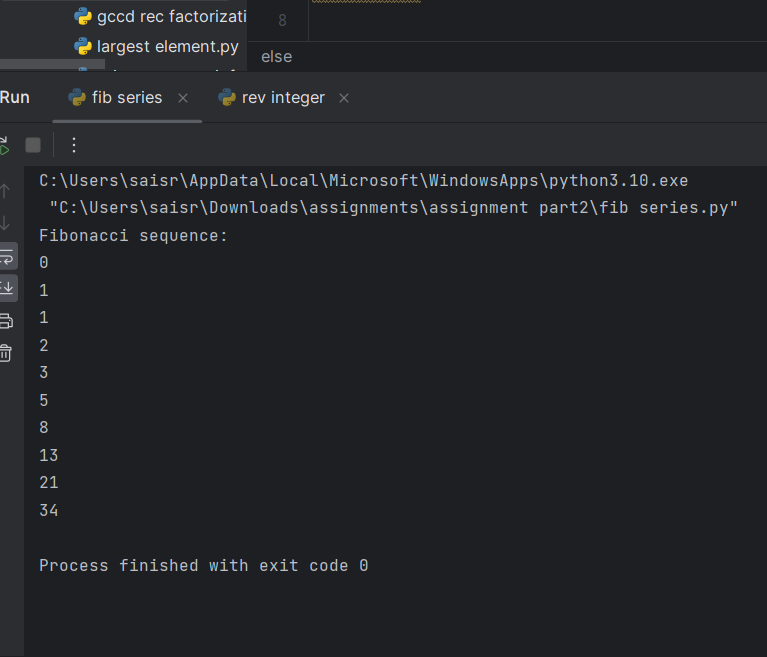
ASSIGNMENT-1

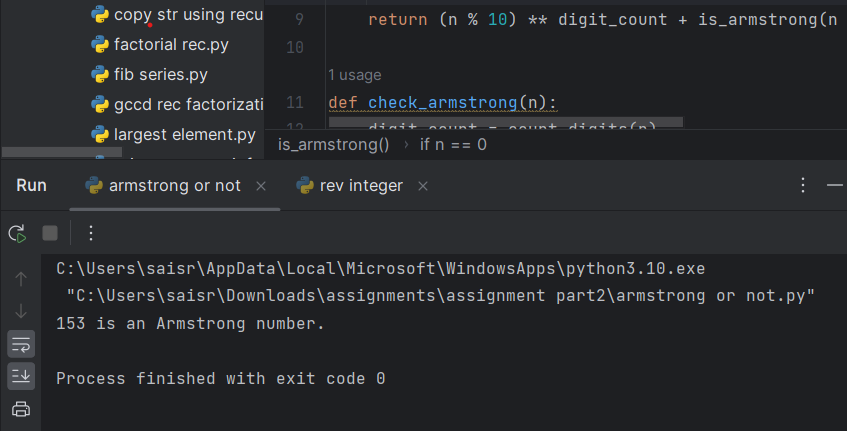
1. Write a program to Print Fibonacci Series using recursion.

def recur\_fibo(n):  
 if n <= 1:  
 return n  
 else:  
 return(recur\_fibo(n-1) + recur\_fibo(n-2))  
  
nterms = 10  
  
if nterms <= 0:  
 print("Plese enter a positive integer")  
else:  
 print("Fibonacci sequence:")  
 for i in range(nterms):  
 print(recur\_fibo(i))



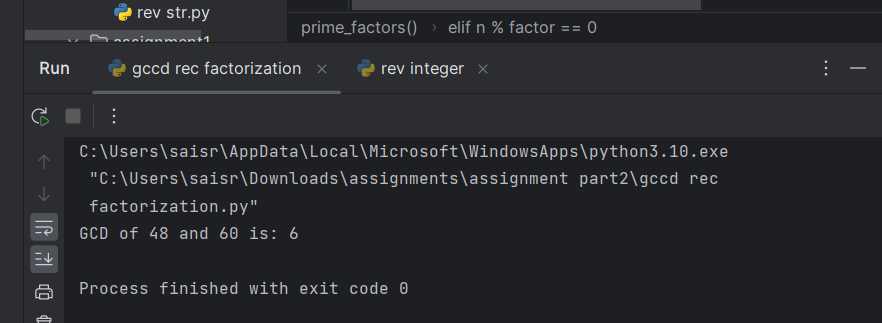
1. Write a program to check the given no is Armstrong or not using recursive function.

def count\_digits(n):  
 if n == 0:  
 return 0  
 return 1 + count\_digits(n // 10)  
  
def is\_armstrong(n, digit\_count):  
 if n == 0:  
 return 0  
 return (n % 10) \*\* digit\_count + is\_armstrong(n // 10, digit\_count)  
  
def check\_armstrong(n):  
 digit\_count = count\_digits(n)  
 sum\_of\_powers = is\_armstrong(n, digit\_count)  
 return sum\_of\_powers == n  
  
# Example usage:  
num = 153  
if check\_armstrong(num):  
 print(num, "is an Armstrong number.")  
else:  
 print(num, "is not an Armstrong number.")



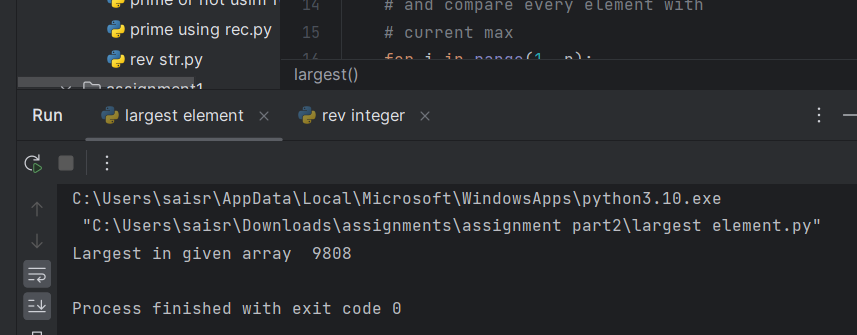
1. Write a program to find the GCD of two numbers using recursive factorization

def prime\_factors(n, factor=2):  
 if n <= 1:  
 return []  
 elif n % factor == 0:  
 return [factor] + prime\_factors(n // factor, factor)  
 else:  
 return prime\_factors(n, factor + 1)  
  
def gcd\_recursive(a, b):  
 if b == 0:  
 return a  
 return gcd\_recursive(b, a % b)  
  
def gcd(a, b):  
 # Get prime factors of both numbers  
 factors\_a = prime\_factors(a)  
 factors\_b = prime\_factors(b)  
  
 # Find common factors  
 common\_factors = set(factors\_a) & set(factors\_b)  
  
 # Calculate GCD by multiplying common factors  
 result = 1  
 for factor in common\_factors:  
 result \*= factor  
  
 return result  
  
# Example usage:  
num1 = 48  
num2 = 60  
print("GCD of", num1, "and", num2, "is:", gcd(num1, num2)) # Output: 12



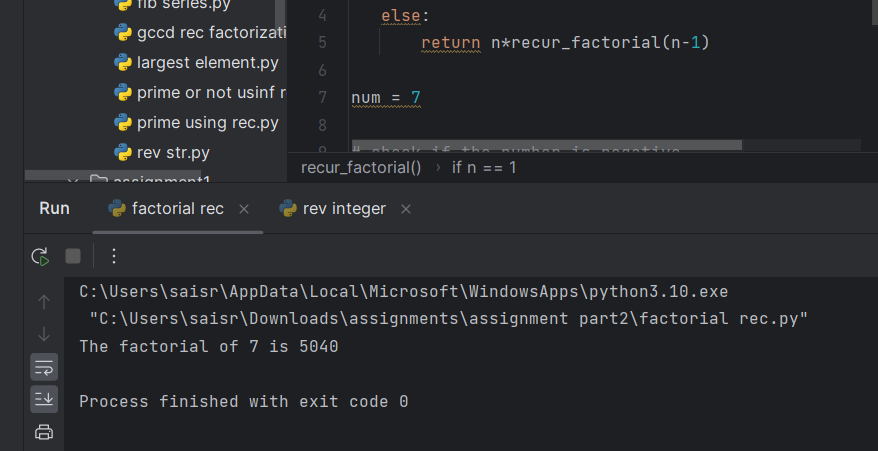
4. Write a program to get the largest element of an array

# Python3 program to find maximum  
# in arr[] of size n  
  
# python function to find maximum  
# in arr[] of size n  
  
  
def largest(arr, n):  
  
 # Initialize maximum element  
 max = arr[0]  
  
 # Traverse array elements from second  
 # and compare every element with  
 # current max  
 for i in range(1, n):  
 if arr[i] > max:  
 max = arr[i]  
 return max  
  
  
# Driver Code  
arr = [10, 324, 45, 90, 9808]  
n = len(arr)  
Ans = largest(arr, n)  
print("Largest in given array ", Ans)



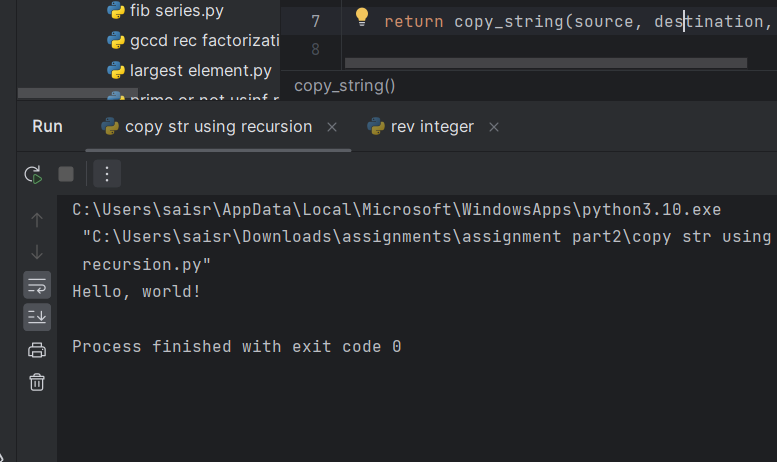
5.Write a program to find the Factorial of a number using recursion.

def recur\_factorial(n):  
 if n == 1:  
 return n  
 else:  
 return n\*recur\_factorial(n-1)  
  
num = 7  
  
# check if the number is negative  
if num < 0:  
 print("Sorry, factorial does not exist for negative numbers")  
elif num == 0:  
 print("The factorial of 0 is 1")  
else:  
 print("The factorial of", num, "is", recur\_factorial(num))



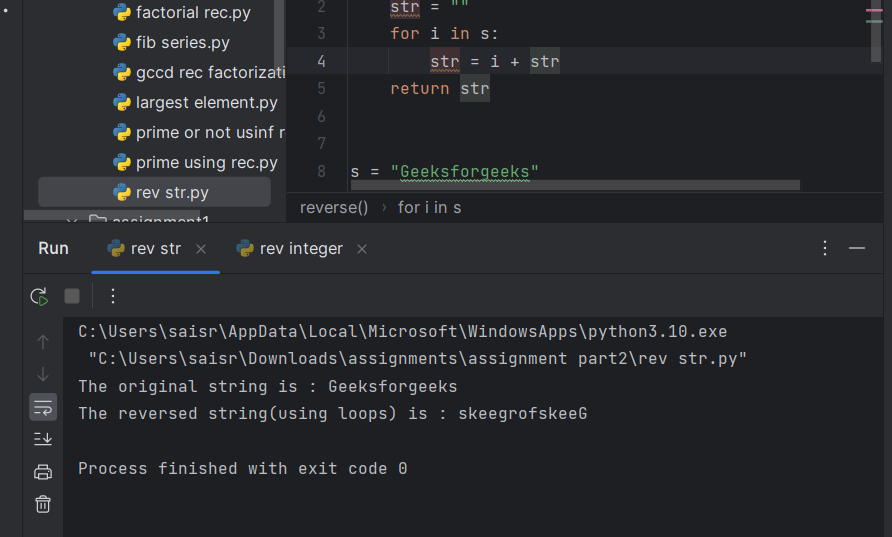
6. Write a program for to copy one string to another using recursion

def copy\_string(source, destination, index=0):  
 if index >= len(source):  
 return destination  
  
 destination += source[index]  
  
 return copy\_string(source, destination, index + 1)  
  
  
source\_string = "Hello, world!"  
destination\_string = ""  
copied\_string = copy\_string(source\_string, destination\_string)  
print(copied\_string)



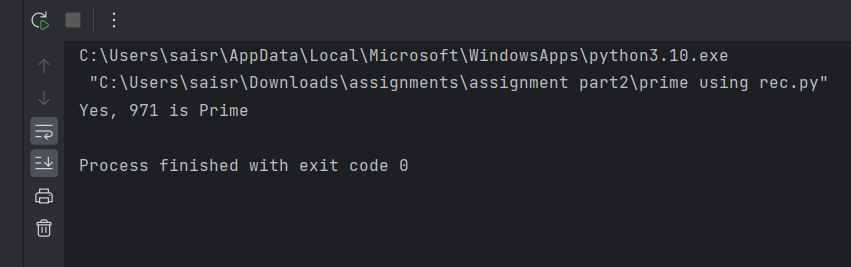
7. Write a program to print the reverse of a string using recursion

def reverse(s):  
 str = ""  
 for i in s:  
 str = i + str  
 return str  
  
  
s = "Geeksforgeeks"  
  
print("The original string is : ", end="")  
print(s)  
  
print("The reversed string(using loops) is : ", end="")  
print(reverse(s))



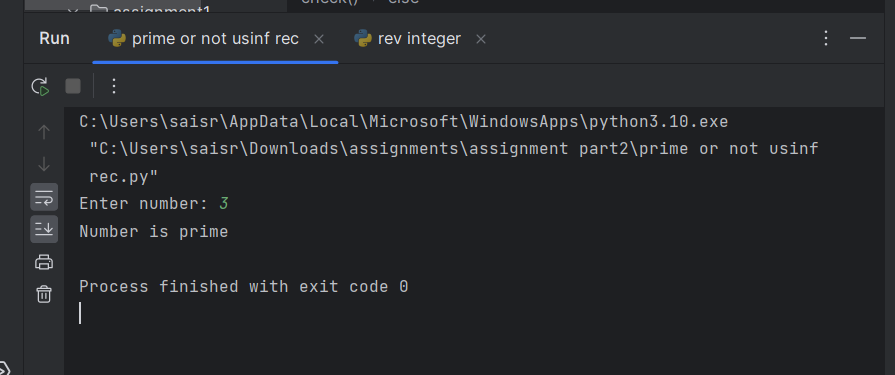
8. Write a program to generate all the prime numbers using recursion

def Prime\_Number(n, i=2):  
 if n == i:  
 return True  
 elif n % i == 0:  
 return False  
 return Prime\_Number(n, i + 1)  
  
  
n = 971  
if Prime\_Number(n):  
 print("Yes,", n, "is Prime")  
else:  
 print("No,", n, "is not a Prime")



9. Write a program to check a number is a prime number or not using recursion.

def check(n, div = None):  
 if div is None:  
 div = n - 1  
 while div >= 2:  
 if n % div == 0:  
 print("Number not prime")  
 return False  
 else:  
 return check(n, div-1)  
 else:  
 print("Number is prime")  
 return 'True'  
n=int(input("Enter number: "))  
check(n)



10.Write a program for to check whether a given String is Palindrome or not using recursion

str\_1 = input ("Enter the string to check if it is a palindrome: ")  
  
str\_1 = str\_1.casefold ()  
  
rev\_str = reversed (str\_1)  
  
if list (str\_1) == list (rev\_str):  
  
 print ("The string is a palindrome.")  
  
else:  
  
 print ("The string is not a palindrome.")

