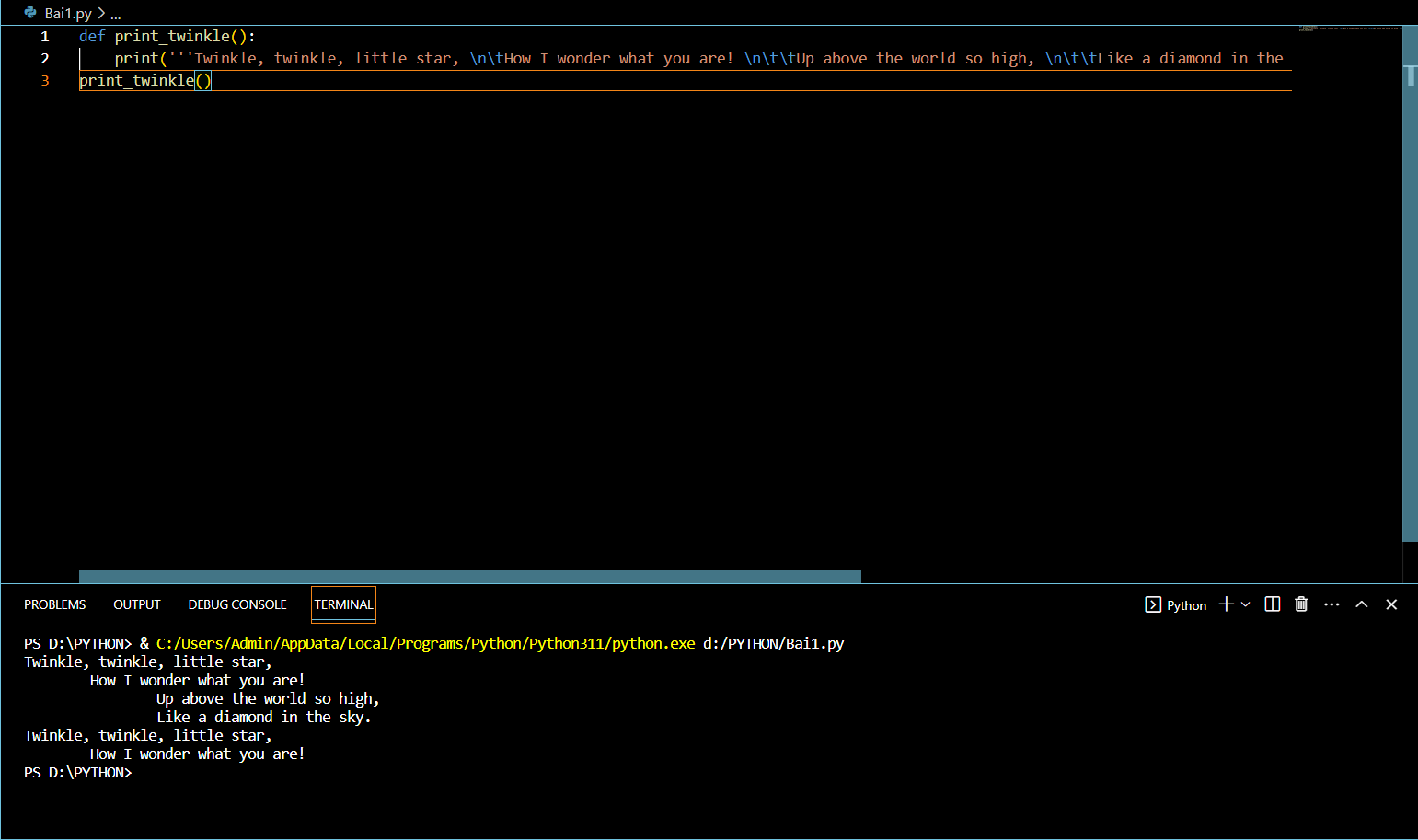
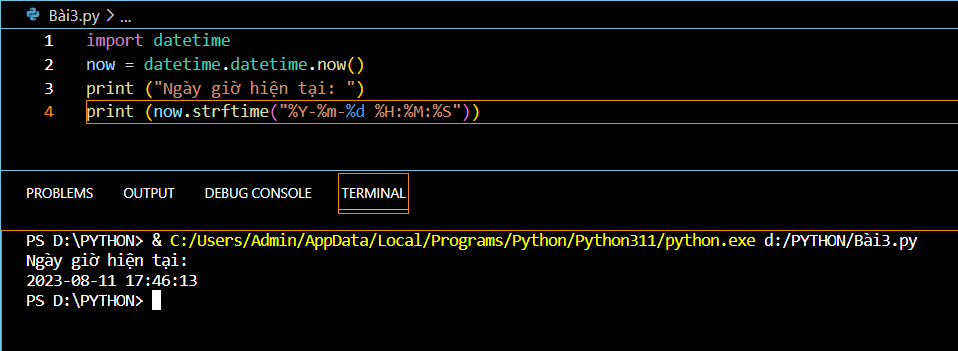
**LAB1**

**Bài 1: Hoàn thành 25/150 bài tập cơ bản ở trang :**

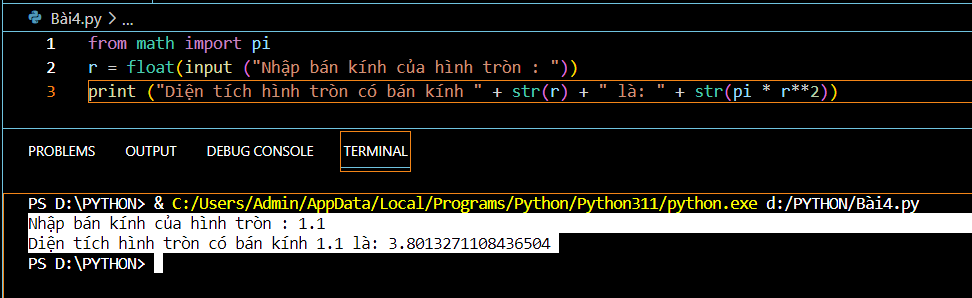
Bài 1:

****Bài 2:

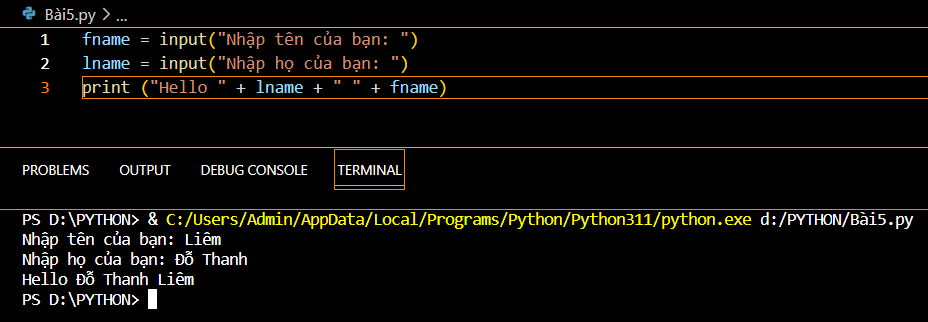
Bài 3:



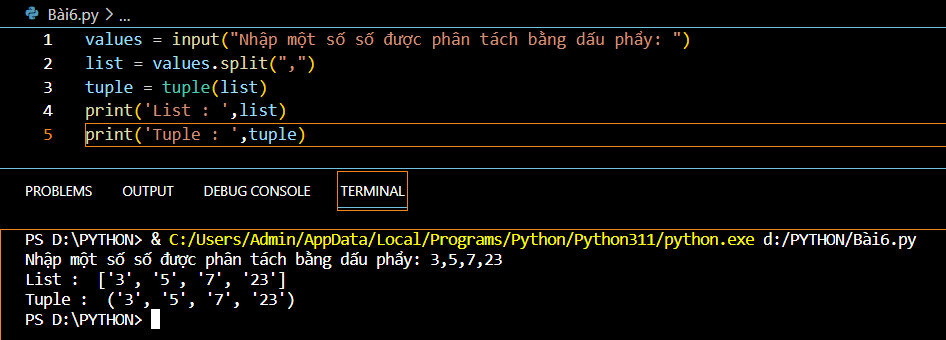
Bài 4:



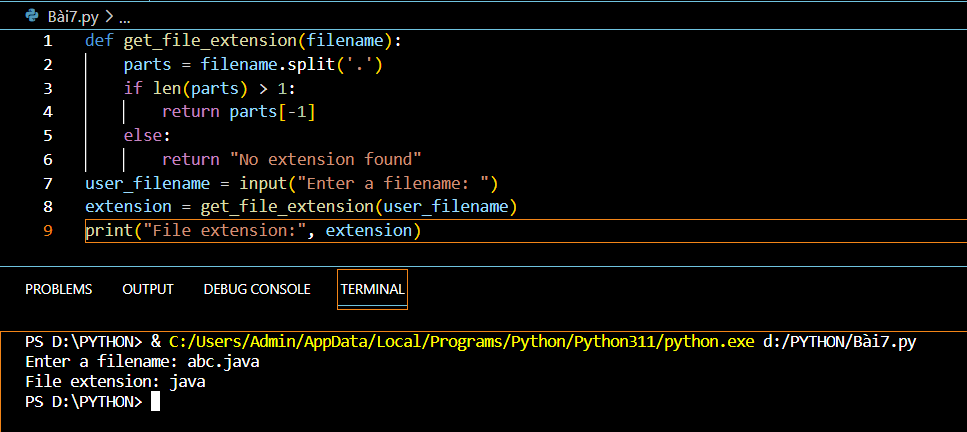
Bài 5:



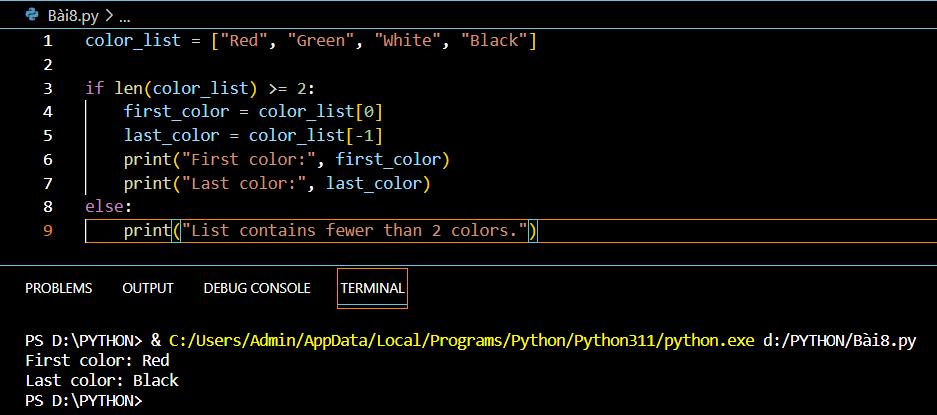
Bài 6:



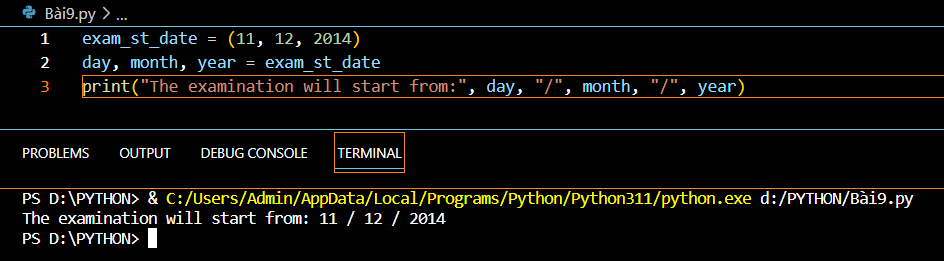
Bài 7:



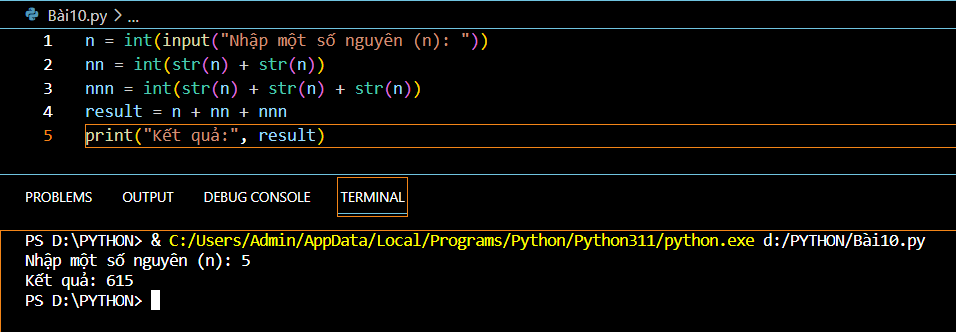
Bài 8:

****

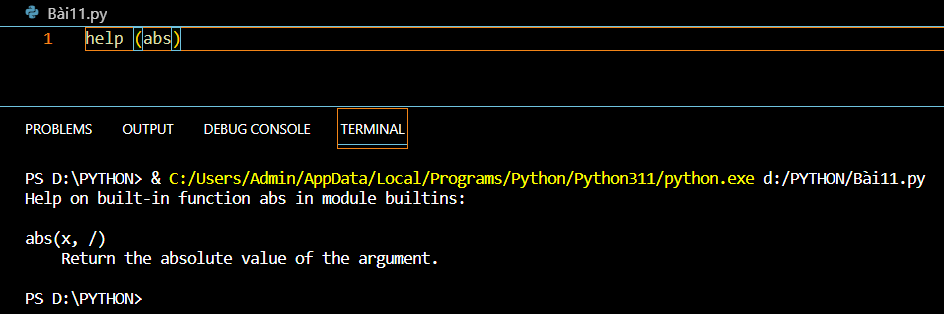
Bài 9:



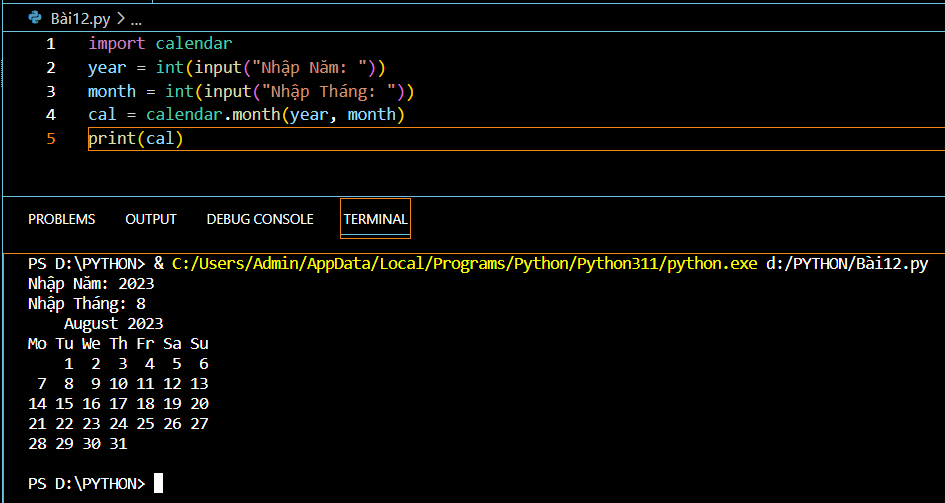
Bài 10:



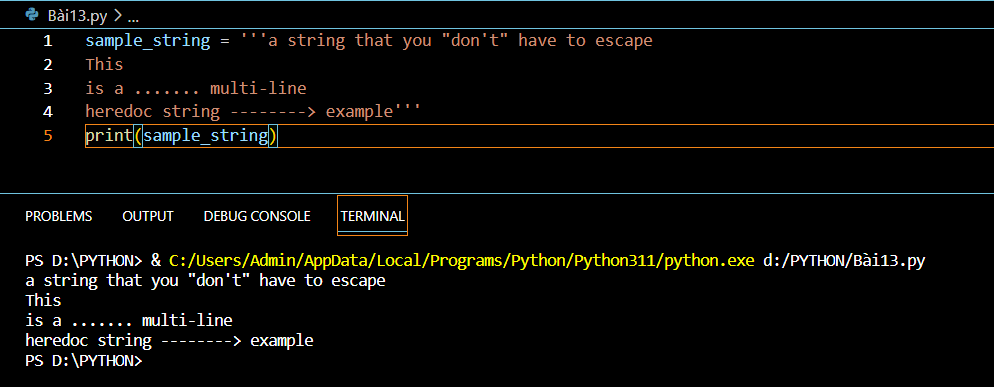
Bài 11:



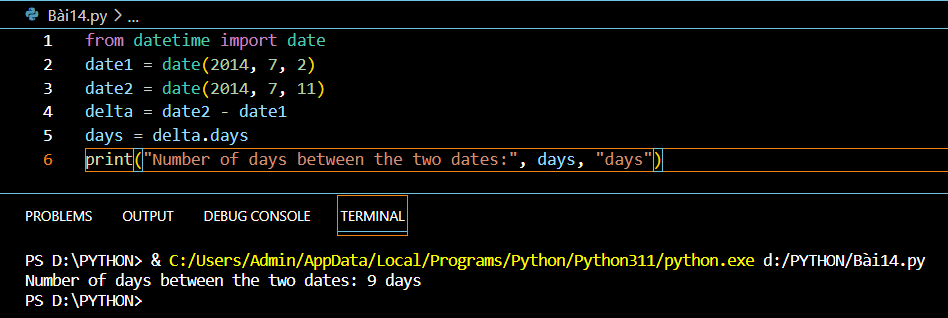
Bài 12:



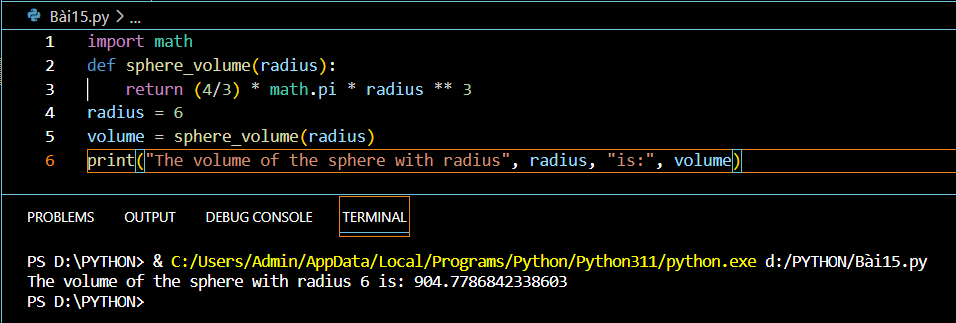
Bài 13:



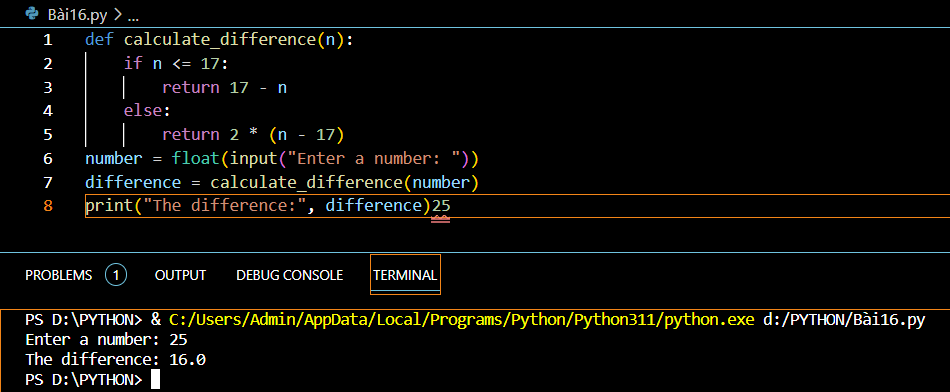
Bài 14:



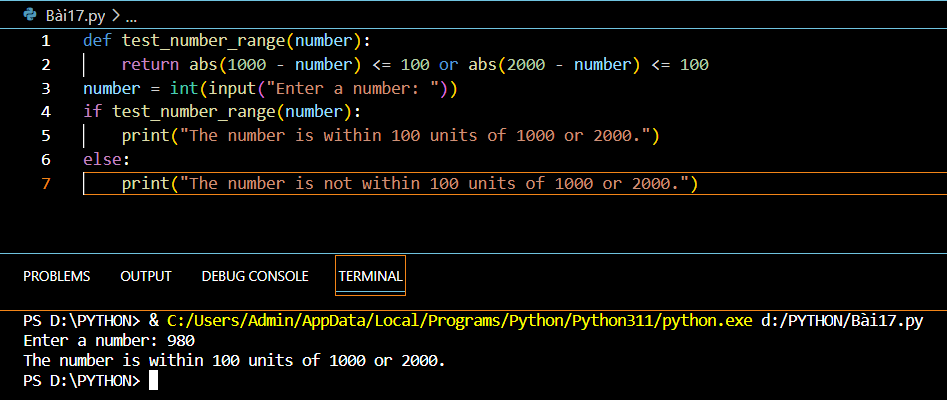
Bài 15:



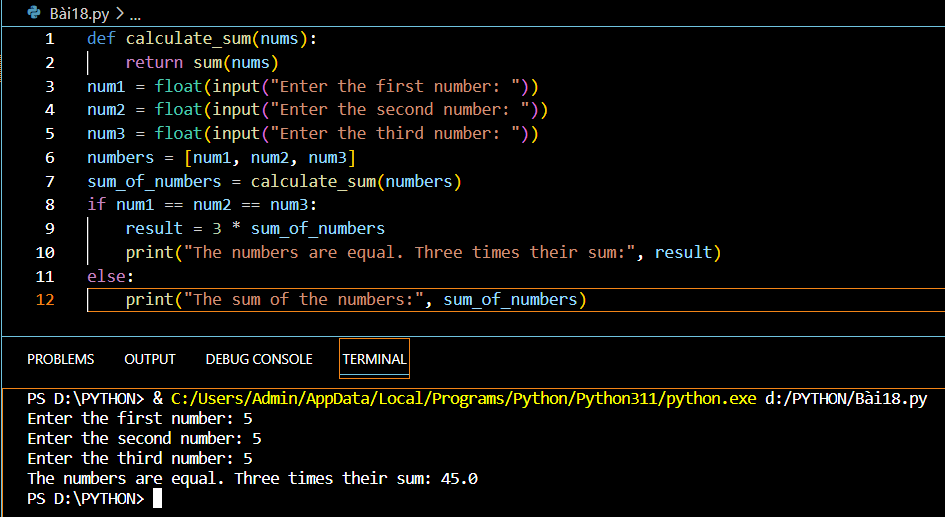
Bài 16:



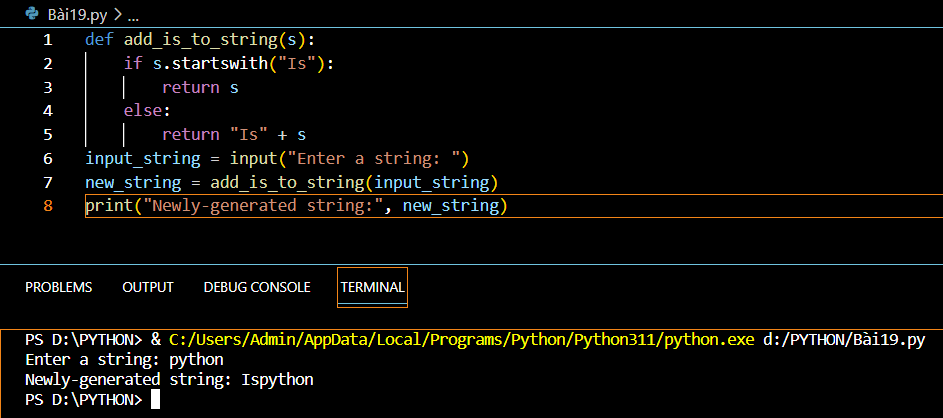
Bài 17:

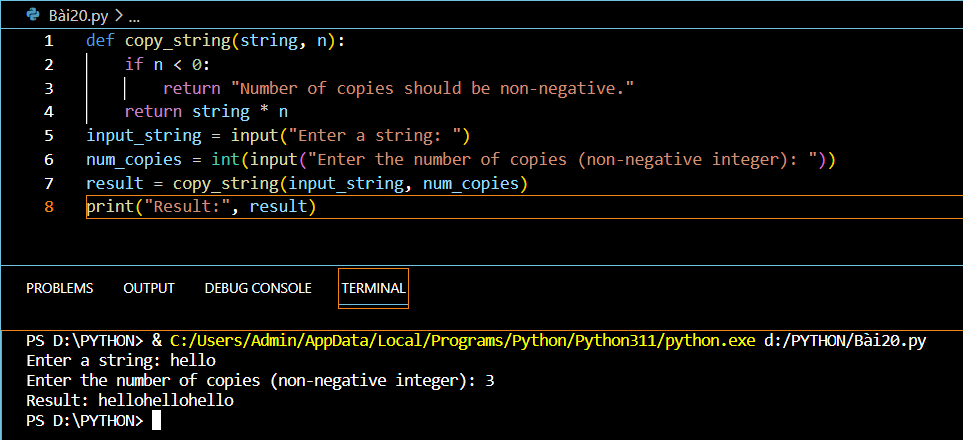


Bài 18:



Bài 19

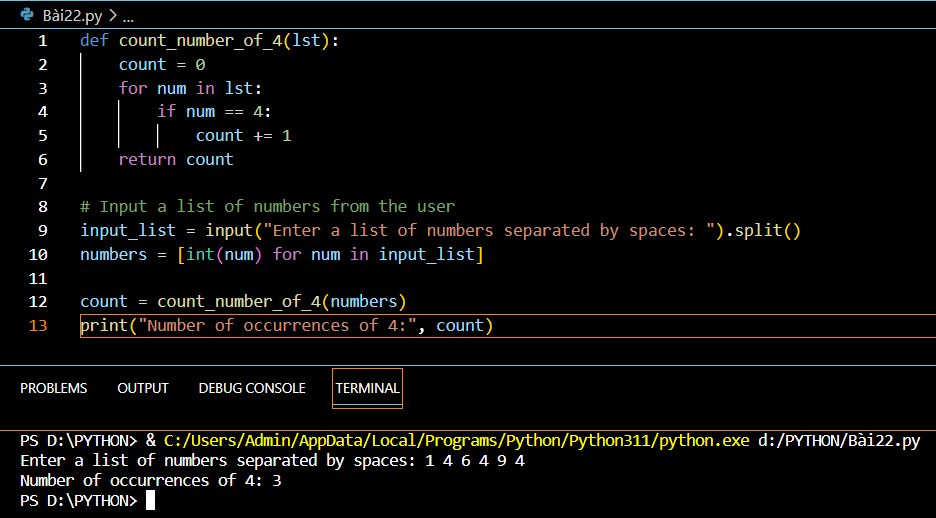


Bài 20:  


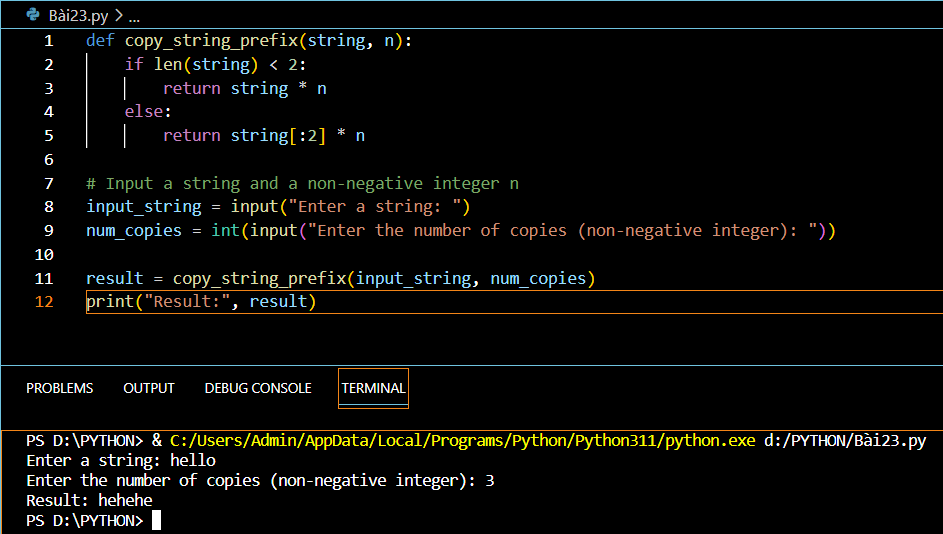
Bài 21:



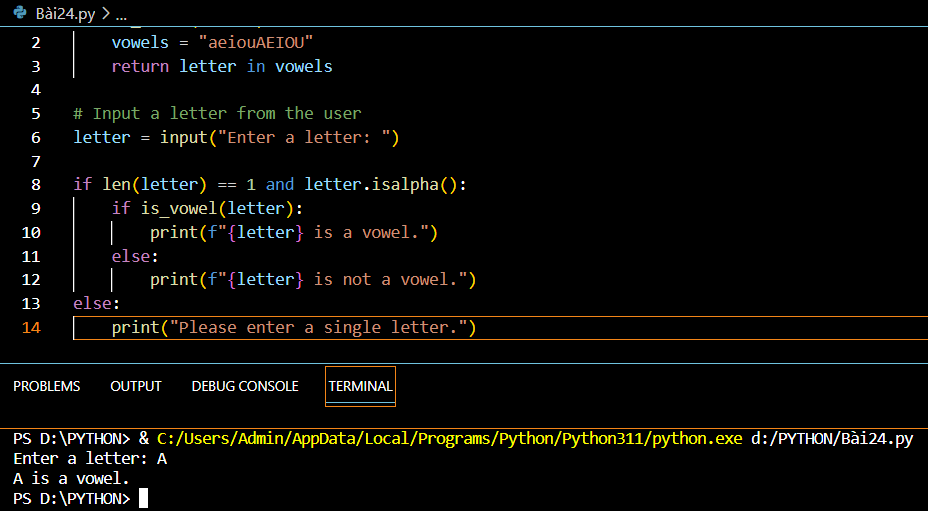
Bài 22:



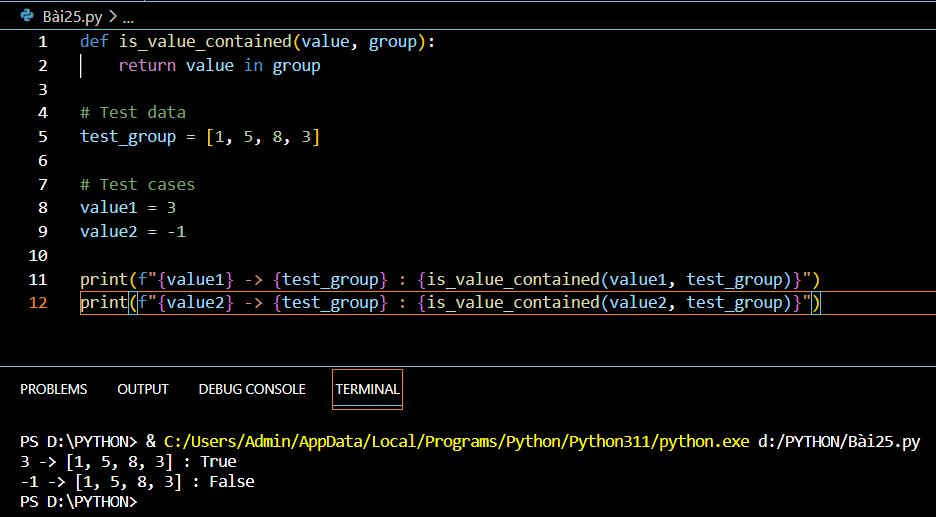
Bài 23:



Bài 24:



Bài 25:



**Bài 2:**

import math

# 1. Tính:

def add(a, b):

    return a + b

def divide(a, b):

    return a / b

def power(a, b):

    return a \*\* b

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

def rectangle\_area(length, width):

    return length \* width

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

def is\_prime(num):

    if num <= 1:

        return False

    for i in range(2, int(math.sqrt(num)) + 1):

        if num % i == 0:

            return False

    return True

def prime\_numbers\_in\_range(start, end):

    primes = [num for num in range(start, end + 1) if is\_prime(num)]

    return primes

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

def is\_perfect\_square(n):

    return math.isqrt(n) \*\* 2 == n

def is\_fibonacci(num):

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

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

def fibonacci\_recursive(n):

    if n <= 0:

        return 0

    elif n == 1:

        return 1

    else:

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

def fibonacci\_iterative(n):

    fib = [0, 1]

    for i in range(2, n + 1):

        fib.append(fib[i - 1] + fib[i - 2])

    return fib[n]

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

def sum\_fibonacci\_recursive(n):

    if n <= 0:

        return 0

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

def sum\_fibonacci\_iterative(n):

    fib\_sum = 0

    for i in range(n + 1):

        fib\_sum += fibonacci\_iterative(i)

    return fib\_sum

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

def sum\_square\_root(n):

    total = 0

    for i in range(1, n + 1):

        total += math.sqrt(i)

    return total

# 8. Giải phương trình bậc 2: ax^2 + bx + c = 0

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

    delta = b \*\* 2 - 4 \* a \* c

    if delta < 0:

        return "No real roots"

    elif delta == 0:

        root = -b / (2 \* a)

        return root

    else:

        root1 = (-b + math.sqrt(delta)) / (2 \* a)

        root2 = (-b - math.sqrt(delta)) / (2 \* a)

        return root1, root2

# 9. Tính n!

def factorial(n):

    if n == 0:

        return 1

    return n \* factorial(n - 1)

# 10. In \* dạng tam giác dưới

def print\_triangle(rows):

    for i in range(1, rows + 1):

        print("\*" \* i)

# 11. Đổi giờ - phút - giây

def seconds\_to\_time(seconds):

    hours = seconds // 3600

    remaining\_seconds = seconds % 3600

    minutes = remaining\_seconds // 60

    seconds = remaining\_seconds % 60

    return f"{hours}:{minutes}:{seconds}"

# 12. Cho một mảng số nguyên

arr = [2, 5, 8, 13, 21, 34, 55, 89, 144]

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

odd\_non\_divisible\_by\_5 = [num for num in arr if num % 2 != 0 and num % 5 != 0]

print("a)", odd\_non\_divisible\_by\_5)

# b) Xuất tất cả các số Fibonacci

def is\_perfect\_square(n):

    return math.isqrt(n) \*\* 2 == n

def is\_fibonacci(num):

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

fibonacci\_nums = [num for num in arr if is\_fibonacci(num)]

print("b)", fibonacci\_nums)

# c) Tìm số nguyên tố lớn nhất

def is\_prime(num):

    if num <= 1:

        return False

    for i in range(2, int(math.sqrt(num)) + 1):

        if num % i == 0:

            return False

    return True

prime\_max = max([num for num in arr if is\_prime(num)])

print("c)", prime\_max)

# d) Tìm số Fibonacci bé nhất

fibonacci\_min = min([num for num in arr if is\_fibonacci(num)])

print("d)", fibonacci\_min)

# e) Tính trung bình các số lẻ

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

average\_odd = sum(odd\_numbers) / len(odd\_numbers)

print("e)", average\_odd)

# f) Tính tích các phần tử là số lẻ không chia hết cho 3 trong mảng

odd\_not\_divisible\_by\_3 = [num for num in arr if num % 2 != 0 and num % 3 != 0]

product\_odd\_not\_divisible\_by\_3 = 1

for num in odd\_not\_divisible\_by\_3:

    product\_odd\_not\_divisible\_by\_3 \*= num

print("f)", product\_odd\_not\_divisible\_by\_3)

# g) Đổi chỗ 2 phần tử của danh sách, đầu vào là 2 vị trí cần đổi chỗ

def swap\_elements(lst, pos1, pos2):

    lst[pos1], lst[pos2] = lst[pos2], lst[pos1]

arr\_copy = arr.copy()

swap\_elements(arr\_copy, 1, 3)

print("g)", arr\_copy)

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

arr\_reverse = arr.copy()

arr\_reverse.reverse()

print("h)", arr\_reverse)

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

unique\_numbers = list(set(arr))

unique\_numbers.sort()

second\_largest = unique\_numbers[-2]

print("i)", second\_largest)

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

def sum\_digits(number):

    return sum(int(digit) for digit in str(number))

total\_digit\_sum = sum(sum\_digits(num) for num in arr)

print("j)", total\_digit\_sum)

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

def count\_occurrences(numbers, target):

count = 0

for num in numbers:

if num == target:

count += 1

return count

# Example usage

my\_list = [2, 3, 4, 2, 5, 2, 6]

target\_number = 2

occurrences = count\_occurrences(my\_list, target\_number)

print(f"The number {target\_number} appears {occurrences} times in the list.")

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

def numbers\_appearing\_n\_times(lst, n):

    return [num for num in lst if lst.count(num) == n]

numbers\_appearing\_2\_times = numbers\_appearing\_n\_times(arr, 2)

print("l)", numbers\_appearing\_2\_times)

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

def most\_common\_numbers(lst):

    from collections import Counter

    counter = Counter(lst)

    max\_count = max(counter.values())

    most\_common = [num for num, count in counter.items() if count == max\_count]

    return most\_common

most\_common\_nums = most\_common\_numbers(arr)

print("m)", most\_common\_nums)