1. Cho 1 mảng số nguyên. Hãy xuất ra màn hình những dãy 3 số mà số thứ 3 bằng tổng của 2 số đầu tiên. Lưu ý mảng được xoay vòng nghĩa là sau vị trí n-1 sẽ là vị trí 0. Các dãy được xuất theo thứ tự từ trái qua phải. Ví dụ mảng số nguyên [ 2, 2, 3, 5, 1, 1 ] có 2 dãy thỏa yêu cầu được xuất ra màn hình là 2 + 3 = 5 và 1 + 1 = 2

def find\_triplets(arr):

n = len(arr) # Lấy độ dài của mảng

result = [] # Khởi tạo danh sách kết quả

for i in range(n):

# Lấy các phần tử theo thứ tự xoay vòng

a = arr[i] # Phần tử thứ nhất

b = arr[(i + 1) % n] # Phần tử thứ hai (xoay về đầu nếu vượt qua cuối mảng)

# b = arr[(i + 1) % n] giải thích:

# - (i + 1) là chỉ số tiếp theo trong mảng.

# - % n đảm bảo chỉ số nằm trong phạm vi [0, n-1].

# - Ví dụ: nếu i = n-1 (phần tử cuối), thì (i + 1) % n = 0 (phần tử đầu).

c = arr[(i + 2) % n] # Phần tử thứ ba

# Kiểm tra nếu số thứ ba bằng tổng của hai số đầu tiên

# Ví dụ sử dụng

arr = [2, 2, 3, 5, 1, 1] # Mảng số nguyên đầu vào

triplets = find\_triplets(arr) # Tìm các dãy thỏa mãn điều kiện

for triplet in triplets:

print(triplet) # Xuất từng dãy ra màn hình

1. Giải các bài tập sau sử dụng biểu thức lambda, filter:

Cho: danh\_sach = [5, 7, 6, 9, 14, 26, 21, 17, 44]

* 1. Xuất các số lẻ không chia hết cho 3

danh\_sach = [5, 7, 6, 9, 14, 26, 21, 17, 44]

# Lọc các số lẻ không chia hết cho 3

so\_le\_khong\_chia\_het\_cho\_3 = list(filter(lambda x: x % 2 != 0 and x % 3 != 0, danh\_sach))

print("Các số lẻ không chia hết cho 3:", so\_le\_khong\_chia\_het\_cho\_3)

* 1. Xuất các số nguyên tố

is\_prime = lambda x: x > 1 and all(x % i != 0 for i in range(2, int(x\*\*0.5) + 1))

# Lọc các số nguyên tố

so\_nguyen\_to = list(filter(is\_prime, danh\_sach))

print("Các số nguyên tố:", so\_nguyen\_to)

* 1. Tính tổng các số nguyên tố của mảng

tong\_nguyen\_to = sum(filter(is\_prime, danh\_sach)) print("Tổng các số nguyên tố:", tong\_nguyen\_to)

1. Bài tập về chuỗi: viết 2 cách trong đó có 1 cách dùng vòng lặp
   1. Đảo ngược 1 chuỗi, s= “hello” 🡪 olleh
   2. Đảo ngược các từ trong chuỗi “how are you” 🡪 “you are how”
   3. Kiểm tra một chuỗi có đối xứng hay không? madam, abeeba
   4. Đếm tần suất xuất hiện của một từ trong đoạn
2. Hàm đệ quy
   1. Tính n!
   2. Tính số thứ n của dãy Fibonacci: 1, 1, 2, 3, 5, 8, 13, …, n
   3. Tính tổng số lẻ trong mảng

# Bài tập về chuỗi và hàm đệ quy

# 3. Bài tập về chuỗi

# a. Đảo ngược một chuỗi

# Cách 1: Sử dụng cắt chuỗi

s = "hello"

dao\_nguoc\_1 = s[::-1]

print("Chuỗi đảo ngược (cắt chuỗi):", dao\_nguoc\_1)

# Cách 2: Sử dụng vòng lặp

s = "hello"

dao\_nguoc\_2 = ""

for char in s:

dao\_nguoc\_2 = char + dao\_nguoc\_2

print("Chuỗi đảo ngược (vòng lặp):", dao\_nguoc\_2)

# b. Đảo ngược các từ trong chuỗi

# Cách 1: Sử dụng split và join

chuoi = "how are you"

dao\_tu\_1 = " ".join(chuoi.split()[::-1])

print("Đảo ngược các từ (split & join):", dao\_tu\_1)

# Cách 2: Sử dụng vòng lặp

tu\_list = chuoi.split()

dao\_tu\_2 = ""

for tu in reversed(tu\_list):

dao\_tu\_2 += tu + " "

print("Đảo ngược các từ (vòng lặp):", dao\_tu\_2.strip())

# c. Kiểm tra chuỗi có đối xứng hay không

# Cách 1: Sử dụng cắt chuỗi

chuoi\_kiem\_tra = "madam"

la\_doi\_xung\_1 = chuoi\_kiem\_tra == chuoi\_kiem\_tra[::-1]

print("Chuỗi đối xứng (cắt chuỗi):", la\_doi\_xung\_1)

# Cách 2: Sử dụng vòng lặp

la\_doi\_xung\_2 = True

for i in range(len(chuoi\_kiem\_tra) // 2):

if chuoi\_kiem\_tra[i] != chuoi\_kiem\_tra[-(i + 1)]:

la\_doi\_xung\_2 = False

break

print("Chuỗi đối xứng (vòng lặp):", la\_doi\_xung\_2)

# d. Đếm tần suất xuất hiện của một từ trong đoạn

van\_ban = "how are you you are how how"

tu\_can\_dem = "how"

tan\_suat = van\_ban.split().count(tu\_can\_dem)

print(f"Tần suất của từ '{tu\_can\_dem}':", tan\_suat)

# 4. Hàm đệ quy

# a. Tính n!

def tinh\_giai\_thua(n):

if n == 0 or n == 1:

return 1

return n \* tinh\_giai\_thua(n - 1)

n = 5

print(f"Giai thừa của {n}:", tinh\_giai\_thua(n))

# b. Tính số thứ n của dãy Fibonacci

def fibonacci(n):

if n == 1 or n == 2:

return 1

return fibonacci(n - 1) + fibonacci(n - 2)

n = 7

print(f"Số Fibonacci thứ {n}:", fibonacci(n))

# c. Tính tổng số lẻ trong mảng

def tong\_so\_le(arr, n=None):

if n is None:

n = len(arr)

if n == 0:

return 0

if arr[n - 1] % 2 != 0:

return arr[n - 1] + tong\_so\_le(arr, n - 1)

return tong\_so\_le(arr, n - 1)

arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]

print("Tổng các số lẻ trong mảng:", tong\_so\_le(arr))

1. Viết hàm tính giá trị trung bình (của các số đưa vào)

3,4,5 🡪 4.0

3,4,5,6 🡪 4.5

# 5. Hàm tính giá trị trung bình

def tinh\_trung\_binh(\*args):

if len(args) == 0:

return 0

return sum(args) / len(args)

# Ví dụ sử dụng

print("Trung bình của 3, 4, 5:", tinh\_trung\_binh(3, 4, 5))

print("Trung bình của 3, 4, 5, 6:", tinh\_trung\_binh(3, 4, 5, 6))