

TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ



BÁO CÁO THỰC HÀNH
KỸ THUẬT LẬP TRÌNH

MÃ HỌC PHẦN: ELE20004

SINH VIÊN THỰC HIỆN : ĐÀO QUANG HUY

MÃ SỐ SINH VIÊN : 235752021610051

GIẢNG VIÊN HƯỚNG DẪN : Thầy MAI THẾ ANH

NGHỆ AN - 2024

LỜI NÓI ĐẦU

Lập trình là một môn học và lĩnh vực nghiên cứu trong khoa học máy tính, liên quan đến việc thiết kế, viết mã nguồn, kiểm thử và bảo trì các phần mềm. Lập trình cho phép con người giao tiếp với máy tính để thực hiện các tác vụ hoặc giải quyết các vấn đề thông qua các chương trình máy tính.

MỤC LỤC

Trang

BÀI 1. Thực hiện các thuật toán bằng phần mềm Flowgorithm.....	
Hình 1. Bắt đầu và kết thúc chương trình trong flowgorithm.....	
Hình 2. Chương trình đã thêm các khối chức năng.....	
Hình 3. Chương trình sau khi định nghĩa cho các khối chức năng thêm vào....	
Hình 4. Cửa sổ điều khiển hiện thị kết quả.....	
Hình 5. Sơ đồ giải thuật bài toán tính diện tích hình tròn.....	
Hình 6. Giải thuật bài toán nhập và in số lượng chai sử dụng vòng lặp for.....	
Hình 7. Sử dụng hàm cho giải thuật tính diện tích hình tròn.....	
Hình 8. Sơ đồ bài toán tính diện tích tam giác.....	
Hình 9. Xây dựng thuật toán kiểm tra tính nguyên tố.....	
BÀI 2. Các cú pháp, kiểu dữ liệu, lệnh điều khiển trong lập trình Python.	
Hình 1. Sửa lỗi thụt lề và dấu hai chấm ở đoạn mã và chạy kết quả.....	
Hình 2. Chương trình nhập hai điểm và tính khoảng cách	
Hình 3. Chương trình nhập vào một số và kiểm tra số chẵn hay lẻ.....	
Hình 4. Chương trình in số nghịch đảo và kết quả dưới dạng thập phân.....	
Hình 5. Nhập số tự nhiên và in kết quả giảm dần	
Hình 6. Các số chia hết cho n (thu được và in trên màn hình , cách nhau bằng dấu phẩy).....	
Hình 7. Chương trình tạo một dictionary bao gồm cả 1 và n.....	
Hình 8. Chương trình in dãy số Fibonacci và tìm tổng dãy đã in.....	
Hình 9. Chương trình đếm số thứ tự trong 1 xâu ký tự và lưu ký tự vào từ điển.....	
Hình 10. Chương trình sử dụng các phương thức split và join để tách nhập xâu ký tự.....	
Hình 11. Viết chương trình kết nối các danh sách vào từ điển.....	
Hình 12. Chương trình kiểm tra tính hợp lệ của mật khẩu mà người dùng nhập vào.	
Hình 13. Chương trình giải phương trình bậc 2: $ax^2 + bx + c = 0$	
BÀI 3. Lập trình hàm trong Python.....	
Hình 1. Viết hàm sum() tính tổng hai số.	
Hình 2. Viết hàm sum() với kết quả trả về.....	
Hình 3. Tìm và sửa lỗi chương trình.	
Hình 4. Viết chương trình có phạm vi biến như sau.....	
Hình 5. Viết chương trình và xem sự thay đổi của biến.	
Hình 6. Viết chương trình và giải thích việc truyền tham số của hàm.....	

Hình 7. Định nghĩa hàm có thể chấp nhận input là số nguyên và in ra màn hình.....	
Hình 8. Viết chương trình để di chuyển một con robot.....	
Hình 9. Chương trình máy tính thực hiện phép tính đơn giản.	
Hình 10. Viết hàm tính chu vi và diện tích hình tròn.....	
Hình 11. Viết hàm tính số tiền lãi nhận được từ số vốn gửi ngân hàng.....	
BÀI 4. Các kiểu dữ liệu có cấu trúc trong Python.....	
Hình 1. Nhập chuỗi S và in ra từng kí tự của S, mỗi kí tự trên một dòng.....	
Hình 2. Bỏ qua không in những kí tự “không nhìn thấy(space và tab)” ở hình 1.....	
Hình 3. Các kí tự ở dạng in hoa ở hình 1.....	
Hình 4. Danh sách có mỗi phần tử cách nhau bởi dấu space hoặc tab và in ra màn hình.....	
Hình 5. Nhập 1 danh sách các từ từ bàn phím và in theo thứ tự ngược lại thứ tự vừa nhập.....	
Hình 6. Nhập một tên người từ bàn phím, hãy tách phân họ và tên riêng của người đó và in chúng ra màn hình	
Hình 7. Nhập một chuỗi từ bàn phím và loại bỏ tất cả các chữ số khỏi chuỗi và in lại nội dung chuỗi mới ra màn hình.....	
Hình 8. Nhập dãy các từ từ bàn phím và in ra từ dài nhất trong dãy vừa nhập, in ra mọi từ có cùng độ dài nhất.	
Hình 9. Nhập một list từ bàn phím.	
Hình 10. Cắt list: lấy list nhưng bỏ phần tử đầu và cuối.	
Hình 11. Thêm phần tử vào list.	
Hình 12. Bỏ phần tử khỏi list.....	
Hình 13. Tìm kiếm phần tử trong list.....	
Hình 14. Sắp xếp các phần tử trong list.	
Hình 15. Nhập liên tiếp các từ tiếng Anh viết tách nhau bởi dấu cách. Nhập chuỗi đầu vào và tách các từ sau đó in ra màn hình các từ đó theo từ điển.	
Hình 16. Chuỗi các số nhị phân viết liên tiếp được nối nhau bởi dấu phẩy. Nhập chuỗi đầu vào sau đó in ra những giá trị được nhập.	
Hình 17. Nhập số n, in ra màn hình các số nguyên dương nhỏ hơn n có tổng các ước số lớn hơn chính nó.....	
Hình 18. Hãy nhập số nguyên , tạo một list gồm các số fibonacci nhỏ hơn n và in ra màn hình.	
Hình 19. Hãy tạo ra tuple P gồm các số nguyên tố nhỏ hơn 1 triệu.....	
Hình 20. Nhập n, in n dòng đầu tiên của tam giác pascal.....	
Hình 21. Chuỗi các số nhị phân 4 chữ số, phân tách bởi dấu phẩy, kiểm tra xem chúng có chia hết cho 5 không và in các số chia hết cho 5 thành dãy phân tách bởi dấu phẩy.	
Hình 22. Viết một chương trình tìm tất cả các số trong một đoạn sao cho tất cả các chữ số trong đó là số chẵn và in thành chuỗi cách nhau bởi dấu phẩy trên một dòng.	

Hình 23. Chương trình chấp nhận đầu vào là một câu, đếm số chữ cái và chữ số trong câu đó.....

Hình 24. Chương trình chấp nhận đầu vào là một câu, đếm chữ hoa, chữ thường.

Hình 25. Sử dụng một danh sách để lọc các số lẻ từ danh sách được người dùng nhập vào.....

Hình 26. Chương trình tính số tiền thực hiện của một tài khoản ngân hàng dựa trên nhật ký giao dịch được nhập vào từ giao diện điều khiển.

BÀI 5. Thiết kế module trong Python.....

Hình 1. Một Module toán học gọi là mymath và sử dụng module này từ một tập lệnh riêng biệt.....

Hình 2. Sử dụng thư viện tiêu chuẩn của Python (datetime).....

Hình 3. Viết chương trình sử dụng thư viện NumPy để tạo một mảng với các giá trị nằm trong khoảng từ 12 đến 38.....

Hình 4. Chương trình để tạo một mảng với các giá trị nằm trong khoảng từ 12 đến 38 và đảo ngược mảng đã tạo.....

Hình 5. Chương trình tìm phần tử lớn nhất và nhỏ nhất của một danh sách.....

Hình 6. In ra vị trí phần tử lớn nhất và nhỏ nhất tìm được ở bài tập trên.

Hình 7. Chương trình sử dụng thư viện Numpy.....

Hình 8. Xây dựng hàm Sequential_Search và viết chương trình nhập 1 dlist.....

Hình 9. Xây dựng hàm Binary_search và chương trình nhập một list.

Hình 10. Xây dựng hàm BubbleSort dưới dạng module.....

Hình 11. Chương trình sử dụng thư viện NumPy và sắp xếp theo yêu cầu.....

Hình 12. Sử dụng thư viện NumPy để sắp xếp id cho sinh viên.....

BÀI 6. Lập trình hướng đối tượng trong Python.....

BÀI 7. Thao tác trên tập tin và thư mục trong Python.....

BÀI 8. Lập trình giao diện trong Python.....

Bài 1. Thực hiện các thuật toán bằng phần mềm Flowgorithm

1.1. Mục đích

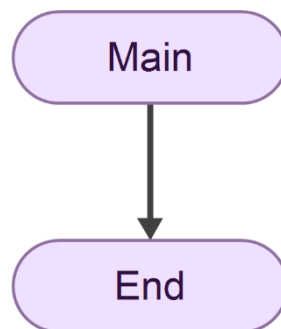
- Sử dụng phần mềm flowgorithm trong thiết kế và biểu diễn thuật toán.
- Xây dựng thuật toán cho các bài toán cụ thể trên flowgorithm.

1.2. Các bước thực hiện và kết quả

1.2.1. Xây dựng thuật toán hiển thị chuỗi ký tự với Flowgorithm

B1: Tạo chương trình mới

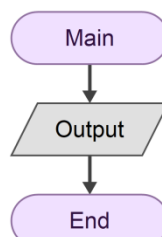
- Khi bắt đầu một sơ đồ mới, chúng ta sẽ thấy hai hình chữ nhật tròn được gọi là “terminals”. Những biểu tượng này đại diện cho sự bắt đầu và kết thúc chương trình của bạn.
- Nhiều sơ đồ thuật toán hiển thị văn bản “Begin” trong terminal. Flowgorithm sử dụng văn bản “Main”. Hầu hết các ngôn ngữ lập trình bắt đầu với các nỗ lực “Main” và Flowgorithm cũng vậy.



Hình 1. Bắt đầu và kết thúc chương trình

B2: Thêm các khối chức năng

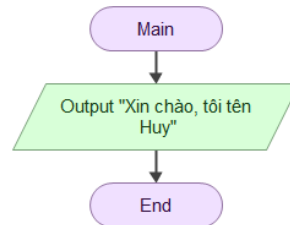
- Tất cả mọi thứ trong một sơ đồ được thể hiện bằng các khối hình. Các khối được thêm vào giữa terminals Main và End.
- Để thêm các hình dạng, di chuyển con trỏ chuột của bạn trên một dòng. Nếu có thể thêm một khối, dòng sẽ chuyển sang màu cam.
- Bấm đúp hoặc bấm chuột phải để thêm hình.
- Hiện thị giao diện với các khối cần thêm.
- Lựa chọn và click vào khối cần thêm sẽ được chương trình như hình 2.



Hình 2. Chương trình đã thêm các khối chức năng

B3. Định nghĩa chức năng cho khối

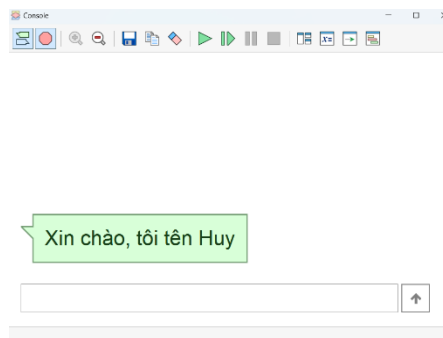
- Cửa sổ "Output Properties" xuất hiện, chúng ta có thể nhập vào các chức năng theo cú pháp quy định, chương trình sau khi định nghĩa chức năng được chỉ ra trong hình 3.



Hình 3. Chương trình sau khi định nghĩa cho các khối chức năng thêm vào

B4. Khởi chạy chương trình đã thiết kế.

- Bấm F5 hoặc nút "Run" trên menu của chương trình.

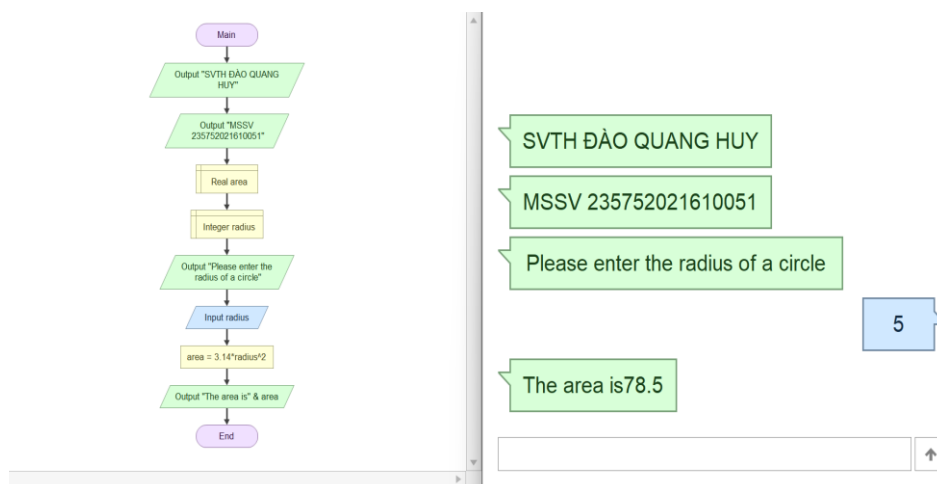


Hình 4. Cửa sổ điều khiển hiển thị kết quả

B5: Xem mã nguồn hoặc các biến sử dụng trong chương trình sử dụng menu "Tools" ở thanh công cụ.

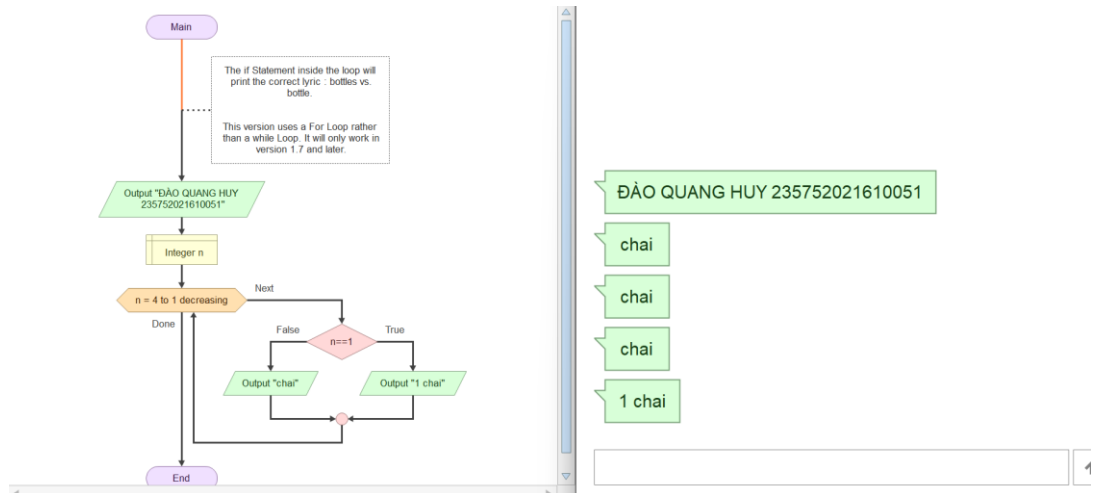
1.2.2. Sử dụng Flowgorithm xây dựng chương trình giải quyết các bài toán.

a. Tính diện tích hình tròn



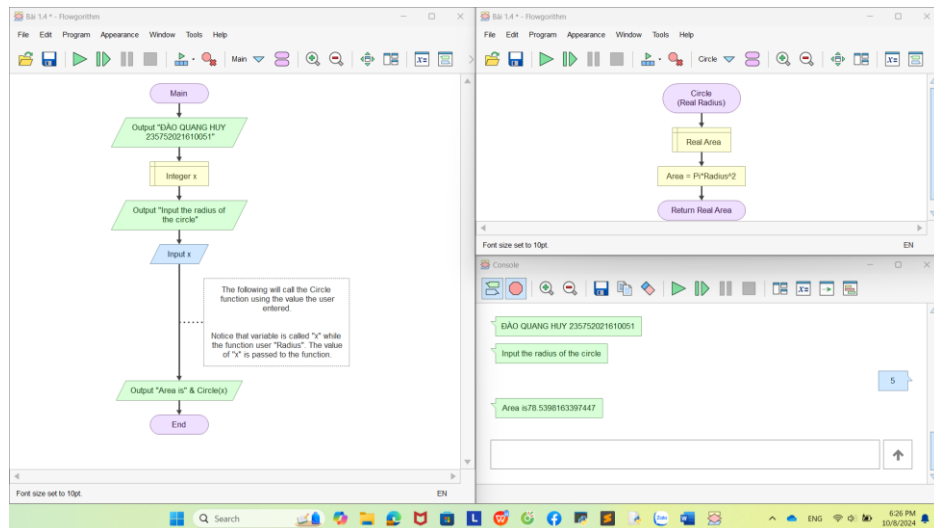
Hình 5. Sơ đồ giải thuật bài toán tính diện tích hình tròn

b. Nhập và in số lượng chai sử dụng vòng lặp for.



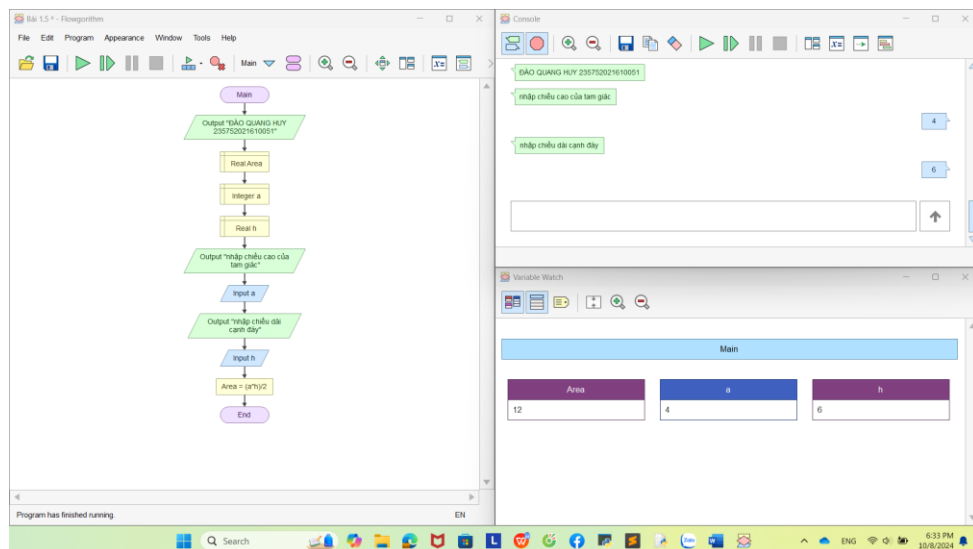
Hình 6. Giải thuật bài toán nhập và in số lượng chai sử dụng vòng lặp for

c. Tính diện tích hình tròn sử dụng hàm (chương trình con).



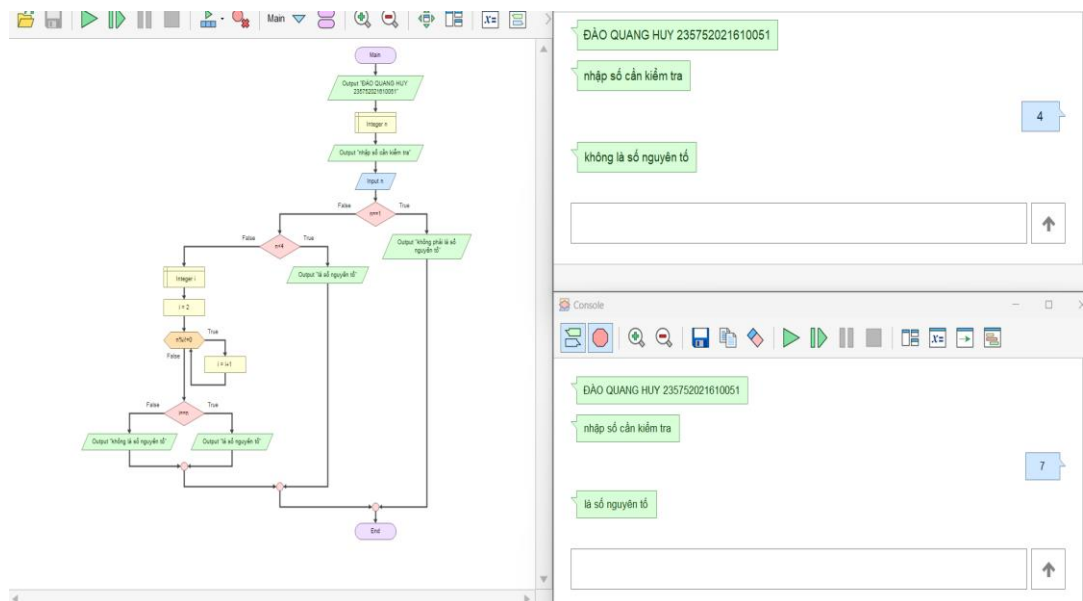
Hình 7. Sử dụng hàm cho giải thuật tính diện tích hình tròn

d. Viết chương trình nhập và cạnh và chiều cao tương ứng của một tam giác và in ra màn hình diện tích tam giác.



Hình 8. Sơ đồ bài toán tính diện tích tam giác

- e. Xây dựng thuật toán kiểm tra tính nguyên tố (một số nguyên dương n là số nguyên tố khi chỉ có hai ước là 1 và chính nó)



Hình 9. Xây dựng thuật toán kiểm tra tính nguyên tố.

1.3. Câu hỏi kiểm tra.

Câu 1: Ý nghĩa các khối sử dụng trong chương trình

- Các khối cơ bản:
 - + Khối bắt đầu (Start): Đánh dấu vị trí bắt đầu của chương trình.
 - + Khối kết thúc (End): Đánh dấu vị trí kết thúc của chương trình.
 - + Khối nhập dữ liệu (Input): Dùng để nhập dữ liệu từ bàn phím vào chương trình.
 - + Khối xuất dữ liệu (Output): Dùng để hiển thị kết quả ra màn hình.
 - + Khối gán giá trị (Assignment): Gán một giá trị cho một biến.
- Các khối điều khiển
 - + Khối điều kiện (If-Then-Else): Kiểm tra một điều kiện. Nếu điều kiện đúng thì thực hiện một nhóm lệnh, ngược lại thì thực hiện một nhóm lệnh khác.
 - + Khối lặp (Loop): Thực hiện một nhóm lệnh nhiều lần cho đến khi một điều kiện nào đó được thỏa mãn.
 - + Khối lặp for: Thực hiện một nhóm lệnh một số lần xác định trước.
 - + Khối lặp while: Thực hiện một nhóm lệnh cho đến khi một điều kiện nào đó trở thành sai.

Câu 2. Cách khai báo nhập dữ liệu cho các biến, các khối chức năng và vòng lặp.

- Khai báo biến
 - + Thêm biến: Trong Flowgorithm, bạn có thể thêm biến bằng cách sử dụng khối "Declare" (Khai báo).
 - + Chọn kiểu dữ liệu: Bạn có thể chọn kiểu dữ liệu cho biến như Integer, Float, String, Boolean, v.v.
 - + Đặt tên biến: Đặt tên cho biến theo quy tắc đặt tên trong lập trình.

- Nhập dữ liệu:
 - + Sử dụng khối Input: Để nhập dữ liệu cho biến, bạn sử dụng khối "Input".
 - + Chọn biến: Trong khối Input, bạn chọn biến mà bạn muốn gán giá trị từ đầu vào.
 - + Thông điệp: Có thể thêm thông điệp để yêu cầu người dùng nhập dữ liệu.
- Khối chức năng:
 - + Tạo khối chức năng: Sử dụng khối "Call" để gọi một hàm hoặc khối chức năng.
 - + Tham số: Nếu hàm cần tham số, bạn có thể khai báo và truyền tham số vào khi gọi hàm.
- Vòng lặp:
 - + Sử dụng khối Loop: Bạn có thể sử dụng khối "For" hoặc "While" để tạo vòng lặp.
 - + Điều kiện lặp: Trong khối vòng lặp, bạn cần xác định điều kiện để tiếp tục lặp lại. Ví dụ: số lần lặp trong khối "For" hoặc điều kiện kiểm tra trong khối "While".
 - + Thao tác bên trong vòng lặp: Bạn có thể thực hiện các thao tác hoặc gọi các khối chức năng bên trong vòng lặp.

Câu 3. Xây dựng hàm thực hiện chương trình con trong Flowgorithm.

- Mở Flowgorithm
 - + Mở ứng dụng Flowgorithm trên máy tính của bạn.
- Tạo Hàm Mới
 - + Chọn khối "Function": Trong Flowgorithm, chọn khối "Function" từ thanh công cụ.
 - + Đặt tên cho hàm: Bạn cần đặt tên cho hàm của mình. Ví dụ: TínhTổng.
 - + Khai báo tham số (nếu cần): Nếu hàm của bạn cần tham số đầu vào, bạn có thể khai báo chúng trong phần "Parameters". Ví dụ, nếu hàm cần một tham số là số nguyên n, bạn sẽ thêm một tham số có tên n và kiểu Integer.
- Xây dựng Logic trong Hàm
 - + Thêm các khối lệnh: Bạn có thể thêm các khối lệnh như Input, Output, Assignment, Loop, v.v. để xây dựng logic của hàm.
- Gọi Hàm
 - + Quay trở lại chương trình chính: Sau khi hoàn thành hàm, bạn quay lại chương trình chính.
 - + Sử dụng khối "Call": Để gọi hàm, bạn sử dụng khối "Call" và chọn hàm mà bạn đã tạo.
 - + Truyền tham số (nếu có): Nếu hàm cần tham số, bạn phải cung cấp giá trị cho nó khi gọi hàm.

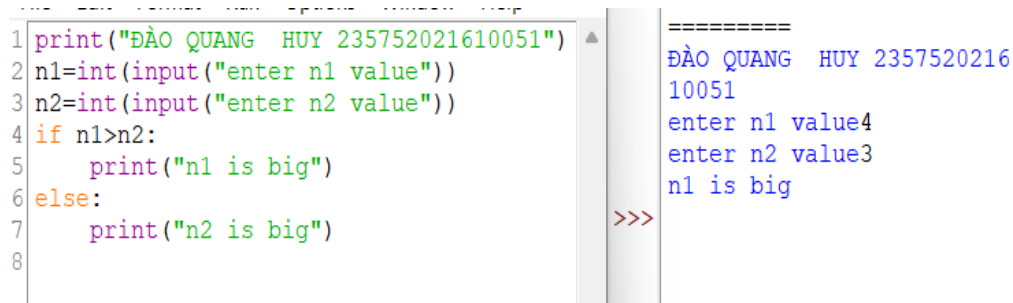
Bài 2. Các cú pháp , kiểu dữ liệu , lệnh điều khiển trong lập trình Python.

1.1. Mục đích.

- Việc sử dụng các cú pháp, kiểu dữ liệu và lệnh điều khiển trong lập trình Python là nền tảng để xây dựng các chương trình máy tính. Mỗi yếu tố này đóng vai trò quan trọng trong việc tạo ra các chương trình có cấu trúc rõ ràng, hiệu quả và dễ bảo trì.
- Viết các chương trình rõ ràng , dễ hiểu mã nguồn sẽ được tổ chức một cách logic và dễ bảo trì.
- Tránh các lỗi ,việc tuân thủ cú pháp và sử dụng kiểu dữ liệu phù hợp sẽ giúp giảm thiểu các lỗi khi chạy chương trình .
- Tạo ra các chương trình mạnh mẽ và linh hoạt , có thể xử lý nhiều tình huống khác nhau và đáp ứng các yêu cầu của người dùng.

1.2. Tiến hành thực hành.

1.2.1. Viết đoạn chương trình và sửa lỗi.



```
1 print("ĐÀO QUANG HUY 235752021610051")
2 n1=int(input("enter n1 value"))
3 n2=int(input("enter n2 value"))
4 if n1>n2:
5     print("n1 is big")
6 else:
7     print("n2 is big")
8
```

=====

ĐÀO QUANG HUY 235752021610051

enter n1 value4

enter n2 value3

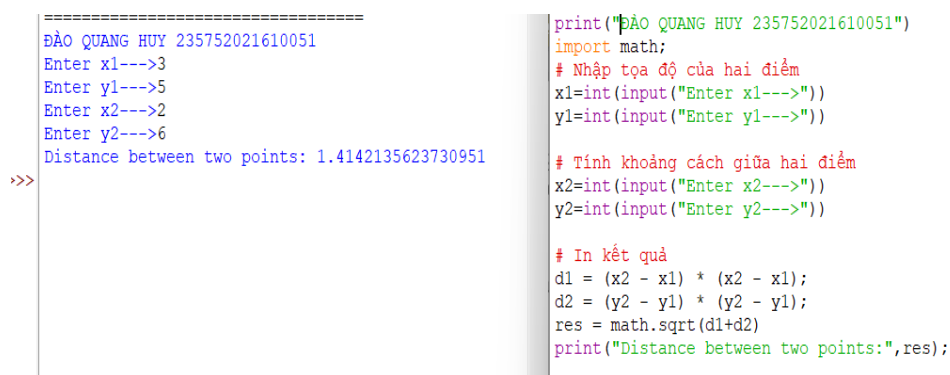
n1 is big

>>>

Hình 1. Sửa lỗi thực tế và dấu hai chấm ở đoạn mã và chạy kết quả

Chú ý: Việc sửa lỗi thực tế đã giúp đoạn mã chạy chính xác và in ra kết quả đúng. Đây là một lỗi rất phổ biến khi bắt đầu học Python, vì vậy hãy chú ý đến việc thực tế đúng cách trong các chương trình của bạn. Làm như vậy đoạn mã trong chương trình mới chạy được .

1.2.2. Viết chương trình nhập hai điểm và tính khoảng cách.



```
=====
ĐÀO QUANG HUY 235752021610051
Enter x1--->3
Enter y1--->5
Enter x2--->2
Enter y2--->6
Distance between two points: 1.4142135623730951
>>>
```

```
print("ĐÀO QUANG HUY 235752021610051")
import math;
# Nhập tọa độ của hai điểm
x1=int(input("Enter x1--->"))
y1=int(input("Enter y1--->"))

# Tính khoảng cách giữa hai điểm
x2=int(input("Enter x2--->"))
y2=int(input("Enter y2--->"))

# In kết quả
d1 = (x2 - x1) * (x2 - x1);
d2 = (y2 - y1) * (y2 - y1);
res = math.sqrt(d1+d2)
print("Distance between two points:",res);
```

Hình 2. Chương trình nhập hai điểm và tính khoảng cách.

Chú ý:

- Chương trình bắt đầu bằng việc nhập thư viện (math.)
- Người dùng nhập tọa độ của hai điểm.
- Chương trình tính bình phương của hiệu các tọa độ x và y.
- Sau đó, chương trình tính căn bậc hai của tổng hai bình phương này để tìm ra khoảng cách giữa hai điểm.
- Cuối cùng, kết quả được in ra màn hình.

1.2.3. Viết chương trình nhập vào một số và kiểm tra số đó là chẵn hay lẻ, in thông báo ra màn hình .

```
= RESTART: D:\Thực hành lập trình\Bài 2 Cú pháp
, dữ liệu ,lệnh điều khiển python\Bài 2.3.py
ĐÀO QUANG HUY MSSV 235752021610051
Enter a number--->6
Even Number

>>

= RESTART: D:\Thực hành lập trình\Bài 2 Cú pháp
, dữ liệu ,lệnh điều khiển python\Bài 2.3.py
ĐÀO QUANG HUY MSSV 235752021610051
Enter a number--->3
ODD Number

>>
```

```
File Edit Format Run Options Window Help
print("ĐÀO QUANG HUY MSSV 235752021610051")
n=int(input("Enter a number--->"))
if n % 2 == 0:
    print ("Even Number");
else:
    print ("ODD Number");
```

Hình 3. Chương trình kiểm tra số chẵn hay lẻ

Chú thích:

- Cách thức hoạt động của đoạn mã trên.
 - + Chương trình yêu cầu người dùng nhập vào một số nguyên.
 - + Số nguyên đó được kiểm tra xem có chia hết cho 2 không (tức là có phải là số chẵn không).
 - + Nếu số đó chia hết cho 2, chương trình in ra "EVEN Number".
 - + Ngược lại, chương trình in ra "ODD Number".

1.2.4. Viết chương trình in ra màn hình số nghịch đảo và kết quả dưới dạng thập phân của một số tự nhiên trong khoảng (a,b).

```
ĐÀO QUANG HUY MSSV 235752021610051
i: 1 j: 2
1 / 2
0.5
i: 1 j: 3
1 / 3
0.3333333333333333
i: 1 j: 4
1 / 4
0.25
i: 1 j: 5
1 / 5
0.2
i: 1 j: 6
1 / 6
0.16666666666666666
i: 1 j: 7
1 / 7
0.14285714285714285
i: 1 j: 8
1 / 8
0.125
i: 1 j: 9
1 / 9
0.11111111111111111
>>>
```

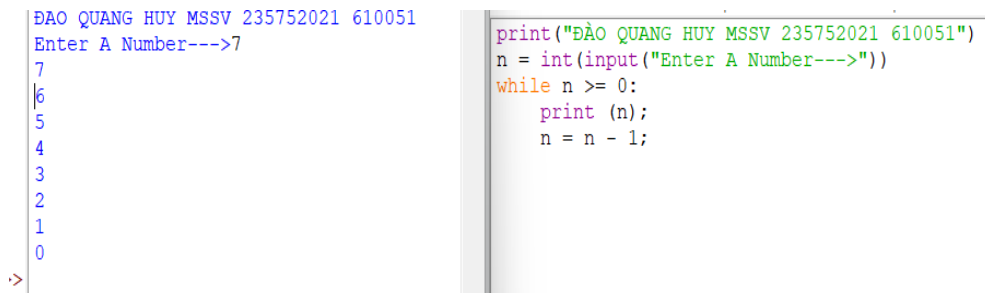
```
print("ĐÀO QUANG HUY MSSV 235752021610051")
i=1;
for j in range(2,10) :
    print("i:",i,"j:",j)
    print(i,"/",j)
    print (i/j);
```

Hình 4 .Chương trình in số nghịch đảo và kết quả dưới dạng thập phân

Chú thích:

- Khai báo biến i:
 - + `i=1;`: Dòng này khai báo một biến có tên là i và gán giá trị ban đầu cho nó là 1.
- Vòng lặp for:
 - + `for j in range(2,10):`: Dòng này bắt đầu một vòng lặp for. Biến j sẽ lần lượt nhận các giá trị từ 2 đến 9 (không bao gồm 10). Mỗi lần lặp, các lệnh bên trong vòng lặp sẽ được thực hiện.
- In ra thông tin:
 - + `print("i:",i,"j:",j)`: In ra giá trị hiện tại của i và j trong mỗi lần lặp.
 - + `print(i,"/",j)`: In ra phép chia i/j dưới dạng chuỗi.
 - + `print (i/j);`: Thực hiện phép chia i/j và in ra kết quả.

1.2.5. Viết chương trình nhập vào một số tự nhiên $n > 0$, in ra màn hình các số tự nhiên giảm dần từ n đến 0, mỗi ký tự in trên 1 hàng.



```
ĐÀO QUANG HUY MSSV 235752021 610051
Enter A Number--->7
7
6
5
4
3
2
1
0
>
```

```
print("ĐÀO QUANG HUY MSSV 235752021 610051")
n = int(input("Enter A Number--->"))
while n >= 0:
    print(n)
    n = n - 1;
```

Hình 5. Nhập số tự nhiên và in kết quả giảm dần (n đến 0), 1 ký tự trên 1 hàng.

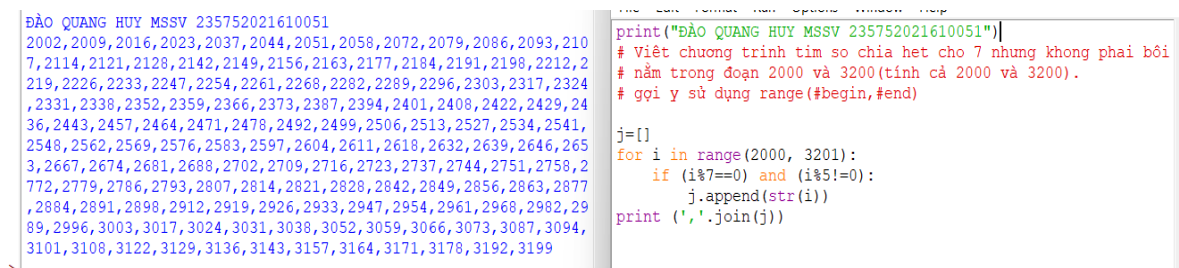
Chú thích :

- `n = int(input("Enter A Number--->"));`
+ `input("Enter A Number--->")`: Hiển thị thông báo "Enter A Number--->" trên màn hình và chờ người dùng nhập vào một giá trị.
+ `int()`: Chuyển đổi giá trị nhập vào (dưới dạng chuỗi) thành số nguyên và gán cho biến `n`.
- `while n >= 0:`
+ Đây là một vòng lặp `while`. Vòng lặp này sẽ tiếp tục chạy cho đến khi điều kiện `n >= 0` trở thành sai (tức là khi `n` nhỏ hơn 0).
- `print(n);`
+ In ra giá trị hiện tại của biến `n` trên một dòng mới.
- `n = n - 1;`
+ Giảm giá trị của biến `n` đi 1 đơn vị. Sau mỗi lần lặp, giá trị của `n` sẽ giảm dần.

Chú ý:

- Cách hoạt động của chương trình:
Nhập số :
Người dùng nhập vào một số nguyên dương n .
Vòng lặp:
+Chương trình kiểm tra xem n có lớn hơn hoặc bằng 0 không.
+Nếu đúng, chương trình in ra giá trị của n và giảm n đi 1.
+Quá trình này lặp lại cho đến khi n nhỏ hơn 0.

1.2.6. Viết chương trình tìm tất cả các số chia hết cho 7 nhưng không phải bội số của 5, nằm trong đoạn 2000 và 3200 (tính cả 2000 và 3200) . Các số thu được sẽ được in thành chuỗi trên một dòng, cách nhau bằng dấu phẩy.



```
ĐÀO QUANG HUY MSSV 235752021610051
2002, 2009, 2016, 2023, 2037, 2044, 2051, 2058, 2072, 2079, 2086, 2093, 210
7, 2114, 2121, 2128, 2142, 2149, 2156, 2163, 2177, 2184, 2191, 2198, 2212, 2
219, 2226, 2233, 2247, 2254, 2261, 2268, 2282, 2289, 2296, 2303, 2317, 2324
, 2331, 2338, 2352, 2359, 2366, 2373, 2387, 2394, 2401, 2408, 2422, 2429, 24
36, 2443, 2457, 2464, 2471, 2478, 2492, 2499, 2506, 2513, 2527, 2534, 2541,
2548, 2562, 2569, 2576, 2583, 2597, 2604, 2611, 2618, 2632, 2639, 2646, 265
3, 2667, 2674, 2681, 2688, 2702, 2709, 2716, 2723, 2737, 2744, 2751, 2758, 2
772, 2779, 2786, 2793, 2807, 2814, 2821, 2828, 2842, 2849, 2856, 2863, 2877
, 2884, 2891, 2898, 2912, 2919, 2926, 2933, 2947, 2954, 2961, 2968, 2982, 29
89, 2996, 3003, 3017, 3024, 3031, 3038, 3052, 3059, 3066, 3073, 3087, 3094,
3101, 3108, 3122, 3129, 3136, 3143, 3157, 3164, 3171, 3178, 3192, 3199
>
```

```
print("ĐÀO QUANG HUY MSSV 235752021610051")
# Viết chương trình tìm số chia hết cho 7 nhưng không phải bội
# nằm trong đoạn 2000 và 3200 (tính cả 2000 và 3200).
# gọi y sử dụng range(begin, end)

j=[]
for i in range(2000, 3201):
    if (i%7==0) and (i%5!=0):
        j.append(str(i))
print(','.join(j))
```

Hình 6. Các số chia hết cho n (thu được và in trên màn hình, cách nhau bằng dấu phẩy)

Chú thích : Các bước thực hiện :

- Khởi tạo danh sách.
 - + Tạo một danh sách rỗng (j) để lưu trữ các số thỏa mãn điều kiện.
- Vòng lặp for.
 - + Vòng lặp này sẽ lặp qua tất cả các số nguyên từ 2000 đến 3200 (bao gồm cả 3200). Biến i sẽ lần lượt nhận giá trị của các số trong khoảng này.
- Kiểm tra điều kiện .
 - + $i \% 7 == 0$: Kiểm tra xem số i có chia hết cho 7 không. Nếu chia hết thì kết quả của phép chia lấy dư sẽ bằng 0.
 - + $i \% 5 != 0$: Kiểm tra xem số i có chia hết cho 5 không. Nếu không chia hết thì kết quả của phép chia lấy dư sẽ khác 0.
 - + Nếu cả hai điều kiện trên đều đúng (tức là số i chia hết cho 7 nhưng không chia hết cho 5) thì khối lệnh bên trong if sẽ được thực hiện.
- Thêm số vào danh sách.
 - + Nếu số i thỏa mãn điều kiện, chúng ta chuyển đổi i thành chuỗi bằng str(i) và thêm vào danh sách j bằng phương thức append().
- In kết quả .

Sau khi kết thúc vòng lặp, chúng ta sử dụng ', '.join(j) để nối tất cả các phần tử trong danh sách j thành một chuỗi, với các phần tử được phân cách bởi dấu phẩy. Sau đó, chuỗi kết quả được in ra màn hình.

1.2.7. Với số nguyên n nhất định , hãy viết chương trình để tạo ra một dictionary chứa (i,i*i) như là số nguyên từ 1 đến n(bao gồm cả 1 và n) sau đó in ra dictionary này

```
ĐÀO QUANG HUY MSSV 235752021610051
Nhập vào một số:8
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}
>>

print("ĐÀO QUANG HUY MSSV 235752021610051")
n=int(input("Nhập vào một số:"))
d=dict()
for i in range(1,n+1):
    d[i]=i*i

print (d)
```

Hình 7. Chương trình tạo một dictionary (chứa (i,i*i) bao gồm cả 1 và n .

Chú thích :

- Nhập số nguyên n.
 - + Dòng này yêu cầu người dùng nhập vào một số nguyên và gán giá trị đó cho biến n. Số nguyên này sẽ xác định giới hạn của các số mà chúng ta muốn tạo cặp (i, i*i).
- Khởi tạo từ điển.
 - + Dòng này tạo một từ điển rỗng và gán cho biến d. Từ điển sẽ được sử dụng để lưu trữ các cặp (key, value) mà chúng ta muốn tạo.
- Vòng lặp For
 - + Vòng lặp này sẽ lặp qua các giá trị của i từ 1 đến n (bao gồm cả n). Trong mỗi lần lặp, biến i sẽ nhận một giá trị mới.
- Tạo cặp (key, value) và thêm vào từ điển:
 - + Trong mỗi lần lặp, chúng ta tạo một cặp (key, value) mới.
 - i là key (khóa) của cặp.
 - i*i là value (giá trị) tương ứng với key i.
 - + Cặp này được thêm vào từ điển d.
- In kết quả . Cuối cùng, chúng ta in ra toàn bộ từ điển d để xem kết quả.

1.2.8. Viết chương trình in ra màn hình dãy số Fibonacci nhỏ hơn 4.000.000, tìm tổng các số chẵn trong dãy đã in.

```
ĐÀO QUANG HUY MSSV 235752021610051
1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4
181 6765 10946 17711 28657 46368 75025 121393 196418 317
811 514229 832040 1346269 2178309 3524578 5702887
sum of prime numbers term in fibonacci series: 4613732

print("ĐÀO QUANG HUY MSSV 235752021610051")
a, b = 1, 2
total = 0
print(a, end=" ")
while (a <= 4000000-1):
    if a % 2 == 0:
        total += a
    a, b = b, a+b
    print(a, end=" ")
print("\n sum of prime numbers term in fibonacci series: ", total)
```

Hình 8. Chương trình in dãy số Fibonacci và tìm tổng dãy đã in

- **Chú thích:**
 - + **a, b = 1, 2:** Khởi tạo hai biến a và b với giá trị ban đầu lần lượt là 1 và 2, đây là hai số Fibonacci đầu tiên.
 - + **total = 0:** Khởi tạo biến total bằng 0 để lưu trữ tổng các số chẵn.
 - + **print(a, end=""):** In ra số Fibonacci đầu tiên (là 1) và không xuống dòng.
 - + **while (a <= 4000000-1):** Vòng lặp while sẽ tiếp tục cho đến khi giá trị của a lớn hơn hoặc bằng 3.999.999 (4.000.000 - 1).
 - + **if a % 2 == 0:** Kiểm tra xem a có phải là số chẵn không (chia hết cho 2).
 - + **total += a:** Nếu a là số chẵn, thì cộng a vào total.
 - + **a, b = b, a+b:** Cập nhật giá trị của a và b để tính số Fibonacci tiếp theo.
 - + **print(a, end=""):** In ra số Fibonacci mới tính được và không xuống dòng.
 - + **print("\n sum of prime numbers term in fibonacci series: ", total):** Lưu ý: Sau khi kết thúc vòng lặp, in ra một dòng mới và thông báo tổng các số chẵn trong dãy Fibonacci.

1.2.9. Viết chương trình đếm số ký tự trong 1 xâu ký tự nhập từ bàn phím, lưu các ký tự vào cấu trúc từ điển .

```
print("ĐÀO QUANG HUY MSSV 235752021610051")
str=input("Enter a String:")
dict = {}
for n in str:
    keys = dict.keys()
    if n in keys:
        dict[n] +=1
    else:
        dict[n] =1
print (dict)

ĐÀO QUANG HUY MSSV 235752021610051
Enter a String:Đào Quang Huy
{'Đ': 1, 'à': 1, 'o': 1, ' ': 2, 'Q': 1, 'u': 2, 'a': 1, 'n': 1, 'g': 1, 'H': 1, 'y': 1}
```

Hình 9.1. Chương trình đếm số ký tự trong 1 xâu ký tự

Chú thích :

- Dòng lệnh **str = input("Enter a String:")** yêu cầu người dùng nhập vào một chuỗi và lưu chuỗi đó vào biến str.
- **dict = {}** tạo một từ điển rỗng để lưu trữ các ký tự và số lần xuất hiện của chúng.
- **for n in str:** là vòng lặp để duyệt qua từng ký tự trong chuỗi nhập vào. Mỗi ký tự được gán cho biến n.
- **keys = dict.keys()** lấy tất cả các khóa (ký tự) trong từ điển. Tuy nhiên, dòng này là thừa vì bạn có thể kiểm tra trực tiếp sự tồn tại của khóa trong từ điển mà không cần gọi dict.keys().
- **if n in keys:** kiểm tra xem ký tự n đã có trong từ điển chưa. Nếu có, ta sẽ tăng giá trị của nó lên 1 (tức là tăng số lần xuất hiện của ký tự).

- Nếu ký tự chưa có trong từ điển, ta thêm ký tự vào từ điển với giá trị là 1 (lần xuất hiện đầu tiên).

```

print('Đào quang huy MSSV 235752021610051')
str=input("Enter a String")
dict = {}
for i in str:
    dict[i] = str.count(i)
print (dict)

```

```

Đào quang huy MSSV 235752021610051
Enter a String:Đào quang huy
{' ': 3, 'd': 1, 'a': 2, 'o': 1, 'q': 1, 'u': 2, 'n': 1, 'g': 1, 'h': 1, 'y': 1}

```

Hình 9.2. Chương trình đếm số ký tự trong 1 xây ký tự

Chú thích :

- Chúng ta cũng có thể viết mã code theo cách này để đếm số ký tự trong 1 chuỗi ký tự nhập từ bàn phím . Kết quả khi chạy chương trình vẫn như trên.

1.2.10. Viết chương trình sử dụng các phương thức split và join để tách nhập chuỗi ký tự.

```

print('Đào quang huy MSSV 235752021610051')
a="hi i am dao quang huy"
b=a.split()
print (b)
c=" ".join(b)
print(c)

```

```

py
Đào quang huy MSSV 235752021610051
['hi', 'i', 'am', 'dao', 'quang', 'huy']
hi i am dao quang huy

```

Hình 10. Chương trình sử dụng split và join để tách nhập chuỗi ký tự.

Chú thích:

- Hàm split() được sử dụng để chia một chuỗi thành một danh sách các từ. Mặc định, split() sẽ chia chuỗi tại mỗi khoảng trắng (space) và tạo thành một danh sách các từ.
- Trong trường hợp này, chuỗi "hi i am dao quang huy" sẽ được chia thành các từ: ['hi', 'i', 'am', 'dao', 'quang', 'huy'].
- print(c): Dòng này sẽ in ra chuỗi đã được nối lại từ các từ trong danh sách b.
- **split()** chia một chuỗi thành các từ và trả về một danh sách.
- **join()** nối các phần tử trong một danh sách thành một chuỗi, sử dụng chuỗi mà bạn chọn làm "nối kết" (ở đây là dấu cách " ").

1.2.11. Viết chương trình kết nối các danh sách vào từ điển.

```

print('Đào quang huy MSSV 235752021610051')
l=[1,'python',4,7]
k=['cse',2,'guntur',8]
m=[]
m.append(l);
m.append(k);
print (m)
d={1:l,2:k,'combine_list':m}
print(d)

```

```

Đào quang huy MSSV 235752021610051
[[1, 'python', 4, 7], ['cse', 2, 'guntur', 8]]
{1: [1, 'python', 4, 7], 2: ['cse', 2, 'guntur', 8], 'combine_list': [[1, 'python', 4, 7], ['cse', 2, 'guntur', 8]]}

```

Hình 11. Chương trình kết nối các danh sách vào từ điển .

Chú thích :

- l = [1, 'python', 4, 7]: Danh sách l chứa các phần tử: số nguyên và chuỗi.
- k = ['cse', 2, 'guntur', 8]: Danh sách k chứa các chuỗi và số nguyên.
- m = []: Tạo một danh sách rỗng m.
- m.append(l): Thêm danh sách l vào m, kết quả là m = [[1, 'python', 4, 7]].
- m.append(k): Thêm danh sách k vào m, kết quả là m = [[1, 'python', 4, 7], ['cse', 2, 'guntur', 8]].
- d = {1: l, 2: k, 'combine_list': m}: Tạo một từ điển d có ba phần tử:

- + Khóa 1 trở đến danh sách l.
- + Khóa 2 trở đến danh sách k.
- + Khóa 'combine_list' trở đến danh sách m.

1.2.12. Một website yêu cầu người dùng nhập tên người dùng và mật khẩu để đăng ký. Viết chương trình để kiểm tra tính hợp lệ của mật khẩu mà người dùng nhập vào.

```
print('Đào quang huy MSSV 235752021610051')
import re

def check_password(password):
    """Kiểm tra tính hợp lệ của một mật khẩu

    Args:
        password: Mật khẩu cần kiểm tra

    Returns:
        True nếu mật khẩu hợp lệ, False nếu không
    """

    # Biểu thức chính quy để kiểm tra các tiêu chí
    pattern = r"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[!@#$%^])[A-Za-z\d!@#$%^]{6,12}$"
    return re.match(pattern, password) is not None

def main():
    """Chương trình chính"""

    passwords = input("Nhập các mật khẩu (cách nhau bởi dấu phẩy): ")
    password_list = passwords.split(',')

    valid_passwords = []
    for password in password_list:
        if check_password(password):
            valid_passwords.append(password)

    print("Các mật khẩu hợp lệ:", ', '.join(valid_passwords))

if __name__ == "__main__":
    main()
```

Hình 12. Tạo website yêu cầu người dùng nhập mật khẩu, kiểm tra tính hợp lệ.

Chú thích :

- Chương trình trên yêu cầu người dùng nhập 1 số mật khẩu từ bàn phím . Sau đó chương trình sẽ kiểm tra tính hợp lệ của mật khẩu và in ra kết quả hợp lệ từ số mật khẩu mà người dùng đã nhập vào.

1.2.13. Viết chương trình giải phương trình bậc 2: $ax^2 + bx + c = 0$, với các hệ số a, b, c nhập từ bàn phím.

```
print("SVTH:ĐÀO QUANG HUY MSV 235752021610051")
import math
print("giải hệ phương trình ax^2+bx+c=0")
a=float(input("nhập giá trị của a là:"))
b=float(input("nhập giá trị của b là:"))
c=float(input("nhập giá trị của c là:"))

if(a==0):
    if(b==0):
        if(c==0):
            print("Phương trình vô số nghiệm")
        else:
            print("Phương trình vô nghiệm")
    else:
        x = -c / b
        print(f"Phương trình có nghiệm duy nhất: x = {x}")
else:
    delta=b**2-4*a*c
    if delta>0:
        x1=(-b + math.sqrt(delta)) / (2*a)
        x2=(-b - math.sqrt(delta)) / (2*a)
        print("Phương trình có nghiệm kép:x = {x}")
        print(f"x1={x1}")
        print(f"x2={x2}")
    elif delta == 0:
        x = -b / (2*a)
        print(f"Phương trình có nghiệm kép:x = {x}")
    else:
        print("Phương trình vô nghiệm")
```

Hình 13. Chương trình giải phương trình bậc 2 với các hệ số nhập từ bàn phím .

Chú thích:

- Khi $a = 0$: Đây là phương trình bậc 1, nếu $b = 0$ nữa thì tùy thuộc vào giá trị của c sẽ có vô số nghiệm hoặc vô nghiệm.
- Khi $a \neq 0$: Bạn đang giải phương trình bậc 2. Phương trình này có thể có:
 - + 2 nghiệm phân biệt khi $\Delta > 0$.
 - + 1 nghiệm kép khi $\Delta = 0$.
 - + Vô nghiệm khi $\Delta < 0$.

1.3. Câu hỏi kiểm tra.

Câu 1. Các kiểu biến, khai báo và đặt tên biến trong python;

Bài 3. Lập trình hàm trong Python

1.1. Mục đích:

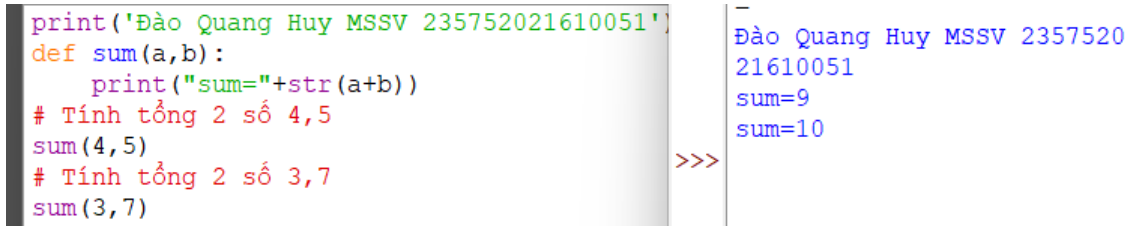
Giúp sinh viên nắm bắt các kiến thức trong lập trình python sử dụng hàm.

1.2. Cơ sở lý thuyết:

Xem các quy tắc khai báo hàm, gọi hàm, giá trị trả về, tham số truyền vào, tham số mặc định, phạm vi của biến trong python, sử dụng các hàm có sẵn trong các thư viện của python.

1.3. Các bước tiến hành:

1.3.1. Viết hàm sum() tính tổng hai số.



```
print('Đào Quang Huy MSSV 235752021610051')
def sum(a,b):
    print("sum="+str(a+b))
# Tính tổng 2 số 4,5
sum(4,5)
# Tính tổng 2 số 3,7
sum(3,7)
```

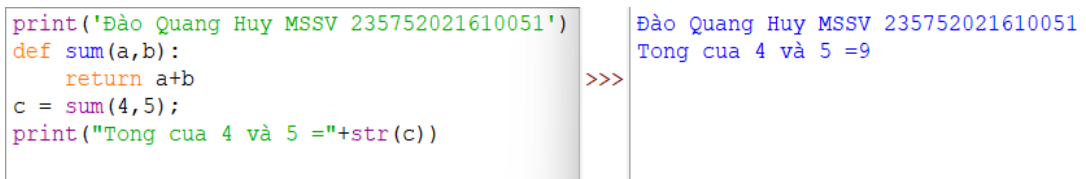
```
Đào Quang Huy MSSV 2357520
21610051
sum=9
sum=10
```

Hình 1. Chương trình tính tổng hai số.

Chú thích:

- Hàm sum(a, b) nhận hai tham số a và b, sau đó tính tổng của chúng bằng cách sử dụng biểu thức a + b.
- Kết quả tổng này được chuyển đổi thành chuỗi (vì print yêu cầu đối số là chuỗi) thông qua hàm str() và sau đó in ra màn hình theo định dạng "sum=...".
- Khi gọi sum(4, 5), Python sẽ thay thế a bằng 4 và b bằng 5. Kết quả là 4 + 5 = 9, sau đó in ra "sum=9".
- Khi gọi sum(3, 7), Python sẽ thay thế a bằng 3 và b bằng 7. Kết quả là 3 + 7 = 10, sau đó in ra "sum=10".

1.3.2. Viết hàm sum() với kết quả trả về.



```
print('Đào Quang Huy MSSV 235752021610051')
def sum(a,b):
    return a+b
c = sum(4,5)
print("Tong cua 4 và 5 =" + str(c))
```

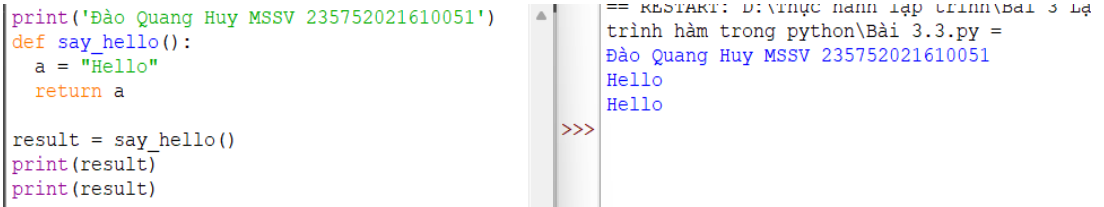
```
Đào Quang Huy MSSV 235752021610051
Tong cua 4 và 5 =9
```

Hình 2. Chương trình tính tổng và trả về kết quả.

Chú thích:

- Hàm này nhận hai đối số a và b, và trả về kết quả của a + b (tổng của hai số).
- Bạn sử dụng return để trả về giá trị, điều này giúp bạn có thể lưu kết quả tính toán vào một biến hoặc sử dụng nó trong các tính toán khác
- c = sum(4, 5) gọi hàm sum với các tham số 4 và 5. Kết quả trả về là 9 và được lưu vào biến c.
- Dòng print("Tong cua 4 và 5 = " + str(c)) chuyển giá trị của biến c thành chuỗi thông qua str(c) và in kết quả ra màn hình. Điều này giúp kết quả của phép tính được hiển thị một cách dễ hiểu.

1.3.3. Tìm và sửa lỗi chương trình.



```
print('Đào Quang Huy MSSV 235752021610051')
def say_hello():
    a = "Hello"
    return a

result = say_hello()
print(result)
print(result)
```

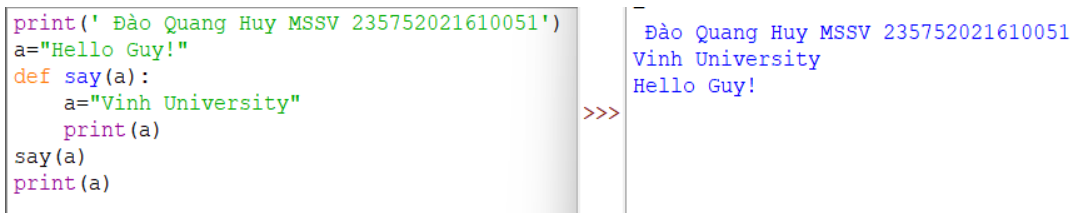
== RESTART: D:\THỰC HÀNH LẬP TRÌNH\BÀI 3 LẬP TRÌNH HÀM TRONG PYTHON\BÀI 3.3.py ==
Đào Quang Huy MSSV 235752021610051
Hello
Hello

Hình 3. Chương trình gọi một hàm trong Python.

Chú thích :

- Hàm say_hello() không nhận tham số. Khi gọi, nó sẽ thực thi các lệnh trong thân hàm.
- Biến a được gán giá trị "Hello", và hàm trả về giá trị này bằng cách sử dụng return a
- Khi bạn gọi say_hello(), Python sẽ thực thi hàm và trả về chuỗi "Hello". Kết quả này được lưu vào biến result.
- Dòng print(result) sẽ in giá trị của result, tức là "Hello", ra màn hình.
- Dòng print(result) lần thứ hai cũng sẽ in ra "Hello" vì biến result vẫn giữ giá trị đó sau lần gọi hàm.

1.3.4. Viết chương trình có phạm vi biến như sau.



```
print('Đào Quang Huy MSSV 235752021610051')
a="Hello Guy!"
def say(a):
    a="Vinh University"
    print(a)
say(a)
print(a)
```

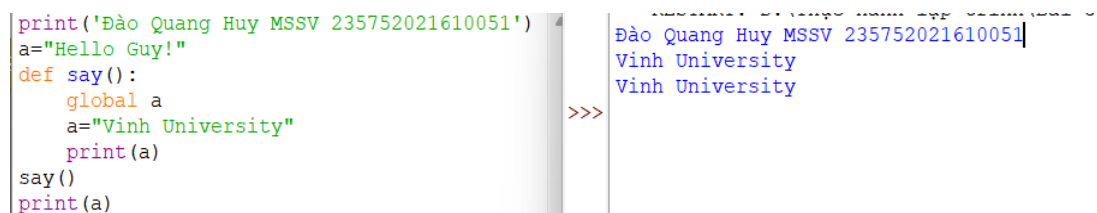
Đào Quang Huy MSSV 235752021610051
Vinh University
Hello Guy!

Hình 4. Chương trình ví dụ về biến cục bộ và biến toàn cục.

Chú thích :

- Biến toàn cục: a = "Hello Guy!" được khai báo bên ngoài hàm, có thể được truy cập và thay đổi ở bất kỳ đâu trong chương trình (trừ khi bị một biến cục bộ che khuất).
- Biến cục bộ: Trong hàm say(a), khi bạn gán a = "Vinh University", biến a trong hàm là cục bộ, chỉ có hiệu lực trong hàm và không ảnh hưởng đến biến a toàn cục.

1.3.5. Viết chương trình sau và xem sự thay đổi của biến.



```
print('Đào Quang Huy MSSV 235752021610051')
a="Hello Guy!"
def say():
    global a
    a="Vinh University"
    print(a)
say()
print(a)
```

Đào Quang Huy MSSV 235752021610051
Vinh University
Vinh University

Hình 5. Chương trình sử dụng từ khóa global để thay đổi giá trị của biến toàn cục từ trong hàm.

Chú thích :

- Biến a được khai báo ngoài hàm với giá trị "Hello Guy!". Đây là một biến toàn cục.
- Trong hàm say(), từ khóa global a chỉ ra rằng biến a mà bạn đang sử dụng trong hàm là biến toàn cục, không phải biến cục bộ. Điều này cho phép bạn thay đổi giá trị của biến a toàn cục từ trong hàm.
- Bạn gán giá trị "Vinh University" cho biến toàn cục a trong hàm, do đó giá trị của a toàn cục sẽ bị thay đổi.

1.3.6. Viết chương trình sau và giải thích việc truyền tham số của hàm.

```
print('Đào Quang Huy MSSV 235752021610051')
def get_sum(*num):
    tmp=0
    # duyệt các tham số
    for i in num:
        tmp +=i
    return tmp
result = get_sum(1,2,3,4,5)
print(result)
```

Đào Quang Huy MSSV 235752021610051
15

Hình 6. Chương trình truyền một số lượng tham số không xác định vào một hàm.

Chú thích :

- Trong Python, khi bạn sử dụng dấu * trước một tham số trong định nghĩa hàm, nó cho phép bạn truyền **hiều tham số** mà không cần khai báo từng tham số một. Tất cả các giá trị bạn truyền vào sẽ được **gói lại** thành một **tuple** và lưu trong biến num. Ví dụ: `get_sum(1, 2, 3, 4, 5)` sẽ tạo ra một tuple `num = (1, 2, 3, 4, 5)`.
- Vòng lặp này sẽ duyệt qua từng phần tử trong tuple num và cộng dồn giá trị của từng phần tử vào biến tmp.
- Biến tmp ban đầu được khởi tạo là 0. Mỗi giá trị trong tuple num được cộng vào tmp để tính tổng.
- Sau khi vòng lặp kết thúc, hàm `get_sum` trả về tổng của các tham số.

1.3.7. Định nghĩa hàm có thể chấp nhận input là số nguyên và in "Đây là một số chẵn" nếu nó chẵn và in "Đây là một số lẻ" nếu là số lẻ.

```
print('Đào Quang Huy MSSV 235752021610051')
def checkValue(n):
    if n%2==0:
        print("Đây là một số chẵn")
    else:
        print("Đây là một số lẻ")
checkValue(7)
```

Đào Quang Huy MSSV 235752021610051
Đây là một số lẻ

Hình 7. Chương trình kiểm tra số nhập vào là chẵn hay lẻ.

Chú thích:

- **Hàm `checkValue(n)`** này nhận một tham số n và kiểm tra xem nó có phải là số chẵn hay không.
- Để kiểm tra số chẵn, bạn dùng phép chia lấy dư `n % 2`. Nếu `n % 2 == 0`, nghĩa là số chia hết cho 2, và do đó là số chẵn.
- Nếu không chia hết cho 2 (`n % 2 != 0`), số đó là số lẻ.
- Khi gọi `checkValue(7)`, hàm sẽ kiểm tra giá trị `n = 7`.
- `7 % 2` cho kết quả là 1 (vì 7 không chia hết cho 2), nên điều kiện `if n % 2 == 0` không thỏa mãn. Chương trình sẽ thực hiện phần else và in ra "Đây là một số lẻ".

1.3.8. Một Robot di chuyển trong mặt phẳng bắt đầu từ điểm đầu tiên (0,0). Robot có thể di chuyển theo hướng UP, DOWN, LEFT và RIGHT với những bước nhất định.

Chú thích :

- `pos = [0, 0]` khởi tạo tọa độ của điểm bắt đầu là (0, 0).
- `while True`: tạo một vòng lặp vô hạn, cho phép người dùng nhập các lệnh di chuyển.
- `s = input()` yêu cầu người dùng nhập một chuỗi chỉ dẫn di chuyển.
- `if not s`: break dừng vòng lặp khi không có đầu vào (người dùng nhấn Enter mà không nhập gì).

- movement = s.split(" ") chia chuỗi đầu vào thành một danh sách chứa hai phần: hướng di chuyển và số bước.
- direction = movement[0] lấy hướng di chuyển (ví dụ: "UP", "DOWN", "LEFT", "RIGHT").
- steps = int(movement[1]) lấy số bước (số nguyên) từ phần thứ hai trong danh sách.
- "UP": tăng giá trị trục y. "DOWN": giảm giá trị trục y. "LEFT": giảm giá trị trục x. "RIGHT": tăng giá trị trục x.
- Cuối cùng, sau khi vòng lặp kết thúc, chương trình tính khoảng cách từ vị trí hiện tại pos đến gốc tọa độ (0, 0) bằng công thức: $\text{distance} = \sqrt{x^2 + y^2}$. Sử dụng math.sqrt() để tính căn bậc 2.

```
print("SVTH: ĐÀO QUANG HUY MSSV 235752021610051")
import math
pos = [0,0]
while True:
    s = input()
    if not s:
        break
    movement = s.split(" ")
    direction = movement[0]
    steps = int(movement[1])
    if direction=="UP":
        pos[0]+=steps
    elif direction=="DOWN":
        pos[0]-=steps
    elif direction=="LEFT":
        pos[1]-=steps
    elif direction=="RIGHT":
        pos[1]+=steps
    else:
        pass
    #####
print (int(round(math.sqrt(pos[1]**2+pos[0]**2))))
```

```
SVTH: ĐÀO QUANG HUY MSSV 235752021610
051
UP 3
DOWN 2
LEFT 4
RIGHT 6
```

```
>>>
```

- round() làm tròn kết quả và int() chuyển đổi nó thành số nguyên.

Hình 8. Chương trình di chuyển theo dõi một điểm trên mặt phẳng.

1.3.9. Chương trình máy tính thực hiện các phép tính đơn giản.

```
print("ĐÀO QUANG HUY MSSV 235752021610051")
# Hàm cộng hai số
def add(x, y):
    return x + y

# Hàm trừ hai số
def subtract(x, y):
    return x - y

# Hàm nhân hai số
def multiply(x, y):
    return x * y

# Hàm chia hai số
def divide(x, y):
    return x / y

print("Chọn phép toán.")
print("1. Cộng")
print("2. Trừ")
print("3. Nhân")
print("4. Chia")

# Nhập lựa chọn từ người dùng
choice = input("Nhập lựa chọn (1/2/3/4):")

num1 = int(input("Nhập số thứ nhất: "))
num2 = int(input("Nhập số thứ hai: "))

if choice == '1':
    print(num1, "+", num2, "=", add(num1, num2))
elif choice == '2':
    print(num1, "-", num2, "=", subtract(num1, num2))
elif choice == '3':
    print(num1, "*", num2, "=", multiply(num1, num2))
elif choice == '4':
    print(num1, "/", num2, "=", divide(num1, num2))
else:
    print("Nhập không hợp lệ")
```

```
>>>
```

```
Python 3.12.6 (tags/v3.12.6:a4a2d2b,
Sep 6 2024, 20:11:23) [MSC v.1940 6
4 bit (AMD64)] on win32
Type "help", "copyright", "credits"
or "license()" for more information.
```

```
= RESTART: D:\Thực hành lập trình\Bài
3 Lập trình hàm trong python\Bài 3
.9.py
```

```
ĐÀO QUANG HUY MSSV 235752021610051
Chọn phép toán.
```

```
1. Cộng
2. Trừ
3. Nhân
4. Chia
Nhập lựa chọn (1/2/3/4):3
Nhập số thứ nhất: 16
Nhập số thứ hai: 8
16 * 8 = 128
```

```
>>>
```

Hình 9.1 Chương trình máy tính một số phép tính đơn giản.

Chú thích :

- Định nghĩa các hàm toán học:
 - + Hàm `add(x, y)` trả về tổng của hai số `x` và `y`.
 - + Hàm `subtract(x, y)` trả về hiệu của hai số `x` và `y`.
 - + Hàm `multiply(x, y)` trả về tích của hai số `x` và `y`.
 - + Hàm `divide(x, y)` trả về thương của hai số `x` và `y`. Lưu ý: chương trình chưa xử lý lỗi chia cho 0.
- Chương trình hiển thị các phép toán có sẵn cho người dùng: cộng, trừ, nhân, chia.
- Người dùng được yêu cầu nhập lựa chọn phép toán (choice) và hai số (num1 và num2).
- Dựa trên lựa chọn (choice), chương trình sẽ thực hiện phép toán tương ứng:
 - + Nếu choice là '1', thực hiện phép cộng bằng hàm `add()`.
 - + Nếu choice là '2', thực hiện phép trừ bằng hàm `subtract()`.
 - + Nếu choice là '3', thực hiện phép nhân bằng hàm `multiply()`.
 - + Nếu choice là '4', thực hiện phép chia bằng hàm `divide()`.
 - + Nếu người dùng nhập lựa chọn không hợp lệ, chương trình in thông báo lỗi.

Chú ý :

- Trong trường hợp người dùng nhập số 0 cho phép chia, chương trình sẽ gặp lỗi chia cho 0, vì vậy bạn nên thêm một kiểm tra để xử lý tình huống này. Bạn có thể cập nhật hàm `divide` để kiểm tra nếu `y` bằng 0.

```
# Hàm chia hai số
def divide(x, y):
    if y == 0:
        return "Lỗi! Không thể chia cho 0"
    else:
        return x / y
```

Hình 9.2. Thêm hàm `divide` để kiểm tra nếu `y` bằng 0.

1.3.10. Viết hàm “def `Tinh(R):`” tính chu vi và diện tích hình tròn, với bán kính `R` được nhập từ bàn phím, và kiểm tra giá trị bán kính đầu vào là hợp lệ.

Gợi ý: sử dụng thư viện “`import math`” và `math.pi` cho số pi nếu cần.

```
print('Đào Quang Huy MSSV 235752021610051')
import math

def Tinh(R):
    # Kiểm tra giá trị bán kính
    if R <= 0:
        return "Bán kính phải là số dương."
    else:
        # Tính chu vi và diện tích
        chu_vi = 2 * math.pi * R
        dien_tich = math.pi * R**2
        return chu_vi, dien_tich

# Nhập bán kính từ người dùng
while True:
    try:
        R = float(input("Nhập bán kính hình tròn: "))
        break
    except ValueError:
        print("Vui lòng nhập một số thực dương.")

# Gọi hàm và in kết quả
ket_qua = Tinh(R)
if isinstance(ket_qua, str):
    print(ket_qua)
else:
    print("Chu vi hình tròn:", ket_qua[0])
    print("Diện tích hình tròn:", ket_qua[1])
```

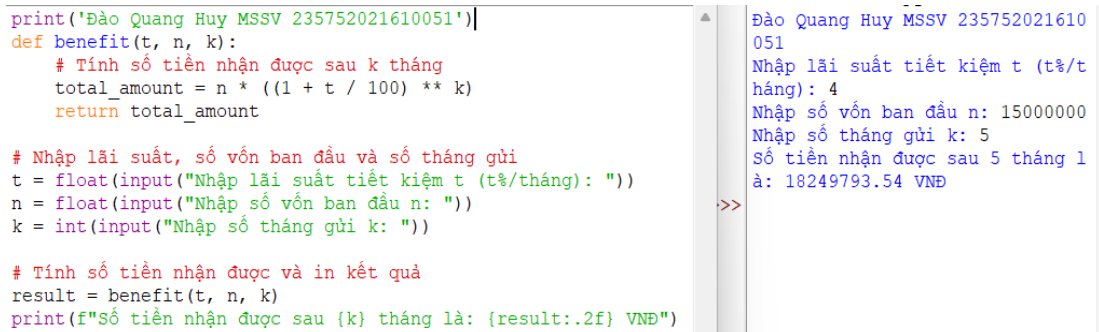
Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep 6 2024, 20:11:23) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: D:\Thực hành lập trình\Bài 3 Lập trình hàm trong python\Bài 3.10.py
Đào Quang Huy MSSV 235752021610051
Nhập bán kính hình tròn: 16
Chu vi hình tròn: 100.53096491487338
Diện tích hình tròn: 804.247719318987
>>> |

Hình 10. Chương trình tính chu vi và diện tích từ giá trị nhập từ bàn phím .

Chú thích :

- **Hàm Tinh(R): Kiểm tra bán kính hợp lệ:**
 - + Nếu bán kính R nhỏ hơn hoặc bằng 0, hàm trả về thông báo lỗi "Bán kính phải là số dương."
 - + **Tính chu vi và diện tích:**
Nếu bán kính hợp lệ, chương trình sẽ tính chu vi của hình tròn bằng công thức $chu_vi = 2 * \text{math.pi} * R$ và diện tích bằng công thức $dien_tich = \text{math.pi} * R^2$. math.pi cung cấp giá trị của số pi (π).
- Sử dụng vòng lặp while True để yêu cầu người dùng nhập bán kính cho đến khi giá trị hợp lệ được nhập.
- try và except được sử dụng để xử lý các lỗi khi người dùng nhập không phải là số thực. Nếu người dùng nhập không hợp lệ (ví dụ, chữ cái thay vì số), chương trình sẽ yêu cầu nhập lại.
- Sau khi người dùng nhập bán kính hợp lệ, hàm Tinh(R) được gọi và kết quả trả về sẽ được lưu trong biến ket_qua.
- Nếu kết quả trả về là một chuỗi (điều này xảy ra khi bán kính không hợp lệ), chương trình sẽ in thông báo lỗi. Nếu kết quả là một tuple chứa chu vi và diện tích, chương trình sẽ in ra kết quả.

1.3.11. Biết lãi suất tiết kiệm là t%/tháng (nhập t từ bàn phím). Nhập số vốn ban đầu n và số tháng gửi k. Tính số tiền nhận được sau k tháng sử dụng cấu trúc hàm
`def benefit(t,n,k):`



```
print('Đào Quang Huy MSSV 235752021610051')
def benefit(t, n, k):
    # Tính số tiền nhận được sau k tháng
    total_amount = n * ((1 + t / 100) ** k)
    return total_amount

# Nhập lãi suất, số vốn ban đầu và số tháng gửi
t = float(input("Nhập lãi suất tiết kiệm t (t%/tháng): "))
n = float(input("Nhập số vốn ban đầu n: "))
k = int(input("Nhập số tháng gửi k: "))

# Tính số tiền nhận được và in kết quả
result = benefit(t, n, k)
print(f"Số tiền nhận được sau {k} tháng là: {result:.2f} VNĐ")
```

Đào Quang Huy MSSV 235752021610051
Nhập lãi suất tiết kiệm t (t%/tháng): 4
Nhập số vốn ban đầu n: 15000000
Nhập số tháng gửi k: 5
Số tiền nhận được sau 5 tháng là: 18249793.54 VNĐ

Hình 11. Chương trình tính số tiền nhận được sau một khoảng thời gian gửi tiết kiệm, với lãi suất được nhập vào theo tháng.

Chú thích :

- Hàm `benefit(t, n, k):`
 - + t: Lãi suất tiết kiệm hàng tháng (theo phần trăm).
 - + n: Số vốn ban đầu (số tiền bạn gửi vào tài khoản tiết kiệm).
 - + k: Số tháng gửi tiết kiệm.
- Ta có công thức :

$$\text{total_amount} = n \times \left(1 + \frac{t}{100}\right)^k$$

- Ở đây, số tiền sau k tháng được tính theo công thức của lãi suất kép. Lãi suất hàng tháng được cộng vào số tiền gốc mỗi tháng.
- Lãi suất t được nhập dưới dạng phần trăm.
- Số vốn ban đầu n là số tiền bạn gửi vào.
- Số tháng gửi k là khoảng thời gian bạn sẽ gửi tiền vào tài khoản.
- Gọi hàm `benefit(t, n, k)` để tính tổng số tiền nhận được sau k tháng.
- Kết quả được in ra với định dạng 2 chữ số thập phân, với đơn vị là VNĐ.

Bài 4. Các kiểu dữ liệu có cấu trúc trong Python

1.1. Mục đích

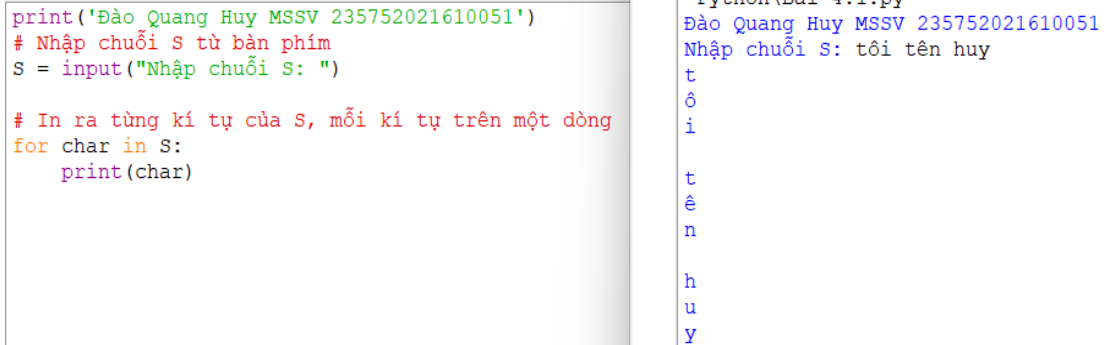
Giúp sinh viên nắm bắt các kiểu dữ liệu có cấu trúc trong lập trình python.

1.2. Cơ sở lý thuyết

Xem các quy tắc sử dụng các kiểu dữ liệu: chuỗi, số, list, tuple, set và dictionary trong python.

1.3. Quá trình thực hiện.

1.3.1. Nhập chuỗi S và in ra từng ký tự của S, mỗi ký tự trên một dòng.



```
print('Đào Quang Huy MSSV 235752021610051')
# Nhập chuỗi S từ bàn phím
S = input("Nhập chuỗi S: ")

# In ra từng ký tự của S, mỗi ký tự trên một dòng
for char in S:
    print(char)
```

Đào Quang Huy MSSV 235752021610051
Nhập chuỗi S: tôi tên huy
t
ô
i

t
ê
n

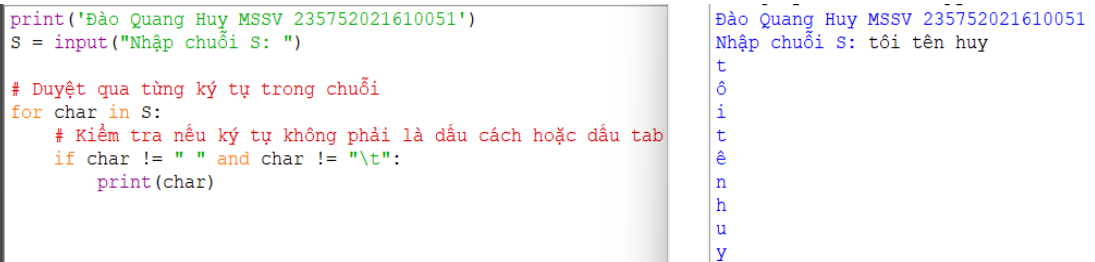
h
u
y

Hình 1. Nhập chuỗi và in các ký tự trên mỗi dòng riêng biệt.

Chú thích :

- Dòng `S = input ("Nhập chuỗi S: ")` yêu cầu người dùng nhập vào một chuỗi bất kỳ và lưu vào biến S. Lệnh `input ()` nhận dữ liệu từ bàn phím dưới dạng chuỗi.
- Câu lệnh `for char in S:` sẽ duyệt qua từng ký tự trong chuỗi S.
- Mỗi lần lặp, biến `char` sẽ nhận giá trị là một ký tự trong chuỗi và lệnh `print(char)` sẽ in ký tự đó ra màn hình.
- Mỗi lần `print(char)` sẽ in một ký tự lên một dòng mới, do `print ()` mặc định kết thúc mỗi lần in bằng dấu xuống dòng (`\n`).

1.3.2. Chỉnh sửa ví dụ trên: hãy bỏ qua không in ra những ký tự “không nhìn thấy” (dấu space và dấu tab).



```
print('Đào Quang Huy MSSV 235752021610051')
S = input("Nhập chuỗi S: ")

# Duyệt qua từng ký tự trong chuỗi
for char in S:
    # Kiểm tra nếu ký tự không phải là dấu cách hoặc dấu tab
    if char != " " and char != "\t":
        print(char)
```

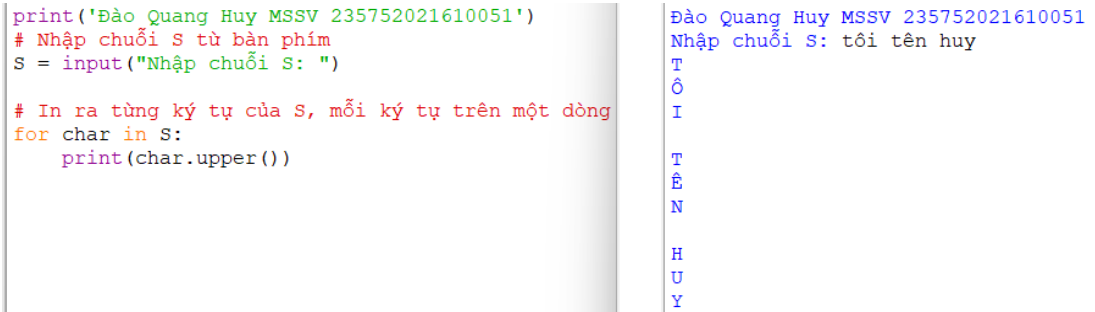
Đào Quang Huy MSSV 235752021610051
Nhập chuỗi S: tôi tên huy
t
ô
i
t
ê
n
h
u
y

Hình 2. Thực hiện việc duyệt qua từng ký tự trong chuỗi và in ra các ký tự

Chú thích:

- Dòng `S = input ("Nhập chuỗi S: ")` yêu cầu người dùng nhập một chuỗi bất kỳ, và giá trị này được lưu vào biến S.
- Vòng lặp `for char in S:` sẽ duyệt qua từng ký tự của chuỗi S. Mỗi lần lặp, ký tự hiện tại sẽ được lưu vào biến `char`.
- Câu lệnh `if char != " " and char != "\t":` kiểm tra xem ký tự hiện tại có phải là dấu cách hoặc dấu tab không. Nếu không phải, chương trình sẽ in ký tự đó ra.
- Lệnh `print(char)` sẽ in ra ký tự nếu điều kiện trên thỏa mãn (tức là ký tự không phải là dấu cách hoặc dấu tab).

1.3.3. Chỉnh sửa ví dụ ở bài 1: hãy các kí tự ở dạng IN HOA.



```
print('Đào Quang Huy MSSV 235752021610051')
# Nhập chuỗi S từ bàn phím
s = input("Nhập chuỗi S: ")

# In ra từng ký tự của S, mỗi ký tự trên một dòng
for char in s:
    print(char.upper())
```

Đào Quang Huy MSSV 235752021610051
Nhập chuỗi S: tôi tên huy
T
Ô
I

T
Ê
N

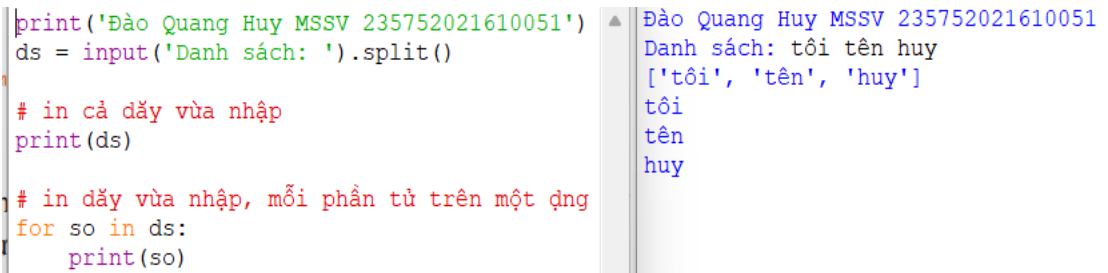
H
U
Y

Hình 3. Nhập chuỗi và in các kí tự ở dạng in hoa.

Chú thích:

- **s = input("Nhập chuỗi S: ")**: Dòng này yêu cầu người dùng nhập một chuỗi, sau đó lưu chuỗi đó vào biến s.
- **for char in s:**: Vòng lặp này duyệt qua từng ký tự trong chuỗi s. Mỗi lần lặp, biến char sẽ chứa một ký tự trong chuỗi s.
- **print(char.upper())**: Mỗi ký tự char được chuyển thành chữ hoa bằng phương thức upper() và sau đó in ra màn hình. Phương thức upper() không thay đổi giá trị ban đầu của chuỗi mà chỉ trả về một bản sao với tất cả các ký tự là chữ hoa

1.3.4. Nhập một danh sách thêm 1 dòng mỗi phần tử cách nhau bởi dấu trống hoặc dấu cách sau đó in ra dãy vừa nhập ra màn hình.



```
print('Đào Quang Huy MSSV 235752021610051')
ds = input('Danh sách: ').split()

# in cả dãy vừa nhập
print(ds)

# in dãy vừa nhập, mỗi phần tử trên một dòng
for so in ds:
    print(so)
```

Đào Quang Huy MSSV 235752021610051
Danh sách: tôi tên huy
['tôi', 'tên', 'huy']
tôi
tên
huy

Hình 4. Yêu cầu người dùng nhập một chuỗi danh sách, sau đó xử lý chuỗi đó thành các phần tử riêng biệt và in ra theo hai cách:

Chú thích:

- **ds = input("Danh sách: ").split()**: Dòng này yêu cầu người dùng nhập một chuỗi. Sau đó, phương thức .split() được gọi để chia chuỗi thành các phần tử dựa trên dấu cách mặc định (space). Kết quả là một danh sách (list) các phần tử mà người dùng đã nhập.
- **print(ds)**: In ra danh sách đầy đủ mà người dùng đã nhập dưới dạng một danh sách Python (các phần tử sẽ được in ra theo định dạng ['phần tử 1', 'phần tử 2', ..., 'phần tử N']).
- **for so in ds**: Duyệt qua từng phần tử trong danh sách ds. Mỗi phần tử so được in ra một dòng riêng biệt.

1.3.5. Chỉnh sửa ví dụ ở bài 4: nhập 1 danh sách các từ từ bàn phím, in ra các từ đó theo thứ tự ngược lại thứ tự vừa nhập .

```
print('Đào Quang Huy MSSV 235752021610051')
# Nhập danh sách các từ từ bàn phím
danh_sach = input("Nhập danh sách các từ: ").split()

# Đảo ngược thứ tự của danh sách
danh_sach.reverse()

# In ra danh sách các từ đã đảo ngược
print("Danh sách các từ sau khi đảo ngược: ", end="")
for sach in danh_sach:
    print(sach, end=" ")

>>> |
```

Đào Quang Huy MSSV 235752021610051
Nhập danh sách các từ: đào quang huy
Danh sách các từ sau khi đảo ngược: huy quang
đào

Chú thích:

- Dòng này yêu cầu người dùng nhập một chuỗi các từ, sau đó phương thức `.split()` sẽ tách chuỗi này thành một danh sách các từ, dựa trên dấu cách (space).
- **`danh_sach.reverse()`**: Phương thức `.reverse()` được sử dụng để đảo ngược thứ tự các phần tử trong danh sách `danh_sach`.
- **`print("Danh sách các từ sau khi đảo ngược: ", end="")`**: In dòng chữ "Danh sách các từ sau khi đảo ngược: " mà không xuống dòng. Tham số `end=""` trong `print()` đảm bảo rằng không có ký tự xuống dòng sau khi in ra chuỗi này.
- **Vòng lặp `for sach in danh_sach`**: Vòng lặp duyệt qua từng phần tử trong danh sách `danh_sach` (các từ đã được đảo ngược). Mỗi từ được in ra trên cùng một dòng, cách nhau bằng một dấu cách nhờ tham số `end=" "` trong `print()`, đảm bảo không có dấu xuống dòng giữa các từ

1.3.6. Nhập một tên người từ bàn phím, hãy tách phần họ và tên riêng của người đó và in chúng ra màn hình (giả thiết họ và tên riêng chỉ gồm một âm).

```
print('Đào Quang Huy MSSV 235752021610051')
ho_ten = input("Họ tên của mi là chi:")

#Tìm vị trí của khoảng trắng(giả sử họ và tên chỉ con cách nhau
# bởi một khoảng trắng
vi_tri_khoang_trang = ho_ten.find(" ")

# Tách họ và tên
ho = ho_ten[:vi_tri_khoang_trang]
ten = ho_ten[vi_tri_khoang_trang+1:]

# In ra kết quả
print("Họ:", ho)
print("Tên:", ten)

>>> |
```

Đào Quang Huy MSSV 235752021610051
Họ tên của mi là chi:đào quang huy
Họ: đào
Tên: quang huy

Hình 6. Nhập họ tên từ người dùng, sau đó tách họ và tên bằng cách tìm vị trí của khoảng trắng và sử dụng chỉ số để cắt chuỗi.

Chú thích:

- **`ho_ten = input("Họ tên của mi là chi:")`**: Dòng này yêu cầu người dùng nhập họ và tên vào chương trình.
- **`vi_tri_khoang_trang = ho_ten.find(" ")`**: Phương thức `.find(" ")` được sử dụng để tìm vị trí đầu tiên của khoảng trắng trong chuỗi. Nếu chuỗi có khoảng trắng (giữa họ và tên), phương thức này sẽ trả về chỉ số vị trí của khoảng trắng đó. Nếu không có khoảng trắng, `.find()` sẽ trả về -1.
- **`ho = ho_ten[:vi_tri_khoang_trang]`**: Cắt chuỗi từ đầu đến vị trí của khoảng trắng, lấy phần họ.
- **`ten = ho_ten[vi_tri_khoang_trang+1:]`**: Cắt chuỗi từ vị trí sau khoảng trắng cho đến hết chuỗi, lấy phần tên.

1.3.7. Nhập một chuỗi từ bàn phím, hãy loại bỏ tất cả các chữ số khỏi chuỗi và in lại nội dung chuỗi mới ra màn hình.

<pre>print('Đào Quang Huy MSSV 235752021610051') chuoi = input("Nhập chuỗi đi thẳng tên: ") chuoi_moi = "" for ky_tu in chuoi: if not ky_tu.isdigit(): chuoi_moi += ky_tu print("Chuỗi mới: ", chuoi_moi)</pre>	<pre>>>> Đào Quang Huy MSSV 235752021610051 Nhập chuỗi đi thẳng tên: daoquanghuy2005 Chuỗi mới: daoquanghuy</pre>
---	--

Hình 7. Chương trình loại bỏ các chữ số khỏi chuỗi nhập vào.

Chú thích :

- `chuoi = input("Nhập chuỗi đi thẳng tên: ")`: Dòng này yêu cầu người dùng nhập một chuỗi và lưu chuỗi đó vào biến `chuoi`.
- `chuoi_moi = ""`: Tạo một chuỗi rỗng `chuoi_moi` để lưu trữ kết quả sau khi loại bỏ các chữ số từ chuỗi ban đầu.
- Vòng lặp `for ky_tu in chuoi`: Vòng lặp này sẽ duyệt qua từng ký tự (`ky_tu`) trong chuỗi `chuoi`.
- `if not ky_tu.isdigit()`: Dòng này kiểm tra nếu ký tự không phải là chữ số. Nếu đúng, ký tự đó sẽ được giữ lại trong chuỗi mới. Phương thức `.isdigit()` trả về `True` nếu ký tự là chữ số, và `False` nếu không phải là chữ số.
- `chuoi_moi += ky_tu`: Nếu ký tự không phải là chữ số, chúng ta sẽ cộng (thêm) ký tự đó vào chuỗi `chuoi_moi`.
- `print("Chuỗi mới: ", chuoi_moi)`: Sau khi vòng lặp hoàn thành, chương trình sẽ in ra chuỗi mới (`chuoi_moi`) mà đã loại bỏ tất cả các chữ số.

1.3.8. Nhập một dãy các từ từ bàn phím, hãy in ra từ dài nhất trong dãy vừa nhập, in ra mọi từ có cùng độ dài nhất.

<pre>print('Đào Quang Huy MSSV 235752021610051') def tim_tu_dai_nhat(danh_sach_tu): """Tìm và in ra các từ dài nhất trong một danh sách các từ.""" # Kiểm tra nếu danh sách trống if not danh_sach_tu: print("Danh sách trống! Không có từ nào để tìm.") return # Tìm độ dài lớn nhất của các từ do_dai_lon_nhat = max(len(tu) for tu in danh_sach_tu) # In ra các từ có độ dài bằng độ dài lớn nhất print("Các từ dài nhất:") for tu in danh_sach_tu: if len(tu) == do_dai_lon_nhat: print(tu) # Nhập danh sách các từ từ bàn phím danh_sach_tu = input("Nhập danh sách các từ: ").split() # Gọi hàm để tìm và in ra các từ dài nhất tim_tu_dai_nhat(danh_sach_tu)</pre>	<pre>Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep 6 2024, 20:11:23) [MSC v.1940 6 4 bit (AMD64)] on win32 Type "help", "copyright", "credits" or "license()" for more information. >>> = RESTART: D:\Thực hành lập trình\Bài i 4 Các kiểu dữ liệu có cấu trúc tro ng Python\Bài 4.8.py Đào Quang Huy MSSV 235752021610051 Nhập danh sách các từ: đào quang huy Các từ dài nhất: quang >>></pre>
--	---

Hình 8. Chương trình nhập vào chuỗi danh sách và in ra từ dài nhất.

Chú thích :

- `danh_sach_tu = input("Nhập danh sách các từ (cách nhau bởi dấu cách): ").split()`: Chúng ta yêu cầu người dùng nhập một chuỗi các từ, cách nhau bằng dấu cách. Hàm `input()` nhận chuỗi đầu vào, sau đó `.split()` sẽ tách chuỗi thành một danh sách các từ.

- `do_dai_lon_nhat = max(len(tu) for tu in danh_sach_tu)`: Sử dụng hàm `max()` với một biểu thức tạo ra độ dài của từng từ trong danh sách. Hàm `max()` sẽ trả về độ dài lớn nhất trong số các độ dài của các từ.
- Sau khi tìm được độ dài lớn nhất, vòng lặp `for tu in danh_sach_tu` được sử dụng để duyệt qua các từ trong danh sách. Nếu độ dài của một từ bằng với độ dài lớn nhất (`len(tu) == do_dai_lon_nhat`), chương trình sẽ in ra từ đó.

1.3.9. Nhập một list từ bàn phím.

```
print('Đào Quang Huy MSSV 235752021610051')
# Nhập một danh sách từ bàn phím
input_list = input("Nhập các phần tử : ")
# Chia chuỗi thành danh sách
my_list = input_list.split()

# In danh sách
print("Danh sách bạn đã nhập:", my_list)
```

```
Đào Quang Huy MSSV 235752021610051
Nhập các phần tử : đào quang huy
Danh sách bạn đã nhập: ['đào', 'quang', 'huy']
>>> |
```

Hình 9. Nhập một list từ bàn phím.

Chú thích:

- Hàm `input()` sẽ nhận một chuỗi từ người dùng.
- `split()` sẽ tách chuỗi này thành các phần tử con dựa trên dấu cách (hoặc bạn có thể thay dấu cách bằng dấu phân cách khác nếu cần).
- `map(int, ...)` sẽ chuyển các phần tử này thành kiểu số nguyên.
- Cuối cùng, `list()` sẽ tạo một danh sách từ các phần tử đã được chuyển đổi.

1.3.10. Cắt list: lấy list nhưng bỏ phần tử đầu và cuối.

```
print('Đào Quang Huy MSSV 235752021610051')
my_list = [16, 8, 2005]

# Cắt danh sách bỏ phần tử đầu và phần tử cuối
new_list = my_list[1:-1]

print(new_list)
```

```
Đào Quang Huy MSSV 235752021610051
[8]
>>> |
```

Hình 10. Cắt bỏ phần tử đầu và cuối trong danh sách .

Chú thích :

- `my_list[1:-1]`:
 - + Chỉ định bắt đầu từ chỉ số 1 (phần tử thứ 2) và kết thúc tại chỉ số -1 (phần tử cuối cùng nhưng không bao gồm nó).
 - + 1 là chỉ số của phần tử thứ 2 (do chỉ số bắt đầu từ 0).
 - + -1 đại diện cho phần tử cuối cùng của danh sách, nhưng khi dùng trong slicing, nó sẽ không bao gồm phần tử ở chỉ số -1.

1.3.11. Thêm phần tử vào list.

<pre>my_list = [1, 2, 3] # Thêm phần tử vào cuối danh sách my_list.append(4) print(my_list)</pre>	<pre>my_list = [1, 2, 3] # Thêm phần tử 0 vào vị trí chỉ số 0 (đầu danh sách) my_list.insert(0, 0) print(my_list)</pre>
Kết quả:	Kết quả:
<pre>csharp [1, 2, 3, 4]</pre>	<pre>csharp [0, 1, 2, 3]</pre>

Hình 11. 1 Sử dụng append và insert.

<pre>my_list = [1, 2, 3] # Thêm các phần tử từ danh sách khác my_list.extend([4, 5, 6]) print(my_list)</pre>	<pre>my_list = [1, 2, 3] # Cộng thêm một danh sách mới vào danh sách hiện tại my_list = my_list + [4, 5] print(my_list)</pre>
Kết quả:	Kết quả:
<pre>csharp [1, 2, 3, 4, 5, 6]</pre>	<pre>csharp [1, 2, 3, 4, 5]</pre>

Hình 11.2 Sử dụng extend và cộng thêm vào list.

Chú thích :

- `append()`: Thêm một phần tử vào cuối danh sách.
- `insert()`: Chèn phần tử vào một vị trí cụ thể trong danh sách.
- `extend()`: Thêm tất cả phần tử của một danh sách khác vào danh sách hiện tại.
- Toán tử `+`: Cộng hai danh sách lại thành một danh sách mới.

1.3.12. Bỏ phần tử khỏi list

<pre>print('Đào Quang Huy MSSV 235752021610051') # Tạo một danh sách my_list = [16, 8, 2005] # Bỏ phần tử có giá trị 16 my_list.remove(16) # In danh sách sau khi bỏ print("Danh sách sau khi bỏ phần tử:", my_list)</pre>	<pre>>>> Đào Quang Huy MSSV 235752021610051 Danh sách sau khi bỏ phần tử: [8, 2005]</pre>
--	--

Hình 12. Bỏ phần tử khỏi list.

Chú thích :

- `my_list.remove(16)` sẽ tìm phần tử có giá trị 16 trong danh sách và xóa nó.
- Sau khi xóa phần tử 16, danh sách còn lại là `[8, 2005]`.

1.3.13. Tìm kiếm phần tử trong list.

```
print('Đào Quang Huy MSSV 235752021610051')
my_list = [16, 8, 2005]

# Kiểm tra xem phần tử 8 có trong danh sách không
if 8 in my_list:
    print("Phần tử 8 có trong danh sách.")
else:
    print("Phần tử 8 không có trong danh sách.")
```

>>> Đào Quang Huy MSSV 235752021610051
Phần tử 8 có trong danh sách.

Hình 13. Tìm kiếm phần tử trong list.

Chú thích :

- Toán tử in giúp kiểm tra xem phần tử có tồn tại trong danh sách hay không. Nếu phần tử tồn tại, kết quả là True, ngược lại là False.

1.3.14. Sắp xếp các phần tử trong list.

<pre>my_list = [16, 8, 2005, 1] # Sắp xếp danh sách tại chỗ my_list.sort() print("Danh sách sau khi sắp xếp:", my_list)</pre>	<pre>my_list = [16, 8, 2005, 1] # Sắp xếp giảm dần tại chỗ my_list.sort(reverse=True) print("Danh sách sau khi sắp xếp giảm dần:", my_list)</pre>
<p>Kết quả:</p> <pre>less Danh sách sau khi sắp xếp: [1, 8, 16, 2005]</pre>	<p>Kết quả:</p> <pre>less Danh sách sau khi sắp xếp giảm dần: [2005, 16, 8, 1]</pre>

Hình 14. Sắp xếp danh sách theo thứ tự tăng dần và giảm dần.

Chú thích :

- sort(): Sắp xếp danh sách tại chỗ, thay đổi danh sách gốc.
- sorted(): Trả về một danh sách mới đã được sắp xếp mà không thay đổi danh sách gốc.

1.3.15. Người dùng nhập từ bàn phím liên tiếp các từ tiếng Anh viết tách nhau bởi dấu cách. Hãy nhập chuỗi đầu vào và tách thành các từ sau đó in ra màn hình các từ đó theo thứ tự từ điển.

```
print('Đào Quang Huy MSSV 235752021610051')
# Nhập chuỗi từ bàn phím
input_string = input("Nhập chuỗi các từ tiếng Anh: ")

# Tách chuỗi thành danh sách các từ
words = input_string.split()

# Sắp xếp danh sách các từ theo thứ tự từ điển
sorted_words = sorted(words)

# In ra các từ theo thứ tự từ điển
print("Các từ theo thứ tự từ điển:")
for word in sorted_words:
    print(word)
```

>>> Đào Quang Huy MSSV 235752021610051
Nhập chuỗi các từ tiếng Anh: hello
hi teacher coffee eat
Các từ theo thứ tự từ điển:
coffee
eat
hello
hi
teacher

Hình 15. Chương trình sắp xếp danh sách theo thứ tự từ điển .

Chú thích :

- input(): Nhận đầu vào từ người dùng.
- split(): Tách chuỗi thành các từ, mặc định tách theo dấu cách.
- sort(): Sắp xếp danh sách các từ theo thứ tự từ điển (theo mặc định, sort() sẽ sắp xếp theo thứ tự từ điển ASCII, với chữ cái viết hoa sẽ đứng trước chữ cái viết thường).
- Vòng lặp for: Duyệt qua danh sách các từ đã sắp xếp và in ra từng từ.

1.3.16. Người dùng nhập từ bàn phím chuỗi các số nhị phân viết liên tiếp được nối nhau bởi dấu phẩy. Hãy nhập chuỗi đầu vào sau đó in ra những giá trị được nhập.

<pre>print('Đào Quang Huy MSSV 235752021610051') # Nhập chuỗi các số nhị phân từ bàn phím input_string = input("Nhập chuỗi các số nhị phân, # Tách chuỗi thành các số nhị phân binary_numbers = input_string.split(',') # In ra các giá trị đã nhập print("Các số nhị phân đã nhập:") for binary in binary_numbers: print(binary)</pre>	<pre>liệu có cấu trúc trong Python\Bài 4.16.py Đào Quang Huy MSSV 235752021610051 Nhập chuỗi các số nhị phân, cách nhau bằng dấu phẩy : 1101,0101,1010 Các số nhị phân đã nhập: 1101 0101 1010 >>> </pre>
--	---

Hình 16.1. Tách chuỗi thành các số nhị phân .

```
# Kiểm tra tính hợp lệ của các số nhị phân
for binary in binary_numbers:
    if all(c in '01' for c in binary): # Kiểm tra nếu chỉ chứa các ký tự 0 hoặc 1
        print(binary)
    else:
        print(f"{binary} không phải là số nhị phân hợp lệ.")
```

Hình 16.2. Kiểm tra danh sách nhập vào có phải số nhị phân không.

Chú thích :

- input(): Nhận đầu vào từ người dùng.
- split(): Tách chuỗi thành các từ, mặc định tách theo dấu cách.
- sort(): Sắp xếp danh sách các từ theo thứ tự từ điển (theo mặc định, sort() sẽ sắp xếp theo thứ tự từ điển ASCII, với chữ cái viết hoa sẽ đứng trước chữ cái viết thường).
- Vòng lặp for: Duyệt qua danh sách các từ đã sắp xếp và in ra từng từ.

1.3.17. Nhập số n, in ra màn hình các số nguyên dương nhỏ hơn n có tổng các ước số lớn hơn chính nó.

<pre>print('Đào Quang Huy MSSV 235752021610051') def sum_of_divisors(num): total = 0 for i in range(1, num): if num % i == 0: total += i return total # Nhập số nguyên dương n n = int(input("Nhập số nguyên dương n: ")) print(f"Các số nguyên dương nhỏ hơn {n} có tổng các ước số lớn hơn chính nó:") for i in range(1, n): if sum_of_divisors(i) > i: print(i)</pre>	<pre>Đào Quang Huy MSSV 23575202161 0051 Nhập số nguyên dương n: 16 Các số nguyên dương nhỏ hơn 16 có tổng các ước số lớn hơn chí nh nó: 12 >>> </pre>
---	--

Hình 17. Chương trình in ra các số nguyên dương nhỏ hơn n.

- Hàm `sum_of_divisors(num)`: Hàm này tính tổng các ước số của số `num` (không tính chính nó). Ví dụ, các ước số của 12 là 1, 2, 3, 4, 6.
- Vòng lặp chính: Vòng lặp từ 1 đến $n-1$, kiểm tra xem tổng các ước số của mỗi số có lớn hơn chính nó không. Nếu có, in ra số đó.

1.3.18. Hãy nhập số nguyên `n`, tạo một list gồm các số fibonacci nhỏ hơn `n` và in ra màn hình.

```
print('Đào Quang Huy MSSV 235752021610051')
def fibonacci_less_than(n):
    fib_list = []
    a, b = 0, 1
    while a < n:
        fib_list.append(a)
        a, b = b, a + b
    return fib_list

# Nhập số nguyên n
n = int(input("Nhập số nguyên n: "))

# Tạo danh sách các số Fibonacci nhỏ hơn n
fibonacci_numbers = fibonacci_less_than(n)

# In ra danh sách các số Fibonacci
print(f"Các số Fibonacci nhỏ hơn {n}:")
print(fibonacci_numbers)
```

```
Đào Quang Huy MSSV 235752021610051
Nhập số nguyên n: 16
Các số Fibonacci nhỏ hơn 16:
[0, 1, 1, 2, 3, 5, 8, 13]
```

Hình 18. Chương trình tạo danh sách các số Fibonacci nhỏ hơn một số nguyên `nnn` mà người dùng nhập vào

Chú thích:

- Hàm `fibonacci_less_than(n)`:
 - + Đây là một hàm dùng để tạo danh sách các số Fibonacci nhỏ hơn `nnn`.
 - + Biến `a` và `b`: Khởi tạo hai số đầu tiên của dãy Fibonacci là 0 và 1.
 - + Vòng lặp `while a < n`: Lặp cho đến khi số Fibonacci hiện tại (là `a`) lớn hơn hoặc bằng `nnn`. Mỗi lần lặp, số `a` sẽ được thêm vào danh sách `fib_list`.
 - + Sau mỗi lần lặp, giá trị của `a` và `b` sẽ được cập nhật: `a` nhận giá trị của `b`, còn `b` nhận giá trị là tổng của `a` và `b` (tạo ra số Fibonacci tiếp theo).
- Chương trình yêu cầu người dùng nhập một số nguyên `nnn`.
 - + Sau đó, nó gọi hàm `fibonacci_less_than(n)` để tạo ra danh sách các số Fibonacci nhỏ hơn `nnn`.
 - + Cuối cùng, danh sách các số Fibonacci sẽ được in ra màn hình.

1.3.19. Hãy tạo ra tuple `P` gồm các số nguyên tố nhỏ hơn 1 triệu.

```
print('Đào Quang Huy MSSV 235752021610051')
def sieve_of_eratosthenes(limit):
    is_prime = [True] * limit
    is_prime[0] = is_prime[1] = False # 0 và 1 không phải là số nguyên tố
    for i in range(2, int(limit**0.5) + 1):
        if is_prime[i]:
            for j in range(i * i, limit, i):
                is_prime[j] = False
    return tuple(i for i in range(limit) if is_prime[i])

# Tạo tuple P chứa các số nguyên tố nhỏ hơn 1 triệu
P = sieve_of_eratosthenes(1_000_000)

# In ra kích thước của tuple và một vài số nguyên tố
print(f"Số lượng số nguyên tố nhỏ hơn 1 triệu: {len(P)}")
print(f"Các số nguyên tố nhỏ hơn 1 triệu (10 số đầu tiên): {P[:100]}")
```

```
Đào Quang Huy MSSV 235752021610051
Số lượng số nguyên tố nhỏ hơn 1 triệu: 78498
Các số nguyên tố nhỏ hơn 1 triệu (10 số đầu tiên): (2,
3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 5
3, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107,
109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167,
173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229,
233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283,
293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359,
367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431,
433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491,
499, 503, 509, 521, 523, 541)
```

Hình 19. Chương trình tìm tất cả các số nguyên tố nhỏ hơn một giá trị `limit` mà bạn cung cấp

Chú thích :

- Hàm `sieve_of_eratosthenes(limit)`: Đây là hàm sử dụng Thuật toán Sàng Eratosthenes để tìm các số nguyên tố nhỏ hơn một giới hạn `limit`.
- Khởi tạo mảng `is_prime`: Đây là một danh sách chứa các giá trị boolean, trong đó mỗi phần tử biểu thị việc một số có phải là số nguyên tố hay không. Ban đầu, tất cả các số được giả định là nguyên tố (True), trừ 0 và 1 (không phải số nguyên tố).
- Sàng các số:
 - + Với mỗi số `i` từ 2 đến căn bậc hai của `limit`, nếu `i` là một số nguyên tố (tức là `is_prime[i]` là True), thì chúng ta đánh dấu tất cả các bội số của `i` từ i^2 đến `limit` là không phải số nguyên tố (đặt `is_prime[j]` là False).
 - + Quá trình này giúp loại bỏ tất cả các số không phải là nguyên tố.
 - + Trả về kết quả: Sau khi sàng, hàm trả về một tuple chứa các số nguyên tố nhỏ hơn `limit` (những số mà vẫn giữ giá trị True trong mảng `is_prime`).
- Tạo tuple P: Dùng hàm `sieve_of_eratosthenes(1_000_000)` để tạo ra một tuple chứa tất cả các số nguyên tố nhỏ hơn 1 triệu.
- Số lượng số nguyên tố: Sử dụng `len(P)` để in ra số lượng các số nguyên tố nhỏ hơn 1 triệu.
- In ra các số nguyên tố đầu tiên: In ra 100 số nguyên tố đầu tiên trong tuple P bằng cách sử dụng `P[:100]`.

1.3.20. Nhập n, in n dòng đầu tiên của tam giác pascal.

```
print('Đào Quang Huy MSSV 235752021610051')
def print_pascals_triangle(n):
    triangle = []

    for i in range(n):
        row = [1] * (i + 1) # Khởi tạo hàng với tất cả các giá trị là 1
        for j in range(1, i): # Tính các giá trị trung gian
            row[j] = triangle[i - 1][j - 1] + triangle[i - 1][j]
        triangle.append(row)

    for row in triangle:
        print(" ".join(map(str, row))) # In ra từng hàng

# Nhập số nguyên n
n = int(input("Nhập số nguyên n: "))

# In n dòng đầu tiên của tam giác Pascal
print_pascals_triangle(n)
```

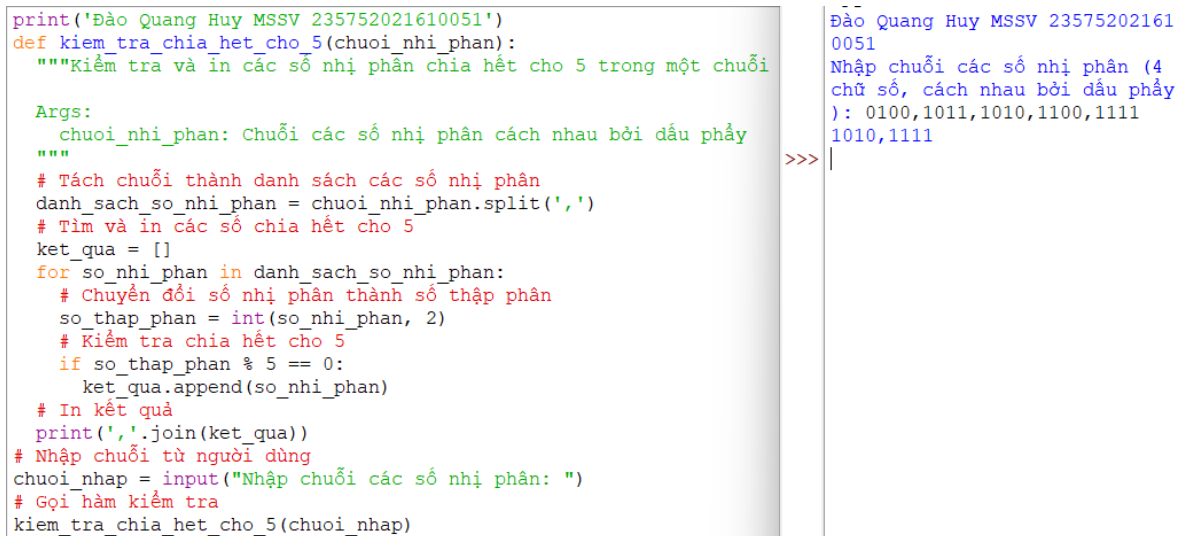
Đào Quang Huy MSSV 235752021610051
Nhập số nguyên n: 8
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1

Hình 20. Chương trình in ra tam giác Pascal với n dòng.

Chú thích :

- Hàm `print_pascals_triangle(n)`:
 - + Khởi tạo tam giác: Dùng một danh sách `triangle` để lưu các hàng của tam giác.
 - + Xây dựng từng hàng: Mỗi hàng bắt đầu với một danh sách chứa các số 1. Đối với các số ở giữa hàng (từ chỉ số 1 đến $i-1$), ta tính giá trị bằng cách lấy tổng của hai số ở hàng trước đó (tương ứng với các chỉ số liền kề).
 - + In tam giác Pascal: Cuối cùng, chương trình in từng hàng của tam giác ra màn hình, sử dụng `map(str, row)` để chuyển các số trong hàng thành chuỗi và `join(" ")` để nối chúng lại với nhau, ngăn cách bằng dấu cách.
- Phần nhập liệu và gọi hàm:
 - + Chương trình yêu cầu người dùng nhập một số nguyên `nnn`, sau đó gọi hàm `print_pascals_triangle(n)` để in ra `nnn` dòng đầu tiên của tam giác Pascal.

1.3.21. Viết một chương trình chấp nhận đầu vào là chuỗi các số nhị phân 4 chữ số, phân tách bởi dấu phẩy, kiểm tra xem chúng có chia hết cho 5 không. Sau đó in các số chia hết cho 5 thành dãy phân tách bởi dấu phẩy.



```
print('Đào Quang Huy MSSV 235752021610051')
def kiem_tra_chia_het_cho_5(chuoi_nhi_phan):
    """Kiểm tra và in các số nhị phân chia hết cho 5 trong một chuỗi"""
    Args:
        chuoi_nhi_phan: Chuỗi các số nhị phân cách nhau bởi dấu phẩy
    """
    # Tách chuỗi thành danh sách các số nhị phân
    danh_sach_so_nhi_phan = chuoi_nhi_phan.split(',')
    # Tìm và in các số chia hết cho 5
    ket_qua = []
    for so_nhi_phan in danh_sach_so_nhi_phan:
        # Chuyển đổi số nhị phân thành số thập phân
        so_thap_phan = int(so_nhi_phan, 2)
        # Kiểm tra chia hết cho 5
        if so_thap_phan % 5 == 0:
            ket_qua.append(so_nhi_phan)
    # In kết quả
    print(', '.join(ket_qua))
# Nhập chuỗi từ người dùng
chuoi_nhap = input("Nhập chuỗi các số nhị phân: ")
# Gọi hàm kiểm tra
kiem_tra_chia_het_cho_5(chuoi_nhap)
```

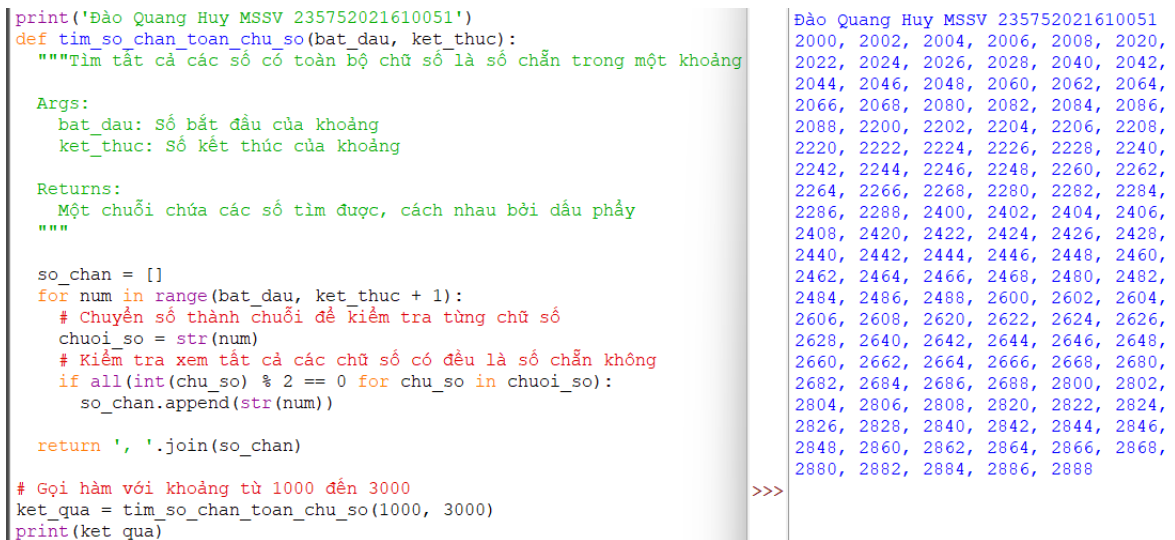
```
Đào Quang Huy MSSV 235752021610051
Nhập chuỗi các số nhị phân (4 chữ số, cách nhau bởi dấu phẩy): 0100,1011,1010,1100,1111
1010,1111
```

Hình 21. Chương trình kiểm tra số nhị phân 4 chữ số có chia hết cho 5 hay không.

Chú thích :

- Hàm `kiem_tra_chia_het_cho_5(chuoi_nhi_phan)`:
 - + Tách chuỗi: Chuỗi nhập vào được tách thành một danh sách các số nhị phân bằng phương thức `split(',')`.
 - + Chuyển đổi từ nhị phân sang thập phân: Dùng `int(so_nhi_phan, 2)` để chuyển mỗi số nhị phân trong danh sách thành số thập phân.
 - + Kiểm tra chia hết cho 5: Dùng toán tử `%` để kiểm tra xem số thập phân có chia hết cho 5 hay không.
 - + Lưu số nhị phân chia hết cho 5: Nếu số thập phân chia hết cho 5, số nhị phân tương ứng được thêm vào danh sách `ket_qua`.
 - + In kết quả: Cuối cùng, chương trình in ra các số nhị phân trong `ket_qua`, phân tách bằng dấu phẩy.
- Chương trình yêu cầu người dùng nhập vào chuỗi các số nhị phân (có 4 chữ số, phân tách bằng dấu phẩy).
 - + Sau khi nhập dữ liệu, chương trình kiểm tra các số nhị phân và in ra các số nhị phân chia hết cho 5.

1.3.22. Viết một chương trình tìm tất cả các số trong đoạn 1000 và 3000 (tính cả 2 số này) sao cho tất cả các chữ số trong số đó là số chẵn. In các số tìm được thành chuỗi cách nhau bởi dấu phẩy, trên một dòng.



```
print('Đào Quang Huy MSSV 235752021610051')
def tim_so_chan_toan_chu_so(bat_dau, ket_thuc):
    """Tìm tất cả các số có toàn bộ chữ số là số chẵn trong một khoảng

    Args:
        bat_dau: Số bắt đầu của khoảng
        ket_thuc: Số kết thúc của khoảng

    Returns:
        Một chuỗi chứa các số tìm được, cách nhau bởi dấu phẩy
    """

    so_chan = []
    for num in range(bat_dau, ket_thuc + 1):
        # Chuyển số thành chuỗi để kiểm tra từng chữ số
        chuoi_so = str(num)
        # Kiểm tra xem tất cả các chữ số có đều là số chẵn không
        if all(int(chuoi_so) % 2 == 0 for chuoi_so in chuoi_so):
            so_chan.append(str(num))

    return ', '.join(so_chan)

# Gọi hàm với khoảng từ 1000 đến 3000
ket_qua = tim_so_chan_toan_chu_so(1000, 3000)
print(ket_qua)
```

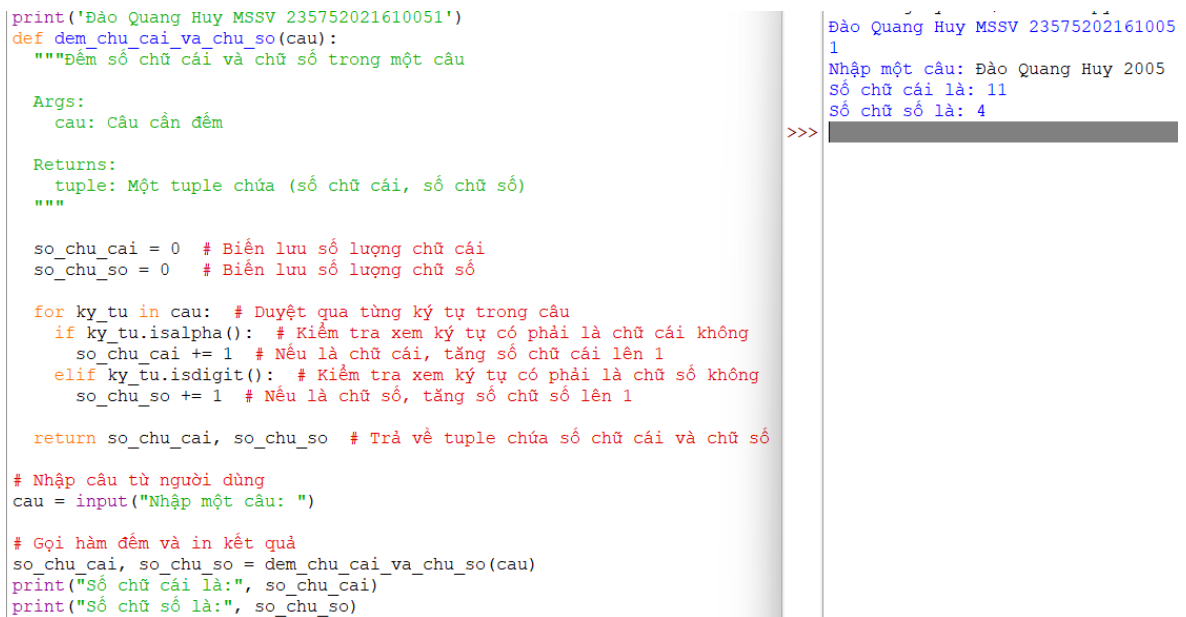
Đào Quang Huy MSSV 235752021610051
 2000, 2002, 2004, 2006, 2008, 2020,
 2022, 2024, 2026, 2028, 2040, 2042,
 2044, 2046, 2048, 2060, 2062, 2064,
 2066, 2068, 2080, 2082, 2084, 2086,
 2088, 2200, 2202, 2204, 2206, 2208,
 2220, 2222, 2224, 2226, 2228, 2240,
 2242, 2244, 2246, 2248, 2260, 2262,
 2264, 2266, 2268, 2280, 2282, 2284,
 2286, 2288, 2400, 2402, 2404, 2406,
 2408, 2420, 2422, 2424, 2426, 2428,
 2440, 2442, 2444, 2446, 2448, 2460,
 2462, 2464, 2466, 2468, 2480, 2482,
 2484, 2486, 2488, 2600, 2602, 2604,
 2606, 2608, 2620, 2622, 2624, 2626,
 2628, 2640, 2642, 2644, 2646, 2648,
 2660, 2662, 2664, 2666, 2668, 2680,
 2682, 2684, 2686, 2688, 2800, 2802,
 2804, 2806, 2808, 2820, 2822, 2824,
 2826, 2828, 2840, 2842, 2844, 2846,
 2848, 2860, 2862, 2864, 2866, 2868,
 2880, 2882, 2884, 2886, 2888

Hình 22. Chương trình tìm tất cả các số trong một khoảng mà tất cả các chữ số của chúng đều là số chẵn.

Chú thích :

- Bạn tạo một danh sách rỗng `so_chan` để lưu trữ các số có tất cả chữ số là số chẵn.
- Dùng vòng lặp `for` để duyệt qua tất cả các số trong khoảng từ `bat_dau` đến `ket_thuc`. Lưu ý rằng `range(bat_dau, ket_thuc + 1)` giúp bao gồm cả `ket_thuc`.
- Chuyển mỗi số `num` thành chuỗi (`str(num)`) để dễ dàng kiểm tra từng chữ số của số đó.
- `all()` là một hàm trong Python dùng để kiểm tra xem tất cả các điều kiện trong một iterable (danh sách, chuỗi, v.v.) có đúng hay không. Nếu tất cả điều kiện đều đúng, hàm `all()` trả về `True`, ngược lại trả về `False`.
- Mỗi chữ số trong chuỗi `chuoi_so` sẽ được kiểm tra với điều kiện `int(chuoi_so) % 2 == 0`. Điều này có nghĩa là chữ số đó phải là số chẵn (0, 2, 4, 6, 8).
- Nếu tất cả các chữ số đều là số chẵn, hàm `all()` trả về `True`, và số `num` sẽ được thêm vào danh sách `so_chan`.
- Nếu tất cả các chữ số của số `num` là số chẵn, bạn thêm số đó vào danh sách `so_chan`.
- Sau khi vòng lặp kết thúc, hàm sẽ trả về chuỗi chứa tất cả các số có chữ số là số chẵn, cách nhau bởi dấu phẩy. Bạn dùng `', '.join(so_chan)` để nối các phần tử trong danh sách `so_chan` thành một chuỗi, mỗi phần tử được ngăn cách bằng dấu phẩy.

1.3.23. Viết một chương trình chấp nhận đầu vào là một câu, đếm số chữ cái và chữ số trong câu đó.



```
print('Đào Quang Huy MSSV 235752021610051')
def dem_chu_cai_va_chu_so(cau):
    """Đếm số chữ cái và chữ số trong một câu

    Args:
        cau: Câu cần đếm

    Returns:
        tuple: Một tuple chứa (số chữ cái, số chữ số)
    """

    so_chu_cai = 0 # Biến lưu số lượng chữ cái
    so_chu_so = 0 # Biến lưu số lượng chữ số

    for ky_tu in cau: # Duyệt qua từng ký tự trong câu
        if ky_tu.isalpha(): # Kiểm tra xem ký tự có phải là chữ cái không
            so_chu_cai += 1 # Nếu là chữ cái, tăng số chữ cái lên 1
        elif ky_tu.isdigit(): # Kiểm tra xem ký tự có phải là chữ số không
            so_chu_so += 1 # Nếu là chữ số, tăng số chữ số lên 1

    return so_chu_cai, so_chu_so # Trả về tuple chứa số chữ cái và chữ số

# Nhập câu từ người dùng
cau = input("Nhập một câu: ")

# Gọi hàm đếm và in kết quả
so_chu_cai, so_chu_so = dem_chu_cai_va_chu_so(cau)
print("Số chữ cái là:", so_chu_cai)
print("Số chữ số là:", so_chu_so)
```

Đào Quang Huy MSSV 235752021610051
 1
 Nhập một câu: Đào Quang Huy 2005
 Số chữ cái là: 11
 Số chữ số là: 4

Hình 23. Chương trình đếm số lượng chữ cái và số nhập vào từ người dùng.

Chú thích :

- Bạn khởi tạo hai biến để lưu trữ số lượng chữ cái và chữ số:
 + so_chu_cai sẽ đếm số chữ cái (a-z, A-Z).
 + so_chu_so sẽ đếm số chữ số (0-9).
- Vòng lặp for này duyệt qua từng ký tự trong chuỗi cau (câu mà người dùng nhập vào). Hàm input() sẽ trả về một chuỗi, và bạn sẽ kiểm tra từng ký tự trong chuỗi này.
- Hàm isalpha() trả về True nếu ký tự là một chữ cái (bao gồm cả chữ cái hoa và chữ cái thường). Nếu đúng, bạn tăng biến so_chu_cai lên 1.
- Hàm isdigit() trả về True nếu ký tự là một chữ số (0-9). Nếu đúng, bạn tăng biến so_chu_so lên 1.
- Sau khi duyệt qua tất cả các ký tự trong câu, hàm trả về một tuple chứa hai giá trị: số chữ cái và số chữ số.
- Bạn yêu cầu người dùng nhập vào một câu thông qua hàm input(). Sau đó, gọi hàm dem_chu_cai_va_chu_so để đếm số chữ cái và chữ số trong câu đó. Kết quả được in ra màn hình.

1.3.24. Viết một chương trình chấp nhận đầu vào là một câu, đếm chữ hoa, chữ thường.

Chú thích :

- Bạn khởi tạo hai biến so_chu_hoa và so_chu_thuong để lưu trữ số lượng chữ hoa và chữ thường. Cả hai biến này đều bắt đầu với giá trị 0.
- Vòng lặp for này duyệt qua từng ký tự trong chuỗi cau (câu mà người dùng nhập vào). Hàm input() sẽ trả về một chuỗi, và bạn sẽ kiểm tra từng ký tự trong chuỗi này.
- Hàm isupper() trả về True nếu ký tự là một chữ cái in hoa (A-Z). Nếu ký tự là chữ hoa, biến so_chu_hoa sẽ được tăng thêm 1.
- Hàm islower() trả về True nếu ký tự là một chữ cái in thường (a-z). Nếu ký tự là chữ thường, biến so_chu_thuong sẽ được tăng thêm 1.

```

print('Đào Quang Huy 235752021610051')
def dem_chu_hoa_thuong(cau):
    """Đếm số lượng chữ hoa và chữ thường trong một câu

    Args:
        cau: Câu cần đếm

    Returns:
        tuple: Một tuple chứa (số chữ hoa, số chữ thường)
    """
    so_chu_hoa = 0 # Biến lưu số lượng chữ hoa
    so_chu_thuong = 0 # Biến lưu số lượng chữ thường

    for ky_tu in cau: # Duyệt qua từng ký tự trong câu
        if ky_tu.isupper(): # Kiểm tra xem ký tự có phải là chữ hoa không
            so_chu_hoa += 1 # Nếu là chữ hoa, tăng số chữ hoa lên 1
        elif ky_tu.islower(): # Kiểm tra xem ký tự có phải là chữ thường không
            so_chu_thuong += 1 # Nếu là chữ thường, tăng số chữ thường lên 1

    return so_chu_hoa, so_chu_thuong # Trả về tuple chứa số chữ hoa và chữ thường

# Nhập câu từ người dùng
cau = input("Nhập một câu: ")

# Gọi hàm đếm và in kết quả
so_chu_hoa, so_chu_thuong = dem_chu_hoa_thuong(cau)
print("Số chữ hoa là:", so_chu_hoa)
print("Số chữ thường là:", so_chu_thuong)

```

Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep 6 2024, 20:11:23) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:\Thực hành lập trình\Bài 4 Các kiểu dữ liệu có cấu trúc trong Python\Bài 4.24.py
Đào Quang Huy 235752021610051
Nhập một câu: Đào Quang Huy
Số chữ hoa là: 3
Số chữ thường là: 8
>>>

Hình 24. Chương trình đếm số lượng chữ thường và chữ hoa trong câu.

1.3.25. Sử dụng một danh sách để lọc các số lẻ từ danh sách được người dùng nhập vào.

```

print('Đào Quang Huy MSSV 235752021610051')
def loc_so_le(danh_sach):
    """Hàm lọc các số lẻ từ một danh sách

    Args:
        danh_sach: Danh sách các số nguyên

    Returns:
        list: Danh sách các số lẻ
    """
    so_le = [] # Khởi tạo danh sách rỗng để lưu các số lẻ
    for so in danh_sach: # Duyệt qua từng số trong danh sách
        if so % 2 != 0: # Kiểm tra nếu số đó là số lẻ (số không chia hết cho 2)
            so_le.append(so) # Nếu là số lẻ, thêm vào danh sách so_le
    return so_le # Trả về danh sách các số lẻ

# Nhập danh sách từ người dùng
danh_sach_nhap = input("Nhập danh sách các số (cách nhau bằng dấu phẩy): ")

# Chuyển đổi chuỗi nhập vào thành một danh sách các số nguyên
danh_sach = [int(so) for so in danh_sach_nhap.split(',')] # Tách chuỗi và chuyển

# Gọi hàm lọc và in kết quả
ket_qua = loc_so_le(danh_sach) # Gọi hàm lọc các số lẻ
print("Danh sách các số lẻ:", ket_qua) # In danh sách các số lẻ

```

Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep 6 2024, 20:11:23) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:\Thực hành lập trình\Bài 4 Các kiểu dữ liệu có cấu trúc trong Python\Bài 4.25.py
Đào Quang Huy MSSV 235752021610051
Nhập danh sách các số (cách nhau bằng dấu phẩy): 16,8,2005
Danh sách các số lẻ: [2005]
>>>

Hình 25. Chương trình tìm số lẻ từ danh sách người dùng nhập vào.

Chú thích:

- **Khởi tạo hàm `loc_so_le(danh_sach)`:**
 - + Hàm này nhận vào một đối số là `danh_sach`, một danh sách chứa các số nguyên.
 - + Mục tiêu của hàm là lọc ra tất cả các số lẻ trong danh sách và trả về một danh sách mới chứa các số lẻ.
- Hàm duyệt qua từng số trong danh sách `danh_sach` bằng vòng lặp `for`.
- `so % 2 != 0` kiểm tra xem một số `so` có phải là số lẻ không.
 - + Nếu `so` chia cho 2 có dư (tức là không chia hết cho 2), thì số này là số lẻ.
 - + Nếu điều kiện này đúng (số lẻ), chúng ta sẽ thêm số đó vào danh sách `so_le`.
- Nếu điều kiện trên đúng (số `so` là số lẻ), số đó sẽ được thêm vào danh sách `so_le` bằng cách sử dụng phương thức `append()`.
- `danh_sach_nhap.split(',')` tách chuỗi người dùng nhập vào thành một danh sách các chuỗi con, mỗi chuỗi đại diện cho một số (ví dụ: ["1", "2", "3", "4", "5"]).
- Sau đó, `int(so)` chuyển mỗi chuỗi thành một số nguyên, và kết quả là một danh sách các số nguyên (ví dụ: [1, 2, 3, 4, 5]).

1.3.26. Viết chương trình tính số tiền thực của một tài khoản ngân hàng dựa trên nhật ký giao dịch được nhập vào từ giao diện điều khiển.

```
print('Đào Quang Huy MSSV 235752021610051')
def tinh_so_du(nhat_ky):
    """Tính số dư tài khoản dựa trên nhật ký giao dịch

    Args:
        nhat_ky: Danh sách các giao dịch (D: gửi, W: rút)

    Returns:
        int: Số dư cuối cùng của tài khoản
    """

    so_du = 0 # Khởi tạo số dư ban đầu là 0
    for giao_dich in nhat_ky: # Duyệt qua từng giao dịch trong nhật ký
        loai_giao_dich, so_tien = giao_dich.split() # Tách loại giao dịch và số tiền
        so_tien = int(so_tien) # Chuyển số tiền thành số nguyên
        if loai_giao_dich == 'D': # Nếu là giao dịch gửi tiền
            so_du += so_tien # Thêm số tiền vào số dư
        elif loai_giao_dich == 'W': # Nếu là giao dịch rút tiền
            so_du -= so_tien # Trừ số tiền khỏi số dư
    return so_du # Trả về số dư cuối cùng

# Nhập nhật ký giao dịch từ người dùng
nhat_ky = [] # Khởi tạo danh sách nhật ký giao dịch
while True:
    giao_dich = input("Nhập giao dịch (D <số tiền> hoặc W <số tiền>, nhập 'quit' để kết thúc): ")
    if giao_dich.lower() == 'quit': # Nếu người dùng nhập 'quit', thoát vòng lặp
        break
    nhat_ky.append(giao_dich) # Thêm giao dịch vào danh sách nhật ký

# Tính số dư và in kết quả
so_du_cuoi = tinh_so_du(nhat_ky) # Gọi hàm tính số dư cuối cùng
print("Số dư cuối cùng của tài khoản:", so_du_cuoi) # In số dư cuối cùng
```

Hình 26.1 Chương trình tính số dư ngân hàng dựa vào nhật lý giao dịch.

Chú thích :

- so_du là biến dùng để lưu số dư tài khoản. Ban đầu, số dư được khởi tạo là 0.
- Dòng này duyệt qua từng giao dịch trong danh sách nhat_ky.
- Mỗi giao dịch được tách thành hai phần: loại giao dịch (D hoặc W) và số tiền.
- Phương thức split() sẽ tách chuỗi thành các phần từ cách nhau bởi dấu cách.
+ Ví dụ: "D 1000" sẽ được tách thành loai_giao_dich = 'D' và so_tien = '1000'.
- Số tiền sau khi tách ra là một chuỗi, vì vậy ta cần chuyển nó thành số nguyên bằng cách sử dụng int().
- Người dùng sẽ nhập các giao dịch một cách liên tiếp, mỗi giao dịch có dạng D <số tiền> (gửi tiền) hoặc W <số tiền> (rút tiền).
- Nếu người dùng nhập 'quit', chương trình sẽ dừng nhập và kết thúc vòng lặp.

```
Đào Quang Huy MSSV 235752021610051
Nhập giao dịch (D <số tiền> hoặc W <số tiền>, nhập 'quit' để kết thúc): D 1000
Nhập giao dịch (D <số tiền> hoặc W <số tiền>, nhập 'quit' để kết thúc): W 200
Nhập giao dịch (D <số tiền> hoặc W <số tiền>, nhập 'quit' để kết thúc): D 1000
Nhập giao dịch (D <số tiền> hoặc W <số tiền>, nhập 'quit' để kết thúc): W 900
Nhập giao dịch (D <số tiền> hoặc W <số tiền>, nhập 'quit' để kết thúc): quit
Số dư cuối cùng của tài khoản: 900
```

Hình 26.2. Ví dụ chạy chương trình cho đoạn mã trên.

Bài 5. Thiết kế module trong Python

1.1. Mục đích .

Giúp sinh viên nắm bắt việc thiết kế module trong lập trình python, sử dụng module thư viện numpy trong các ứng dụng.

1.2. Cơ sở lý thuyết .

Xem các quy tắc khai báo, thiết kế và sử dụng module trong python, các thuật toán tìm kiếm, sắp xếp, cài đặt và sử dụng thư viện numpy.

1.3. Các bước tiến hành.

1.3.1. Sử dụng module. Định nghĩa một module toán học gọi là mymath và sử dụng module này từ một tập lệnh riêng biệt.

```
# File: mymath.py

def square(n):
    return n * n

def cube(n):
    return n * n * n

def average(values):
    nvals = len(values)
    sum = 0.0
    for v in values:
        sum += v
    return float(sum) / nvals
```

Hình 1.1. Tạo 1 module có tên mymath.

Chú thích:

- Hàm square(n):Hàm này nhận đầu vào là một số n và trả về bình phương của số đó (tức là n^2).
- Hàm cube(n) :Hàm này nhận đầu vào là một số n và trả về lập phương của số đó (tức là n^3).
- Hàm average(values) :Hàm này nhận đầu vào là một danh sách hoặc một iterable values (một tập hợp các giá trị). Nó tính tổng các giá trị trong danh sách đó và chia cho số lượng các giá trị để trả về giá trị trung bình (hay còn gọi là "mean").Đầu tiên, hàm tính tổng các phần tử trong values. Sau đó, nó chia tổng đó cho số lượng phần tử để tính trung bình.

```
print('Đào Quang Huy MSSV 235752021610051')
# File: main.py

import mymath

# Danh sách các giá trị
values = [2, 4, 6, 8, 10]

# Tính bình phương của từng giá trị trong danh sách
print('Squares:')
for v in values:
    print(mymath.square(v)) # Gọi hàm square từ mymath

# Tính lập phương của từng giá trị trong danh sách
print('Cubes:')
for v in values:
    print(mymath.cube(v)) # Gọi hàm cube từ mymath

# Tính trung bình của danh sách
print('Average: ' + str(mymath.average(values))) # Gọi hàm average từ mymath
```

Hình 1.2. Chương trình chạy cho module mymath trên.


```

print('Đào Quang Huy MSSV 235752021610051')
# File: main.py

import mymath as mt # Nhập module mymath với tên tắt là mt

# Danh sách các giá trị
values = [2, 4, 6, 8, 10]

# Tính bình phương của từng giá trị trong danh sách
print('Squares:')
for v in values:
    print(mt.square(v)) # Gọi hàm square từ mymath (sử dụng tên tắt mt)

# Tính lập phương của từng giá trị trong danh sách
print('Cubes:')
for v in values:
    print(mt.cube(v)) # Gọi hàm cube từ mymath (sử dụng tên tắt mt)

# Tính trung bình của danh sách
print('Average: ' + str(mt.average(values))) # Gọi hàm average từ mymath

```

```

Đào Quang Huy MSSV 235752021610051
1
Squares:
4
16
36
64
100
Cubes:
8
64
216
512
1000
Average: 6.0
>>>

```

Hình 1.3. Dùng tên viết tắt để gọi tên cho module mymath.

Chú thích :

- Chúng ta có thể gán cho module mymath một tên viết tắt để dễ dàng trong việc gọi hàm.

1.3.2. Sử dụng thư viện tiêu chuẩn của python (datetime).

```

print('Đào Quang Huy MSSV 235752021610051')
import datetime as dt
format = '%Y-%m-%dT%H:%M:%S'
t1 = dt.datetime.strptime('2005-08-16T14:45:52', format)
print('Day ' + str(t1.day))
print('Month ' + str(t1.month))
print('Minute ' + str(t1.minute))
print('Second ' + str(t1.second))
# Define todays date and time
t2 = dt.datetime.now()
diff = t2 - t1
print('How many days difference? ' + str(diff.days))

```

```

Đào Quang Huy MSSV 235752021610051
Day 16
Month 8
Minute 45
Second 52
How many days difference? 7039
>>>

```

Hình 2. Sử dụng thư viện datetime của Python để làm việc với ngày và giờ.

Chú thích :

- Bạn nhập khẩu module datetime với bí danh dt, giúp việc gọi các hàm trong module này trở nên ngắn gọn hơn.
- Bạn định nghĩa một chuỗi định dạng format = '%Y-%m-%dT%H:%M:%S' để mô tả cách ngày giờ được thể hiện trong chuỗi.
+ %Y là năm (4 chữ số), %m là tháng (2 chữ số), %d là ngày (2 chữ số), %H là giờ (24 giờ), %M là phút, %S là giây.
- Định dạng này sẽ dùng để chuyển đổi chuỗi '2005-08-16T14:45:52' thành đối tượng datetime.
- Bạn sử dụng hàm strptime() để chuyển đổi chuỗi '2005-08-16T14:45:52' thành đối tượng datetime theo định dạng đã chỉ định.
- Bạn truy cập các thành phần của đối tượng t1 (là một đối tượng datetime), chẳng hạn như ngày (t1.day), tháng (t1.month), phút (t1.minute), và giây (t1.second).
- Bạn sử dụng dt.datetime.now() để lấy thời gian hiện tại và lưu vào biến t2.
- Bạn tính sự khác biệt giữa t2 (thời gian hiện tại) và t1 (ngày giờ đã cho). Phép trừ giữa hai đối tượng datetime sẽ trả về một đối tượng timedelta, từ đó bạn có thể lấy số ngày bằng cách truy cập thuộc tính .days.

1.3.3. Viết chương trình sử dụng thư viện NumPy để tạo một mảng với các giá trị nằm trong khoảng từ 12 đến 38.

```
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Users\LENOVO>python --version
Python 3.12.6

C:\Users\LENOVO>pip --version
pip 24.2 from C:\Users\LENOVO\AppData\Local\Programs\Python\Python312\Lib\site-packages\pip (python 3.12)

C:\Users\LENOVO>pip install numpy
Requirement already satisfied: numpy in c:\users\lenovo\appdata\local\programs\python\python312\lib\site-packages (2.1.3)

[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\LENOVO>
```

Hình 3.1. Cài đặt chương trình numpy để chạy chương trình.

Chú ý :

- Mở command prompt và kiểm tra phiên bản python bằng cú pháp.
- .
- .
- .
- .

```
print('Đào Quang Huy MSSV 235752021610051')
import numpy as np # Nhập thư viện NumPy

# Tạo mảng với các giá trị từ 12 đến 37 (38 không được bao gồm)
x = np.arange(12, 38)

# In mảng ra màn hình
print(x)
```

```
>>> |
Đào Quang Huy MSSV 23575
2021610051
[12 13 14 15 16 17 18 19
 20 21 22 23 24 25 26 27
 28 29 30 31 32 33 34 35
 36 37]
```

Hình 3.2. Sử dụng thư viện numpy để chạy chương trình.

Chú thích:

- np.arange(12, 38): Hàm arange() tạo một mảng có các giá trị bắt đầu từ 12 đến 37 (vì tham số kết thúc 38 sẽ không được bao gồm). Mặc định, các giá trị trong mảng sẽ được tạo ra cách đều nhau, với bước nhảy là 1.

1.3.4. Viết chương trình để tạo một mảng với các giá trị nằm trong khoảng từ 12 đến 38 và đảo ngược mảng đã tạo (phần tử đầu tiên trở thành cuối cùng).

```
print('Đào Quang Huy MSSV 235752021610051')
import numpy as np

# Tạo mảng với các giá trị từ 12 đến 38 (bao gồm cả 12,
arr = np.arange(12, 39)

# Đảo ngược mảng
reversed_arr = arr[::-1]

# In mảng ban đầu và mảng đã đảo ngược ra màn hình
print("Mảng ban đầu:")
print(arr)
print("\nMảng sau khi đảo ngược:")
print(reversed_arr)
```

```
>>> |
Đào Quang Huy MSSV 235752021610051
1
Mảng ban đầu:
[12 13 14 15 16 17 18 19 20 21 22
 23 24 25 26 27 28 29 30 31 32 33
 34 35
 36 37 38]

Mảng sau khi đảo ngược:
[38 37 36 35 34 33 32 31 30 29 28
 27 26 25 24 23 22 21 20 19 18 17
 16 15
 14 13 12]
```

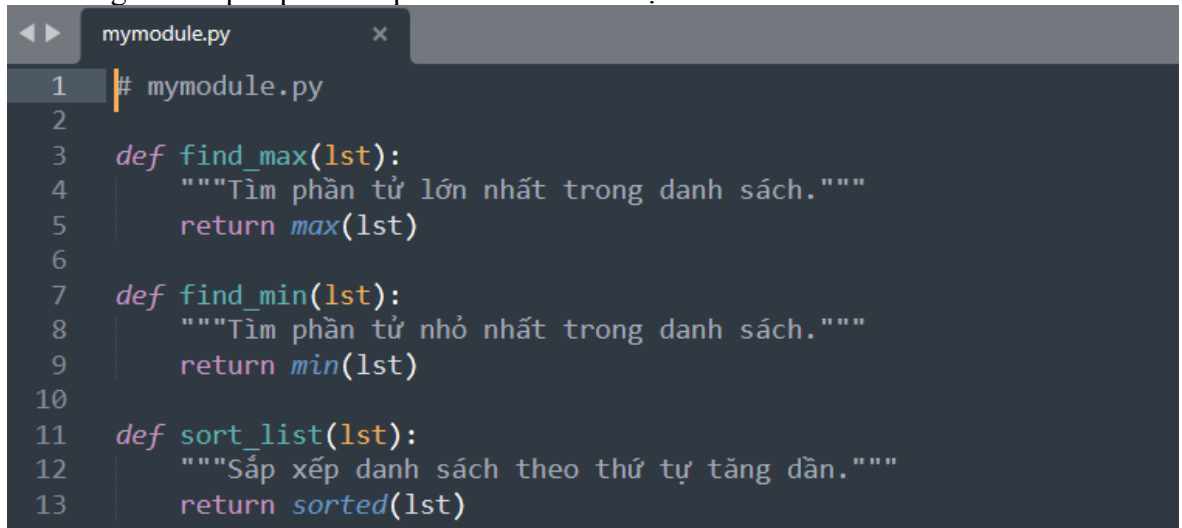
Hình 4. Chương trình đảo ngược mảng giá trị sử dụng thư viện numpy.

Chú thích:

- np.arange(12, 39): Tạo một mảng với các giá trị từ 12 đến 38.
- arr[::-1]: Đây là một kỹ thuật slicing trong Python. Cách viết [::-1] cho phép bạn đảo ngược mảng. Cụ thể:
 - + Dấu : chỉ định lấy toàn bộ phần tử của mảng.
 - + Dấu -1 chỉ định bước nhảy âm, có nghĩa là duyệt mảng từ cuối lên đầu, tức là đảo ngược mảng.

1.3.5. Viết chương trình tìm phần tử lớn nhất và nhỏ nhất của một danh sách

- Số lượng và giá trị của list được nhập từ bàn phím
- Phương thức sắp xếp và tìm phần tử lớn nhất được viết thành module

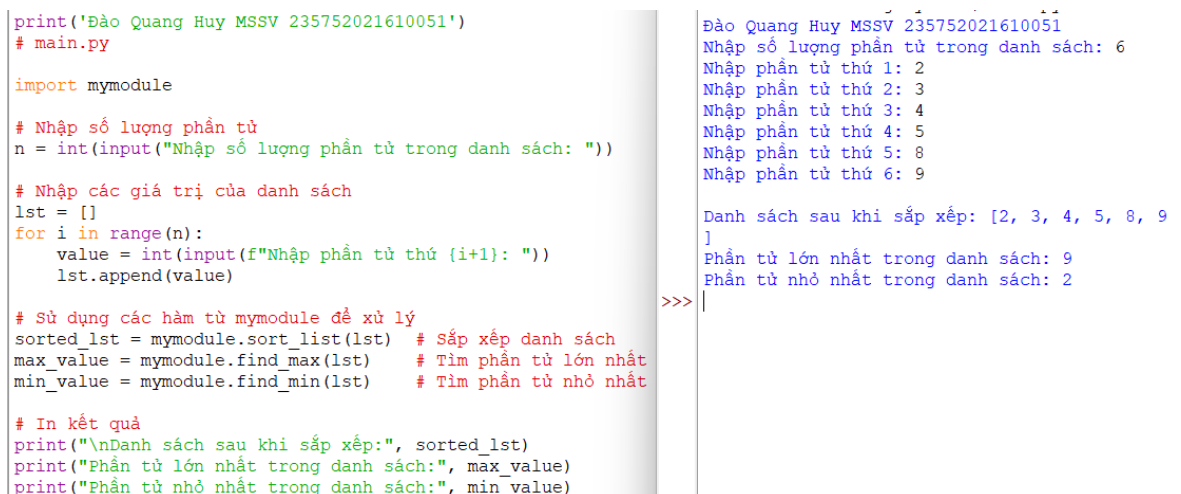


```
1 # mymodule.py
2
3 def find_max(lst):
4     """Tìm phần tử lớn nhất trong danh sách."""
5     return max(lst)
6
7 def find_min(lst):
8     """Tìm phần tử nhỏ nhất trong danh sách."""
9     return min(lst)
10
11 def sort_list(lst):
12     """Sắp xếp danh sách theo thứ tự tăng dần."""
13     return sorted(lst)
```

Hình 5.1. Tạo 1 module có tên mymodule để chạy chương trình python.

Chú thích:

- Mở sublime text và tạo 1 module có tên mymodule.
- find_max(lst): Hàm này nhận một danh sách lst và trả về phần tử lớn nhất trong danh sách bằng cách sử dụng hàm max().
- find_min(lst): Hàm này nhận một danh sách lst và trả về phần tử nhỏ nhất trong danh sách bằng cách sử dụng hàm min().
- sort_list(lst): Hàm này nhận một danh sách lst và trả về danh sách đã được sắp xếp theo thứ tự tăng dần bằng cách sử dụng hàm sorted().



```
print('Đào Quang Huy MSSV 235752021610051')
# main.py

import mymodule

# Nhập số lượng phần tử
n = int(input("Nhập số lượng phần tử trong danh sách: "))

# Nhập các giá trị của danh sách
lst = []
for i in range(n):
    value = int(input(f"Nhập phần tử thứ {i+1}: "))
    lst.append(value)

# Sử dụng các hàm từ mymodule để xử lý
sorted_lst = mymodule.sort_list(lst) # Sắp xếp danh sách
max_value = mymodule.find_max(lst)   # Tìm phần tử lớn nhất
min_value = mymodule.find_min(lst)   # Tìm phần tử nhỏ nhất

# In kết quả
print("\nDanh sách sau khi sắp xếp:", sorted_lst)
print("Phần tử lớn nhất trong danh sách:", max_value)
print("Phần tử nhỏ nhất trong danh sách:", min_value)
```

Đào Quang Huy MSSV 235752021610051
Nhập số lượng phần tử trong danh sách: 6
Nhập phần tử thứ 1: 2
Nhập phần tử thứ 2: 3
Nhập phần tử thứ 3: 4
Nhập phần tử thứ 4: 5
Nhập phần tử thứ 5: 8
Nhập phần tử thứ 6: 9

Danh sách sau khi sắp xếp: [2, 3, 4, 5, 8, 9]
Phần tử lớn nhất trong danh sách: 9
Phần tử nhỏ nhất trong danh sách: 2
>>> |

Hình 5.2 Chạy chương trình python từ module đã tạo.

Chú thích :

- Chúng ta sử dụng input() để nhập số lượng phần tử n và giá trị các phần tử cho danh sách lst.
- Sau đó, các giá trị này được lưu vào danh sách lst.
- **Sử dụng module mymodule:**
 - + Hàm sort_list(lst) được gọi để sắp xếp danh sách lst theo thứ tự tăng dần.
 - + Hàm find_max(lst) tìm phần tử lớn nhất.
 - + Hàm find_min(lst) tìm phần tử nhỏ nhất.

1.3.6. In ra vị trí phần tử lớn nhất và nhỏ nhất tìm được ở bài tập trên.

```
# Tìm chỉ số (vị trí) của phần tử lớn nhất và nhỏ nhất
max_index = lst.index(max_value)
min_index = lst.index(min_value)

# In kết quả
print("\nDanh sách sau khi sắp xếp:", sorted_lst)
print("Phần tử lớn nhất trong danh sách:", max_value, "vị trí:", max_index)
print("Phần tử nhỏ nhất trong danh sách:", min_value, "vị trí:", min_index)
```

Hình 6.1. Đoạn mã chạy tìm vị trí từ ví dụ câu 5.

Chú ý:

- Vẫn sử dụng đoạn code như câu 5 và thêm 2 hàm vào cuối để tìm vị trí phần tử lớn nhất và nhỏ nhất.
- Kết quả tìm số lớn nhất và nhỏ nhất vẫn như câu 5 và tìm được thêm vị trí của số đó trong chương trình.

```
Đào Quang Huy MSSV 235752021610051
Nhập số lượng phần tử trong danh sách: 6
Nhập phần tử thứ 1: 2
Nhập phần tử thứ 2: 3
Nhập phần tử thứ 3: 4
Nhập phần tử thứ 4: 5
Nhập phần tử thứ 5: 8
Nhập phần tử thứ 6: 9

Danh sách sau khi sắp xếp: [2, 3, 4, 5, 8, 9]
Phần tử lớn nhất trong danh sách: 9 vị trí: 5
Phần tử nhỏ nhất trong danh sách: 2 vị trí: 0
```

Hình 6.2. Kết quả sau khi thêm hàm vào câu 5 để tìm vị trí.

1.3.7. Viết chương trình sử dụng thư viện NumPy để tạo một mảng có cấu trúc từ tên sinh viên, chiều cao, lớp và các kiểu dữ liệu của họ. Bây giờ sắp xếp các mảng theo chiều cao.

```
print('Đào Quang Huy MSSV 235752021610051')
import numpy as np

# Tạo một mảng có cấu trúc với các cột: Tên, Chiều cao, Lớp
dt = np.dtype([('Tên', 'U50'), ('Chiều cao', 'f4'), ('Lớp', 'U20')])

# Dữ liệu về sinh viên
data = np.array([('Đào Quang Huy', 1.75, '12A1'),
                 ('Ngô Mạnh Nguyên', 1.60, '12B1'),
                 ('Đào Mạnh Nguyên', 1.80, '12A2'),
                 ('Ngô Quang Huy', 1.65, '12C1')], dtype=dt)

# Sắp xếp mảng theo chiều cao (cột 'Chiều cao')
sorted_data = np.sort(data, order='Chiều cao')

# In ra mảng đã sắp xếp
print(sorted_data)
```

```
Đào Quang Huy MSSV 235752021610051
1
[('Ngô Mạnh Nguyên', 1.6 , '12B1')
 ('Ngô Quang Huy', 1.65, '12C1')
 ('Đào Quang Huy', 1.75, '12A1')
 ('Đào Mạnh Nguyên', 1.8 , '12A2')
 ]
```

Hình 7. Chương trình sắp xếp tăng dần theo chiều cao từ thư viện numpy.

Chú thích:

- numpy: Đây là thư viện cốt lõi của Python cho tính toán khoa học, đặc biệt là các phép toán trên mảng.

- dt: Biến này đại diện cho một kiểu dữ liệu cấu trúc (structured dtype) với ba trường:
 - + 'Tên': 'U50': Một chuỗi Unicode có độ dài tối đa 50 ký tự để lưu tên sinh viên.
 - + 'Chiều cao': 'f4': Một số thực đơn độ chính xác (float) 4 byte để lưu chiều cao.
 - + 'Lớp': 'U20': Một chuỗi Unicode có độ dài tối đa 20 ký tự để lưu lớp học.
- data: Đây là một mảng NumPy chứa dữ liệu về các sinh viên. Mỗi phần tử trong mảng là một tuple gồm tên, chiều cao và lớp. Kiểu dữ liệu của mảng được quy định bởi dt đã định nghĩa ở trên.
- np.sort(data, order='Chiều cao'): Hàm này được sử dụng để sắp xếp mảng data theo thứ tự tăng dần của cột 'Chiều cao'. Kết quả được gán vào biến sorted_data.

1.3.8. Xây dựng hàm “Sequential_Search(dlist, item)” (giải thuật tìm kiếm tuyến tính) dưới dạng module. Viết chương trình nhập một dlist n phần tử từ bàn phím và tìm kiếm phần tử item bất kỳ.

```

1  # sequential_search.py
2
3  def Sequential_Search(dlist, item):
4      """
5          Hàm tìm kiếm tuyến tính trong danh sách dlist để tìm phần tử item.
6          Trả về chỉ số của phần tử nếu tìm thấy, hoặc -1 nếu không tìm thấy.
7          """
8      for index in range(len(dlist)):
9          if dlist[index] == item:
10             return index # Trả về chỉ số của phần tử tìm thấy
11     return -1 # Trả về -1 nếu không tìm thấy phần tử
12

```

Hình 8.1. Tạo module sequential_Search mã nguồn của hàm tìm kiếm tuyến tính.

Chú thích:

- for index in range(len(dlist)):: Duyệt qua từng phần tử trong danh sách dlist. len(dlist) trả về số lượng phần tử trong danh sách, và index sẽ thay đổi từ 0 đến len(dlist)-1.
- if dlist[index] == item:: Nếu phần tử tại chỉ số index của danh sách dlist bằng với giá trị item mà bạn muốn tìm, hàm sẽ thực hiện bước tiếp theo.
- return index: Nếu tìm thấy phần tử, hàm sẽ trả về chỉ số index của phần tử đó trong danh sách.
- return -1: Nếu vòng lặp kết thúc mà không tìm thấy phần tử, hàm sẽ trả về -1 để thông báo rằng phần tử không tồn tại trong danh sách.

```

print('Đào Quang Huy MSSV 235752021610051')
# main.py

import sequential_search

def main():
    # Nhập số lượng phần tử trong danh sách
    n = int(input("Nhập số lượng phần tử trong danh sách: "))

    # Nhập danh sách các phần tử
    dlist = []
    print(f"Nhập {n} phần tử vào danh sách:")
    for i in range(n):
        element = input(f"Phần tử {i+1}: ")
        dlist.append(element)

    # Nhập phần tử cần tìm
    item = input("Nhập phần tử cần tìm: ")

    # Gọi hàm tìm kiếm tuyến tính
    result = sequential_search.Sequential_Search(dlist, item)

    # Hiển thị kết quả tìm kiếm
    if result != -1:
        print(f"Phần tử '{item}' được tìm thấy tại vị trí {result}.")
    else:
        print(f"Phần tử '{item}' không có trong danh sách.")

if __name__ == "__main__":
    main()

```

>>>

```

Đào Quang Huy MSSV 235752021610051
Nhập số lượng phần tử trong danh sách: 5
Nhập 5 phần tử vào danh sách:
Phần tử 1: năm
Phần tử 2: huy
Phần tử 3: nguyên
Phần tử 4: anh
Phần tử 5: tuần
Nhập phần tử cần tìm: năm
Phần tử 'năm' được tìm thấy tại vị trí 0.

```

Hình 8.2. Chương trình chạy cho module sequential_search.

Chú thích :

- Dòng `n = int(input("Nhập số lượng phần tử trong danh sách: "))` yêu cầu người dùng nhập số lượng phần tử trong danh sách. Giá trị này được lưu vào biến `n`.
- hàm trình tạo một danh sách rỗng `dlist = []`.
- Sau đó, sử dụng vòng lặp `for` để nhập từng phần tử và thêm vào danh sách `dlist`. Các phần tử được yêu cầu nhập vào dưới dạng chuỗi.
- Dòng `item = input("Nhập phần tử cần tìm: ")` yêu cầu người dùng nhập phần tử cần tìm trong danh sách.
- Hàm `sequential_search.Sequential_Search(dlist, item)` được gọi để tìm kiếm phần tử `item` trong danh sách `dlist`. Kết quả trả về (chỉ số hoặc -1 nếu không tìm thấy) được lưu trong biến `result`.
- Nếu hàm tìm kiếm trả về một chỉ số (không phải -1), nghĩa là phần tử đã được tìm thấy trong danh sách. Chương trình sẽ in ra thông báo và vị trí của phần tử.
- Nếu trả về -1, chương trình sẽ in ra thông báo rằng phần tử không có trong danh sách.

1.3.9. Xây dựng hàm “*binary_search(list, value)*” (giải thuật tìm kiếm nhị phân) dưới dạng module. Viết chương trình nhập một list `n` phần tử từ bàn phím và tìm kiếm phần tử `value` bất kỳ.

```

# binary_search.py
def binary_search(sorted_list, value):
    left, right = 0, len(sorted_list) - 1 # Khởi tạo các chỉ số trái và phải
    while left <= right: # Khi nào vẫn còn khoảng cách giữa trái và phải
        mid = (left + right) // 2 # Tính chỉ số giữa (mid)

        # Kiểm tra phần tử ở giữa
        if sorted_list[mid] == value: # Nếu phần tử ở giữa bằng giá trị cần tìm
            return mid # Trả về chỉ số của phần tử tìm thấy
        elif sorted_list[mid] < value: # Nếu phần tử giữa nhỏ hơn giá trị cần tìm
            left = mid + 1 # Thay đổi trái để tìm trong nửa sau
        else: # Nếu phần tử giữa lớn hơn giá trị cần tìm
            right = mid - 1 # Thay đổi phải để tìm trong nửa trước

    return -1 # Nếu không tìm thấy giá trị, trả về -1

```

Hình 9.1. Tạo module giải thuật toán tìm kiếm nhị phân (binary search).

Chú thích :

- sorted_list: Là danh sách đã được sắp xếp (lưu ý là danh sách phải được sắp xếp tăng dần để thuật toán tìm kiếm nhị phân hoạt động chính xác).
- value: Là giá trị bạn muốn tìm trong danh sách.
- left = 0: Đại diện cho chỉ số đầu tiên của danh sách.
- right = len(sorted_list) - 1: Đại diện cho chỉ số cuối cùng của danh sách.
- Vòng lặp while (left <= right): Vòng lặp này sẽ tiếp tục chạy miễn là left nhỏ hơn hoặc bằng right, nghĩa là còn khoảng cách giữa hai chỉ số này.
- mid = (left + right) // 2: Tính chỉ số giữa của đoạn danh sách mà chúng ta đang tìm kiếm. Sử dụng phép chia lấy phần nguyên (//) để có chỉ số nguyên.
- Nếu sorted_list[mid] == value: Nếu giá trị tại chỉ số giữa bằng với giá trị cần tìm (value), thì trả về chỉ số mid (chỉ số của giá trị tìm thấy).
- Nếu sorted_list[mid] < value: Nếu giá trị tại chỉ số giữa nhỏ hơn giá trị cần tìm, tức là giá trị cần tìm nằm ở nửa sau của danh sách. Ta sẽ thay đổi left = mid + 1 để tìm kiếm trong nửa sau.
- Nếu sorted_list[mid] > value: Nếu giá trị tại chỉ số giữa lớn hơn giá trị cần tìm, tức là giá trị cần tìm nằm ở nửa trước của danh sách. Ta sẽ thay đổi right = mid - 1 để tìm kiếm trong nửa trước.
- Nếu vòng lặp kết thúc mà không tìm thấy giá trị (tức là không còn khoảng cách giữa left và right), hàm sẽ trả về -1, báo hiệu rằng giá trị không có trong danh sách.

```
print('Đào Quang Huy MSSV 235752021610051')
import binary_search # Import module chứa hàm tìm kiếm nhị phân

def main():
    # Nhập số lượng phần tử
    n = int(input("Nhập số lượng phần tử trong danh sách: "))

    # Nhập các phần tử
    lst = []
    print("Nhập các phần tử:")
    for i in range(n):
        lst.append(int(input(f"Phần tử {i + 1}: ")))

    # Sắp xếp danh sách
    lst.sort()

    # Nhập giá trị cần tìm
    value = int(input("Nhập giá trị cần tìm: "))

    # Sử dụng hàm binary_search từ module binary_search
    result = binary_search.binary_search(lst, value)

    # In kết quả
    if result != -1:
        print(f"Giá trị {value} được tìm thấy tại chỉ mục {result}.")
    else:
        print(f"Giá trị {value} không có trong danh sách.")

# Chạy chương trình chính
if __name__ == "__main__":
    main()
```

```
Đào Quang Huy MSSV 235752021610051
Nhập số lượng phần tử trong danh sách: 5
Nhập các phần tử:
Phần tử 1: 16
Phần tử 2: 8
Phần tử 3: 2005
Phần tử 4: 20
Phần tử 5: 15
Nhập giá trị cần tìm: 15
Giá trị 15 được tìm thấy tại chỉ mục 1.
>>>
```

Hình 9.2. Chương trình sử dụng module binary_search để thực hiện việc tìm kiếm nhị phân trong danh sách được nhập từ bàn phím.

Chú ý:

- Trong Python (và nhiều ngôn ngữ lập trình khác), chỉ số của các phần tử trong danh sách bắt đầu từ 0.
- Do đó, danh sách trên được sắp xếp theo thứ tự tăng dần 8, 15, 16, 20, 2005. Cho nên khi tìm số 15 thì kết quả sẽ là ở chỉ số thứ 1, vì trong python các phần tử trong danh sách sẽ bắt đầu từ không như đã nêu ở trên.

Chú thích:

- Đầu tiên, bạn nhập số lượng phần tử trong danh sách. Dòng input() sẽ lấy dữ liệu từ người dùng và int() chuyển dữ liệu đó thành kiểu số nguyên.
- n là biến lưu số lượng phần tử mà người dùng muốn nhập vào.

- Bạn khai báo một danh sách rỗng `lst = []`.
- Sau đó, chương trình sẽ yêu cầu người dùng nhập vào `n` phần tử. Lệnh `for i in range(n)` sẽ lặp lại `n` lần để người dùng nhập đủ số lượng phần tử.
- Mỗi lần nhập, phần tử sẽ được chuyển thành số nguyên bằng `int()` và thêm vào danh sách `lst` bằng phương thức `append()`.
- Sau khi người dùng nhập xong các phần tử, bạn sắp xếp danh sách `lst` theo thứ tự tăng dần bằng phương thức `sort()`. Sau đó in ra danh sách đã được sắp xếp để người dùng xem.
- Tiếp theo, chương trình yêu cầu người dùng nhập giá trị cần tìm trong danh sách. Người dùng sẽ nhập giá trị này và nó được chuyển thành số nguyên nhờ `int()`.
- Tại đây, bạn gọi hàm `binary_search(lst, value)` để thực hiện tìm kiếm nhị phân. Hàm `binary_search` sẽ tìm giá trị `value` trong danh sách `lst` đã được sắp xếp.
- Kết quả trả về sẽ là chỉ số của phần tử trong danh sách nếu tìm thấy, hoặc `-1` nếu không tìm thấy.

1.3.10. Xây dựng hàm “*bubbleSort (nlist)*” (giải thuật sắp xếp nổi bọt) dưới dạng module. Viết chương trình nhập một *nlist* `n` phần tử từ bàn phím và sắp xếp.

Sample Data: [14,46,43,27,57,41,45,21,70]

Expected Result: [14, 21, 27, 41, 43, 45, 46, 57, 70]

1.3.11. Viết chương trình sử dụng thư viện NumPy để tạo một mảng có cấu trúc từ tên sinh viên, chiều cao, lớp và các kiểu dữ liệu của họ. Bây giờ sắp xếp theo lớp, sau đó chiều cao nếu lớp bằng nhau.

```
import numpy as np

# Tạo mảng có cấu trúc với các trường tên, chiều cao và lớp
data_type = np.dtype([('name', 'U50'), ('height', 'f4'), ('class', 'U20')])

# Tạo mảng dữ liệu
students = np.array([('Đào Quang Huy', 180, '12A'),
                    ('Ngô Mạnh Nguyên', 160, '11B'),
                    ('Nguyễn Hữu Kiều', 165, '12A'),
                    ('Nguyễn Thái Huy', 175, '11B'),
                    ('Lê Văn Hoàng', 168, '12B')],
                    dtype=data_type)

# Sắp xếp theo lớp trước, sau đó theo chiều cao nếu lớp giống nhau
sorted_students = np.sort(students, order=['class', 'height'])

# In kết quả
for student in sorted_students:
    print(f"Tên: {student['name']}, Chiều cao: {student['height']} cm, Lớp: {student['class']}")
```

Hình 11.1. Sử dụng thư viện numpy để chạy chương trình sắp xếp.

Chú thích:

- Dòng đầu dùng để gọi thư viện **numpy** một thư viện phổ biến trong Python dùng để làm việc với các mảng và ma trận, đặc biệt là các phép toán số học hiệu quả.
- Đây là cách khai báo một kiểu dữ liệu cấu trúc cho mảng. Nó cho phép mỗi phần tử trong mảng chứa nhiều trường dữ liệu khác nhau (tương tự như một bảng dữ liệu).
 - + 'name': là tên của sinh viên, được lưu dưới dạng chuỗi ký tự ('U50' có nghĩa là chuỗi ký tự tối đa 50 ký tự).
 - + 'height': chiều cao của sinh viên, lưu dưới dạng số thực 4 byte ('f4').
 - + 'class': lớp của sinh viên, lưu dưới dạng chuỗi ký tự ('U20' có nghĩa là chuỗi ký tự tối đa 20 ký tự).
- tạo một mảng dữ liệu tên là `students` chứa các thông tin của 5 sinh viên, bao gồm tên, chiều cao và lớp học.

- Sử dụng hàm `np.sort()` để sắp xếp mảng `students`.
- Tham số `order` chỉ định thứ tự ưu tiên khi sắp xếp: VD: `order=['class', 'height']`
- Sau khi sắp xếp xong, kết quả sẽ được lưu vào biến `sorted_students`
- Dòng `for student in sorted_students`: sẽ lặp qua tất cả các sinh viên trong mảng đã sắp xếp.
- `student['name']`, `student['height']`, và `student['class']` dùng để truy xuất thông tin của từng sinh viên từ mảng có cấu trúc.

```
Tên: Ngô Mạnh Nguyên, Chiều cao: 160.0 cm, Lớp: 11B
Tên: Nguyễn Thái Huy, Chiều cao: 175.0 cm, Lớp: 11B
Tên: Nguyễn Hữu Kiều, Chiều cao: 165.0 cm, Lớp: 12A
Tên: Đào Quang Huy, Chiều cao: 180.0 cm, Lớp: 12A
Tên: Lê Văn Hoàng, Chiều cao: 168.0 cm, Lớp: 12B
```

Hình 11.2. Kết quả sau khi chạy chương trình trên.

1.3.12. Viết chương trình sử dụng thư viện NumPy để sắp xếp id sinh viên với chiều cao tăng dần của sinh viên từ id sinh viên và chiều cao đã cho. In các chỉ số nguyên mô tả thứ tự sắp xếp theo nhiều cột và dữ liệu được sắp xếp (sử dụng hàm `lexsort()`).

<pre>import numpy as np # Tạo mảng ID sinh viên và chiều cao student_ids = np.array([101, 102, 103, 104, 105]) heights = np.array([170, 160, 165, 175, 168]) # Sử dụng lexsort để sắp xếp theo chiều cao (chỉ số tăng dần) sorted_indices = np.lexsort((heights,)) # In các chỉ số sắp xếp print("Chỉ số sắp xếp (theo chiều cao):", sorted_indices) # Dữ liệu sau khi sắp xếp theo chiều cao sorted_student_ids = student_ids[sorted_indices] sorted_heights = heights[sorted_indices] # In kết quả sắp xếp print("Dữ liệu sau khi sắp xếp:") for id, height in zip(sorted_student_ids, sorted_heights): print(f"ID: {id}, Chiều cao: {height} cm")</pre>	<pre>>>> Chỉ số sắp xếp (theo chiều cao): [1 2 4 0 3] Dữ liệu sau khi sắp xếp: ID: 102, Chiều cao: 160 cm ID: 103, Chiều cao: 165 cm ID: 105, Chiều cao: 168 cm ID: 101, Chiều cao: 170 cm ID: 104, Chiều cao: 175 cm</pre>
---	--

Hình 12. Chương trình sử dụng thư viện Numpy để sắp xếp theo thứ tự tăng dần.

Chú thích:

- `student_ids`: chứa các ID của sinh viên (ví dụ: 101, 102, 103, ...).
- `heights`: chứa chiều cao của sinh viên tương ứng (ví dụ: 170 cm, 160 cm, 165 cm, ...).
- `np.lexsort()` là một hàm đặc biệt trong NumPy giúp sắp xếp các phần tử theo nhiều tiêu chí.
- Chúng ta chỉ cần sắp xếp theo chiều cao, vì vậy ta truyền `heights` vào hàm `lexsort()`.
- Hàm này sẽ trả về một danh sách chỉ số (`sorted_indices`) cho biết thứ tự của các sinh viên sau khi sắp xếp chiều cao từ thấp đến cao.
- Nếu chiều cao là `[170, 160, 165, 175, 168]`, `lexsort()` sẽ trả về một mảng chỉ số sắp xếp, ví dụ: `[1, 2, 4, 0, 3]`.
- Điều này có nghĩa là sinh viên có chiều cao 160 cm (ID: 102) sẽ đứng đầu, tiếp theo là 165 cm (ID: 103), 168 cm (ID: 105), và cuối cùng là 175 cm (ID: 104).
- Cuối cùng, chúng ta sử dụng vòng lặp `for` để in ra ID và chiều cao của các sinh viên sau khi đã sắp xếp.

BÀI 6. LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

1.1. Mục đích.

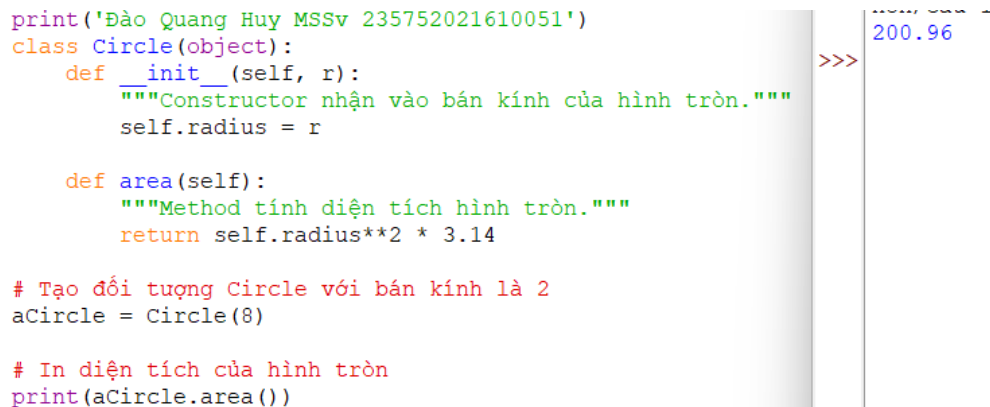
Giúp sinh viên nắm bắt về lập trình hướng đối tượng trong lập trình python.

1.2. Cơ sở lý thuyết.

Xem các quy tắc xây dựng class, các phương thức và thuộc tính của đối tượng.

1.3. Các bước tiến hành.

1.3.1. Định nghĩa một class có tên là Circle có thể được xây dựng từ bán kính. Circle có một method có thể tính diện tích. Gợi ý: Sử dụng def methodName(self) để định nghĩa method.



```
print('Đào Quang Huy MSSv 235752021610051')
class Circle(object):
    def __init__(self, r):
        """Constructor nhận vào bán kính của hình tròn."""
        self.radius = r

    def area(self):
        """Method tính diện tích hình tròn."""
        return self.radius**2 * 3.14

# Tạo đối tượng Circle với bán kính là 2
aCircle = Circle(2)

# In diện tích của hình tròn
print(aCircle.area())
```

Hình 1. Định nghĩa hàm để xây dựng chương trình tính diện tích.

Chú thích:

- class Circle(object):: Dòng này khai báo một lớp mới tên là Circle. Lớp này kế thừa từ lớp object (lớp gốc của mọi đối tượng trong Python).
- __init__: Đây là phương thức đặc biệt (constructor) được gọi tự động khi bạn tạo một đối tượng mới của lớp Circle.
- self: Tham chiếu đến chính đối tượng đang được tạo.
- r: Tham số truyền vào, đại diện cho bán kính của hình tròn.
- self.radius = r: Gán giá trị của tham số r cho thuộc tính radius của đối tượng. Thuộc tính này sẽ lưu trữ bán kính của hình tròn.
- area: Đây là một phương thức của lớp Circle dùng để tính diện tích hình tròn.
- return self.radius**2 * 3.14: Tính diện tích bằng công thức $S = \pi r^2$ và trả về kết quả.
- aCircle = Circle(2): Tạo một đối tượng mới của lớp Circle với tên là aCircle và truyền giá trị 2 cho tham số r của constructor. Điều này có nghĩa là ta đang tạo một hình tròn có bán kính bằng 2.
- print(aCircle.area()): Gọi phương thức area() của đối tượng aCircle để tính diện tích và in kết quả ra màn hình.

1.3.2. Định nghĩa class có tên là *Hinhchunhat* được xây dựng bằng chiều dài và chiều rộng. Class *Hinhchunhat* có method để tính diện tích.

```
class Hinhchunhat:
    def __init__(self, chieudai, chieurong):
        """Khởi tạo một hình chữ nhật mới.

        Args:
            chieudai: Chiều dài của hình chữ nhật.
            chieurong: Chiều rộng của hình chữ nhật.
        """
        self.chieudai = chieudai
        self.chieurong = chieurong

    def tinh_dien_tich(self):
        """Tính diện tích hình chữ nhật.

        Returns:
            Diện tích của hình chữ nhật.
        """
        return self.chieudai * self.chieurong
# Tạo một hình chữ nhật có chiều dài 5 và chiều rộng 3
hinh_chu_nhat = Hinhchunhat(5, 3)

# Tính diện tích và in kết quả
dien_tich = hinh_chu_nhat.tinh_dien_tich()
print("Diện tích hình chữ nhật là:", dien_tich)
```

```
>>> |
Đầu 2.py
Diện tích hình chữ nhật là: 15
```

Hình 2. Định nghĩa hàm có tên *Hinhchunhat* để tính diện tích.

Chú thích :

- `class Hinhchunhat:`: Khai báo một lớp mới tên là *Hinhchunhat*.
- `__init__(self, chieudai, chieurong)`: Đây là phương thức khởi tạo (constructor). Khi tạo một đối tượng hình chữ nhật, phương thức này sẽ được gọi để gán giá trị cho chiều dài và chiều rộng.
- `self.chieudai = chieudai`: Gán giá trị của tham số `chieudai` cho thuộc tính `chieudai` của đối tượng.
- `self.chieurong = chieurong`: Gán giá trị của tham số `chieurong` cho thuộc tính `chieurong` của đối tượng.
- `tinh_dien_tich(self)`: Phương thức này tính diện tích hình chữ nhật bằng cách nhân chiều dài với chiều rộng.
- `hinh_chu_nhat = Hinhchunhat(5, 3)`: Tạo một đối tượng `hinh_chu_nhat` của lớp *Hinhchunhat* với chiều dài là 5 và chiều rộng là 3.
- `dien_tich = hinh_chu_nhat.tinh_dien_tich()`: Gọi phương thức `tinh_dien_tich()` của đối tượng `hinh_chu_nhat` để tính diện tích và gán kết quả vào biến `dien_tich`.
- `print("Diện tích hình chữ nhật là:", dien_tich)`: In ra kết quả diện tích.

1.3.3. Định nghĩa class *Nguoi* và 2 class con của nó: *Nam*, *Nu*. Tất cả các class có method "*getGender*" có thể in "*Nam*" cho class *Nam* và "*Nữ*" cho class *Nu*.

```
print('Đào Quang Huy MSSV 235752021610051')
# Class cơ sở
class Nguoi(object):
    def getGender(self):
        return "Unknown"

# Class Nam kế thừa từ Nguoi
class Nam(Nguoi):
    def getGender(self):
        return "Nam"

# Class Nu kế thừa từ Nguoi
class Nu(Nguoi):
    def getGender(self):
        return "Nữ"

# Tạo đối tượng từ các lớp con
aNam = Nam()
aNu = Nu()

# In kết quả từ phương thức getGender của các đối tượng
print(aNam.getGender()) # In ra "Nam"
print(aNu.getGender()) # In ra "Nữ"
```

```
>>> |
Đào Quang Huy MSSV 235752021610051
Nam
Nữ
```

Hình 3. Định nghĩa hàm, tổng quan về các class và kế thừa trong Python:

Chú thích:

- Class **Nguoi**: Đây là lớp cơ sở, có phương thức `getGender()` trả về "Unknown".
+ `getGender(self)`: Đây là phương thức của class **Nguoi**. Phương thức này chỉ trả về giá trị mặc định là "Unknown", vì class này không xác định rõ giới tính.
+ `self`: Trong Python, `self` là tham chiếu đến đối tượng hiện tại (instance) của class. Nó được dùng để truy cập các thuộc tính và phương thức của đối tượng.
- Class **Nam**: Đây là một class con, kế thừa từ class **Nguoi**. Class này ghi đè phương thức `getGender()` của class **Nguoi** để trả về "Nam".
- Khi bạn gọi phương thức `getGender()` từ đối tượng của class **Nam**, nó sẽ trả về "Nam" thay vì "Unknown" như trong class **Nguoi**.
- Class **Nu**: Đây là một class con khác, cũng kế thừa từ class **Nguoi**. Tương tự class **Nam**, class **Nu** cũng ghi đè phương thức `getGender()` để trả về "Nữ".
- Khi gọi `aNam.getGender()`, Python sẽ tìm phương thức `getGender()` trong class **Nam**. Vì class **Nam** đã ghi đè phương thức này, kết quả trả về sẽ là "Nam".
- Khi gọi `aNu.getGender()`, Python sẽ tìm phương thức `getGender()` trong class **Nu**. Vì class **Nu** đã ghi đè phương thức này, kết quả trả về sẽ là "Nữ".

1.3.4. Viết chương trình Python dưới dạng class để chuyển đổi một số La Mã thành một số nguyên.

```
print('Đào Quang Huy MSSV 235752021610051')
class RomanToInteger:
    def __init__(self):
        # Mappings of Roman numerals to integer values
        self.roman_map = {
            'I': 1, 'V': 5, 'X': 10, 'L': 50,
            'C': 100, 'D': 500, 'M': 1000
        }

    def convert(self, roman: str) -> int:
        # Kết quả ban đầu là 0
        result = 0
        # Duyệt qua các ký tự trong chuỗi La Mã
        for i in range(len(roman)):
            # Lấy giá trị của ký tự hiện tại
            current_value = self.roman_map[roman[i]]

            # Kiểm tra nếu ký tự hiện tại có giá trị nhỏ hơn ký tự tiếp theo
            # thì trừ đi giá trị của ký tự hiện tại, nếu không cộng vào
            if i + 1 < len(roman) and current_value < self.roman_map[roman[i + 1]]:
                result -= current_value
            else:
                result += current_value

        return result

# Test chương trình
if __name__ == "__main__":
    roman_converter = RomanToInteger()

    # Thử nghiệm với một số La Mã
    roman_numeral = "XVI"
    print(f"Số La Mã {roman_numeral} chuyển thành số nguyên là: {roman_converter.convert(roman_numeral)}")

    roman_numeral = "MMV"
    print(f"Số La Mã {roman_numeral} chuyển thành số nguyên là: {roman_converter.convert(roman_numeral)}")
```

Hình 4. Chương trình chuyển đổi số la mã thành 1 số nguyên.

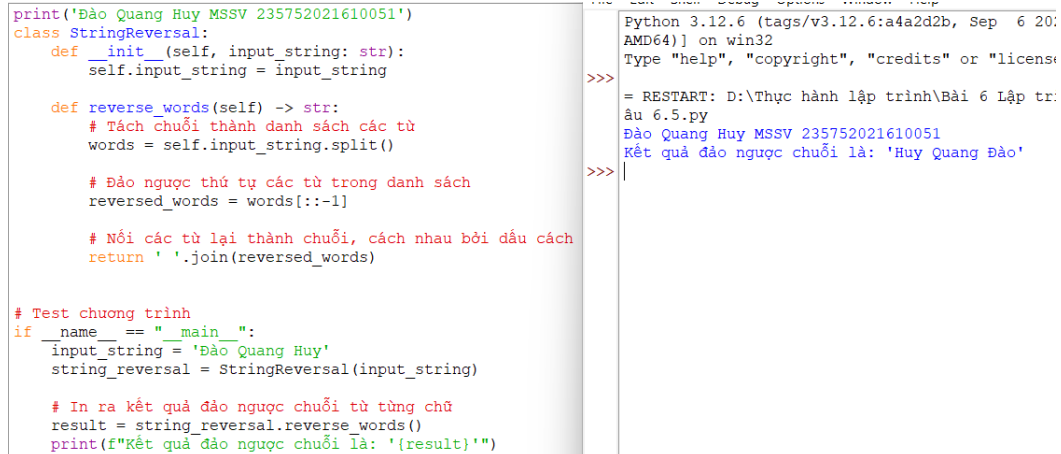
Chú thích:

- Đầu tiên, ta tạo ra một lớp **RomanToInteger**. Lớp này sẽ chứa các phương thức để chuyển đổi số La Mã thành số nguyên.
- Trong phương thức khởi tạo `__init__`, ta tạo một từ điển `roman_map`, nó lưu các ký tự La Mã và giá trị tương ứng của chúng.
- `I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000`.
- Phương thức `convert`: Phương thức này chịu trách nhiệm chính trong việc chuyển đổi một số La Mã (dạng chuỗi) thành số nguyên.
- `result = 0`: Đây là biến lưu kết quả cuối cùng. Ban đầu, chúng ta cho nó bằng 0.
- `for i in range(len(roman))`: Duyệt qua từng ký tự trong chuỗi `roman`.
- `roman[i]` là ký tự thứ `i` trong chuỗi.

- Ta lấy giá trị của ký tự tại vị trí i thông qua từ điển `roman_map`, ví dụ: `current_value = self.roman_map[roman[i]]`.
- Nếu ký tự hiện tại nhỏ hơn ký tự tiếp theo, ta trừ giá trị của ký tự hiện tại đi.
- Nếu ký tự hiện tại lớn hơn hoặc bằng ký tự tiếp theo, ta cộng giá trị của ký tự hiện tại vào kết quả.

1.3.5. Viết chương trình Python dưới dạng class để đảo ngược chuỗi từ từng chữ.

Dữ liệu vào : 'Đào Quang Huy', Đầu ra : 'Huy Quang Đào '



```
print('Đào Quang Huy MSSV 235752021610051')
class StringReversal:
    def __init__(self, input_string: str):
        self.input_string = input_string

    def reverse_words(self) -> str:
        # Tách chuỗi thành danh sách các từ
        words = self.input_string.split()

        # Đảo ngược thứ tự các từ trong danh sách
        reversed_words = words[::-1]

        # Nối các từ lại thành chuỗi, cách nhau bởi dấu cách
        return ' '.join(reversed_words)

# Test chương trình
if __name__ == "__main__":
    input_string = 'Đào Quang Huy'
    string_reversal = StringReversal(input_string)

    # In ra kết quả đảo ngược chuỗi từ từng chữ
    result = string_reversal.reverse_words()
    print(f"Kết quả đảo ngược chuỗi là: '{result}'")
```

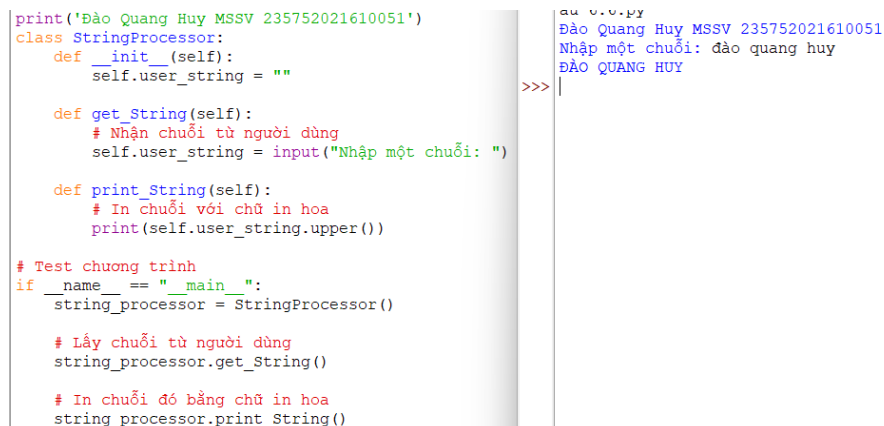
Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep 6 20: AMD64) on win32
Type "help", "copyright", "credits" or "license()"
>>> = RESTART: D:\Thực hành lập trình\Bài 6 Lập trình 6.5.py
Đào Quang Huy MSSV 235752021610051
Kết quả đảo ngược chuỗi là: 'Huy Quang Đào'
>>> |

Hình 5. Chương trình đảo ngược chuỗi từng chữ từ dữ liệu nhập vào.

Chú thích :

- Lớp `StringReversal`: Lớp này có phương thức khởi tạo `__init__` nhận vào chuỗi đầu vào mà bạn muốn đảo ngược thứ tự các từ.
- Phương thức `reverse_words`:
+ Sử dụng `split()` để tách chuỗi thành danh sách các từ.
+ Dùng cú pháp `[::-1]` để đảo ngược thứ tự các từ trong danh sách.
+ Cuối cùng, sử dụng `' '.join(reversed_words)` để nối các từ lại thành chuỗi với dấu cách giữa chúng.
- Với chuỗi 'Đào Quang Huy', khi gọi phương thức `reverse_words()`, kết quả sẽ là 'Huy Quang Đào'.

1.3.6. Viết một class Python có hai phương thức `get_String` và `print_String`. `get_String` chấp nhận một chuỗi từ người dùng và `print_String` in chuỗi đó bằng chữ in hoa.



```
print('Đào Quang Huy MSSV 235752021610051')
class StringProcessor:
    def __init__(self):
        self.user_string = ""

    def get_String(self):
        # Nhận chuỗi từ người dùng
        self.user_string = input("Nhập một chuỗi: ")

    def print_String(self):
        # In chuỗi với chữ in hoa
        print(self.user_string.upper())

# Test chương trình
if __name__ == "__main__":
    string_processor = StringProcessor()

    # Lấy chuỗi từ người dùng
    string_processor.get_String()

    # In chuỗi đó bằng chữ in hoa
    string_processor.print_String()
```

Đào Quang Huy MSSV 235752021610051
Nhập một chuỗi: đào quang huy
ĐÀO QUANG HUY
>>> |

Hình 6. Chương trình in kết quả là chữ in hoa từ dữ liệu nhập vào.

Chú thích :

- Chúng ta tạo một lớp có tên là StringProcessor, lớp này sẽ chứa hai phương thức (method) để thực hiện công việc:
 - + Phương thức get_String: Nhận chuỗi từ người dùng.
 - + Phương thức print_String: In chuỗi đó bằng chữ in hoa.
- Phương thức __init__ (Khởi tạo lớp):
 - + Phương thức này được gọi tự động khi tạo một đối tượng từ lớp StringProcessor.
 - + Nó có nhiệm vụ khởi tạo một thuộc tính **user_string** (chuỗi của người dùng) với giá trị mặc định là một chuỗi rỗng "".
- Phương thức get_String:
 - + Phương thức này yêu cầu người dùng nhập vào một chuỗi thông qua hàm input().
 - + Chuỗi nhập vào sẽ được lưu vào thuộc tính user_string của đối tượng.
- Phương thức print_String:
 - + Phương thức này sẽ in ra giá trị của thuộc tính user_string bằng chữ **in hoa**. Để chuyển chuỗi thành chữ in hoa, chúng ta dùng phương thức .upper().
- Khi bạn tạo một đối tượng từ lớp StringProcessor, ví dụ: string_processor = StringProcessor(), phương thức __init__() sẽ tự động được gọi. Lúc này, chuỗi user_string được khởi tạo với giá trị rỗng "".
- Bạn gọi phương thức get_String() để yêu cầu người dùng nhập một chuỗi. Ví dụ: nếu người dùng nhập "hello world", chuỗi đó sẽ được lưu vào user_string.
- Sau đó, bạn gọi phương thức print_String(). Phương thức này sẽ in giá trị của user_string, nhưng bằng chữ **in hoa**. Ví dụ, nếu user_string = "hello world", kết quả in ra sẽ là "HELLO WORLD".

1.3.7. Viết một class Python có tên Circle được xây dựng theo bán kính và hai phương thức sẽ tính diện tích và chu vi của hình tròn.

```
print('Đào Quang Huy MSSV 235752021610051')
import math

class Circle:
    def __init__(self, radius):
        # Khởi tạo bán kính của hình tròn
        self.radius = radius

    def area(self):
        # Tính diện tích của hình tròn: A = π * r^2
        return math.pi * self.radius ** 2

    def perimeter(self):
        # Tính chu vi của hình tròn: P = 2 * π * r
        return 2 * math.pi * self.radius

# Test chương trình
if __name__ == "__main__":
    # Nhập bán kính từ người dùng
    radius = float(input("Nhập bán kính của hình tròn: "))

    # Tạo đối tượng hình tròn
    circle = Circle(radius)

    # In diện tích và chu vi của hình tròn
    print(f"Diện tích của hình tròn là: {circle.area():.2f}")
    print(f"Chu vi của hình tròn là: {circle.perimeter():.2f}")
```

Đào Quang Huy MSSV 235752021610051
Nhập bán kính của hình tròn: 16
Diện tích của hình tròn là: 804.25
Chu vi của hình tròn là: 100.53

Hình 7. Chương trình tính diện tích và chu vi hình tròn.

Chú thích :

- Lớp Circle được xây dựng với một thuộc tính chính là **bán kính** của hình tròn (radius). Khi bạn tạo một đối tượng của lớp này, bạn phải cung cấp bán kính của hình tròn. Ví dụ: Nếu bán kính là 5, đối tượng sẽ có bán kính là 5.
- Phương thức area (diện tích): Diện tích của hình tròn được tính bằng công thức: $A = \pi \times r^2$ Trong đó:
 - + π (pi) là hằng số (khoảng 3.1416).
 - + r là bán kính của hình tròn.
 - + Phương thức area() sẽ dùng công thức này để tính diện tích.

- Phương thức perimeter (chu vi): Chu vi của hình tròn được tính bằng công thức:
 $P = 2 \times \pi \times r$ Trong đó:
 + π (pi) là hằng số (khoảng 3.1416).
 + r là bán kính của hình tròn.
 + Phương thức perimeter() sẽ dùng công thức này để tính chu vi.

1.3.8. Chương trình ATM đơn giản.

```
print('Đào Quang Huy MSSV 235752021610051')
# Tạo số dư ban đầu
balance = 0

def display_menu():
    """Hiển thị menu các tùy chọn."""
    print("\n===== ATM Menu =====")
    print("1. Xem số dư")
    print("2. Rút tiền")
    print("3. Nạp tiền")
    print("4. Thoát")

def check_balance():
    """Kiểm tra và in số dư hiện tại."""
    print(f"Số dư hiện tại của bạn là: {balance} VND")

def withdraw_money():
    """Hàm để rút tiền."""
    global balance
    amount = float(input("Nhập số tiền muốn rút: "))
    if amount > balance:
        print("Số dư không đủ để thực hiện giao dịch này.")
    elif amount <= 0:
        print("Số tiền rút phải lớn hơn 0.")
    else:
        balance -= amount
        print(f"Bạn đã rút {amount} VND. Số dư mới là: {balance} VND")

def deposit_money():
    """Hàm để nạp tiền."""
    global balance
    amount = float(input("Nhập số tiền muốn nạp: "))
    if amount <= 0:
        print("Số tiền nạp phải lớn hơn 0.")
    else:
        balance += amount
        print(f"Bạn đã nạp {amount} VND. Số dư mới là: {balance} VND")

def main():
    """Chạy chương trình ATM."""
```

Hình 8.1. Chương trình ATM (Khúc đầu).

```
def main():
    """Chạy chương trình ATM."""
    while True:
        display_menu()
        choice = input("Chọn một tùy chọn (1-4): ")

        if choice == "1":
            check_balance()
        elif choice == "2":
            withdraw_money()
        elif choice == "3":
            deposit_money()
        elif choice == "4":
            print("Cảm ơn bạn đã sử dụng dịch vụ. Hẹn gặp lại!")
            break
        else:
            print("Lựa chọn không hợp lệ. Vui lòng chọn lại.")

# Chạy chương trình
main()
```

Hình 8.2. Chương trình ATM (Khúc sau).

Chú thích :

- Lớp ATM: Lớp này có một thuộc tính chính là số dư tài khoản (balance), được khởi tạo khi tạo đối tượng của lớp.
- check_balance(): Kiểm tra và in ra số dư tài khoản.
- deposit(amount): Nhận vào một số tiền và gửi vào tài khoản, nếu số tiền hợp lệ.
- withdraw(amount): Nhận vào một số tiền và rút từ tài khoản, nếu số tiền hợp lệ và đủ trong tài khoản.
- start(): Đây là phương thức chính, tạo ra một giao diện đơn giản với các lựa chọn như kiểm tra số dư, gửi tiền, rút tiền, và thoát chương trình.

Bài 7: Thao tác trên tập tin và thư mục trong Python

1.1. Mục đích

Giúp sinh viên có thể thao tác với các file văn bản sử dụng python

1.2. Cơ sở lý thuyết

Xem các quy tắc mở, đóng file văn bản, đọc và ghi nội dung của các file.

1.3. Các bước thực hiện

1.3.1. Chương trình đọc file và in đảo ngược kết quả.

```
print('Đào Quang Huy MSSV 235752021610051')
# Mở file với mã hóa utf-8
with open('D:/Thực hành lập trình/Bài 7 Thao tác trên tập tin và thư mục trong Python/Câu 7.1.py', 'r', encoding='utf-8') as input_file:
    for line in input_file:
        reversed_line = line[::-1]
        print(reversed_line)
```

Hình 1.1. Chương trình đọc file và đảo ngược file.

Chú thích :

- with open('D:/Thực hành lập trình/Bài 7 Thao tác trên tập tin và thư mục trong Python/Câu 7.1.py', 'r', encoding='utf-8') as input_file::
+ Mở file có đường dẫn 'D:/Thực hành lập trình/Bài 7 Thao tác trên tập tin và thư mục trong Python/Câu 7.1.py' ở chế độ đọc ('r').
+ encoding='utf-8': Chỉ định mã hóa UTF-8 để đảm bảo đọc đúng các ký tự tiếng Việt và các ký tự đặc biệt khác.
+ Sử dụng with để tự động đóng file sau khi kết thúc khối lệnh.
- for line in input_file: Vòng lặp này sẽ duyệt qua từng dòng trong file.
- reversed_line = line[::-1]: Đây là cách ngắn gọn và hiệu quả nhất để đảo ngược một chuỗi trong Python. Slicing [::-1] sẽ lấy tất cả các ký tự trong chuỗi và sắp xếp chúng theo thứ tự ngược lại.
- print(reversed_line): In ra dòng đã được đảo ngược.

Chú ý:

- :: Chỉ định lấy tất cả các phần tử trong chuỗi.
- -1: Chỉ định bước nhảy là -1, tức là lấy các phần tử từ cuối lên đầu.

```
Đào Quang Huy MSSV 235752021610051
)'150016120257532 VSSM yuH gnauQ oàĐ'(tnirp
8-ftu aóh ãm ióv elif òM #
:elif_tupni sa )'8-ftu'=gnidocne , 'r' , 'yp.1.7 uâC/nohtyP gnort cùm uht àv nit p
ật nêrt cát oahT 7 iàB/hnìrt pậl hnàh cựhT/:D'(nepo htiw
:elif_tupni ni enil rof
]1-::[enil = enil_desrever
)enil_desrever(tnirp
```

Hình 1.2. Kết quả sau khi chạy chương trình đảo ngược file.

1.3.2. Chương trình đọc một file, tính số ký tự, số từ và số dòng của file.

```
print('Đào Quang Huy MSSV 235752021610051')
# Mở tệp với chế độ đọc
with open('D:/Thực hành lập trình/Bài 7 Thao tác trên tập tin và thư mục trong Python/Câu 7.2.py', 'r', encoding='utf-8') as file:
    # Khởi tạo các biến để đếm số dòng, số từ và số ký tự
    line_count = 0
    word_count = 0
    char_count = 0

    # Đọc từng dòng trong tệp
    for line in file:
        line_count += 1 # Tăng số dòng lên 1 cho mỗi dòng
        word_count += len(line.split()) # Đếm số từ trong dòng (split() chia theo khoảng trắng)
        char_count += len(line) # Đếm số ký tự trong dòng

    # In kết quả
    print(f"Số dòng trong tệp: {line_count}")
    print(f"Số từ trong tệp: {word_count}")
    print(f"Số ký tự trong tệp: {char_count}")
```

Đào Quang Huy MSSV 235752021610051
Số dòng trong tệp: 18
Số từ trong tệp: 123
Số ký tự trong tệp: 741

Hình 2. Chương trình đọc 1 file đếm số ký tự số từ và số dòng.

Chú thích :

- with open('D:/Thực hành lập trình/Bài 7 Thao tác trên tập tin và thư mục trong Python/Câu 7.2.py', 'r', encoding='utf-8') as file:
+ Mở tệp tại đường dẫn chỉ định ở chế độ đọc ('r').
+ Sử dụng encoding='utf-8' để đảm bảo đọc đúng các ký tự tiếng Việt và các ký tự đặc biệt khác.
+ with tự động đóng tệp sau khi thực hiện xong khối lệnh.
- line_count = 0, word_count = 0, char_count = 0: Khởi tạo các biến để đếm số dòng, số từ và số ký tự, ban đầu đều bằng 0.
- for line in file:: Vòng lặp này sẽ duyệt qua từng dòng trong tệp.
- line_count += 1: Tăng biến line_count lên 1 mỗi khi gặp một dòng mới.
- word_count += len(line.split()): Chia dòng thành các từ bằng split() và đếm số từ.
- char_count += len(line): Tính độ dài của dòng để đếm số ký tự.

1.3.3. Viết chương trình Python để đọc toàn bộ tệp văn bản

```
print('Đào Quang Huy MSSV 235752021610051')
# Mở tệp và đọc toàn bộ nội dung
with open('D:/Thực hành lập trình/Bài 7 Thao tác trên tập tin và thư mục trong Python/Câu 7.1.py', 'r', encoding='utf-8') as file:
    # Đọc toàn bộ nội dung của tệp
    content = file.read()

    # In ra nội dung của tệp
    print(content)
```

Đào Quang Huy MSSV 235752021610051
print('Đào Quang Huy MSSV 235752021610051')
Mở file với mã hóa utf-8
with open('D:/Thực hành lập trình/Bài 7 Thao tác trên tập tin và thư mục trong Python/Câu 7.1.py', 'r', encoding='utf-8') as input_file:
 :
 for line in input_file:
 reversed_line = line[::-1]
 print(reversed_line)

Hình 3. Chương trình đọc toàn bộ tệp file bản nhập từ người dùng .

Chú thích :

- with open('D:/Thực hành lập trình/Bài 7 Thao tác trên tập tin và thư mục trong Python/Câu 7.1.py', 'r', encoding='utf-8') as file:
+ Mở tệp tại đường dẫn chỉ định ở chế độ đọc ('r').
+ Sử dụng encoding='utf-8' để đảm bảo đọc đúng các ký tự tiếng Việt và các ký tự đặc biệt khác.
+ with tự động đóng tệp sau khi thực hiện xong khối lệnh.
- content = file.read(): Đọc toàn bộ nội dung của tệp và gán vào biến content

1.3.4. Chương trình Python để đọc n dòng đầu tiên của tệp.

```
print('Đào Quang Huy MSSV 235752021610051')
# Đọc n dòng đầu tiên của tệp
def read_first_n_lines(file_path, n):
    with open(file_path, 'r', encoding='utf-8') as file:
        # Đọc n dòng đầu tiên
        for i in range(n):
            line = file.readline()
            if line: # Kiểm tra nếu dòng không rỗng
                print(line, end='') # In ra dòng mà không có ký tự '\n' (vì readline() đã có '\n' ở cuối)
            else:
                break # Dừng lại nếu tệp không đủ dòng

# Gọi hàm với đường dẫn tệp và số dòng cần đọc
file_path = 'D:/Thực hành lập trình/Bài 7 Thao tác trên tập tin và thư mục trong Python/Câu 7.1.py'
n = 5 # Đọc 5 dòng đầu tiên
read_first_n_lines(file_path, n)
```

Ln: 8 Col: 5

```
# Mở file với mã hóa utf-8
with open('D:/Thực hành lập trình/Bài 7 Thao tác trên tập tin và thư mục trong Python/Câu 7.1.py', 'r', encoding='utf-8') as input_file:
    for line in input_file:
        reversed_line = line[::-1]
```

Hình 4. Chương trình đọc n dòng từ file yêu cầu từ người dùng.

Chú thích:

- Hàm `read_first_n_lines`:
 - + `file_path`: Đường dẫn đến tệp cần đọc.
 - + `n`: Số dòng cần đọc.
 - + `with open(file_path, 'r', encoding='utf-8') as file::` Mở tệp ở chế độ đọc, sử dụng mã hóa UTF-8.
 - + Vòng lặp `for`: Lặp n lần để đọc từng dòng.
 - + `line = file.readline()`: Đọc một dòng từ tệp.
 - + Kiểm tra dòng: Nếu `line` không rỗng (có dữ liệu), in ra dòng đó.
 - + Dừng vòng lặp: Nếu `line` rỗng (đã đến cuối tệp hoặc đã đọc đủ n dòng), dừng vòng lặp.
- Gọi hàm `read_first_n_lines` để thực hiện việc đọc và in.
- Sử dụng `end=""` để in ra dòng mà không có ký tự xuống dòng thừa.

1.3.5. Chương trình Python để nối văn bản vào tệp và hiển thị văn bản.

```
print('Đào Quang Huy MSSV 235752021610051')
def noi_van_ban_vao_tep(ten_tep, noi_dung_moi):
    # Nối văn bản vào tệp
    with open(ten_tep, 'a', encoding='utf-8') as file:
        file.write(noi_dung_moi + '\n') # Ghi văn bản và xuống dòng mới

    # Hiển thị nội dung của tệp sau khi nối
    with open(ten_tep, 'r', encoding='utf-8') as file:
        noi_dung = file.read() # Đọc toàn bộ nội dung tệp
        print("Nội dung tệp sau khi nối:")
        print(noi_dung)

# Đường dẫn đến tệp và văn bản cần nối
ten_tep = 'D:/Thực hành lập trình/Bài 7 Thao tác trên tập tin và thư mục trong Python/Câu 7.1.py'
noi_dung_moi = "Đào Quang Huy MSSV 235752021610051"

# Gọi hàm để nối văn bản và hiển thị nội dung tệp
noi_van_ban_vao_tep(ten_tep, noi_dung_moi)
```

Hình 5.1. Chương trình nối thêm văn bản vào tệp.

Chú thích :

- `with open(ten_tep, 'a', encoding='utf-8') as file::`
 - + `open(ten_tep, 'a', encoding='utf-8')`: Mở tệp ở chế độ 'a': append (nối thêm). Điều này có nghĩa là nội dung mới sẽ được thêm vào cuối tệp mà không ghi đè lên nội dung cũ.
 - + `encoding='utf-8'`: Chỉ định mã hóa UTF-8 để hỗ trợ các ký tự tiếng Việt và các ký tự đặc biệt khác.

- + with: Đây là một ngữ cảnh quản lý, đảm bảo tệp sẽ được đóng tự động ngay cả khi có lỗi xảy ra.
- file.write(noi_dung_moi + '\n'): Ghi nội dung mới vào tệp. Dấu '\n' được thêm vào cuối để tạo một dòng mới sau khi ghi.
- with open(ten_tep, 'r', encoding='utf-8') as file: Mở lại tệp ở chế độ đọc 'r' để đọc toàn bộ nội dung sau khi đã nối thêm.
- noi_dung = file.read(): Đọc toàn bộ nội dung của tệp và gán vào biến noi_dung.
- print("Nội dung tệp sau khi nối:"): In ra một dòng thông báo cho biết nội dung sắp được hiển thị.
- print(noi_dung): In ra toàn bộ nội dung của tệp đã được đọc.

```
Đào Quang Huy MSSV 235752021610051
Nội dung tệp sau khi nối:
print('Đào Quang Huy MSSV 235752021610051')
# Mở file với mã hóa utf-8
with open('D:/Thực hành lập trình/Bài 7 Thao tác trên tập tin và thư
mục trong Python/Câu 7.1.py', 'r', encoding='utf-8') as input_file:
    for line in input_file:
        reversed_line = line[::-1]
        print(reversed_line)
This is the new text added to the file.

Đào Quang Huy MSSV 235752021610051
```

Hình 5.2. Kết quả sau khi thêm nội dung vào chương trình trên.

1.3.6. Chương trình Python để đọc n dòng cuối cùng của tệp.

```
print('Đào Quang Huy MSSV 235752021610051')
import os
def doc_n_dong_cuoi(ten_tep, n):
    # Kiểm tra nếu tệp tồn tại
    if not os.path.exists(ten_tep):
        print(f"Tệp {ten_tep} không tồn tại!")
        return []

    with open(ten_tep, 'r', encoding='utf-8') as f:
        dong = f.readlines()

        # Nếu tệp có ít dòng hơn n, lấy toàn bộ tệp
        if len(dong) < n:
            print(f"Tệp chỉ có {len(dong)} dòng, lấy tất cả.")
            return dong # Trả về tất cả các dòng trong tệp
        return dong[-n:] # Lấy n dòng cuối

# Ví dụ sử dụng
ten_tep = "D:/Thực hành lập trình/Bài 7 Thao tác trên tập tin và thư mục trong Python/Câu 7.1.py"
n = 3
dong_cuoi = doc_n_dong_cuoi(ten_tep, n)
if dong_cuoi:
    for dong in dong_cuoi:
        print(dong, end='') # In các dòng mà không thêm ký tự xuống dòng
```

Hình 6. Chương trình đọc n dòng cuối của tệp.

Chú thích:

- Nhập module os để thực hiện các thao tác liên quan đến hệ điều hành, cụ thể ở đây là kiểm tra sự tồn tại của tệp.
- Định nghĩa hàm doc_n_dong_cuoi với hai tham số:
+ ten_tep: Tên của tệp cần đọc.
+ n: Số lượng dòng cần đọc từ cuối tệp.
- if not os.path.exists(ten_tep): Kiểm tra xem tệp có tồn tại hay không bằng hàm os.path.exists(). Nếu tệp không tồn tại, in ra thông báo lỗi và trả về một danh sách rỗng.
- with open(ten_tep, 'r', encoding='utf-8') as f::
+ Mở tệp ở chế độ đọc ('r') với mã hóa UTF-8 để hỗ trợ các ký tự đặc biệt.
+ Sử dụng with để đảm bảo tệp được đóng tự động sau khi thực hiện xong các thao tác.
- dong = f.readlines(): Đọc toàn bộ nội dung của tệp và lưu vào danh sách dong, mỗi phần tử trong danh sách là một dòng.

- if len(dong) < n: Kiểm tra xem số lượng dòng trong tệp có nhỏ hơn số lượng dòng cần đọc (n) hay không. Nếu đúng, trả về toàn bộ danh sách dòng (tức là trả về tất cả các dòng có trong tệp).
- return dong[-n:]: Lấy n phần tử cuối cùng của danh sách dòng và trả về. Đây chính là các dòng cuối cùng của tệp.

1.3.7. Viết chương trình Python để đếm số dòng trong tệp văn bản.

```
print('Đào Quang Huy MSSV 235752021610051')
def dem_so_dong(ten_tep):
    try:
        with open(ten_tep, 'r', encoding='utf-8') as file:
            # Đọc tất cả dòng trong tệp và đếm số dòng
            so_dong = sum(1 for line in file)
            return so_dong
    except FileNotFoundError:
        print(f"Tệp {ten_tep} không tồn tại!")
        return 0

# Ví dụ sử dụng
# Thay đổi địa chỉ file nếu cần
ten_tep = "D:/Thực hành lập trình/Bài 7 Thao tác trên tập tin và thư mục trong Python/Câu 7.1.py"
so_dong = dem_so_dong(ten_tep)
if so_dong > 0:
    print(f"Số dòng trong tệp {ten_tep} là: {so_dong}")
```

Hình 7. Chương trình đếm số dòng trong tệp.

Chú thích:

- Định nghĩa hàm dem_so_dong(ten_tep):
 - + Hàm này có một tham số ten_tep, dùng để truyền vào tên (hoặc đường dẫn) của tệp văn bản mà bạn muốn đếm số dòng.
 - + Hàm này sẽ trả về số dòng trong tệp hoặc 0 nếu có lỗi xảy ra (ví dụ tệp không tồn tại).
- Câu lệnh try-except:
 - + try: Mọi thao tác mở và xử lý tệp đều được thực hiện trong khối try để đảm bảo rằng nếu có lỗi xảy ra (chẳng hạn tệp không tồn tại), chương trình sẽ không bị lỗi mà sẽ xử lý lỗi một cách duyên dáng.
 - + except FileNotFoundError: Nếu tệp không tồn tại (lỗi FileNotFoundError), chương trình sẽ bắt lỗi và thông báo cho người dùng biết rằng tệp không tồn tại. Nếu không có lỗi, chương trình sẽ tiếp tục chạy bình thường.
- Mở tệp với with open(ten_tep, 'r', encoding='utf-8') as file::
 - + with là cách mở tệp trong Python, đảm bảo tệp sẽ được đóng tự động sau khi kết thúc khối with mà không cần phải gọi file.close().
 - + encoding='utf-8' chỉ định mã hóa tệp. Nếu tệp sử dụng mã hóa khác (ví dụ 'cp1252'), bạn có thể thay đổi mã hóa cho phù hợp.
- Đếm số dòng bằng sum(1 for line in file):
 - + file là đối tượng tệp, khi bạn lặp qua đối tượng này (vòng lặp for), mỗi lần lặp sẽ trả về một dòng trong tệp.
 - + sum(1 for line in file) là cách sử dụng một biểu thức generator để đếm số dòng.
 - + for line in file lặp qua mỗi dòng trong tệp.
 - + 1 for line có nghĩa là mỗi lần lặp qua một dòng, bạn cộng 1 vào tổng số dòng.
 - + sum(...) sẽ cộng dồn tất cả các giá trị 1 này lại để cho bạn tổng số dòng trong tệp.

1.3.8. Viết chương trình Python để viết nội dung danh sách vào tệp.

```
print('Đào Quang Huy MSSV 235752021610051')
def ghi_danh_sach_vao_tep(ten_tep, danh_sach):
    try:
        # Mở tệp ở chế độ ghi (write) với mã hóa utf-8
        with open(ten_tep, 'w', encoding='utf-8') as file:
            for item in danh_sach:
                # Ghi từng phần tử của danh sách vào tệp, mỗi phần tử trên một dòng
                file.write(str(item) + '\n')
            print(f"Đã ghi nội dung danh sách vào tệp {ten_tep}")
    except Exception as e:
        print(f"Đã xảy ra lỗi: {e}")

# Ví dụ sử dụng
danh_sach = ["Apple", "Banana", "Orange", "Grapes"]
ten_tep = "fruits.txt" # Đường dẫn của tệp cần ghi

ghi_danh_sach_vao_tep(ten_tep, danh_sach)
```

Hình 8. Thêm nội dung danh sách vào tệp.

Chú thích :

- **Hàm ghi_danh_sach_vao_tep(ten_tep, danh_sach):**
 - + ten_tep: Đường dẫn của tệp mà bạn muốn ghi dữ liệu vào (ví dụ: "fruits.txt").
 - + danh_sach: Đây là danh sách chứa các phần tử bạn muốn ghi vào tệp.
- open(ten_tep, 'w', encoding='utf-8'): Hàm open() mở tệp ở chế độ ghi ('w'):
 - + Nếu tệp không tồn tại, Python sẽ tạo tệp mới.
 - + Nếu tệp đã tồn tại, nội dung cũ sẽ bị ghi đè.
- encoding='utf-8': Đảm bảo rằng tệp sẽ được ghi với mã hóa UTF-8.
- for item in danh_sach: Lặp qua từng phần tử trong danh sách.
- file.write(str(item) + '\n'): Ghi từng phần tử vào tệp. Mỗi phần tử được chuyển thành chuỗi (str(item)) và sau đó thêm ký tự xuống dòng ('\n') để mỗi phần tử nằm trên một dòng riêng biệt trong tệp.
- Sử dụng try-except để xử lý các lỗi có thể xảy ra trong quá trình ghi tệp (ví dụ, tệp không thể mở, quyền truy cập bị từ chối, v.v.).
- Nếu có lỗi, thông báo lỗi sẽ được in ra.

1.3.9. Viết chương trình Python để sao chép nội dung của tệp này sang tệp khác.

```
print('Đào Quang Huy MSSV 235752021610051')
def sao_chep_tep(tep_nguon, tep_dich):
    try:
        # Mở tệp nguồn để đọc và tệp đích để ghi
        with open(tep_nguon, 'r', encoding='utf-8') as file_nguon:
            # Đọc toàn bộ nội dung tệp nguồn
            noi_dung = file_nguon.read()

        # Mở tệp đích để ghi nội dung
        with open(tep_dich, 'w', encoding='utf-8') as file_dich:
            # Ghi nội dung vào tệp đích
            file_dich.write(noi_dung)

        print(f"Đã sao chép nội dung từ tệp {tep_nguon} sang tệp {tep_dich}")
    except Exception as e:
        print(f"Đã xảy ra lỗi: {e}")

# Ví dụ sử dụng
tep_nguon = "source.txt" # Đường dẫn ( ghi rỏ ổ đĩa, thư mục tên file)
tep_dich = "destination.txt" # Đường dẫn ( ghi rỏ ổ đĩa, thư mục tên file)

sao_chep_tep(tep_nguon, tep_dich)
```

Hình 9. Chương trình sao chép nội dung từ tệp này sang tệp khác.

Chú thích :

`def sao_chep_tep(tep_nguon, tep_dich) ::` Hàm này nhận hai tham số:

- `tep_nguon`: Đường dẫn đến tệp nguồn cần sao chép.
- `tep_dich`: Đường dẫn đến tệp đích để lưu trữ nội dung sao chép.
- `with open(tep_nguon, 'r', encoding='utf-8') as file_nguon::` Mở tệp nguồn ở chế độ đọc ('r') với mã hóa UTF-8 để hỗ trợ nhiều loại ký tự.
- `with open(tep_dich, 'w', encoding='utf-8') as file_dich::` Mở tệp đích ở chế độ ghi ('w') với mã hóa UTF-8. Nếu tệp đích đã tồn tại, nội dung cũ sẽ bị ghi đè.
- `noi_dung = file_nguon.read()`: Đọc toàn bộ nội dung của tệp nguồn và lưu vào biến `noi_dung`.
- `file_dich.write(noi_dung)`: Ghi nội dung đã đọc vào tệp đích.
- `print(f'Đã sao chép nội dung từ tệp {tep_nguon} sang tệp {tep_dich}')`: In ra thông báo xác nhận khi quá trình sao chép thành công.
- `except Exception as e:` Bắt tất cả các ngoại lệ có thể xảy ra và in ra thông báo lỗi chi tiết.

1.3.10. Viết chương trình python để tìm những từ dài nhất trong văn bản.

```
print('Đào Quang Huy MSSV 235752021610051')
def tim_tu_dai_nhat(van_ban):
    # Tách văn bản thành các từ (sử dụng split để tách theo khoảng trắng)
    tu_list = van_ban.split()

    # Tìm độ dài của từ dài nhất
    max_length = max(len(tu) for tu in tu_list)

    # Lọc ra những từ có độ dài bằng với độ dài dài nhất
    tu_dai_nhat = [tu for tu in tu_list if len(tu) == max_length]

    return tu_dai_nhat

# Ví dụ sử dụng
van_ban = "Đào Quang Huy"
tu_dai_nhat = tim_tu_dai_nhat(van_ban)

print("Những từ dài nhất trong văn bản là:")
for tu in tu_dai_nhat:
    print(tu)
```

Hình 10. Chương trình tìm từ dài nhất trong văn bản.

Chú thích:

- Hàm `tim_tu_dai_nhat(van_ban)`:
 - + Nhận vào một chuỗi văn bản (`van_ban`).
 - + Tách văn bản thành các từ: Sử dụng phương thức `split()` để tách chuỗi văn bản thành danh sách các từ, cách nhau bởi các khoảng trắng.
 - + Tìm độ dài của từ dài nhất: Sử dụng hàm `max()` kết hợp với `len()` để tìm độ dài của từ dài nhất trong danh sách.
 - + Lọc ra những từ có độ dài bằng với độ dài dài nhất: Dùng một comprehension list để lọc ra những từ có độ dài bằng với độ dài dài nhất.

Bài 8. Lập trình giao diện trong Python

1.1. Mục đích

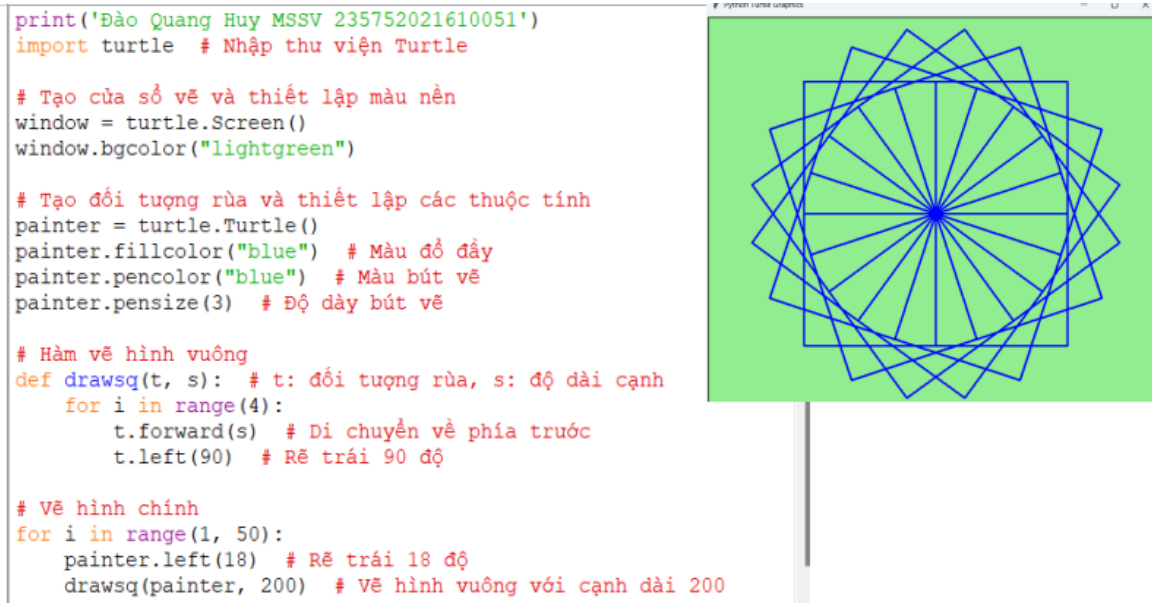
Giúp sinh viên làm quen với lập trình đồ họa và sử dụng thư viện turtle, Tkinter trong python.

1.2. Cơ sở lý thuyết

Xem các quy tắc xây dựng và quản lý layout, widget, hộp thoại, đồ họa trong python

1.3. Các bước tiến hành

1.3.1. Viết chương trình đồ họa sử dụng thư viện turtle, kiểm tra kết quả và giải thích chương trình.



Hình 1. Chương trình sử dụng thư viện turtle để vẽ hình vuông.

Chú thích :

- `import turtle`: Dòng lệnh này nhập thư viện Turtle, cung cấp các hàm và đối tượng cần thiết để vẽ đồ họa.
- `window = turtle.Screen()`: Tạo một cửa sổ vẽ và gán nó cho biến `window`.
- `window.bgcolor("lightgreen")`: Thiết lập màu nền cho cửa sổ là màu xanh lá nhạt.
- `painter = turtle.Turtle()`: Tạo một đối tượng rùa (turtle) và gán nó cho biến `painter`.
- `painter.fillcolor('blue')`: Thiết lập màu đổ đầy cho hình vẽ là màu xanh dương.
- `painter.pencolor('blue')`: Thiết lập màu bút vẽ là màu xanh dương.
- `painter.pensize(3)`: Thiết lập độ dày của bút vẽ là 3 pixel.
- `def drawsq(t, s)`: Định nghĩa một hàm có tên `drawsq` với hai tham số:
 - + `t`: Đối tượng rùa sẽ vẽ hình vuông.
 - + `s`: Độ dài cạnh của hình vuông.
- Hàm này sẽ vẽ một hình vuông bằng cách lặp 4 lần, mỗi lần di chuyển về phía trước một đoạn bằng độ dài cạnh và rẽ trái 90 độ.
- Vòng lặp `for i in range(1,50)`: Lặp 49 lần.
- `painter.left(18)`: Mỗi lần lặp, rùa sẽ rẽ trái 18 độ.
- `drawsq(painter, 200)`: Gọi hàm `drawsq` để vẽ một hình vuông có cạnh dài 200 pixel.

1.3.2. Viết chương trình đồ họa sử dụng thư viện turtle, kiểm tra kết quả và giải thích chương trình.

```
print('Đào Quang Huy MSSV 235752021610051')
import turtle, random # Nhập thư viện Turtle và Random

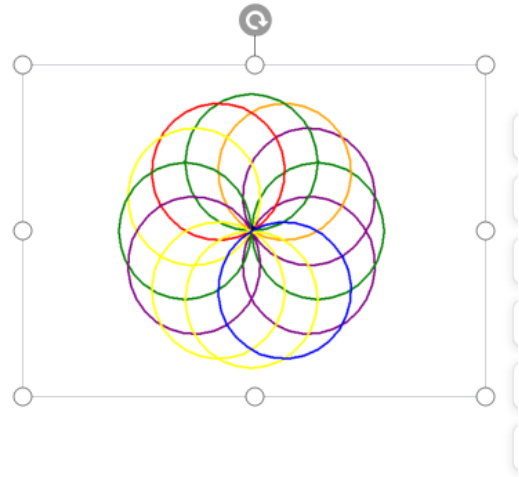
# Danh sách các màu sắc
colors = ["red", "green", "blue", "orange", "purple", "pink", "yellow"]

# Tạo đối tượng rùa và thiết lập độ dày bút vẽ
painter = turtle.Turtle()
painter.pensize(3)

# Vẽ nhiều hình tròn với màu sắc ngẫu nhiên
for i in range(20):
    # Chọn màu ngẫu nhiên
    color = random.choice(colors)
    painter.pencolor(color) # Đặt màu bút vẽ

    # Vẽ hình tròn
    painter.circle(100)

    # Xoay và di chuyển rùa
    painter.right(30)
    painter.left(60)
    painter.setposition(0, 0) # Quay về vị trí xuất phát
```



Hình 2. Sử dụng thư viện turtle để vẽ các màu sắc tùy chọn.

Chú thích:

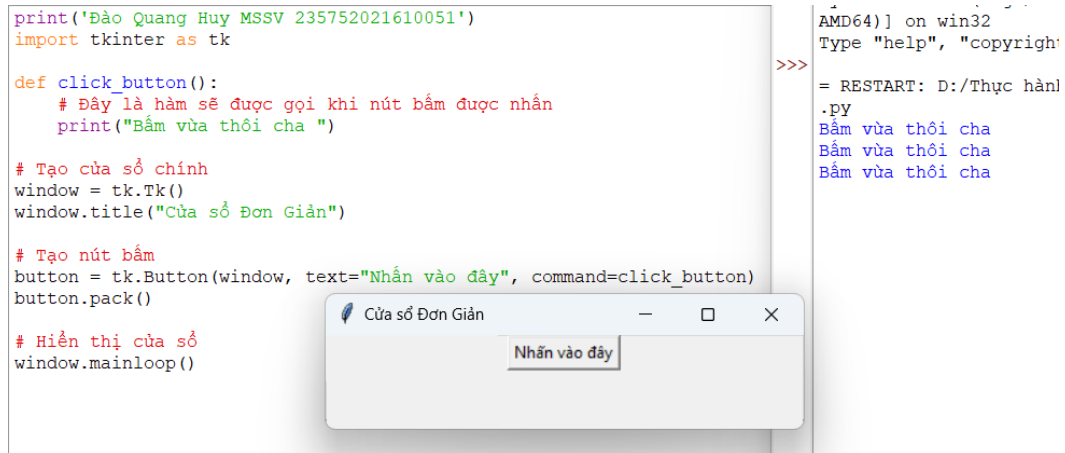
- import turtle, random: Dòng lệnh này nhập hai thư viện:
 - + turtle: Cung cấp các hàm và đối tượng để vẽ đồ họa.
 - + random: Dùng để chọn ngẫu nhiên các màu sắc từ danh sách.
- colors = ["red", "green", "blue", "orange", "purple", "pink", "yellow"]: Tạo một danh sách chứa các màu sắc sẽ được sử dụng để vẽ.
- painter = turtle.Turtle(): Tạo một đối tượng rùa (turtle) và gán nó cho biến painter.
- painter.pensize(3): Thiết lập độ dày của bút vẽ là 3 pixel.
- for i in range(20): Vòng lặp này sẽ thực hiện 20 lần.
 - + color = random.choice(colors): Chọn ngẫu nhiên một màu từ danh sách colors.
 - + painter.pencolor(color): Thiết lập màu bút vẽ cho rùa.
 - + painter.circle(100): Vẽ một hình tròn có bán kính 100 đơn vị.
 - + painter.right(30): Rẽ phải 30 độ.
 - + painter.left(60): Rẽ trái 60 độ.
 - + painter.setposition(0, 0): Đặt rùa về vị trí (0, 0) trên màn hình.

1.3.3. Dựa trên các kết quả đạt được từ các chương trình trên hãy viết chương trình hiển thị hình ảnh đồ họa sau.

Chú ý: Làm tương tự như câu trên (1.3.2.)

1.3.4. Viết chương sử dụng thư viện đồ họa tkinter thực hiện:

- Xây dựng cửa sổ đồ họa window form
- Thêm một widget (button) vào window form
- Xây dựng phương thức xử lý sự kiện phím bấm



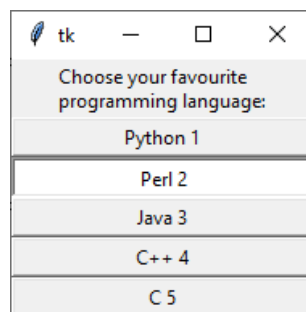
Hình 4. Xây dựng cửa sổ khi ấn vào sẽ in ra nội dung tùy chọn.

Chú thích:

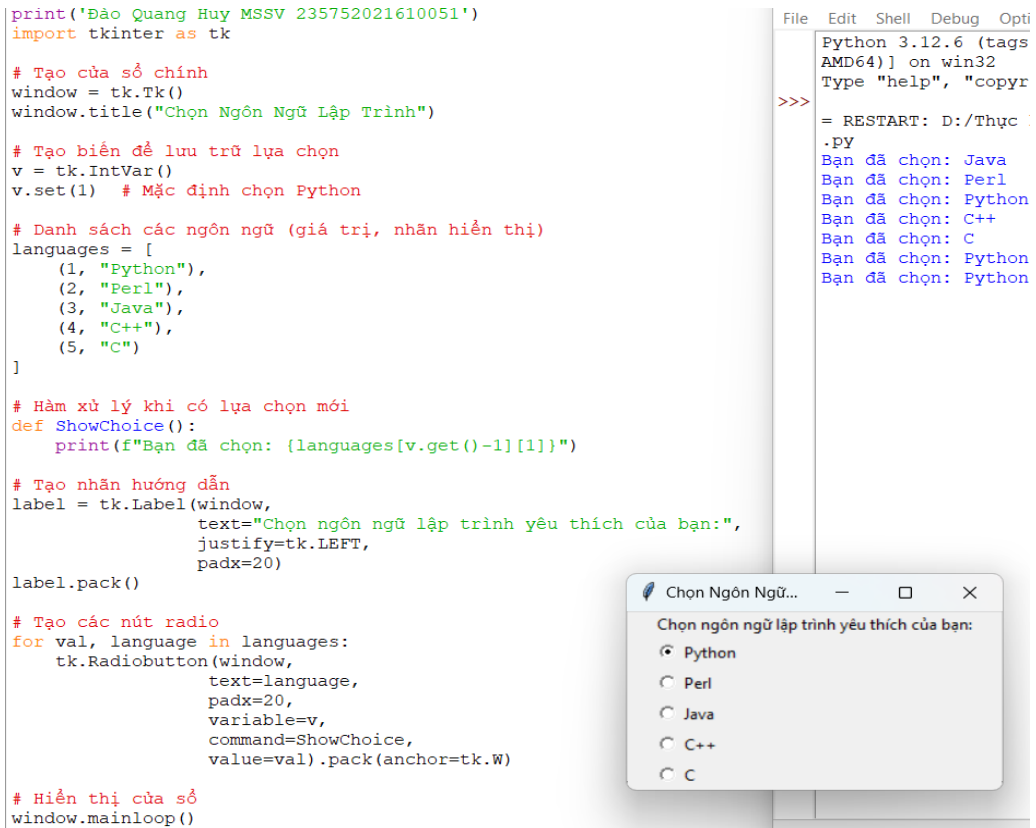
- `import tkinter as tk`: Dòng này nhập thư viện Tkinter và đặt biệt danh là tk để sử dụng cho gọn.
- `def click_button()`: Định nghĩa hàm `click_button()`. Hàm này sẽ được gọi tự động khi nút bấm được nhấn. Trong hàm này, bạn có thể thực hiện bất kỳ hành động nào bạn muốn, ví dụ như in ra màn hình, thực hiện một phép tính, hoặc mở một cửa sổ mới.
- `window = tk.Tk()`: Tạo một đối tượng cửa sổ và gán cho biến `window`.
- `window.title("Cửa sổ Đơn Giản")`: Đặt tiêu đề cho cửa sổ.
- `button = tk.Button(window, text="Nhấn vào đây", command=click_button)`:
 - + Tạo một nút bấm và gán cho biến `button`.
 - + `window`: Cửa sổ cha của nút bấm.
 - + `text`: Văn bản hiển thị trên nút bấm.
 - + `command`: Hàm sẽ được gọi khi nút bấm được nhấn.
- `button.pack()`: Xếp nút bấm vào trong cửa sổ.
- `window.mainloop()`: Bắt đầu vòng lặp chính của cửa sổ, cho phép cửa sổ được hiển thị và xử lý các sự kiện người dùng.

1.3.5. Sử dụng thư viện tkinter thực hiện:

- Xây dựng các radio button cho phép thực hiện các lựa chọn khác nhau.
- Thay thế các radio button thành các indicator như hình.



a).



Hình 5.1. Chương trình yêu cầu chọn lựa các lựa chọn yêu thích.

Chú thích :

- `import tkinter as tk`: Nhập thư viện Tkinter và đặt biệt danh là tk để sử dụng cho gọn.
- `root = tk.Tk()`: Tạo một cửa sổ chính và gán cho biến root.
- `v = tk.IntVar()`: Tạo một biến kiểu số nguyên để lưu trữ giá trị của lựa chọn.
- `v.set(1)`: Gán giá trị mặc định là 1 cho biến v, tương ứng với lựa chọn "Python" trong danh sách.
- `languages`: Một danh sách các tuple, mỗi tuple chứa một ngôn ngữ và một giá trị tương ứng. Giá trị này sẽ được gán cho biến v khi người dùng chọn ngôn ngữ đó.
- `ShowChoice()`: Hàm này sẽ được gọi khi người dùng chọn một ngôn ngữ khác.
- `print(v.get())`: In ra giá trị hiện tại của biến v (tức là giá trị tương ứng với ngôn ngữ đã chọn).
- `tk.Label`: Tạo một nhãn (label) để hiển thị thông báo cho người dùng.
- `justify=tk.LEFT`: Căn chỉnh văn bản trong nhãn sang bên trái.
- `padx=20`: Thêm khoảng cách bên trái và bên phải của nhãn.
- `.pack()`: Xếp nhãn vào trong cửa sổ.
- `for val, language in enumerate(languages)`: Lặp qua danh sách các ngôn ngữ.
- `tk.Radiobutton`: Tạo một nút radio cho mỗi ngôn ngữ.
- `variable=v`: Liên kết nút radio với biến v để cập nhật giá trị khi được chọn.
- `command=ShowChoice`: Gọi hàm ShowChoice khi nút radio được chọn.
- `value=val`: Gán giá trị tương ứng cho nút radio.
- `.pack(anchor=tk.W)`: Xếp nút radio vào trong cửa sổ, căn chỉnh sang bên trái.
- `root.mainloop()`: Bắt đầu vòng lặp chính của cửa sổ, cho phép cửa sổ được hiển thị và xử lý các sự kiện người dùng.

b).