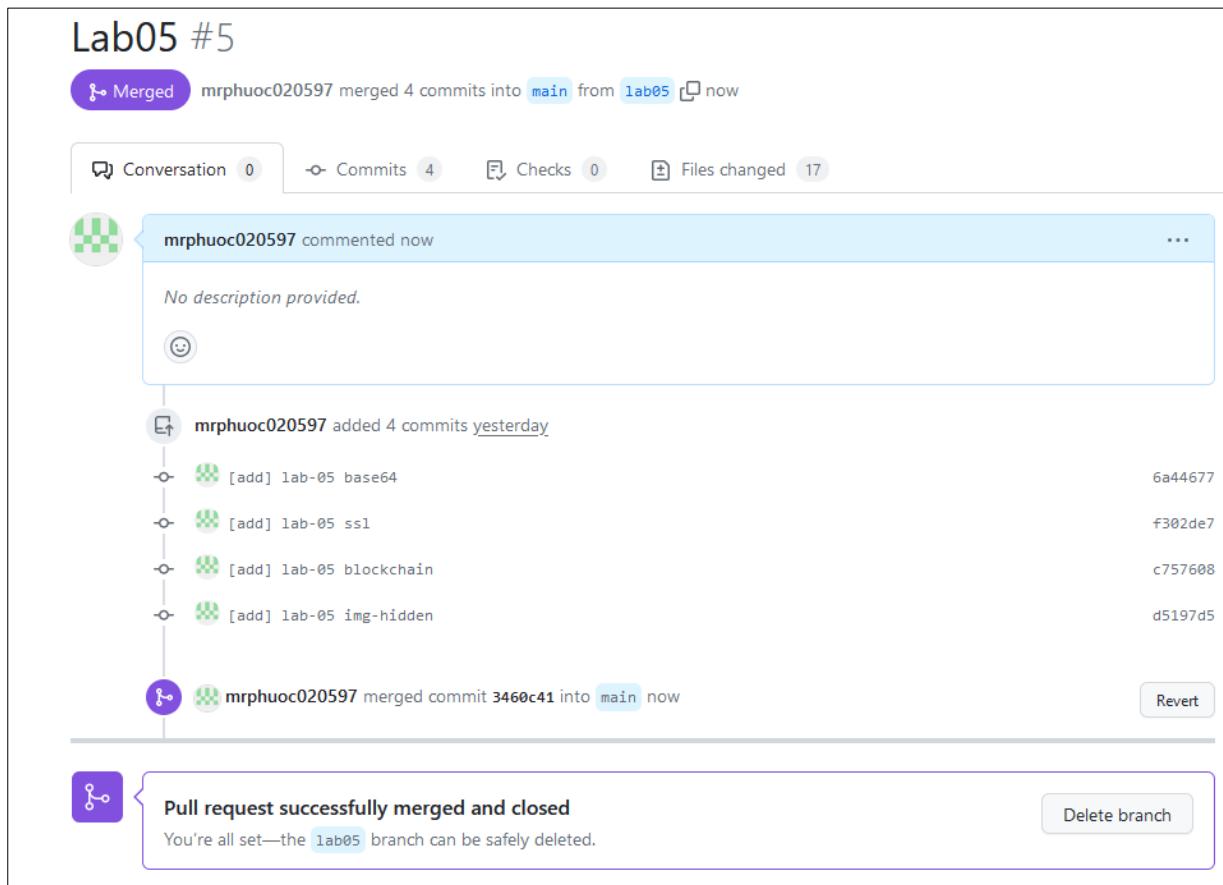
- Commit code:

```
git add .
git commit -m "[add] lab-05 img-hidden"
```

- Push các thay đổi lên remote repo:

```
git push origin lab05
```

- Tiến hành tạo PR từ nhánh lab05 về nhánh main. Review, kiểm tra đủ số lượng file, sau đó tiến hành “Merge pull request”.



## 5.6 BÀI TẬP MỞ RỘNG

- **Câu 01:** Thực hiện tạo giao diện UI desktop cho các bài thực hành phía trên.
- **Câu 02:** Thực hiện tìm hiểu về các kỹ thuật giấu tin trong các phương tiện truyền thông khác như video, văn bản hoặc âm thanh.

# BÀI 6: AN TOÀN MẠNG VÀ GIÁM SÁT HỆ THỐNG

Trong bài thực hành này, chúng ta sẽ khám phá về web server, cơ chế hoạt động của nó và cách tạo ra một web server đơn giản. Bên cạnh đó, chúng ta sẽ tìm hiểu hai thư viện “scapy” và “psutil” trong Python, được sử dụng để kiểm tra và quản lý an toàn cho hệ thống mạng, quét mạng, thay đổi gói tin, và giám sát hệ thống một cách hiệu quả. Chúng ta cũng sẽ ứng dụng và kết hợp các thư viện này để giám sát và gửi thông báo cho quản trị viên hệ thống qua Telegram, từ đó giúp người học nắm vững kiến thức cơ bản cần thiết để nghiên cứu, xây dựng và phát triển các hệ thống bảo đảm an toàn và an ninh thông tin.

## 6.1 WEB SERVER

---

Một web server là phần mềm hoặc thiết bị được sử dụng để lưu trữ, xử lý và cung cấp các trang web, ứng dụng web và tài nguyên khác qua internet. Đây là một thành phần thiết yếu trong hạ tầng internet, cung cấp nền tảng cho người dùng truy cập và tương tác với nhiều loại dữ liệu trực tuyến. Nhiệm vụ chính của web server là nhận các yêu cầu từ trình duyệt web hoặc phần mềm khách hàng thông qua giao thức HTTP (Hypertext Transfer Protocol) hoặc HTTPS (HTTP Secure - phiên bản bảo mật của HTTP), sau đó xử lý những yêu cầu này và trả về các tài nguyên được yêu cầu, như trang web, hình ảnh, video, tệp HTML, CSS, JavaScript, và nhiều loại dữ liệu khác.

Một số web server phổ biến bao gồm Apache HTTP Server, Nginx, Microsoft Internet Information Services (IIS) và OpenLiteSpeed. Mỗi loại có những tính năng và đặc điểm riêng biệt.

## 6.2 NETWORK SCANNER & NETWORK CAPTURE

---

Network Scanner và Network Capture là hai công cụ quan trọng trong lĩnh vực mạng máy tính, được sử dụng để khám phá và phân tích các mạng máy tính khác

nhau. Cả hai công cụ đều là quan trọng trong việc xác định, kiểm tra và bảo vệ mạng máy tính khỏi các mối đe dọa và lỗ hổng bảo mật.

**Network Scanner** là một loại công cụ phần mềm được thiết kế để quét và kiểm tra các thiết bị, máy chủ, hoặc các cổng kết nối mạng trong một mạng cụ thể. Công cụ này thường được sử dụng để tìm kiếm và phân tích các thiết bị có sẵn trong mạng, nhằm thu thập thông tin về địa chỉ IP, cổng mạng, dịch vụ đang chạy, và các thông tin liên quan khác. Một số công cụ quét mạng phổ biến bao gồm Nmap, Angry IP Scanner, Zenmap, và OpenVAS.

**Network Capture** (hay còn được gọi là packet sniffing) là quá trình ghi lại và phân tích các gói tin dữ liệu (network packets) đi qua một mạng máy tính. Công cụ này cho phép người dùng theo dõi và phân tích lưu lượng mạng, xem nội dung của các gói tin, và hiểu rõ hơn về cách thức hoạt động của mạng. Wireshark là một trong những công cụ phổ biến nhất để thực hiện việc capture và phân tích gói tin trong mạng.

Sự khác biệt giữa Network Scanner và Network Capture:

- Network Scanner tập trung vào việc quét và tìm kiếm thông tin về các thiết bị và dịch vụ có sẵn trong mạng.
- Network Capture tập trung vào việc ghi lại và phân tích các gói tin mạng để hiểu rõ về hoạt động và nội dung của dữ liệu truyền qua mạng.

## 6.3 PORT SCANNER

Port Scanner là một công cụ phần mềm được sử dụng để kiểm tra các cổng kết nối mạng trên một máy chủ hoặc thiết bị trong mạng. Công cụ này quét và phân tích các cổng mạng trên một máy tính hoặc một địa chỉ IP để xác định xem các cổng nào đang mở, đóng, hoặc đang lắng nghe các kết nối. Mục tiêu chính của việc sử dụng Port Scanner là tìm ra các dịch vụ hoặc ứng dụng đang chạy trên máy chủ hoặc thiết bị mạng. Kết quả của quá trình quét cổng mạng có thể giúp người dùng:

- Phát hiện lỗ hổng bảo mật: Xác định các cổng mạng mở, mà khi không được quản lý hoặc bảo vệ cẩn thận, có thể tạo ra lỗ hổng bảo mật cho hệ thống.

- Phân tích bảo mật hệ thống: Kiểm tra và xác định cách các dịch vụ hoặc ứng dụng được triển khai trên các cổng kết nối mạng.
- Tìm kiếm dịch vụ và ứng dụng: Xác định các dịch vụ hoặc ứng dụng đang chạy trên một máy chủ hoặc thiết bị cụ thể.

## 6.4 THƯ VIỆN PYTHON SCAPY

---

“**scapy**” là một thư viện mạnh mẽ trong ngôn ngữ lập trình Python được sử dụng để tạo và gửi các gói tin trên mạng, cũng như phân tích các gói tin đã nhận được. Nó cung cấp một số công cụ cho việc tương tác với mạng và làm việc với các giao thức mạng khác nhau. Một số tính năng quan trọng của Scapy:

- Tạo và gửi gói tin: Scapy cho phép bạn tạo ra các gói tin mạng từ các lớp và chúng đi trên mạng.
- Phân tích gói tin: Nó cũng có khả năng phân tích các gói tin mạng mà máy tính nhận được, giúp chúng ta hiểu rõ hơn về cấu trúc và thông tin có trong các gói tin mạng.
- Tích hợp với các giao thức mạng: Scapy hỗ trợ nhiều giao thức mạng phổ biến như IP, TCP, UDP, ICMP, DNS, ARP và nhiều giao thức khác.
- Kiểm tra an ninh mạng (Network Security Testing): Được sử dụng để kiểm tra an ninh mạng bằng cách tạo và gửi các gói tin mạng tùy chỉnh, kiểm tra và thử nghiệm lỗ hổng bảo mật.
- Automation và Network Prototyping: Scapy có thể được sử dụng để tự động hóa các tác vụ liên quan đến mạng, cũng như để tạo các mô hình mạng tạm thời để kiểm tra hoặc mô phỏng.

## 6.5 THƯ VIỆN PYTHON PSUTIL

---

“**psutil**” là một thư viện Python mạnh mẽ được sử dụng để giám sát và thu thập thông tin về hệ thống máy tính hiện đang chạy. Nó cung cấp các công cụ để truy cập thông tin chi tiết về tài nguyên hệ thống như CPU, bộ nhớ, ổ đĩa, mạng và các tiến trình đang chạy. “**psutil**” cung cấp giao diện thuận tiện để thu thập thông tin hệ

thống mà không cần thực hiện các cuộc gọi hệ thống cấp thấp, giúp chúng ta dễ dàng quản lý và giám sát hệ thống trong ứng dụng Python của mình.

## 6.6 BÀI TẬP THỰC HÀNH

### 6.6.1 Bài thực hành 01: Tạo web server

Viết chương trình tạo một web server đơn giản bằng Python. Hướng dẫn:

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

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

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

The screenshot shows the VS Code interface with the 'TERMINAL' tab selected. The terminal window displays the following command and its execution:

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

- Tách nhánh lab06 từ nhánh main.

```
git checkout -b lab06
```

- Tạo folder "lab-06". Trong folder "lab-06" tạo folder "webserver".
- Trong folder "webserver" tạo file "webserver.py".

```
1 import socket
2
3 def handle_request(client_socket, request_data):
4     if "GET /admin" in request_data:
5         response = "HTTP/1.1 200 OK\r\nContent-Type: text/
6         html\r\n\r\nWelcome to the admin page!"
7     else:
8         response = "HTTP/1.1 200 OK\r\nContent-Type: text/
9         html\r\n\r\nHello, this is a simple web server!"
10        client_socket.sendall(response.encode('utf-8'))
11        client_socket.close()
```

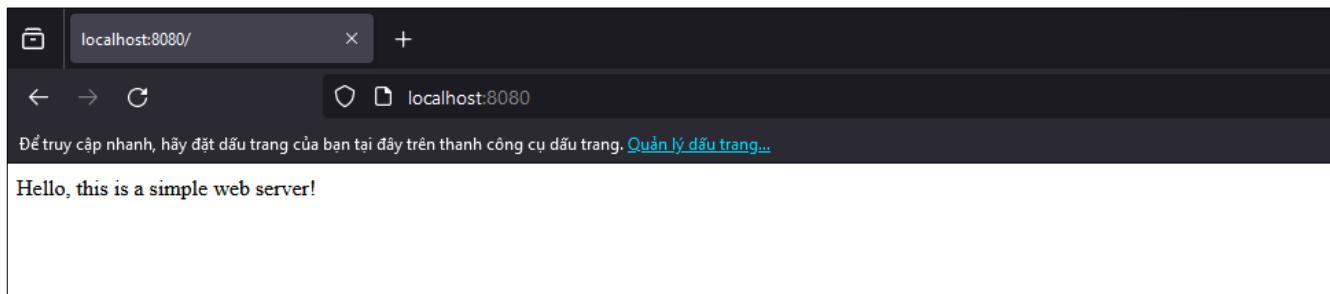
```

11 def main():
12     server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
13     server_socket.bind(('127.0.0.1', 8080))
14     server_socket.listen(5)
15
16     print("Server listening on port 8080...")
17
18     while True:
19         client_socket, client_address = server_socket.accept()
20         print(f"Connection from {client_address}")
21         request_data = client_socket.recv(1024).decode('utf-8')
22         handle_request(client_socket, request_data)
23
24 if __name__ == '__main__':
25     main()

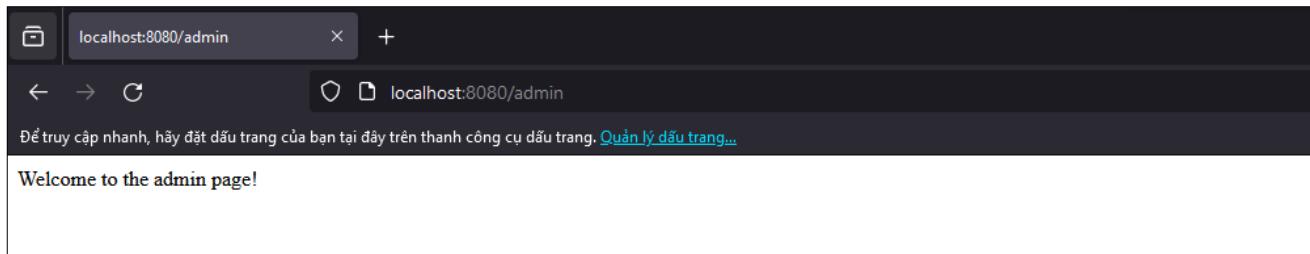
```

- Chạy file “webserver.py”. Vào trình duyệt, gõ địa chỉ: <http://localhost:8080/>

```
python .\webserver.py
```



- Đổi URL thành: <http://localhost:8080/admin>



- Trong folder “webserver” tạo file “webserver-html.py”.

```

1 import socket
2

```

```

3 def handle_request(client_socket, request_data):
4     if "GET /admin" in request_data:
5         response = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n"
6         with open("admin.html", "r") as file:
7             response += file.read()
8     else:
9         response = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n"
10        with open("index.html", "r") as file:
11            response += file.read()
12    client_socket.sendall(response.encode('utf-8'))
13    client_socket.close()
14

```

```

15 def main():
16     server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
17     server_socket.bind(('127.0.0.1', 8080))
18     server_socket.listen(5)
19
20     print("Server listening on port 8080...")
21
22     while True:
23         client_socket, client_address = server_socket.accept()
24         print(f"Connection from {client_address}")
25         request_data = client_socket.recv(1024).decode('utf-8')
26         handle_request(client_socket, request_data)
27
28 if __name__ == '__main__':
29     main()

```

- Trong folder “webserver” tạo file “index.html”.

```

1 This is my website!

```

- Trong folder “webserver” tạo file “admin.html”.

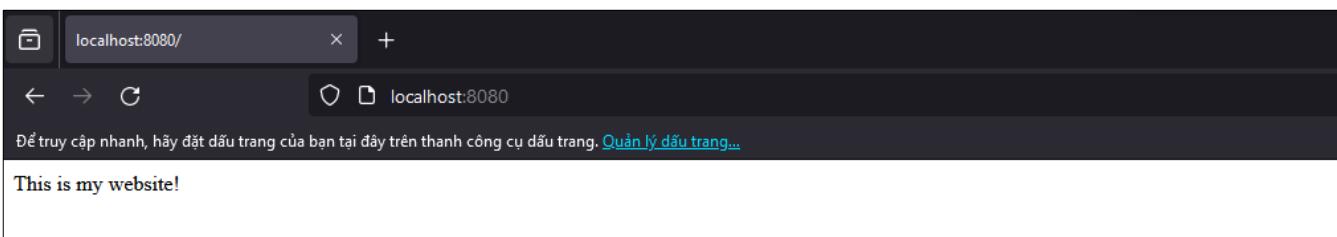
```

1 This is admin only! Please contact the administrator for more information.

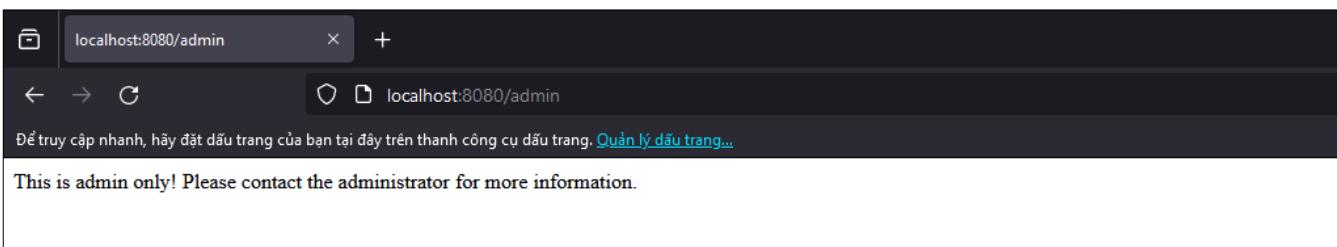
```

- Nhấn Ctrl + C để ngắt tiến trình đang chạy của file “webserver.py”. Tiến hành chạy file “webserver-html.py”, sau đó mở trình duyệt vào địa chỉ <http://localhost:8080> để kiểm tra.

```
python .\webserver-html.py
```



- Chuyển URL thành <http://localhost:8080/admin> và kiểm tra kết quả.



- Commit code:

```
git add .
git commit -m "[add] lab-06 web server"
```

## 6.6.2 Bài thực hành 02: Network Scanner

Viết chương trình thực hiện Network Scanner bằng Python sử dụng thư viện "scapy" và sử dụng API tại <https://api.macvendors.com/> để lấy thông tin về nhà sản xuất từ địa chỉ MAC của thiết bị.

Hướng dẫn:

- Trong folder "lab-06" tạo folder "netscanner". Trong folder "netscanner" tạo file "requirements.txt".

```
1 scapy
```

- Vào Terminal, chạy các lệnh sau:

```
cd .\lab-06\netscanner\
pip install -r ./requirements.txt
```

- Trong folder "netscanner" tạo file "network\_scanner.py".

```

1 import requests
2 from scapy.all import ARP, Ether, srp
3
4 def local_network_scan(ip_range):
5     arp = ARP(pdst=ip_range)
6     ether = Ether(dst="ff:ff:ff:ff:ff:ff")
7     packet = ether/arp
8     result = srp(packet, timeout=3, verbose=0)[0]
9
10    devices = []
11    for sent, received in result:
12        devices.append({
13            'ip': received.psrc,
14            'mac': received.hwsrc,
15            'vendor': get_vendor_by_mac(received.hwsrc)
16        })
17
18    return devices
19

```

```

20 def get_vendor_by_mac(mac):
21     try:
22         response = requests.get(f"https://api.macvendors.com/{mac}")
23         if response.status_code == 200:
24             return response.text
25         else:
26             return "Unknown"
27     except Exception as e:
28         print("Error fetching vendor information:", e)
29     return "Unknown"
30

```

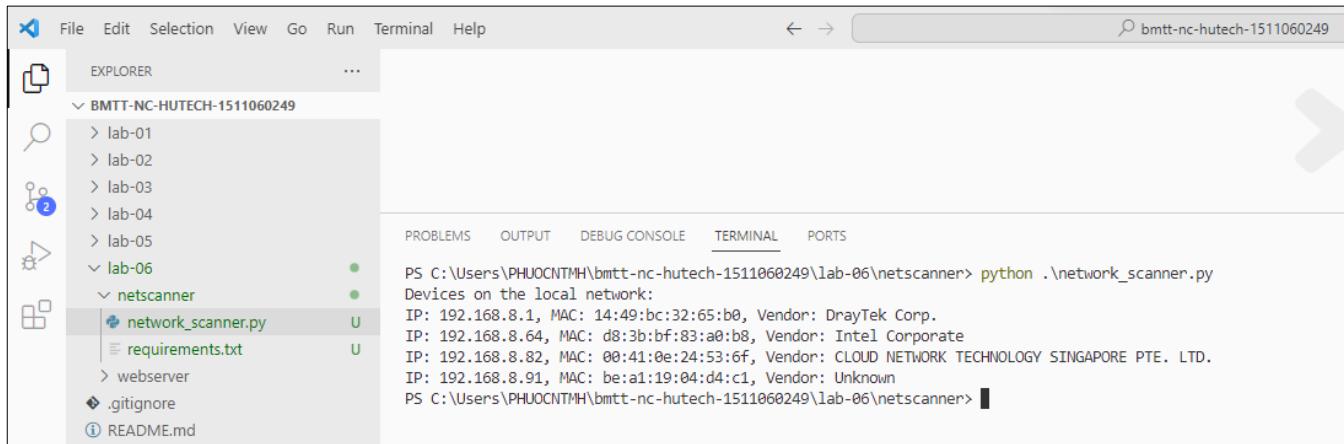
```

31 def main():
32     ip_range = "192.168.8.1/24"
33     devices = local_network_scan(ip_range)
34
35     print("Devices on the local network:")
36     for device in devices:
37         print(f"IP: {device['ip']}, MAC: {device['mac']}, Vendor: {device['vendor']}")
38
39 if __name__ == '__main__':
40     main()

```

- Tải và cài đặt Npcap tại địa chỉ: <https://npcap.com/#download>
  - Chạy file “network\_scanner.py” và kiểm tra kết quả.

```
python .\network_scanner.py
```



- Commit code:

```
git add .
git commit -m "[add] lab-06 network scanner"
```

### **6.6.3 Bài thực hành 03: Network Capture**

Viết chương trình Network Capture bằng Python sử dụng thư viện “scapy”.

## Hướng dẫn:

- Trong folder “lab-06” tạo folder “netcapture”. Trong folder “netcapture” tạo file “network\_capture.py”.

```
1 import subprocess
2 from scapy.all import *
3
4 def get_interfaces():
5     result = subprocess.run(["netsh", "interface", "show", "interface"],
6                           capture_output=True, text=True)
7     output_lines = result.stdout.splitlines()[3:]
8     interfaces = [line.split()[3] for line in output_lines if len(line.split())
9                   () >= 4]
10    return interfaces
```

```

10 def packet_handler(packet):
11     if packet.haslayer(Raw):
12         print("Captured Packet:")
13         print(str(packet))
14
15 # Lấy danh sách các giao diện mạng
16 interfaces = get_interfaces()
17
18 # In danh sách giao diện mạng để người dùng lựa chọn
19 print("Danh sách các giao diện mạng:")
20 for i, iface in enumerate(interfaces, start=1):
21     print(f"{i}. {iface}")
22
23 # Lựa chọn giao diện mạng từ người dùng
24 choice = int(input("Chọn một giao diện mạng (nhập số): "))
25 selected_iface = interfaces[choice - 1]
26
27 # Bắt gói tin trên giao diện mạng được chọn
28 sniff(iface=selected_iface, prn=packet_handler, filter="tcp")

```

- Thực thi file “network\_capture.py” và kiểm tra kết quả.

```
python .\network_capture.py
```

The screenshot shows a terminal window within a code editor interface (VS Code). The command `python .\network_capture.py` is run, and the output is displayed in the terminal tab. The output shows the program listing available interfaces and then capturing traffic on the selected interface (index 4, which is VMware).

```

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-06\netcapture> python .\network_capture.py
Danh sách các giao diện mạng:
1. VMware
2. VMware
3. Ethernet
4. Wi-Fi
Chọn một giao diện mạng (nhập số): 4
Captured Packet:
Ether / IP / TCP 66.203.125.15:https > 192.168.8.64:13714 PA / Raw
Captured Packet:
Ether / IP / TCP 192.168.8.64:13963 > 52.111.240.8:https PA / Raw
Captured Packet:
Ether / IP / TCP 52.111.240.8:https > 192.168.8.64:13963 PA / Raw
Captured Packet:
Ether / IP / TCP 192.168.8.64:13963 > 52.111.240.8:https PA / Raw
Captured Packet:
Ether / IP / TCP 52.111.240.8:https > 192.168.8.64:13963 PA / Raw
Captured Packet:
Ether / IP / TCP 142.251.222.202:https > 192.168.8.64:13382 PA / Raw
Captured Packet:
Ether / IP / TCP 192.168.8.64:13963 > 52.111.240.8:https PA / Raw
Captured Packet:
Ether / IP / TCP 52.111.240.8:https > 192.168.8.64:13963 PA / Raw
Captured Packet:
Ether / IP / TCP 192.168.8.64:13963 > 52.111.240.8:https PA / Raw
Captured Packet:
Ether / IP / TCP 52.111.240.8:https > 192.168.8.64:13963 PA / Raw
Captured Packet:
Ether / IP / TCP 192.168.8.64:13963 > 52.111.240.8:https PA / Raw
Captured Packet:

```

- Commit code:

```
git add .
git commit -m "[add] lab-06 network capture"
```

## 6.6.4 Bài thực hành 04: Port Scanner

Viết chương trình Port Scanner bằng Python sử dụng thư viện “scapy” để quét các cổng phổ biến. Hướng dẫn:

- Trong folder “lab-06” tạo folder “portscanner”. Trong folder “portscanner” tạo file “port\_scanner.py”.

```
1 from scapy.all import *
2
3 COMMON_PORTS = [21, 22, 23, 25, 53, 80, 110, 143, 443, 445, 3389]
4
5 def scan_common_ports(target_domain, timeout=2):
6     open_ports = []
7     target_ip = socket.gethostbyname(target_domain)
8
9     for port in COMMON_PORTS:
10         response = sr1(IP(dst=target_ip)/TCP(dport=port, flags="S"),
11                      timeout=timeout, verbose=0)
12
13         if response and response.haslayer(TCP) and response[TCP].flags ==
14             0x12:
15             open_ports.append(port)
16             send(IP(dst=target_ip)/TCP(dport=port, flags="R"), verbose=0)
17
18     return open_ports
```

```
18 def main():
19     target_domain = input("Enter the target domain: ")
20
21     open_ports = scan_common_ports(target_domain)
22
23     if open_ports:
24         print("Open common ports:")
25         print(open_ports)
```

```

26     else:
27         print("No open common ports found.")
28
29 if __name__ == '__main__':
30     main()

```

- Kiểm tra chương trình bằng cách chạy file “port\_scanner.py”.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-06\portscanner> python .\port_scanner.py
Enter the target domain: hutech.edu.vn
Open common ports:
[80, 443]
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-06\portscanner>

```

- Commit code:

```

git add .
git commit -m "[add] lab-06 port scanner"

```

## 6.6.5 Bài thực hành 05: Lắng nghe và thay đổi gói tin ICMP

Viết chương trình lắng nghe và thay đổi gói tin ICMP bằng Python sử dụng thư viện “scapy”. Hướng dẫn:

- Trong folder “lab-06” tạo folder “icmp”. Trong folder “icmp” tạo file “icmp\_listen.py”.

```

1 from scapy.all import *
2
3 def packet_callback(packet):
4     if packet.haslayer(ICMP):
5         icmp_packet = packet[ICMP]
6         print("ICMP Packet Information:")
7         print(f"Source IP: {packet[IP].src}")
8         print(f"Destination IP: {packet[IP].dst}")
9         print(f"Type: {icmp_packet.type}")
10        print(f"Code: {icmp_packet.code}")
11        print(f"ID: {icmp_packet.id}")

```

```

12        print(f"Sequence: {icmp_packet.seq}")
13        print(f"Load: {icmp_packet.load}")
14        print("=" * 30)
15

```

```

16 def main():
17     sniff(prn=packet_callback, filter="icmp", store=0)
18
19 if __name__ == '__main__':
20     main()

```

- Kiểm tra chương trình:

- Bước 1: Split Terminal. Chạy file “icmp\_listen.py”.

```
python .\icmp_listen.py
```

- Bước 2: Gõ lệnh “ping” tại Terminal.

The screenshot shows two terminal sessions side-by-side. The left terminal session is running the script `icmp_listen.py`, which is printing ICMP packet information for each received packet. The right terminal session is running a `ping` command to the IP address 8.8.8.8, displaying the ping statistics.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-06\icmp> python .\icmp_listen.py
ICMP Packet Information:
Source IP: 192.168.8.64
Destination IP: 8.8.8.8
Type: 8
Code: 0
ID: 1
Sequence: 13
Load: b'abcdefghijklmnopqrstuvwxyz'
=====
ICMP Packet Information:
Source IP: 8.8.8.8
Destination IP: 192.168.8.64
Type: 0
Code: 0
ID: 1
Sequence: 13
Load: b'abcdefghijklmnopqrstuvwxyz'
=====
ICMP Packet Information:
Source IP: 192.168.8.64
Destination IP: 8.8.8.8
Type: 0
Code: 0
ID: 1
Sequence: 14
Load: b'abcdefghijklmnopqrstuvwxyz'
=====
ICMP Packet Information:
Source IP: 8.8.8.8
Destination IP: 192.168.8.64
Type: 0
Code: 0
ID: 1
Sequence: 14
Load: b'abcdefghijklmnopqrstuvwxyz'
=====
ICMP Packet Information:
Source IP: 192.168.8.64
Destination IP: 8.8.8.8
Type: 8
Code: 0
ID: 1
Sequence: 15
Load: b'abcdefghijklmnopqrstuvwxyz'
=====

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249> ping 8.8.8.8
Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=33ms TTL=117
Reply from 8.8.8.8: bytes=32 time=30ms TTL=117
Reply from 8.8.8.8: bytes=32 time=31ms TTL=117
Reply from 8.8.8.8: bytes=32 time=29ms TTL=117

Ping statistics for 8.8.8.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 29ms, Maximum = 33ms, Average = 30ms
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249>

```

- Trong folder “icmp” tạo file “icmp\_change.py”.

```

1 from scapy.all import *
2
3 def modify_icmp_packet(packet):
4     if packet.haslayer(ICMP):
5         icmp_packet = packet[ICMP]
6
7         print("Original ICMP Packet:")
8         print(f"Source IP: {packet[IP].src}")
9         print(f"Destination IP: {packet[IP].dst}")
10        print(f"Type: {icmp_packet.type}")
11        print(f"Code: {icmp_packet.code}")
12        print(f"ID: {icmp_packet.id}")
13        print(f"Sequence: {icmp_packet.seq}")
14        print(f"Load: {icmp_packet.load}")
15
16        new_load = b"This is a modified ICMP packet."
17        new_packet = IP(src=packet[IP].dst, dst=packet[IP].src)/ICMP
18            (type=icmp_packet.type, code=icmp_packet.code, id=icmp_packet.id,
seq=icmp_packet.seq)/new_load
19
20        print("\nModified ICMP Packet:")
21        print(f"Source IP: {new_packet[IP].src}")
22        print(f"Destination IP: {new_packet[IP].dst}")
23        print(f"Type: {new_packet[ICMP].type}")
24        print(f"Code: {new_packet[ICMP].code}")
25        print(f"ID: {new_packet[ICMP].id}")
26        print(f"Sequence: {new_packet[ICMP].seq}")
27        print(f"Load: {new_load}")
28        print("=" * 30)
29
30        send(new_packet)
31
32    def main():
33        sniff(prn=modify_icmp_packet, filter="icmp", store=0)
34
35    if __name__ == '__main__':
36        main()

```

- Kiểm tra chương trình:

- Bước 1: Split Terminal. Chạy file “icmp\_change.py”.

```
python .\icmp_change.py
```

- Bước 2: Gõ lệnh “ping” tại Terminal.

The screenshot shows a terminal window with the following content:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

File "C:\Users\PHUOCNTMH\AppData\Local\Programs\Python\Python312\Lib\site-packages\scapy\packet.py", PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-06\icmp> python .\icmp_change.py
Original ICMP Packet:
Source IP: 192.168.8.64
Destination IP: 8.8.8.8
Type: 8
Code: 0
ID: 1
Sequence: 22
Load: b'abcdefghijklmnopqrstuvwxyz'

Modified ICMP Packet:
Source IP: 8.8.8.8
Destination IP: 192.168.8.64
Type: 8
Code: 0
ID: 1
Sequence: 22
Load: b'This is a modified ICMP packet.'

=====
.
Sent 1 packets.
Original ICMP Packet:
Source IP: 192.168.8.64
Destination IP: 8.8.8.8
Type: 0
Code: 0
ID: 1
Sequence: 22
Load: b'This is a modified ICMP packet.'

Modified ICMP Packet:
Source IP: 8.8.8.8
Destination IP: 192.168.8.64
Type: 0
Code: 0
ID: 1
Sequence: 22
Load: b'This is a modified ICMP packet.'

=====
.
Sent 1 packets.
Original ICMP Packet:
Source IP: 8.8.8.8
Destination IP: 192.168.8.64
Type: 0
Code: 0
ID: 1
Sequence: 22
Load: b'This is a modified ICMP packet.'


PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249> ping 8.8.8.8
Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=34ms TTL=117
Reply from 8.8.8.8: bytes=32 time=30ms TTL=117
Reply from 8.8.8.8: bytes=32 time=30ms TTL=117
Reply from 8.8.8.8: bytes=32 time=32ms TTL=117

Ping statistics for 8.8.8.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 30ms, Maximum = 34ms, Average = 31ms
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249>

```

- Commit code:

```
git add .
git commit -m "[add] lab-06 icmp change"
```

## 6.6.6 Bài thực hành 06: Giám sát hệ thống và gửi tin qua Telegram

Viết chương trình giám sát hệ thống bằng Python sử dụng thư viện “psutil” và gửi tin nhắn giám sát về Telegram thông qua thư viện “python-telegram-bot”. Hướng dẫn:

- Trong folder "lab-06" tạo folder "monitor". Trong folder "monitor" tạo file "requirements.txt".

```
1 psutil
2 python-telegram-bot
```

- Vào Terminal, chạy các lệnh sau:

```
cd .\lab-06\monitor\
pip install -r .\requirements.txt
```

- Trong folder " monitor" tạo file "monitor.py".

```
1 import psutil
2 import platform
3 import socket
4 import logging
5 import time
6
7 # Cấu hình logging
8 logging.basicConfig(level=logging.INFO, filename="system_monitor.log",
9                     format="%(asctime)s - %(levelname)s - %(message)s")
9 logger = logging.getLogger()
10
11 # Hàm ghi log
12 def log_info(category, message):
13     logger.info(f"{category}: {message}")
14     print(f"{category}: {message}")
15
```

```
16 # Hàm giám sát CPU và bộ nhớ
17 def monitor_cpu_memory():
18     cpu_percent = psutil.cpu_percent()
19     memory_info = psutil.virtual_memory()
20
21     log_info("CPU", f"Usage: {cpu_percent}%")
22     log_info("Memory", f"Usage: {memory_info.percent}%")
23
```

```
24 # Hàm giám sát thông tin hệ điều hành
25 def monitor_system():
26     os_info = platform.uname()
27     hostname = socket.gethostname()
28
29     log_info("System Info", f"Hostname: {hostname}")
30     log_info("System Info", f"Operating System: {os_info.system} {os_info.
31         release}")
32     log_info("System Info", f"Python Version: {platform.python_version()}")
33
```

```
33 # Hàm giám sát tình trạng mạng
34 def monitor_network():
35     net_stats = psutil.net_io_counters()
36     log_info("Network", f"Bytes Sent: {net_stats.bytes_sent}, Bytes
37         Received: {net_stats.bytes_recv}")
38 # Hàm giám sát danh sách phần mềm
39 def monitor_software():
40     software_list = psutil.process_iter(attrs=['pid', 'name', 'username'])
41
42     log_info("Software", "Running Software:")
43     for software in software_list:
44         software_name = software.info['name']
45         software_pid = software.info['pid']
46         software_username = software.info['username']
47         log_info("Software", f"PID: {software_pid}, Name: {software_name},
48             Username: {software_username}")
49
```

```
49 # Hàm thực hiện giám sát toàn bộ hệ thống
50 def monitor_system():
51     log_info("System Monitor", "Starting system monitoring...")
52
53     while True:
54         monitor_cpu_memory()
55         monitor_system_info()
56         monitor_network()
57         monitor_software()
58         log_info("System Monitor",
59                 "-----")
60         time.sleep(60)
```

```

60
61 if __name__ == "__main__":
62     monitor_system()

```

- Kiểm tra chương trình:

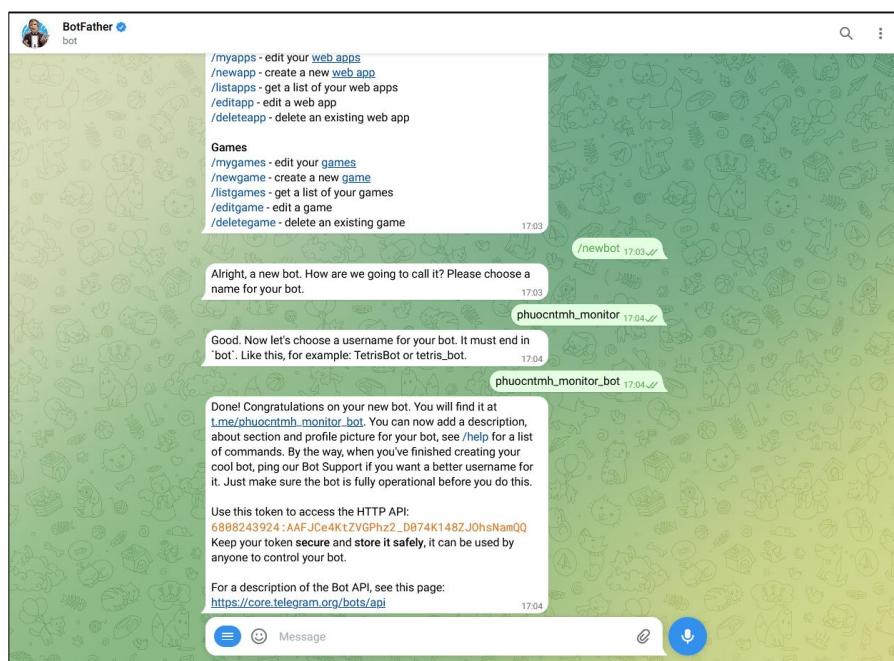
```

PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-06\monitor> python .\monitor.py
System Monitor: Starting system monitoring...
CPU: Usage: 0.0%
Memory: Usage: 48.5%
System Info: Hostname: PHUOCNTMH
System Info: Operating System: Windows 11
System Info: Python Version: 3.12.1
Network: Bytes Sent: 553752475, Bytes Received: 1049332862
Software: Running Software:
Software: PID: 0, Name: System Idle Process, Username: NT AUTHORITY\SYSTEM
Software: PID: 4, Name: System, Username: NT AUTHORITY\SYSTEM
Software: PID: 108, Name: , Username: None
Software: PID: 148, Name: Registry, Username: None
Software: PID: 332, Name: crashpad_handler.exe, Username: PHUOCNTMH\PHUOCNTMH

```

Tiếp theo, chúng ta sẽ tạo Telegram bot để cập nhật thông tin giám sát hệ thống về tin nhắn Telegram.

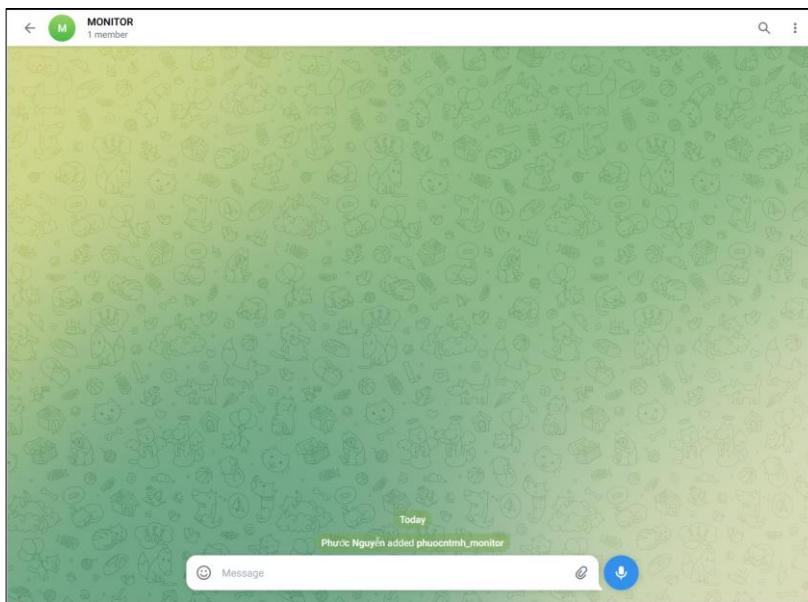
- Sinh viên tự tạo tài khoản Telegram. Login vào Telegram web tại địa chỉ: <https://web.telegram.org/>
- Chat với “BotFather” để tạo một bot: Tại ô Search của Telegram tìm “BotFather” (nhớ chọn cái có tick xanh). Chat “/newbot” vào khung chat với BotFather. Nhập tên cho bot, ở đây tôi chọn là “phuocntmh\_monitor\_bot”.
- Sau khi hoàn tất chúng ta sẽ nhận được 1 token, ghi lại token này.



# 190

## BÀI 6: AN TOÀN MẠNG VÀ GIÁM SÁT HỆ THỐNG

- Tạo mới 1 nhóm trên Telegram và thêm bot vào.



- Lấy "chat\_id" bằng cách truy kiểm tra URL của nhóm chat (có bot). Ví dụ:  
<https://web.telegram.org/k/#-4031606701> → "chat\_id" là "-4031606701".
- Bây giờ, các bước chuẩn bị đã hoàn thành, chúng ta sẽ tạo code bằng Python:
- Trong folder "monitor" tạo file "monitor-bot.py".

```
1 import psutil
2 import logging
3 import time
4 import asyncio
5 from telegram import Bot
6
7 # Thay thế 'YOUR_BOT_TOKEN' bằng mã truy cập API của bot bạn đã tạo
8 BOT_TOKEN = '6808243924:AAFJCe4KtZVGPhz2_D074K148ZJ0hsNamQQ'
9 # Thay thế 'YOUR_CHAT_ID' bằng chat_id của cuộc trò chuyện bạn muốn gửi
10 # thông báo tới
11 CHAT_ID = '-4031606701'
12
13 # Cấu hình logging
14 logging.basicConfig(level=logging.INFO, filename="system_monitor_bot.log",
15 format"%(asctime)s - %(levelname)s - %(message)s")
16 logger = logging.getLogger()
```

```

16 # Hàm ghi log
17 def log_info(category, message):
18     logger.info(f"{category}: {message}")
19     print(f"{category}: {message}")
20
21 # Hàm gửi tin nhắn qua Telegram
22 async def send_telegram_message(message):
23     bot = Bot(token=BOT_TOKEN)
24     await bot.send_message(chat_id=CHAT_ID, text=message)
25

```

```

26 # Hàm giám sát CPU và bộ nhớ
27 def monitor_cpu_memory():
28     cpu_percent = psutil.cpu_percent()
29     memory_info = psutil.virtual_memory()
30
31     log_info("CPU", f"Usage: {cpu_percent}%")
32     log_info("Memory", f"Usage: {memory_info.percent}%")
33
34     # Gửi thông báo qua Telegram
35     message = f"CPU Usage: {cpu_percent}%\nMemory Usage: {memory_info.
36     percent}%""
37     asyncio.run(send_telegram_message(message))
38

```

```

38 # Hàm thực hiện giám sát toàn bộ hệ thống
39 def monitor_system():
40     log_info("System Monitor", "Starting system monitoring...")
41
42     while True:
43         monitor_cpu_memory()
44         log_info("System Monitor",
45             "-----")
46         time.sleep(60)
47
48 if __name__ == "__main__":
49     monitor_system()

```

- Kiểm tra chương trình, chạy file “monitor-bot.py” và kiểm tra tin nhắn Telegram.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\PHUOCNTMH\bmtt-nc-hutech-1511060249\lab-06\monitor> python .\monitor-bot.py
System Monitor: Starting system monitoring...
CPU: Usage: 12.3%
Memory: Usage: 51.1%
System Monitor: -----
```



- Cập nhật file “.gitignore” để không push các file \*.log lên Git. Thêm vào hàng sau:

```
1 *.pyc
2 __pycache__/
3 *.log
```

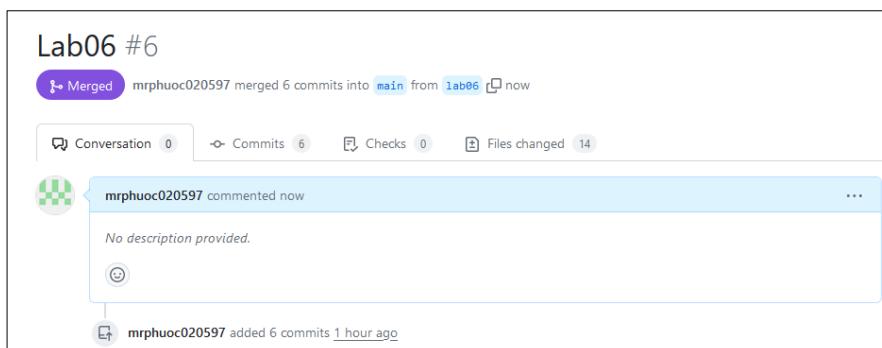
- Commit code:

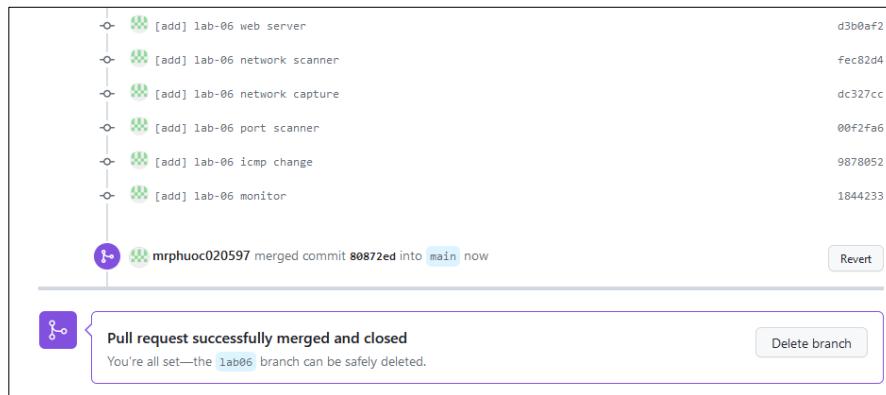
```
git add .
git commit -m "[add] lab-06 monitor"
```

- Push các thay đổi lên remote repo:

```
git push origin lab06
```

- Tiến hành tạo PR từ nhánh lab06 về nhánh main. Review, kiểm tra đủ số lượng file, sau đó tiến hành “Merge pull request”.





## 6.7 BÀI TẬP MỞ RỘNG

**Câu 01:** Xây dựng giao diện UI desktop cho Bài thực hành 02, 03, 04.

**Câu 02:** Cập nhật Bài thực hành 06 để thêm nhiều nội dung được gửi từ bot Telegram, sử dụng ví dụ được thực hành phía trên thông qua thư viện “psutil”.

## TÀI LIỆU THAM KHẢO

- [1] Charles Severance, "Python for Everybody", available at: [py4e.com](http://py4e.com).
- [2] "Python Wiki", available at: [wiki.python.org](http://wiki.python.org).
- [3] "Documentation for Visual Studio Code", available at: [code.visualstudio.com](http://code.visualstudio.com).
- [4] "Simple Git tutorial for beginners", Nulab Backlog, available at: [nulab.com/learn](http://nulab.com/learn).
- [5] Nigel P. Smart, "Cryptography Made Simple", Springer, 2016.
- [6] Lee-Chao, "Networking Systems Design and Development", CRC Press, 2009.
- [7] William Stallings, "Cryptography and Network Security Principles and Practices", Prentice Hall, 2005.
- [8] Văn Thiên Hoàng, Nguyễn Trọng Minh Hồng Phước, "Giáo trình Bảo mật thông tin", HUTECH, 2024.