**ASSIGNMENT-2**

**QUE-1**

**def is\_valid(s: str) -> bool:**

**# Create a stack to keep track of opening brackets**

**stack = []**

**# Dictionary to hold the mapping of closing to opening brackets**

**bracket\_map = {')': '(', '}': '{', ']': '['}**

**# Traverse each character in the string**

**for char in s:**

**if char in bracket\_map.values(): # if it's one of '(', '{', '['**

**stack.append(char)**

**elif char in bracket\_map.keys(): # if it's one of ')', '}', ']'**

**# Check if the stack is not empty and the top of the stack matches the current closing bracket**

**if stack and stack[-1] == bracket\_map[char]:**

**stack.pop()**

**else:**

**return False**

**else:**

**# In case of any other characters which are not brackets**

**return False**

**# If the stack is empty, all brackets are matched**

**return not stack**

**# Example usage**

**s = "{[()]}"**

**print(is\_valid(s)) # Output: True**

**s = "{[(])}"**

**print(is\_valid(s)) # Output: False**

**que-2**

date = input("Enter the date to be checked (dd/mm/yyyy): ")  
c = date.split("/")  
b = list(map(int, c))  
input\_year = b[2]  
  
*# Check if the input year is a leap year*if input\_year % 4 == 0:  
 if input\_year % 100 == 0:  
 if input\_year % 400 == 0:  
 print("%d is a Leap Year" % input\_year)  
 else:  
 print("%d is not a Leap Year" % input\_year)  
 else:  
 print("%d is a Leap Year" % input\_year)  
else:  
 print("%d is not a Leap Year" % input\_year)  
  
*# Find the previous and next leap year*def find\_previous\_leap\_year(year):  
 while True:  
 year -= 1  
 if year % 4 == 0 and (year % 100 != 0 or year % 400 == 0):  
 return year  
  
def find\_next\_leap\_year(year):  
 while True:  
 year += 1  
 if year % 4 == 0 and (year % 100 != 0 or year % 400 == 0):  
 return year  
  
if input\_year % 4 == 0 and (input\_year % 100 != 0 or input\_year % 400 == 0):  
 print("Next leap year:", find\_next\_leap\_year(input\_year))  
else:  
 print("Previous leap year:", find\_previous\_leap\_year(input\_year))  
 print("Next leap year:", find\_next\_leap\_year(input\_year))

**que-3**

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

**"""Check if a number is prime."""**

**if n <= 1:**

**return False**

**if n == 2:**

**return True**

**if n % 2 == 0:**

**return False**

**for i in range(3, int(n\*\*0.5) + 1, 2):**

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

**return False**

**return True**

**def print\_non\_primes(A, B):**

**"""Print all non-prime numbers between A and B inclusive."""**

**non\_primes = []**

**for num in range(A, B + 1):**

**if not is\_prime(num):**

**non\_primes.append(num)**

**print(", ".join(map(str, non\_primes)))**

**# Sample Input**

**A = 12**

**B = 19**

**# Output Non-Prime Numbers between A and B**

**print\_non\_primes(A, B)**

**que-4**

def print\_pattern(n):  
 *"""Prints the specified pattern up to n lines."""* for i in range(1, n + 1):  
 for j in range(1, i + 1):  
 print(j, end=' ')  
 print() *# Move to the next line after each row  
  
# Specify the number of lines for the pattern*n = 5  
print\_pattern(n)

**que-5**

**def get\_total\_balance():**

**# Create an empty dictionary to hold the denomination and number of notes**

**denominations = {}**

**# Loop to get the input from the user**

**for i in range(1, 5):**

**denomination = int(input(f"Enter the {i}st Denomination: "))**

**number\_of\_notes = int(input(f"Enter the {i}st Denomination number of notes: "))**

**denominations[denomination] = number\_of\_notes**

**# Calculate the total balance**

**total\_balance = sum(denom \* count for denom, count in denominations.items())**

**# Print the total available balance**

**print(f"\nTotal Available Balance in ATM: {total\_balance}")**

**# Call the function to execute the program**

**get\_total\_balance()**

**que-6**

def find\_max\_of\_three():  
 *# Prompt the user to enter three numbers* num1 = float(input("Enter the first number: "))  
 num2 = float(input("Enter the second number: "))  
 num3 = float(input("Enter the third number: "))  
  
 *# Find the maximum of the three numbers* if num1 >= num2 and num1 >= num3:  
 max\_num = num1  
 elif num2 >= num1 and num2 >= num3:  
 max\_num = num2  
 else:  
 max\_num = num3  
  
 *# Print the maximum number* print(f"The maximum of the three numbers is: {max\_num}")  
  
  
*# Call the function to execute the program*find\_max\_of\_three()

**que-7**

def fibonacci(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  
  
  
*# Input value for N*N = 8  
  
*# Find and print the Nth Fibonacci number*nth\_fibonacci = fibonacci(N)  
print(f"The {N}th Fibonacci number is: {nth\_fibonacci}")

**que-8**

**def add(x, y):**

**"""This function adds two numbers."""**

**return x + y**

**def subtract(x, y):**

**"""This function subtracts two numbers."""**

**return x - y**

**def multiply(x, y):**

**"""This function multiplies two numbers."""**

**return x \* y**

**def divide(x, y):**

**"""This function divides two numbers."""**

**if y == 0:**

**return "Error! Division by zero."**

**return x / y**

**def calculator():**

**"""This function provides a menu to perform basic arithmetic operations."""**

**print("Select operation:")**

**print("1. Add")**

**print("2. Subtract")**

**print("3. Multiply")**

**print("4. Divide")**

**while True:**

**# Take input from the user**

**choice = input("Enter choice(1/2/3/4): ")**

**# Check if choice is one of the four options**

**if choice in ('1', '2', '3', '4'):**

**num1 = float(input("Enter first number: "))**

**num2 = float(input("Enter second number: "))**

**if choice == '1':**

**print(f"The result is: {add(num1, num2)}")**

**elif choice == '2':**

**print(f"The result is: {subtract(num1, num2)}")**

**elif choice == '3':**

**print(f"The result is: {multiply(num1, num2)}")**

**elif choice == '4':**

**print(f"The result is: {divide(num1, num2)}")**

**else:**

**print("Invalid input")**

**# Ask if the user wants to perform another calculation**

**next\_calculation = input("Do you want to perform another calculation? (yes/no): ")**

**if next\_calculation.lower() != 'yes':**

**break**

**# Run the calculator**

**calculator()**

**que-9**

**def check\_voting\_eligibility():**

**# Prompt the user to enter their age**

**age = int(input("Enter your age: "))**

**# Check if the age is 18 or above**

**if age >= 18:**

**print("You are eligible to vote.")**

**else:**

**print("You are not eligible to vote. You need to be at least 18 years old.")**

**# Run the function to check voting eligibility**

**check\_voting\_eligibility()**

**que-10**

**def reverse\_word(word):**

**reversed\_word = ""**

**# Loop through the word in reverse order**

**for i in range(len(word) - 1, -1, -1):**

**reversed\_word += word[i]**

**return reversed\_word**

**# Input word from the user**

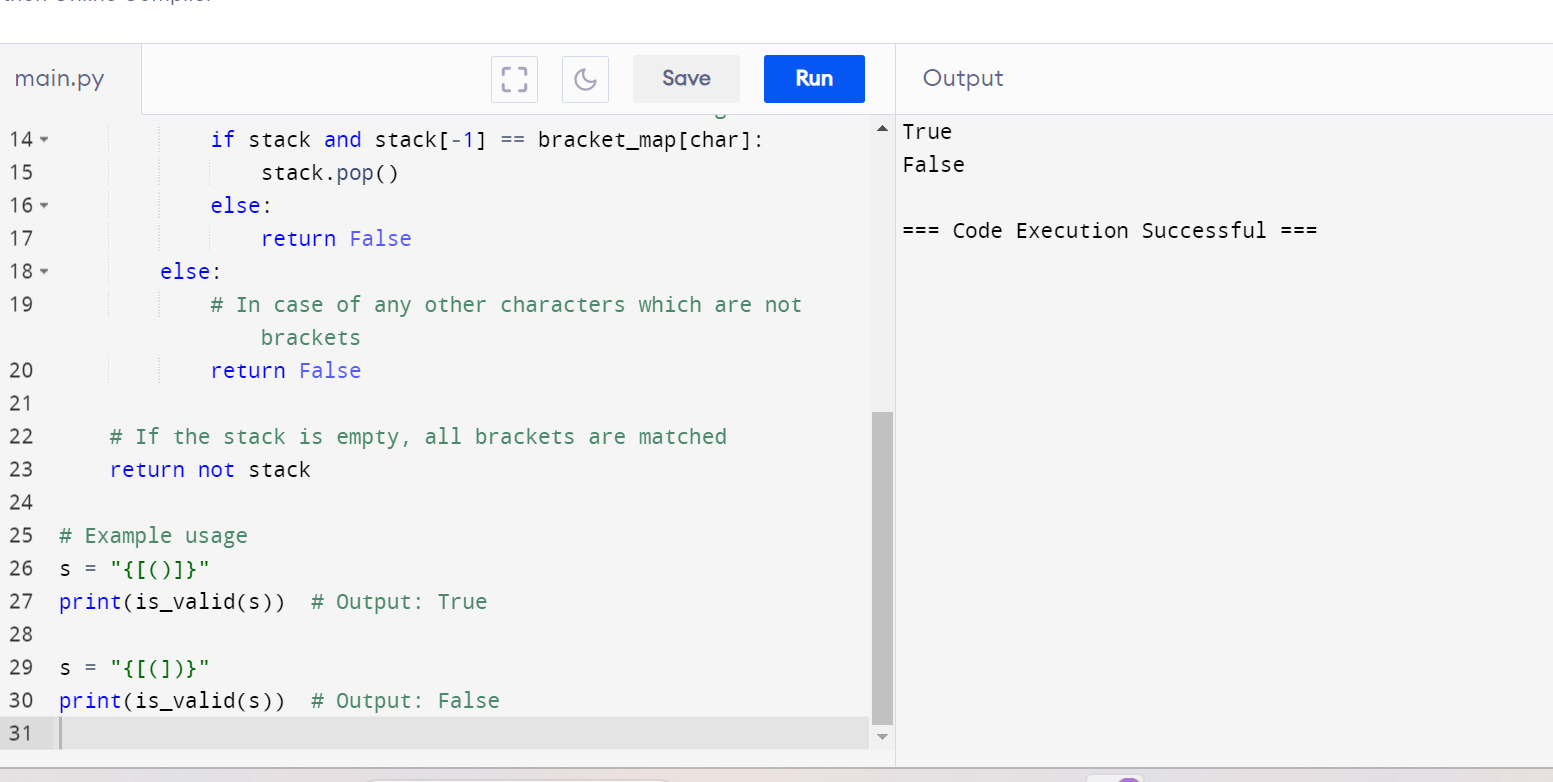
**word = input("Enter a word to reverse: ")**

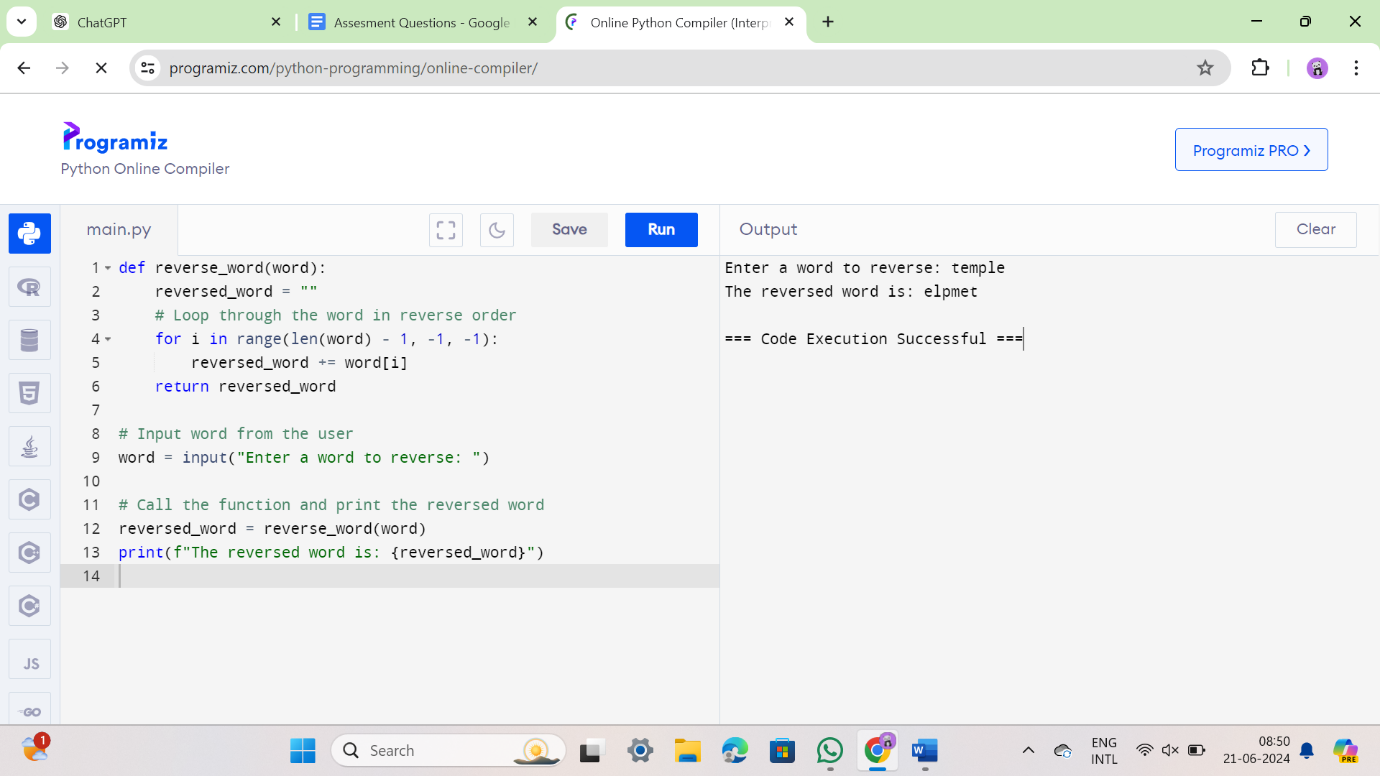
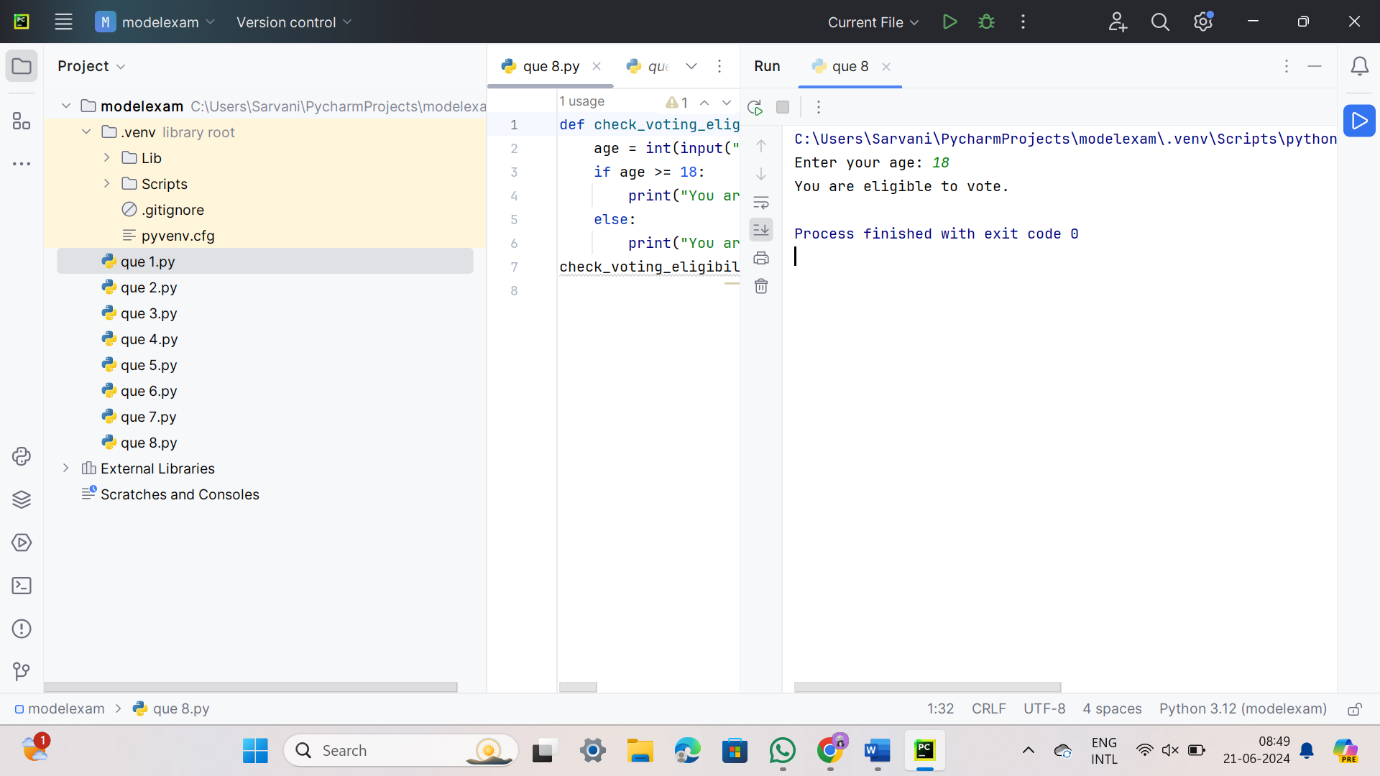
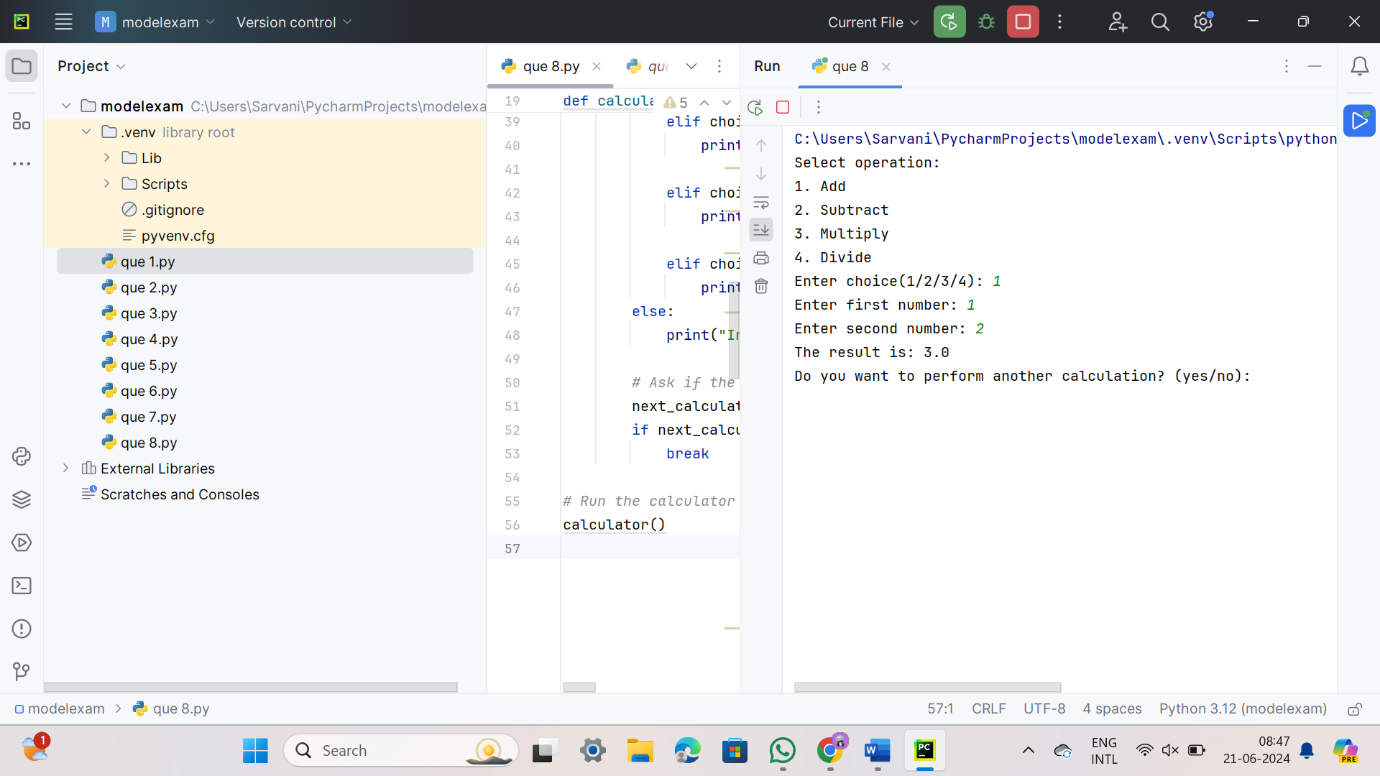
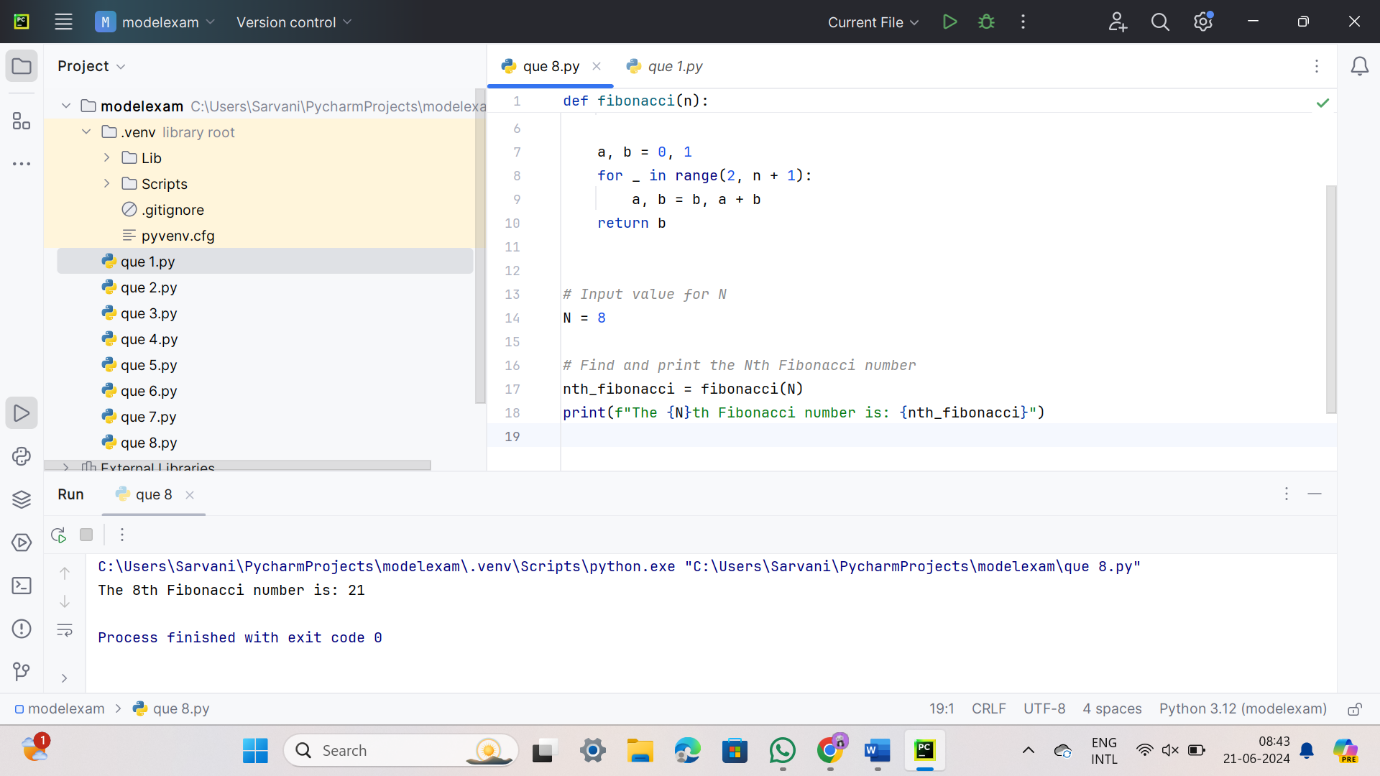
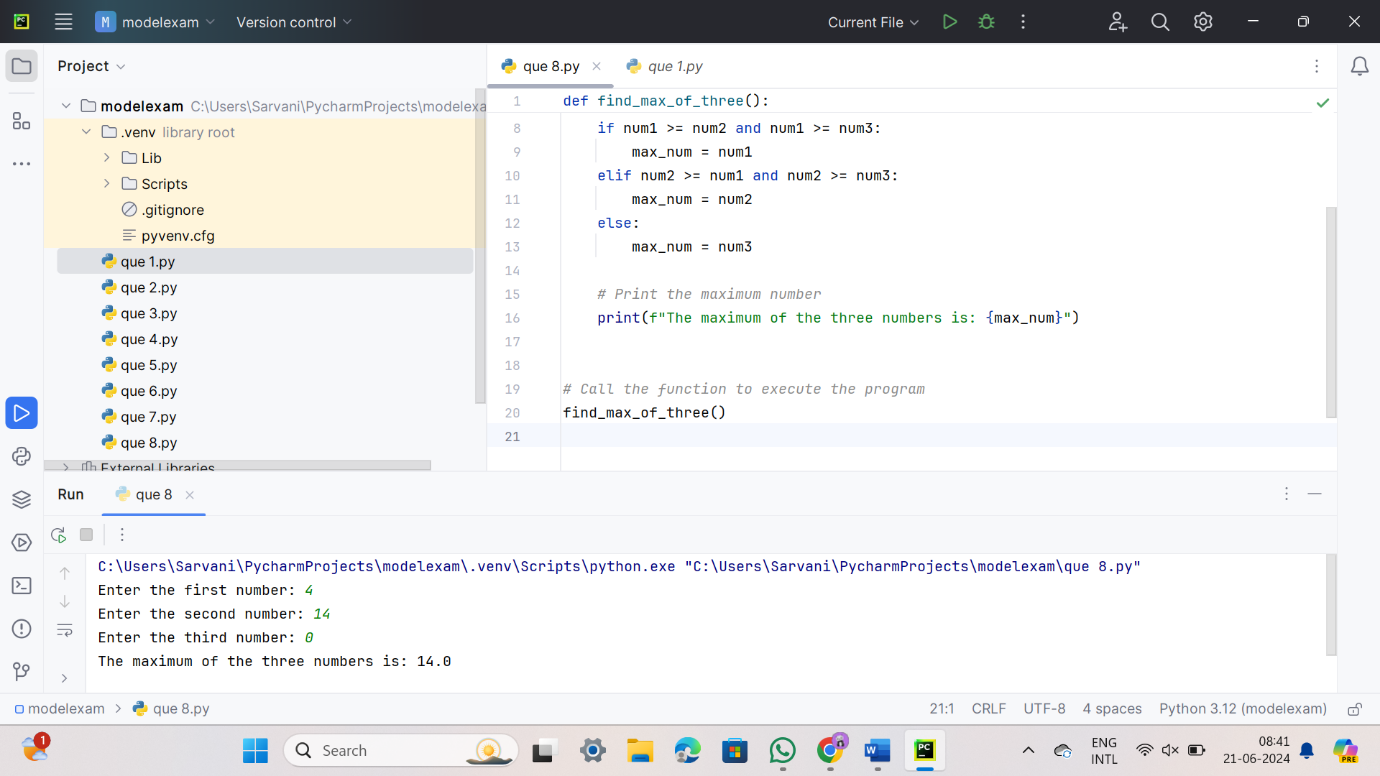
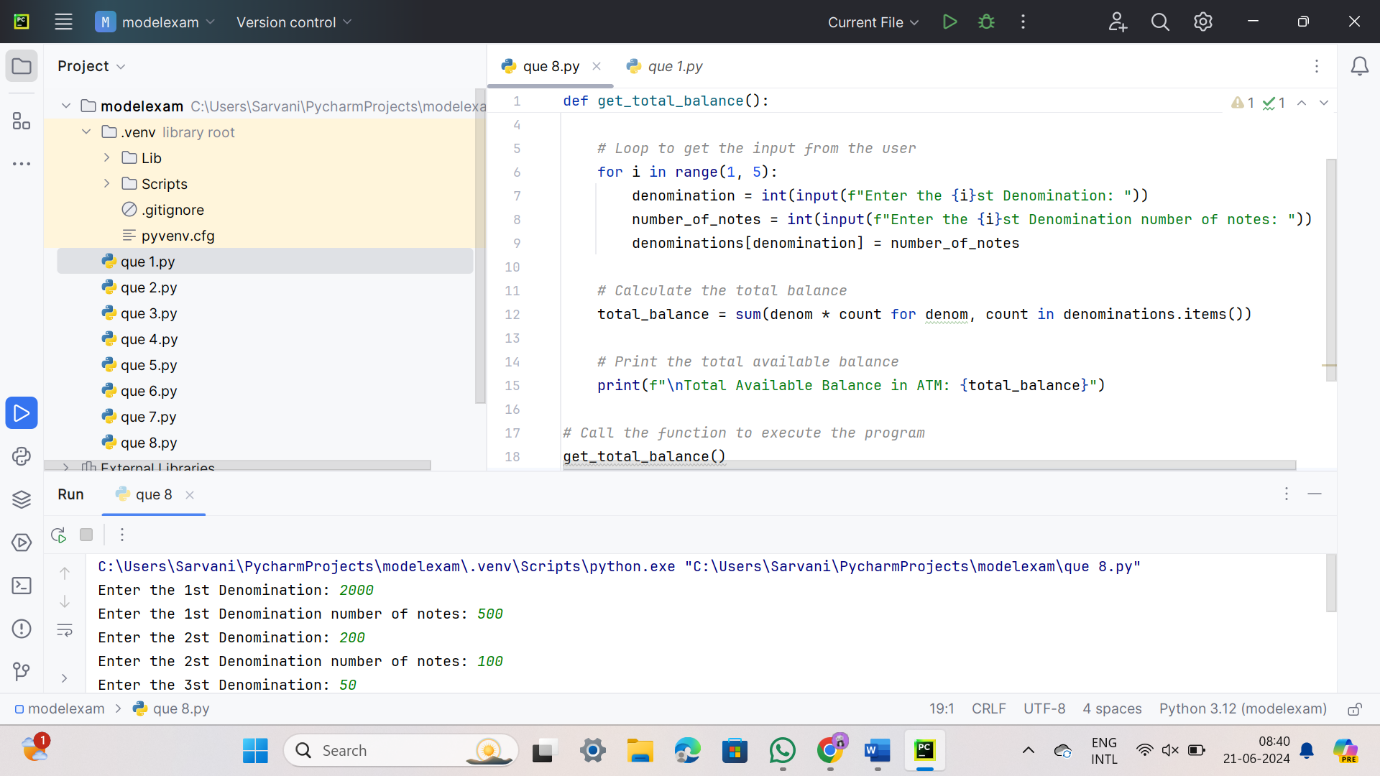
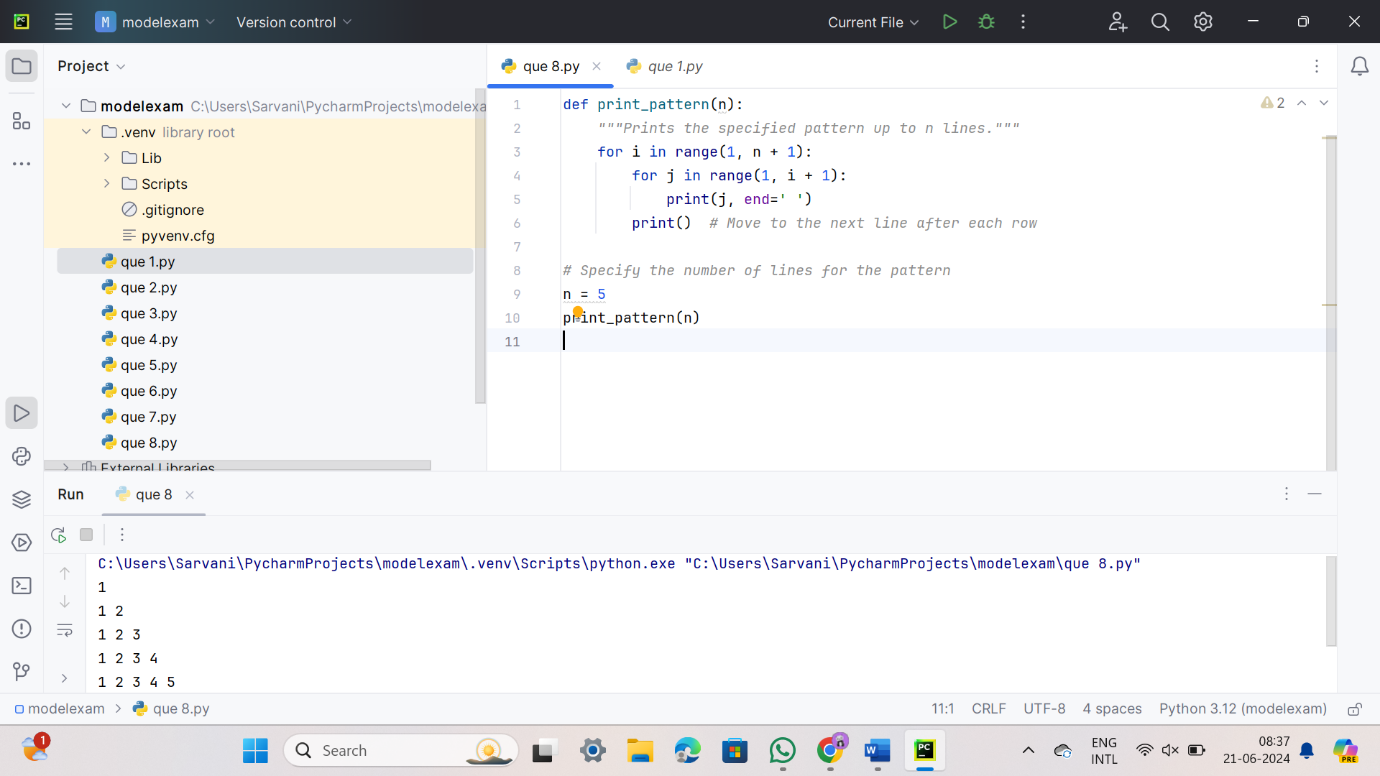
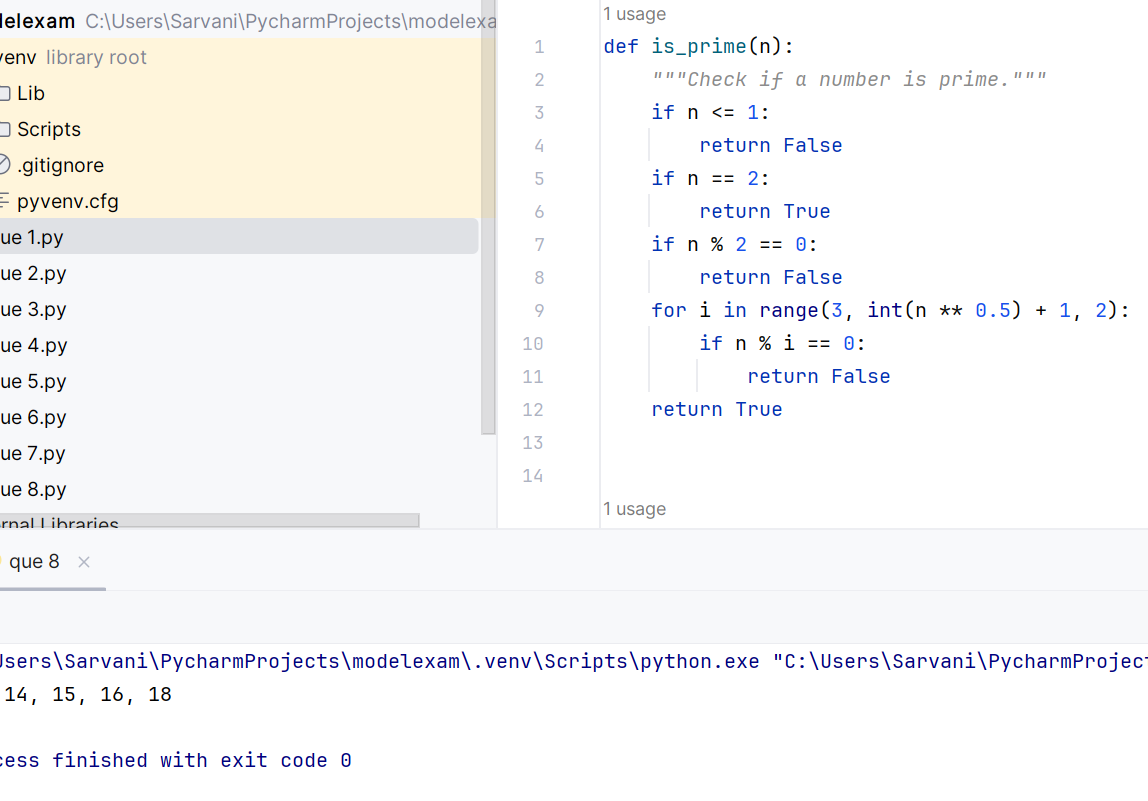
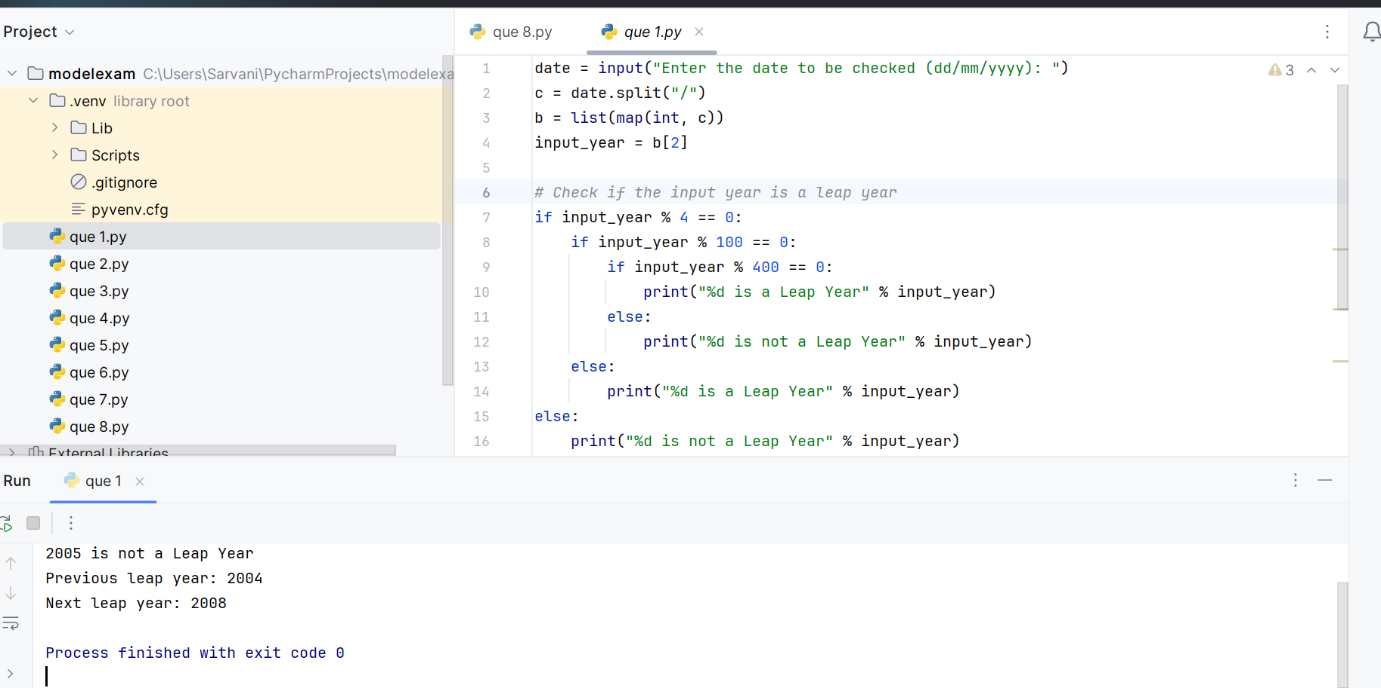
**# Call the function and print the reversed word**

**reversed\_word = reverse\_word(word)**

**print(f"The reversed word is: {reversed\_word}")**

**outputs for que1-10**

****

****