In [36]:
```
def inputNumber(number):   #this function would take the input and would check the number entered is of non-negative integer value
    while True:            #this loop will execute until the numerical and non-negative integer value is entered
        try:
            userInput = int(input(number))    #type casting the data entered to be of int type
            assert userInput >= 0            # Test if user input entered is >=0
        except (ValueError, AssertionError) as error:   #this exception throws an error to the user in case the value entered is string or non-positive
            print("Not a non-negative integer value. Please Try again!!")
            continue    #executes the loop and asks for the user input untill desired condition is met
        else:
            return userInput                #return the value of the input to the function called
            break                           #break the loop in case the user enters non-negative integer value




def factors(execution_val, checkInput=0):    #function defined which would calculate the factors of the number
    if execution_val == "call_out_function":#checks if the fucntion is called from the main body(outside the function) for the first time
        userInput = inputNumber("Input a number to find its factors ?") #prompted the user to enter the input
    else:                                  #checks if the function is called from the inside the other functions e.g is_prime() and hcf()
        userInput = checkInput          #assigns the value of already inputed data from the previous called function
    list_factor=[]                     #intialized an empty list which would store the result of the factors of a number
    for k in range(1,userInput+1):     #iterating through the elements of the list
        if userInput%k==0:             #used logic to append the element in the list if the entered number is divisible by the Kth element
            list_factor.append(k)

    return userInput, list_factor      #return two values to the function called- which is the data input and the list of factors




def is_prime():                      #function defined which would check if the user input number is prime or not
    userInput = inputNumber("Input a number to check if it is prime or not ?")  #prompted the user to enter the input
    list_factor= []                  #intialized an empty list which would store the factors of the given number from function 'factors'
    list_factor= factors("call_in_function",userInput) #call the factors function and pass argument as call_in_function as the first argument to indicate that factors function is called from inside function and pass the second argument as the data input for this fun

    if len(list_factor[1])==2:         #used logic that depicts that prime number will have count ==2 that is the prime number would only be divisible by 1 and itself
        return userInput, True       #return the user input data and True in case the number is prime
    else:
        return userInput, False        #return the user input data and False in case the number is not prime




def hcf():
    x= inputNumber("Input the first number to calculate its HCF ?") #prompted the user to enter the first number for HCF
    y= inputNumber("Input the second number to calculate HCF?") #prompted the user to enter the second number for HCF
    list_factor_x= []  #initialized the empty list to store factors of first input number i.e x
    list_factor_x= factors("call_in_function",x) #call the factors function and pass argument as call_in_function as the first argument to indicate that factors function is called from inside function and pass the second argument as the data input for this function
    list_factor_y= []  #initialized the empty list to store factors of second input number i.e y
    list_factor_y= factors("call_in_function",y) #call the factors function and pass argument as call_in_function as the first argument to indicate that factors function is called from inside function and pass the second argument as the data input for this function
    list_factor_final=[]  #initialized the empty list to store the common values present in both the factor list of x and y
    hcf = 0
    for l in list_factor_x[1]:  #accessing the second index of the list_factor_x which would contain the factors list of x
        if l in list_factor_y[1]: #accessing the second index of the list_factor_y which would contain the factors list of y
            list_factor_final.append(l) #append the list to the list_factor_final in case the element of list_factor_x is present in list_factor_y
    while True:   #exectue the loop to handle the exception and check for the error in case we need the maximum value of the list which is empty
        try:
            max(list_factor_final)   #check for the maximum value of the list_factor_final list
        except ValueError:     #handle the error here and assigned the value of hcf to 0 so that it does not throws the error
            hcf=0
            break
        else:
            hcf = max(list_factor_final)     #assign the value of the highest element in the list_factor_final in case list is not empty
            break

    return x,y,hcf #return the value of first data input, second data input and the value of hcf obtatined from the fucntion




hold_list=[]   #intialized an empty list which would store the result of the return values from the functions in the form of list
hold_list = factors("call_out_function")  #calling function factors and passing call_out_function to indicate that function is being called from the main body
print(f"Solution 1: factors of {hold_list[0]} is {hold_list[1]}") #printing the value of the userinput and the factors list
if(len(hold_list[1])==0):  #addition check to print the following statement in case the list is empty
    print("The factor list is empty!")
hold_list = is_prime()  #calling function is_prime
print(f"Solution 2: The number {hold_list[0]} is prime: {hold_list[1]}") #printing the value of the userinput and boolean values to check if its prime based on the output of the numbers
hold_list = hcf()    #calling function hcf to find highest commom factor
print(f"Solution 3: The HCF of {hold_list[0]} and {hold_list[1]} is: {hold_list[2]}") #printing the value of first user input, second user input and the HCF value
```

```
Input a number to find its factors ?10
Solution 1: factors of 10 is [1, 2, 5, 10]
Input a number to check if it is prime or not ?7
Solution 2: The number 7 is prime: True
Input the first number to calculate its HCF ?12
Input the second number to calculate HCF?18
Solution 3: The HCF of 12 and 18 is: 6
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: