**Bài 1:**

***1. Formatted Twinkle Poem***

print('''

Twinkle, twinkle, little star,

    How I wonder what you are!

        Up above the world so high,

        Like a diamond in the sky.

Twinkle, twinkle, little star,

    How I wonder what you are

''')

***2. Python Version Checker***

import sys

print("Python version")

print(sys.version)

print("Version info.")

print(sys.version\_info)

***3. Current DateTime Display***

import datetime

dt1 = datetime.datetime.now()

print("Current date and time : ")

print(dt1.strftime("%Y-%m-%d %H:%M:%S"))

***4. Circle Area Calculator***

import math

banKinh = float(input("Nhập bán kính hình tròn: "))

dienTich = math.pi \* banKinh \*\* 2

print("Diện tích hình tròn có bán kính", banKinh, "la", dienTich)

***5. Reverse Full Name***

ten = input("Nhập tên cua người dùng: ")

ho = input("Nhập họ của người dùng: ")

print("Ho ten của người dùng là:", ho, ten)

***6. List and Tuple Generator***

values = input("Nhập số nguyên được phân tách bằng dấu phẩy: ")

list = values.split(", ")

tuple = tuple(list)

print('List: ', list)

print('Tuple: ', tuple)

**7. File Extension Extractor**

filename = input("Nhập tên tệp: ")

tenTep = filename.split('.')

print("Phần mở rộng của tập tin là:", tenTep[-1])

**8. First and Last Colors**

color\_list = ["Red","Green","White" ,"Black"]

print("Màu đầu tiên:", color\_list[0],";", "Màu cuối cùng:", color\_list[-1])

**9. Exam Schedule Formatter**

ngayThi = (1, 8, 2024)

print("The examination will start from : %i / %i / %i" % ngayThi)

***10. Number Expansion Calculator***

nhapSoNguyen = int(input("Nhập số nguyên: "))

n1 = int("%s" % nhapSoNguyen)

n2 = int("%s%s" % (nhapSoNguyen, nhapSoNguyen))

n3 = int("%s%s%s" % (nhapSoNguyen, nhapSoNguyen, nhapSoNguyen))

print("Kết quả:", n1 + n2 + n3)

***11. Function Documentation Printer***

***# Print the docstring (documentation) of the 'len' function***

***print(len.\_\_doc\_\_)***

***# Print the docstring (documentation) of the 'sorted' function***

***print(sorted.\_\_doc\_\_)***

***# Print the docstring (documentation) of the 'sum' function***

***print(sum.\_\_doc\_\_)***

***# Print the docstring (documentation) of the 'map' function***

***print(map.\_\_doc\_\_)***

***# Print the docstring (documentation) of the 'filter' function***

***print(filter.\_\_doc\_\_)***

***12. Monthly Calendar Display***

# Import the 'calendar' module

import calendar

# Prompt the user to input the year and month

y = int(input("Input the year : "))

m = int(input("Input the month : "))

# Print the calendar for the specified year and month

print(calendar.month(y, m))

13. Multi-line Here Document

# Use triple double-quotes to create a multi-line string

print("""

a string that you "don't" have to escape

This

is a ....... multi-line

heredoc string --------> example

""")

14. Days Between Dates

# Import the 'date' class from the 'datetime' module

from datetime import date

# Define a start date as July 2, 2014

f\_date = date(2014, 7, 2)

# Define an end date as July 11, 2014

l\_date = date(2014, 7, 11)

# Calculate the difference between the end date and start date

delta = l\_date - f\_date

# Print the number of days in the time difference

print(delta.days)

15. Sphere Volume Calculator

# Define the value of pi

pi = 3.1415926535897931

# Define the radius of the sphere

r = 6.0

# Calculate the volume of the sphere using the formula

V = 4.0/3.0 \* pi \* r\*\*3

# Print the calculated volume of the sphere

print('The volume of the sphere is: ', V)

16. Difference from 17

# Define a function named "difference" that takes an integer parameter "n"

def difference(n):

# Check if n is less than or equal to 17

if n <= 17:

# If n is less than or equal to 17, return the absolute difference between 17 and n

return 17 - n

else:

# If n is greater than 17, return the absolute difference between n and 17 multiplied by 2

return (n - 17) \* 2

# Call the "difference" function with the argument 22 and print the result

print(difference(22))

# Call the "difference" function with the argument 14 and print the result

print(difference(14))

17. Number Range Tester

# Define a function named "near\_thousand" that takes an integer parameter "n"

def near\_thousand(n):

# Check if the absolute difference between 1000 and n is less than or equal to 100

# OR check if the absolute difference between 2000 and n is less than or equal to 100

return ((abs(1000 - n) <= 100) or (abs(2000 - n) <= 100))

# Call the "near\_thousand" function with the argument 1000 and print the result

print(near\_thousand(1000))

# Call the "near\_thousand" function with the argument 900 and print the result

print(near\_thousand(900))

# Call the "near\_thousand" function with the argument 800 and print the result

print(near\_thousand(800))

# Call the "near\_thousand" function with the argument 2200 and print the result

print(near\_thousand(2200))

18. Triple Sum Calculator

# Define a function named "sum\_thrice" that takes three integer parameters: x, y, and z

def sum\_thrice(x, y, z):

# Calculate the sum of x, y, and z

sum = x + y + z

# Check if x, y, and z are all equal (all three numbers are the same)

if x == y == z:

# If they are equal, triple the sum

sum = sum \* 3

# Return the final sum

return sum

# Call the "sum\_thrice" function with the arguments (1, 2, 3) and print the result

print(sum\_thrice(1, 2, 3))

# Call the "sum\_thrice" function with the arguments (3, 3, 3) and print the result

print(sum\_thrice(3, 3, 3))

19. Prefix "Is" String Modifier

# Define a function named "new\_string" that takes a string parameter called "text"

def new\_string(text):

# Check if the length of the "text" is greater than or equal to 2 and if the first two characters of "text" are "Is"

if len(text) >= 2 and text[:2] == "Is":

# If the conditions are met, return the original "text" unchanged

return text

else:

# If the conditions are not met, prepend "Is" to the "text" and return the modified string

return "Is" + text

# Call the "new\_string" function with the argument "Array" and print the result

print(new\_string("Array"))

# Call the "new\_string" function with the argument "IsEmpty" and print the result

print(new\_string("IsEmpty"))

20. String Copy Generator

# Define a function named "larger\_string" that takes two parameters, "text" and "n"

def larger\_string(text, n):

# Initialize an empty string variable named "result"

result = ""

# Use a for loop to repeat the "text" "n" times and concatenate it to the "result"

for i in range(n):

result = result + text

# Return the final "result" string

return result

# Call the "larger\_string" function with the arguments 'abc' and 2, then print the result

print(larger\_string('abc', 2))

# Call the "larger\_string" function with the arguments '.py' and 3, then print the result

print(larger\_string('.py', 3))

21. Even or Odd Checker

# Prompt the user to enter a number and convert the input to an integer

num = int(input("Enter a number: "))

# Calculate the remainder when the number is divided by 2

mod = num % 2

# Check if the remainder is greater than 0, indicating an odd number

if mod > 0:

# Print a message indicating that the number is odd

print("This is an odd number.")

else:

# Print a message indicating that the number is even

print("This is an even number.")

22. Count 4 in List

# Define a function called list\_count\_4 that takes a list of numbers (nums) as a parameter.

def list\_count\_4(nums):

# Initialize a variable count to keep track of the count of occurrences of the number 4.

count = 0

# Iterate through each element (num) in the input list (nums).

for num in nums:

# Check if the current element (num) is equal to 4.

if num == 4:

# If the element is 4, increment the count by 1.

count = count + 1

# Return the final count after iterating through the list.

return count

# Call the list\_count\_4 function with two different input lists and print the results.

print(list\_count\_4([1, 4, 6, 7, 4])) # Output: 2 (There are two occurrences of 4 in the list.)

print(list\_count\_4([1, 4, 6, 4, 7, 4])) # Output: 3 (There are three occurrences of 4 in the list.)

23. String Prefix Copies

# Define a function called substring\_copy that takes two parameters: text (a string) and n (an integer).

def substring\_copy(text, n):

# Set the initial value of flen (substring length) to 2.

flen = 2

# Check if flen is greater than the length of the input text.

if flen > len(text):

# If flen is greater, set it to the length of the text to avoid going out of bounds.

flen = len(text)

# Extract a substring of length flen from the beginning of the text.

substr = text[:flen]

# Initialize an empty string result to store the concatenated substrings.

result = ""

# Iterate n times to concatenate the substring to the result.

for i in range(n):

result = result + substr

# Return the final result after concatenating substrings.

return result

# Call the substring\_copy function with two different inputs and print the results.

print(substring\_copy('abcdef', 2)) # Output: "abab" (substring "ab" repeated 2 times)

print(substring\_copy('p', 3)) # Output: "pp" (substring "p" repeated 3 times)

25. Value in Group Tester

# Define a function called is\_group\_member that takes two parameters: group\_data (a list) and n (an integer).

def is\_group\_member(group\_data, n):

# Iterate through the elements (values) in the group\_data list.

for value in group\_data:

# Check if the current value is equal to the given integer, n.

if n == value:

return True # If found, return True.

return False # If the loop completes and no match is found, return False.

# Call the is\_group\_member function with two different lists and integers, and print the results.

print(is\_group\_member([1, 5, 8, 3], 3)) # Output: True (3 is in the list)

print(is\_group\_member([5, 8, 3], -1)) # Output: False (-1 is not in the list)

Bài 2

1 a.

def tinhTong(a,b):

return a + b

a = float(input("Nhap a: "))

b = float(input("Nhap b: "))

tong = tinhTong(a,b)

print(f"Tong {a} va {b} la {tong}")

b.

def tinhTong(a,b):

    return a / b

a = float(input("Nhap a: "))

b = float(input("Nhap b: "))

tong = tinhTong(a,b)

print(f"Tich {a} va {b} la {tong}")

c.

def tinhTong(a,b):

    return a \*\* b

a = float(input("Nhap a: "))

b = float(input("Nhap b: "))

tong = tinhTong(a,b)

print(f"Luy thua {a} va {b} la {tong}")

2. Tính diện tích hình chữ nhật khi biết bán kính

def tinhDienTich(a,b):

    return a \* b

a = float(input("Nhap chieu dai: "))

b = float(input("Nhap chieu rong: "))

tong = tinhDienTich(a,b)

print(f"Dien tich {a} va {b} la {tong}")

3. Xuất tất cả các số nguyên tố trong 1 khoảng cho trước

def is\_prime(n):

    if n<2:

        return False

    for i in range(2, int(n\*\*0.5)+1):

        if n % 1 == 0:

            return False

    return True

def primes\_in\_range(start, end):

    return [n for n in range(start, end + 1 )if is\_prime(n)]

start = int(input("Nhap so bat dau"))

end = int(input("Nhap so ket thuc"))

print("Cac so nguyen to trong khoang: ", primes\_in\_range(start, end))

4. Kiểm tra 1 số nguyên n có phải là số Fibonacci hay không

import math

def is\_perfect\_square(n):

    s =int(math.sqrt(n))

    return s \* s == n

def is\_fibonacci(n):

    if n < 0:

       return False

    return is\_perfect\_square(5 \* n \* n +4) or is\_perfect\_square(5 \* n \* n -4)

n = int(input("Nhap so Fibonacci "))

if is\_fibonacci(n):

    print(f"{n} la so Fibonacci")

else:

 print(f"{n} ko la so Fibonacci")

5. Tìm số Fibonacci thứ n (dùng đệ quy và không đệ quy)

Dung đệ quy

def fibonacci\_recursive(n):

    if n <= 0:

        return 0

    elif n == 1:

        return 1

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

# Ví dụ:

print(fibonacci\_recursive(10))  # Output: 55

Không dung đệ quy

def fibonacci\_iterative(n):

    if n <= 0:

        return 0

    elif n == 1:

        return 1

    a, b = 0, 1

    for \_ in range(2, n + 1):

        a, b = b, a + b

    return b

# Ví dụ:

print(fibonacci\_iterative(10))  # Output: 55

6. Tính tổng n số Fibonacci đầu tiên (dùng đệ quy và không đệ quy)

1. **Dùng đệ quy**

def fibonacci\_recursive(n):

    if n <= 0:

        return 0

    elif n == 1:

        return 1

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

def sum\_fibonacci\_recursive(n):

    if n <= 0:

        return 0

    return fibonacci\_recursive(n) + sum\_fibonacci\_recursive(n - 1)

# Ví dụ:

print(sum\_fibonacci\_recursive(6))  # Output: 20

2. **Không đệ quy (Dùng vòng lặp - Iterative)**

def sum\_fibonacci\_iterative(n):

    if n <= 0:

        return 0

    elif n == 1:

        return 1

    a, b = 0, 1

    total = a + b

    for \_ in range(2, n):

        a, b = b, a + b

        total += b

    return total

# Ví dụ:

print(sum\_fibonacci\_iterative(6))  # Output: 20

7. Tính tổng căn bậc 2 của n số nguyên đầu tiên

import math

def sum\_sqrt\_iterative(n):

total = 0

# Duyệt từ 1 đến n, tính căn bậc 2 và cộng dồn vào total

for i in range(1, n + 1):

total += math.sqrt(i)

return total

# Ví dụ:

n = 6

print("Tổng không đệ quy:", sum\_sqrt\_iterative(n))

**8. Giải phương trình bậc 2: ax2 + bx + c=0**

import math

def solve\_quadratic(a, b, c):

# Trường hợp a = 0: không phải phương trình bậc 2

if a == 0:

if b == 0:

return "Phương trình có vô số nghiệm" if c == 0 else "Phương trình vô nghiệm"

else:

# Giải phương trình bậc nhất: bx + c = 0

return (-c / b,)

# Tính delta

delta = b\*\*2 - 4 \* a \* c

if delta < 0:

return "Phương trình vô nghiệm"

elif delta == 0:

x = -b / (2 \* a)

return (x,)

else:

sqrt\_delta = math.sqrt(delta)

x1 = (-b + sqrt\_delta) / (2 \* a)

x2 = (-b - sqrt\_delta) / (2 \* a)

return (x1, x2)

# Ví dụ:

print("Phương trình: x^2 - 3x + 2 = 0")

print("Nghiệm:", solve\_quadratic(1, -3, 2))

9. Tính n!

def factorial\_iterative(n):

if n < 0:

return "Giai thừa không xác định với số âm"

result = 1

for i in range(2, n + 1):

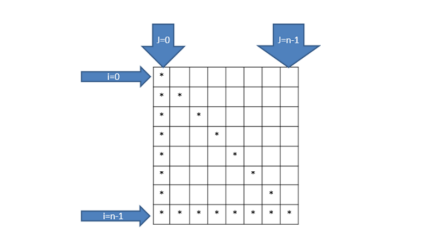
result \*= i

return result

# Ví dụ:

print("5! =", factorial\_iterative(5)) # Kết quả: 120

**10.In \* dạng tam giác dưới như hình bên, đầu vào là số hàng(cột)**

****

**def print\_lower\_triangle\_simple(n):**

**for i in range(1, n + 1):**

**print('\*' \* i)**

**# Ví dụ:**

**n = 5**

**print\_lower\_triangle\_simple(n)**

**# Kết quả:**

**# \***

**# \*\***

**# \*\*\***

**# \*\*\*\***

**# \*\*\*\*\***

11. Đổi giờ - phút – giây: thời gian đầu vào là giây được đổi thành giờ, phút, giây. Xuất kết quả ra màn hình dưới dạng: giờ:phút:giây. Ví dụ: soGiay = 3770 thì xuất ra màn hình 1:2:50.

def convert\_seconds\_to\_hms(soGiay):

gio = soGiay // 3600

phanDu = soGiay % 3600

phut = phanDu // 60

giay = phanDu % 60

# Trả về chuỗi theo định dạng giờ:phút:giây

return f"{gio}:{phut}:{giay}"

# Kiểm thử

soGiay = 3770

result = convert\_seconds\_to\_hms(soGiay)

print(result) # Kết quả mong muốn: 1:2:50

**12.Cho một mảng số nguyên: (nên viết 2-3 cách)**

**```python**

**arr = [1, 2, 3, 4, 5, 8, 13, 15, 16, 25, 27, 29, 30, 31]**

**```**

**Để “viết 2-3 cách”, bạn có thể tham khảo các cách làm khác nhau:**

**- Dùng vòng lặp `for` truyền thống**

**- Dùng \*\*list comprehension\*\* (hoặc kết hợp với hàm `filter`, `map`)**

**- Dùng các hàm sẵn có (như `count`, `sum`, `sorted`, …) hoặc thư viện (`collections.Counter`, …)**

**Mình sẽ gợi ý ít nhất 2 cách (khi phù hợp) cho mỗi ý.**

**---**

**## (a) Xuất tất cả các số lẻ \*\*không chia hết cho 5\*\***

**\*\*Cách 1: Dùng vòng lặp `for`\*\***

**```python**

**odd\_not\_div5 = []**

**for x in arr:**

**if x % 2 != 0 and x % 5 != 0:**

**odd\_not\_div5.append(x)**

**print("a) Số lẻ không chia hết cho 5 (for loop):", odd\_not\_div5)**

**```**

**\*\*Cách 2: Dùng list comprehension\*\***

**```python**

**odd\_not\_div5\_lc = [x for x in arr if x % 2 != 0 and x % 5 != 0]**

**print("a) Số lẻ không chia hết cho 5 (list comprehension):", odd\_not\_div5\_lc)**

**```**

**---**

**## (b) Xuất tất cả các số Fibonacci trong danh sách**

**Muốn kiểm tra số nào là Fibonacci, ta có thể:**

**1. Sinh các số Fibonacci đến giá trị lớn nhất trong `arr` rồi đưa vào một `set`.**

**2. Kiểm tra phần tử `arr` có nằm trong `set` đó hay không.**

**\*\*Bước 1: Sinh các số Fibonacci\*\***

**```python**

**def generate\_fibonacci\_upto(n):**

**"""Trả về một tập hợp (set) các số Fibonacci <= n."""**

**fib\_set = set()**

**a, b = 0, 1**

**fib\_set.add(a)**

**fib\_set.add(b)**

**while b <= n:**

**a, b = b, a + b**

**fib\_set.add(a)**

**return fib\_set**

**```**

**\*\*Bước 2: Lọc các phần tử trong `arr`\*\***

**```python**

**fib\_set = generate\_fibonacci\_upto(max(arr))**

**fib\_in\_arr = [x for x in arr if x in fib\_set]**

**print("b) Các số Fibonacci trong arr:", fib\_in\_arr)**

**```**

**\*\*(Cách khác)\*\*: Dùng tính chất “`n` là Fibonacci nếu và chỉ nếu `5\*n\*n + 4` hoặc `5\*n\*n - 4` là số chính phương”. Tuy nhiên, cách sinh `fib\_set` như trên thường dễ thao tác hơn.**

**---**

**## (c) Tìm số nguyên tố lớn nhất**

**\*\*Cách 1: Dùng hàm kiểm tra nguyên tố + duyệt\*\***

**```python**

**def is\_prime(n):**

**if n < 2:**

**return False**

**for i in range(2, int(n\*\*0.5) + 1):**

**if n % i == 0:**

**return False**

**return True**

**def max\_prime(arr):**

**max\_p = None**

**for x in arr:**

**if is\_prime(x):**

**if max\_p is None or x > max\_p:**

**max\_p = x**

**return max\_p**

**largest\_prime = max\_prime(arr)**

**print("c) Số nguyên tố lớn nhất:", largest\_prime)**

**```**

**\*\*Cách 2: Dùng list comprehension + hàm `max()`\*\***

**```python**

**primes\_in\_arr = [x for x in arr if is\_prime(x)]**

**largest\_prime\_lc = max(primes\_in\_arr) if primes\_in\_arr else None**

**print("c) Số nguyên tố lớn nhất (list comprehension):", largest\_prime\_lc)**

**```**

**---**

**## (d) Tìm số Fibonacci bé nhất trong danh sách**

**Tận dụng `fib\_set` ở (b):**

**```python**

**fib\_in\_arr = [x for x in arr if x in fib\_set]**

**min\_fib = min(fib\_in\_arr) if fib\_in\_arr else None**

**print("d) Số Fibonacci bé nhất:", min\_fib)**

**```**

**---**

**## (e) Tính trung bình các số lẻ**

**\*\*Cách 1: Dùng vòng lặp\*\***

**```python**

**sum\_odd = 0**

**count\_odd = 0**

**for x in arr:**

**if x % 2 != 0:**

**sum\_odd += x**

**count\_odd += 1**

**avg\_odd = sum\_odd / count\_odd if count\_odd > 0 else None**

**print("e) Trung bình các số lẻ (for loop):", avg\_odd)**

**```**

**\*\*Cách 2: Dùng list comprehension\*\***

**```python**

**odd\_numbers = [x for x in arr if x % 2 != 0]**

**avg\_odd\_lc = sum(odd\_numbers)/len(odd\_numbers) if odd\_numbers else None**

**print("e) Trung bình các số lẻ (list comprehension):", avg\_odd\_lc)**

**```**

**---**

**## (f) Tính tích các phần tử là số lẻ \*\*không chia hết cho 3\*\***

**\*\*Cách 1: Dùng vòng lặp\*\***

**```python**

**product = 1**

**has\_valid = False**

**for x in arr:**

**if x % 2 != 0 and x % 3 != 0:**

**product \*= x**

**has\_valid = True**

**product\_result = product if has\_valid else 0 # tuỳ quy ước nếu không có phần tử nào**

**print("f) Tích các số lẻ không chia hết cho 3:", product\_result)**

**```**

**\*\*Cách 2: Dùng `math.prod` (Python 3.8+) kết hợp list comprehension\*\***

**```python**

**import math**

**numbers\_f = [x for x in arr if x % 2 != 0 and x % 3 != 0]**

**product\_f = math.prod(numbers\_f) if numbers\_f else 0**

**print("f) Tích (math.prod):", product\_f)**

**```**

**---**

**## (g) Đổi chỗ 2 phần tử của danh sách (đầu vào là 2 vị trí)**

**Ví dụ ta có hai vị trí `i`, `j` muốn đổi chỗ trong `arr`.**

**\*\*Cách 1: Gán tạm\*\***

**```python**

**def swap\_in\_list(arr, i, j):**

**if i < 0 or i >= len(arr) or j < 0 or j >= len(arr):**

**return # hoặc raise Exception**

**temp = arr[i]**

**arr[i] = arr[j]**

**arr[j] = temp**

**test\_arr\_g = arr[:] # copy**

**swap\_in\_list(test\_arr\_g, 0, 3) # đổi chỗ phần tử đầu (index 0) và index 3**

**print("g) Danh sách sau khi swap (temp):", test\_arr\_g)**

**```**

**\*\*Cách 2: Pythonic\*\***

**```python**

**def swap\_in\_list\_pythonic(arr, i, j):**

**arr[i], arr[j] = arr[j], arr[i]**

**test\_arr\_g2 = arr[:]**

**swap\_in\_list\_pythonic(test\_arr\_g2, 1, 4)**

**print("g) Danh sách sau khi swap (Pythonic):", test\_arr\_g2)**

**```**

**---**

**## (h) Đảo ngược trật tự các phần tử của danh sách**

**\*\*Cách 1: Dùng hàm `reverse()` in-place\*\***

**```python**

**test\_arr\_h = arr[:]**

**test\_arr\_h.reverse()**

**print("h) Danh sách sau khi reverse() in-place:", test\_arr\_h)**

**```**

**\*\*Cách 2: Dùng slicing\*\***

**```python**

**reversed\_arr = arr[::-1]**

**print("h) Danh sách đảo ngược (slicing):", reversed\_arr)**

**```**

**\*\*Cách 3: Dùng hàm `reversed()` (trả về iterator)\*\***

**```python**

**reversed\_arr\_iter = list(reversed(arr))**

**print("h) Danh sách đảo ngược (reversed()):", reversed\_arr\_iter)**

**```**

**---**

**## (i) Xuất tất cả các số \*\*lớn thứ nhì\*\* của danh sách**

**Ta hiểu “lớn thứ nhì” là \*\*giá trị lớn thứ hai về mặt \*\*distinct\*\* (không tính trùng lặp). Sau khi tìm được giá trị đó, in ra mọi vị trí (hoặc mọi “bản sao”) nếu có.**

**\*\*Bước 1\*\*: Tìm giá trị lớn thứ nhất và thứ nhì (phân biệt).**

**\*\*Bước 2\*\*: In tất cả phần tử trong mảng bằng giá trị đó (nếu muốn).**

**\*\*Cách 1: Dùng `sorted` và `set`\*\***

**```python**

**unique\_vals = sorted(set(arr))**

**if len(unique\_vals) < 2:**

**second\_largest\_val = None**

**else:**

**second\_largest\_val = unique\_vals[-2]**

**print("i) Giá trị lớn thứ nhì:", second\_largest\_val)**

**# Nếu muốn xuất mọi phần tử bằng giá trị đó:**

**if second\_largest\_val is not None:**

**all\_second\_largest = [x for x in arr if x == second\_largest\_val]**

**print(" Tất cả các 'bản sao' lớn thứ nhì:", all\_second\_largest)**

**else:**

**print(" Không tồn tại giá trị lớn thứ nhì.")**

**```**

**\*\*Cách 2: Tìm trực tiếp (không sort)\*\***

**```python**

**def find\_second\_largest(arr):**

**largest = None**

**second\_largest = None**

**for x in arr:**

**if largest is None or x > largest:**

**second\_largest = largest**

**largest = x**

**elif x != largest and (second\_largest is None or x > second\_largest):**

**second\_largest = x**

**return second\_largest**

**sl\_val = find\_second\_largest(arr)**

**print("i) Lớn thứ nhì (không sort):", sl\_val)**

**```**

**---**

**## (j) Tính tổng \*\*các chữ số\*\* của tất cả các số trong danh sách**

**\*\*Cách 1: Viết hàm `sum\_of\_digits` rồi cộng dồn\*\***

**```python**

**def sum\_of\_digits(n):**

**# Xét giá trị tuyệt đối để không ảnh hưởng dấu**

**n = abs(n)**

**s = 0**

**while n > 0:**

**s += n % 10**

**n //= 10**

**return s**

**total\_digit\_sum = 0**

**for x in arr:**

**total\_digit\_sum += sum\_of\_digits(x)**

**print("j) Tổng chữ số của tất cả các số:", total\_digit\_sum)**

**```**

**\*\*Cách 2: Dùng list comprehension + `sum()`\*\***

**```python**

**total\_digit\_sum\_lc = sum(sum\_of\_digits(x) for x in arr)**

**print("j) Tổng chữ số (list comprehension):", total\_digit\_sum\_lc)**

**```**

**---**

**## (k) Đếm số lần xuất hiện của \*\*một số\*\* trong danh sách**

**Giả sử ta muốn đếm số lần xuất hiện của `target = 5`.**

**\*\*Cách 1: Dùng vòng lặp\*\***

**```python**

**def count\_occurrences\_loop(arr, target):**

**count = 0**

**for x in arr:**

**if x == target:**

**count += 1**

**return count**

**count\_5\_loop = count\_occurrences\_loop(arr, 5)**

**print("k) Đếm số lần xuất hiện của 5 (loop):", count\_5\_loop)**

**```**

**\*\*Cách 2: Dùng phương thức `list.count`\*\***

**```python**

**count\_5\_builtin = arr.count(5)**

**print("k) Đếm số lần xuất hiện của 5 (builtin):", count\_5\_builtin)**

**```**

**---**

**## (l) Xuất các số xuất hiện \*\*n lần\*\* trong danh sách**

**Giả sử `n = 2`: ta muốn tìm tất cả các số xuất hiện \*\*đúng 2 lần\*\*.**

**\*\*Cách 1: Dùng `collections.Counter`\*\***

**```python**

**from collections import Counter**

**def numbers\_appear\_n\_times(arr, n):**

**freq = Counter(arr)**

**# Lọc ra những số có tần suất == n**

**return [num for num, count in freq.items() if count == n]**

**nums\_appear\_2 = numbers\_appear\_n\_times(arr, 2)**

**print("l) Các số xuất hiện đúng 2 lần:", nums\_appear\_2)**

**```**

**\*\*Cách 2: Tự dùng dict\*\***

**```python**

**def numbers\_appear\_n\_times\_dict(arr, n):**

**freq = {}**

**for x in arr:**

**freq[x] = freq.get(x, 0) + 1**

**return [k for k, v in freq.items() if v == n]**

**nums\_appear\_2\_dict = numbers\_appear\_n\_times\_dict(arr, 2)**

**print("l) Các số xuất hiện đúng 2 lần (dict):", nums\_appear\_2\_dict)**

**```**

**---**

**## (m) Xuất các số xuất hiện \*\*nhiều lần nhất\*\* trong danh sách**

**Ta muốn tìm tất cả các “mode” (các số có tần suất lớn nhất).**

**\*\*Cách 1: Dùng `Counter`\*\***

**```python**

**from collections import Counter**

**def most\_frequent\_numbers(arr):**

**freq = Counter(arr)**

**max\_freq = max(freq.values()) # tần suất lớn nhất**

**return [num for num, count in freq.items() if count == max\_freq]**

**modes = most\_frequent\_numbers(arr)**

**print("m) Các số xuất hiện nhiều lần nhất:", modes)**

**```**

**\*\*Cách 2: Tự dùng dict + tìm max\*\***

**```python**

**def most\_frequent\_numbers\_dict(arr):**

**freq = {}**

**for x in arr:**

**freq[x] = freq.get(x, 0) + 1**

**# Tìm tần suất lớn nhất**

**max\_freq = max(freq.values())**

**# Lọc ra tất cả các số có tần suất = max\_freq**

**result = []**

**for k, v in freq.items():**

**if v == max\_freq:**

**result.append(k)**

**return result**

**modes\_dict = most\_frequent\_numbers\_dict(arr)**

**print("m) Các số xuất hiện nhiều lần nhất (dict):", modes\_dict)**

**```**