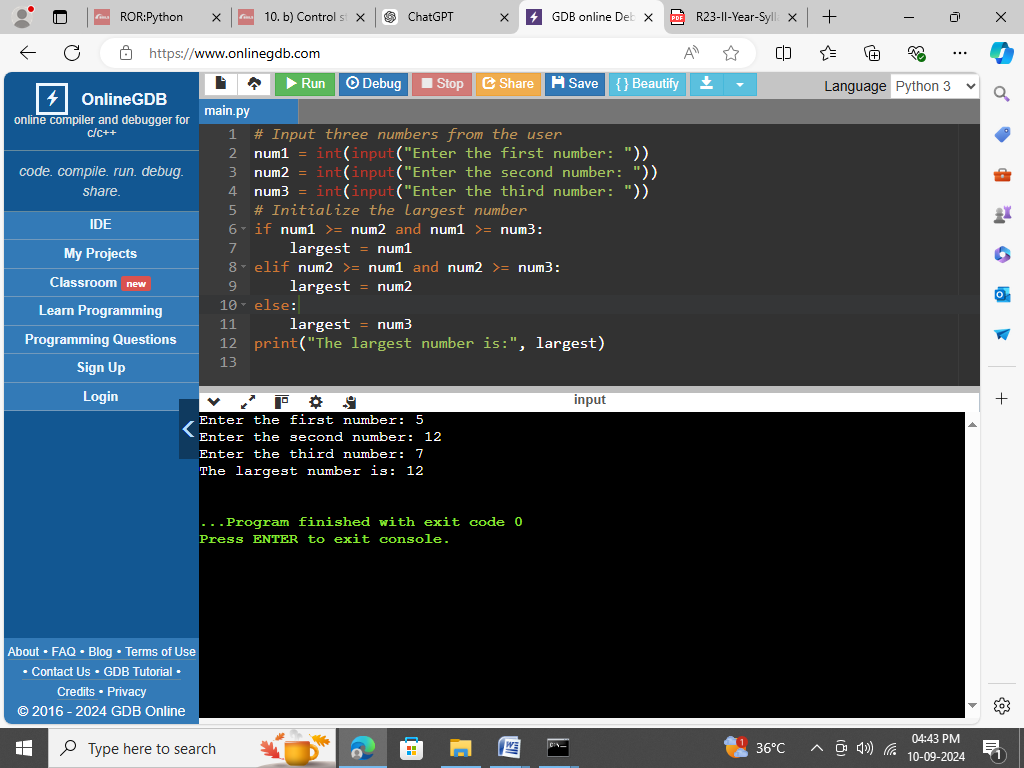
# Aim: Write a program to find the largest element among three Numbers.

<https://onlinegdb.com/s_S888s90>

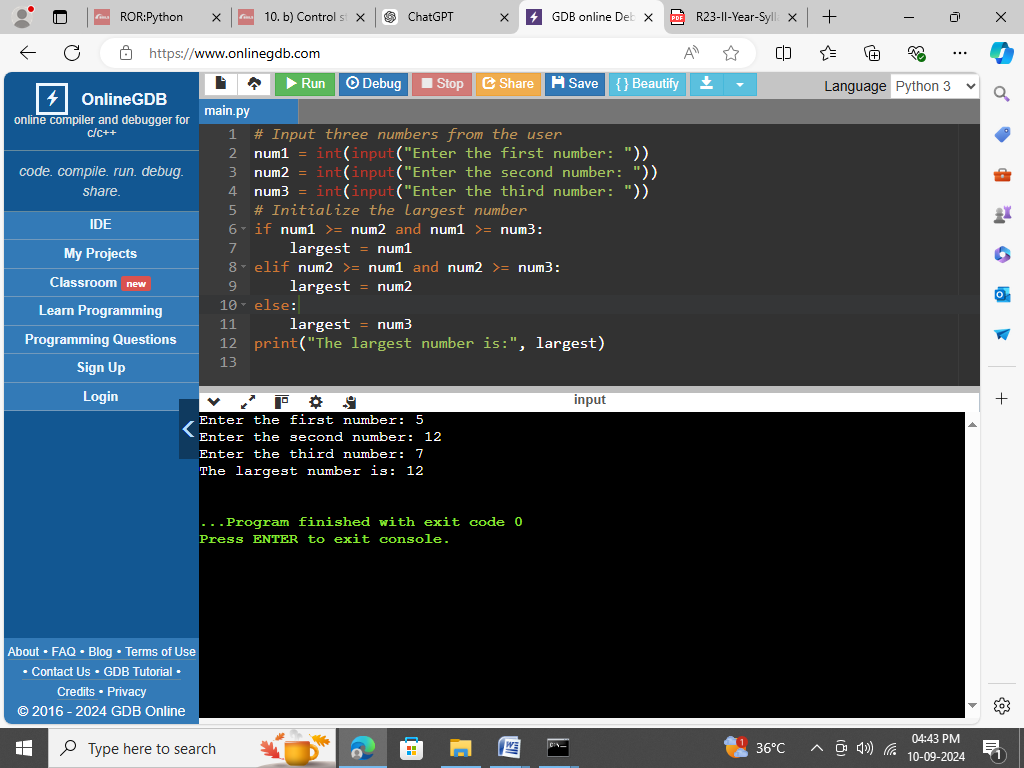
#### Source Code: find\_largest.py



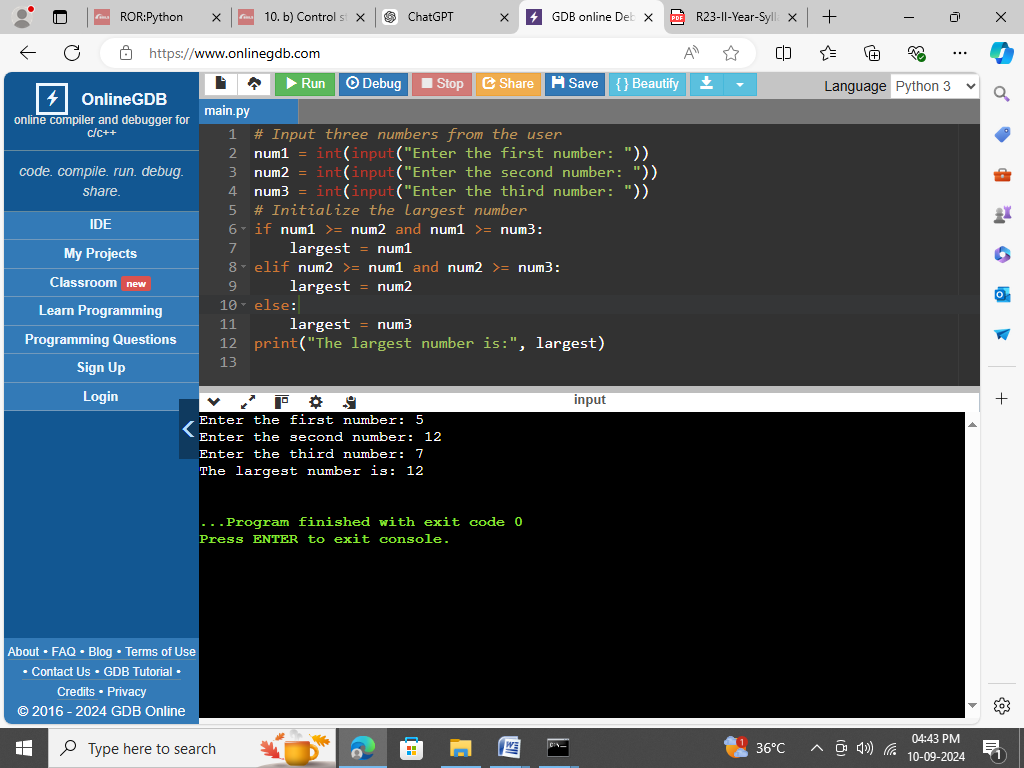
### Result: Successfully executed the largest element among three numbers.

### Run:D:\> python find\_largest.py

### Output:



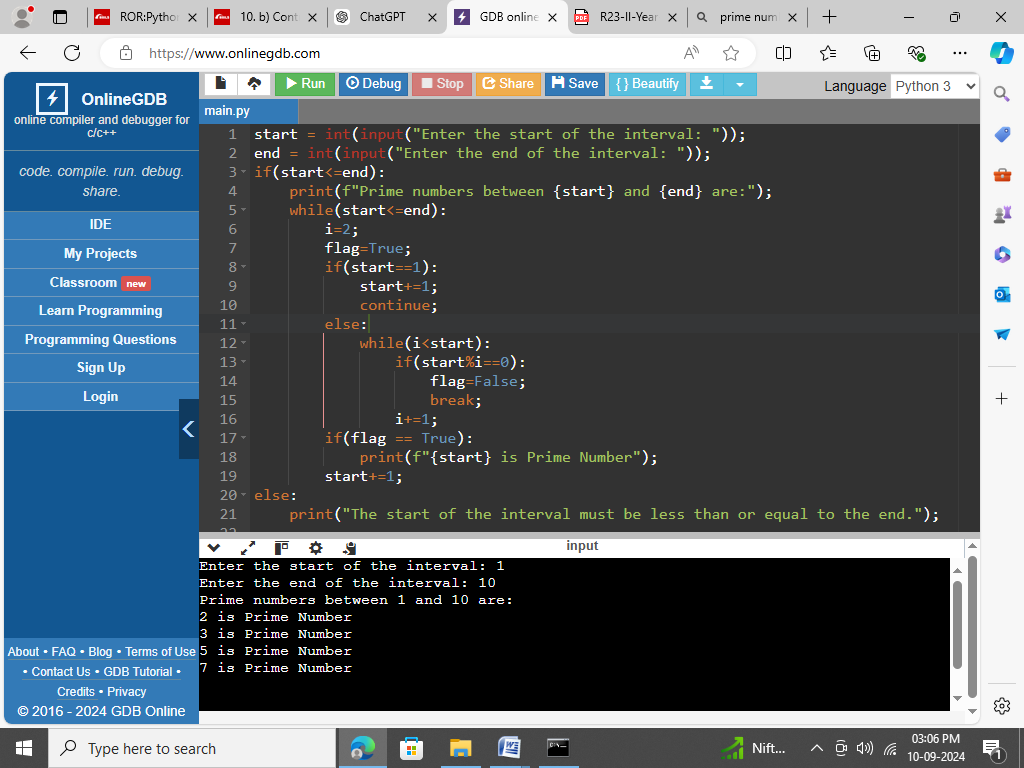
### Screenshot:



# Aim: Write a Program to display all prime numbers within an interval

<https://onlinegdb.com/yScYK7p2V>

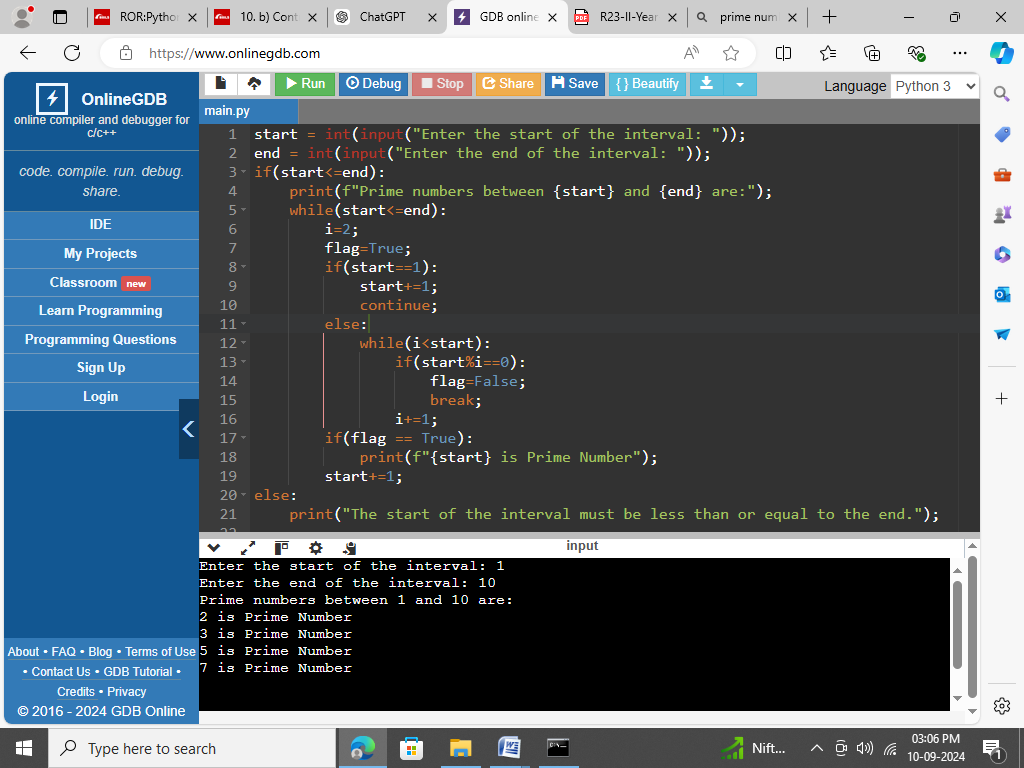
### Source Code: prime\_numbers\_interval.py



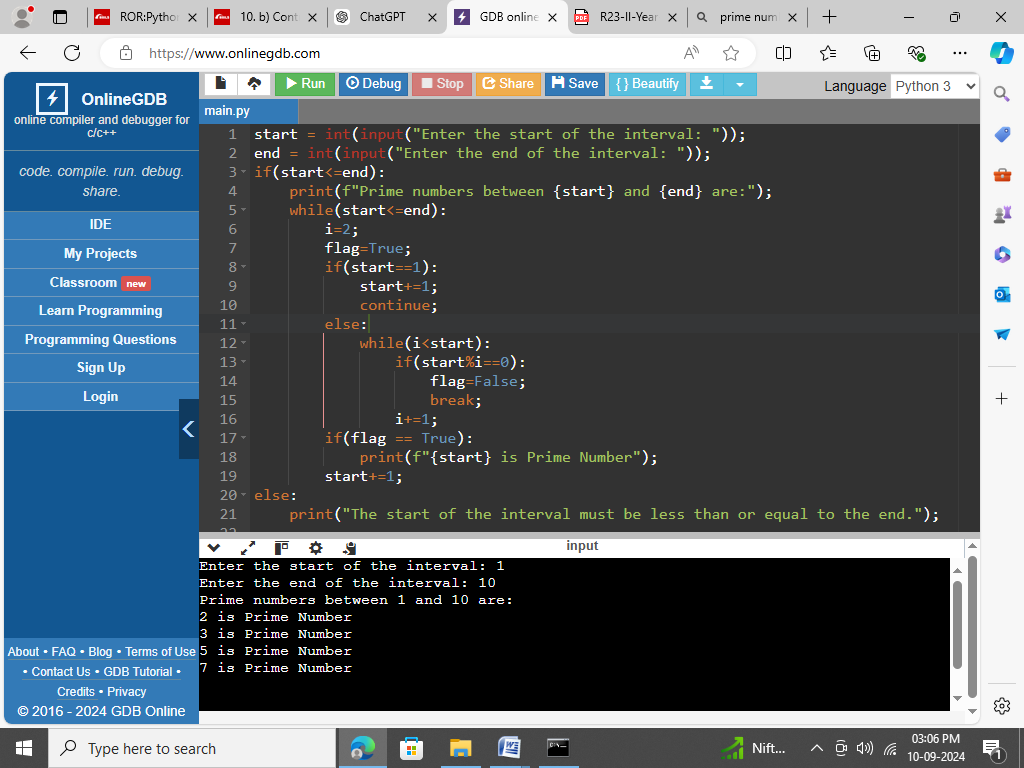
### Result: Successfully executed the all prime numbers within an interval.

### Run: D:\> python prime\_numbers\_interval.py

### Output:-



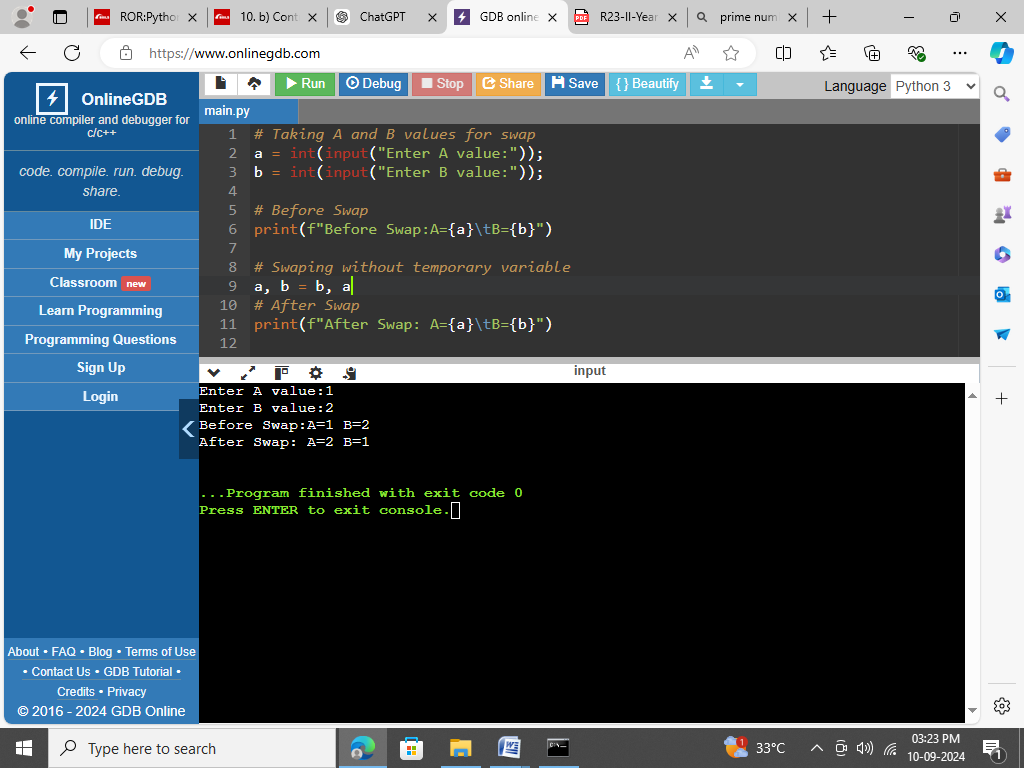
### Screenshot:



# Aim: Write a program to swap two numbers without using a temporary variable.

[**https://onlinegdb.com/LjNWDyxp3**](https://onlinegdb.com/LjNWDyxp3)

### Source Code: swap\_without\_temp.py

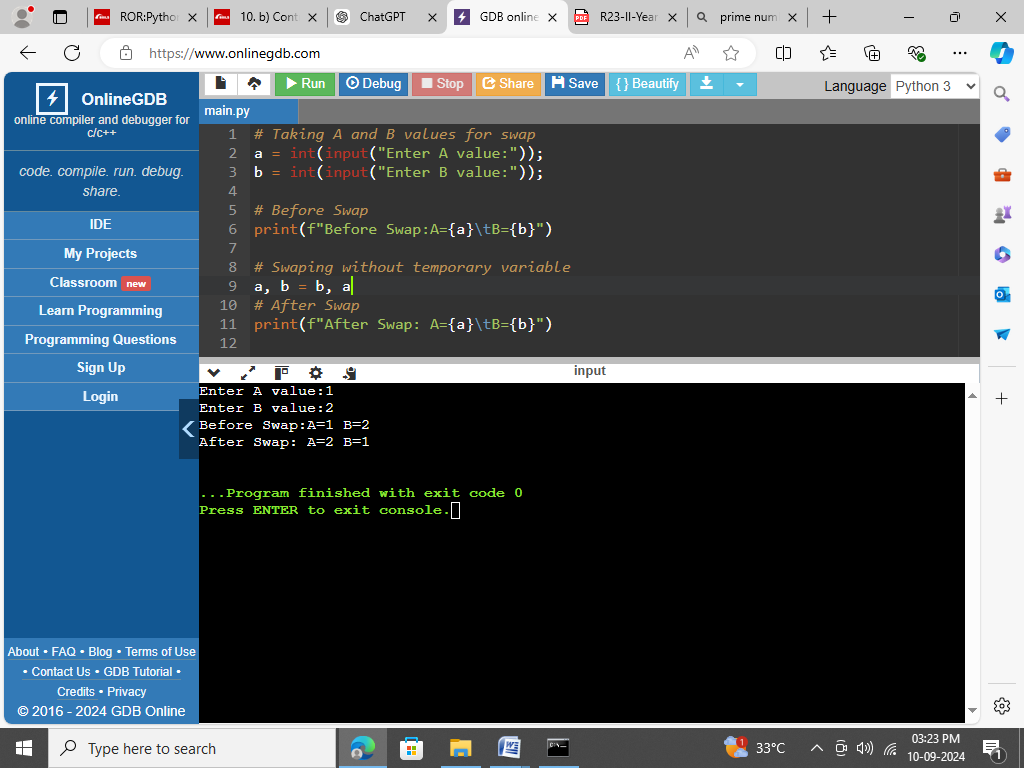


### Result: Successfully executed the swap two numbers without using a temporary variable.

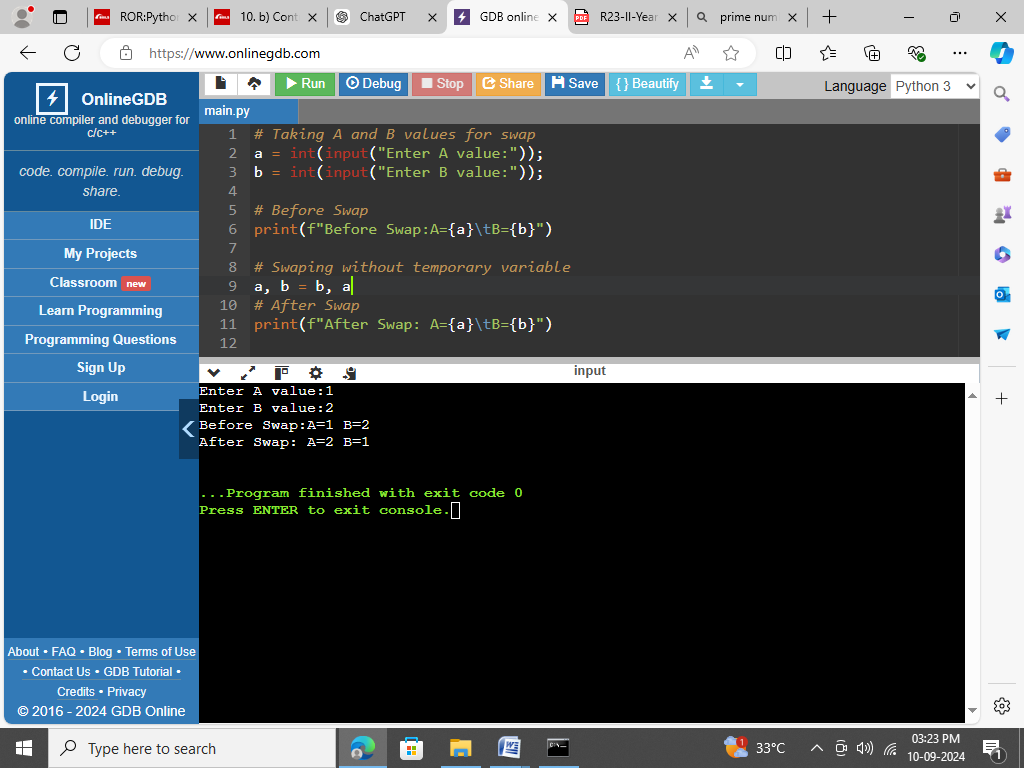
### Run:

D:\> python swap\_without\_temp.py

### Output:



### Screenshot:



# Aim: Demonstrate the following Operators in Python with suitable examples.

## i) Arithmetic Operators

## ii) Relational Operators

## iii) Assignment Operators

## iv) Logical Operators

## v) Bit wise Operators

## vi) Ternary Operator

## vii) Membership Operators

## viii) Identity Operators

<https://onlinegdb.com/VK2ogEjv4>

### Source Code: Operators.py

# Arithmetic Operators

a = 10

b = 3

print(f"A={a}\nB={b}")

print("i) Arithmetic Operators:\n")

print("\tA+B:", a + b)

print("\tA-B:", a - b)

print("\tA\*B:", a \* b)

print("\tA Exponentiation B:", a \*\* b)

print("\t:A/B:", a / b)

print("\tA//B:", a // b)

print("\t:A%b:", a % b)

print("\nii) Relational Operators:\n")

a = 10

b = 3

print("\tA < B :", a < b)

print("\tA <= B :", a <= b)

print("\tA > B :", a > b)

print("\tA >= B :", a >= b)

print("\tA == B :", a >= b)

print("\tA != B :", a <= b)

# Assignment Operators

print("\niii) Assignment Operators\n")

# iii) a. Simple Assignment Operator

i=100

print(f"\ta. Simple Assignment Operator: i={i}")

# iii) b. Compound Assignment Operator

i+=1

print("\n\tb. Compound Assignment Operator:\n")

print("\t\ti += 1 :",i)

i-=1

print("\t\ti -= 1 :",i)

i\*=2

print("\t\ti \*= 2 :",i)

i\*\*=2

print("\t\ti \*\*= 2 :",i)

i/=2

print("\t\ti /= 2 :",i)

i//=2

print("\t\ti //= 2 :",i)

i%=2

print("\t\ti %= 2 :",i)

# parllel Assignment Operators

print("\tc. parallel Assignment Operators:-\n");

p=100

q=200

print(f"\t\tBefore : P={p},Q={q}")

p,q=q,p

print(f"\t\tAfter : P={p},Q={q}")

# Logical Operators

print("\niv) Logical Operators:-\n")

print("\t(1<2) and (1<2) :", (1<2) and (1<2))

print("\t(1<2) or (1>2) :", (1<2) or (1>2))

print("\tnot(1<2):", not(1<2))

# Bitwise Operators

print("\nv) Bitwise Operators:-\n")

print("\t7 & 2:", 7 & 2)

print("\t7 | 2 :", 7 | 2)

print("\t7 ^ 3 :", 7 ^ 2)

print("\t~7:", ~7)

print("\t7<<1:", 7 << 1)

print("\t7>>1:", 7 >> 1)

# Ternary Operator

print("\nvi) Ternary Operator:-\n")

a = 10

b = 5

result = "a is greater" if a > b else "b is greater or equal"

print("\t",result)

# Membership Operators

print("\nvii) Membership Operators:-\n")

list1 = [1, 2, 3, 4, 5]

print("\tMy List:",list1)

print("\t3 in list1:",3 in list1)

print("\t6 not in list1:",6 not in list1)

# viii) Identity Operators

print("\nviii) Identity Operators:-\n")

a = [1, 2, 3]

b = [1, 2, 3]

print("\tA:",a)

print("\tB:",b)

print("\ta is b : ",a is b)

print("\ta is not b :",a is not b)

c = a

print("\ta is c : ",a is c)

print("\ta is not c : ",a is not c)

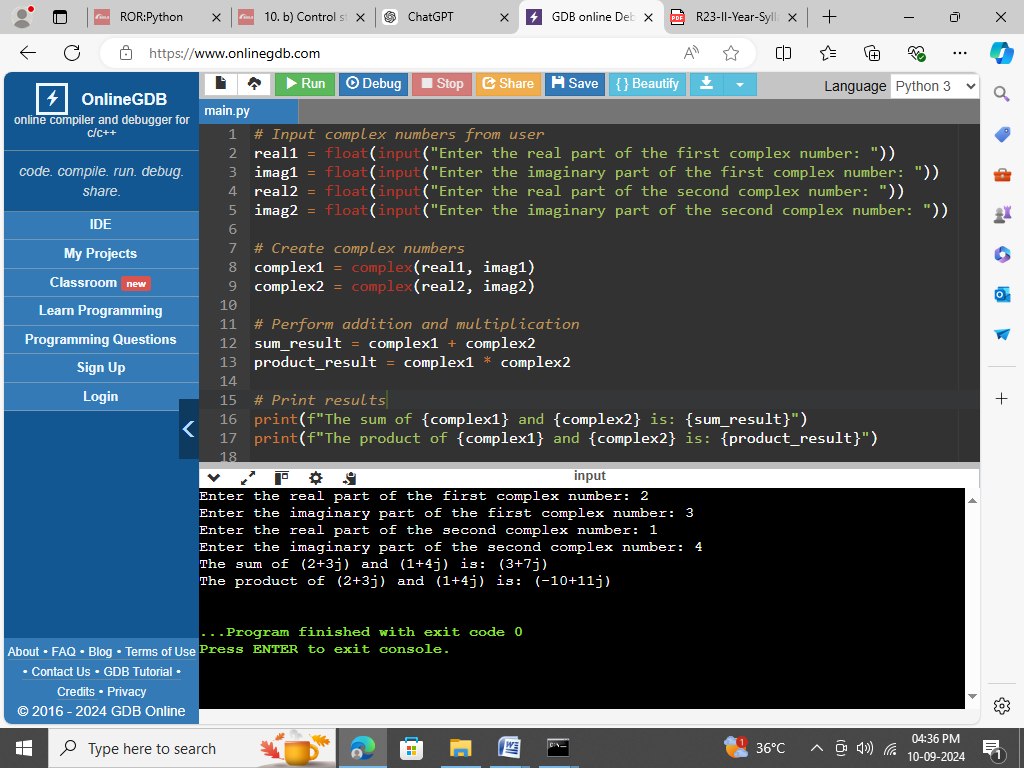
### Result: Successfully executed types of operator examples.

### Run: D:\> python Operators.py

### Output:

# Aim: Write a program to add and multiply complex numbers

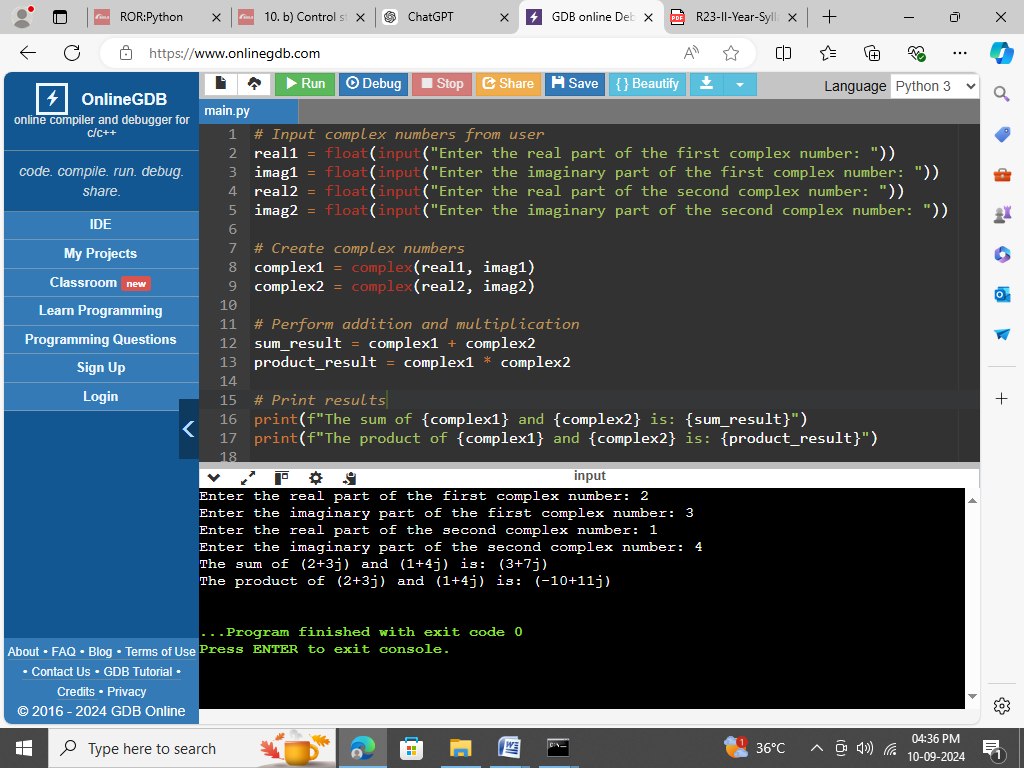
### Source Code: complex\_operations.py



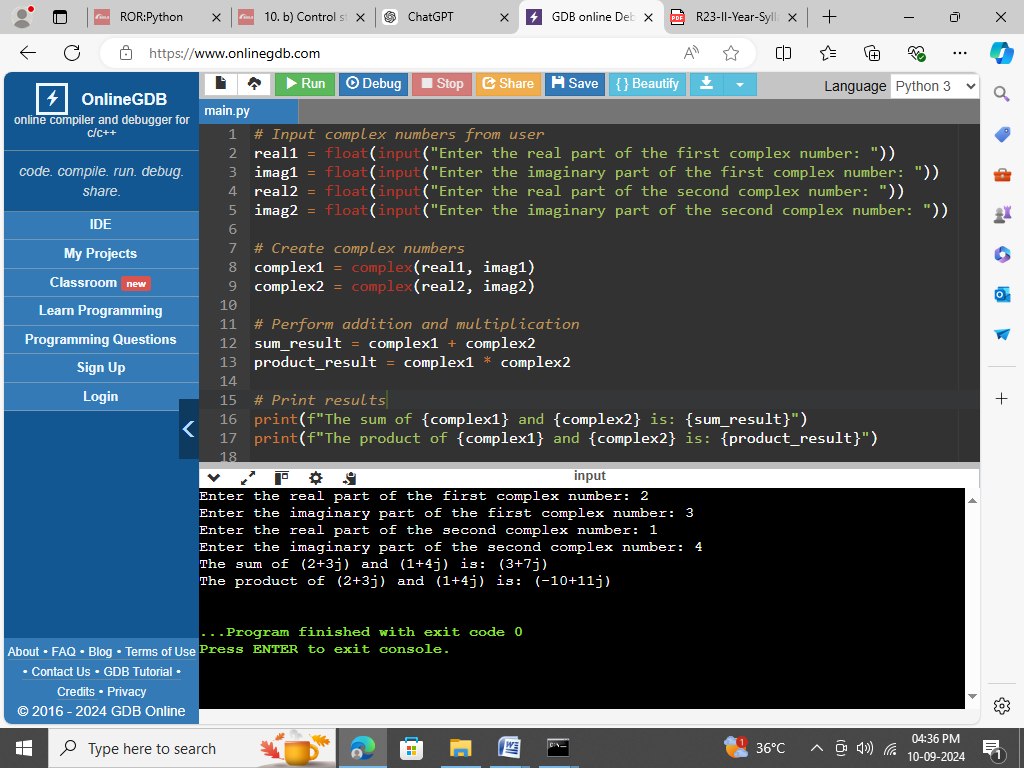
### Result: Successfully executed the add and multiply complex numbers.

### Run: D:\> python complex\_operations.py

### Output:



### Screenshot:



# 6. Write a program to print multiplication table of a given number.

[**https://onlinegdb.com/7l93s5Ry\_**](https://onlinegdb.com/7l93s5Ry_)

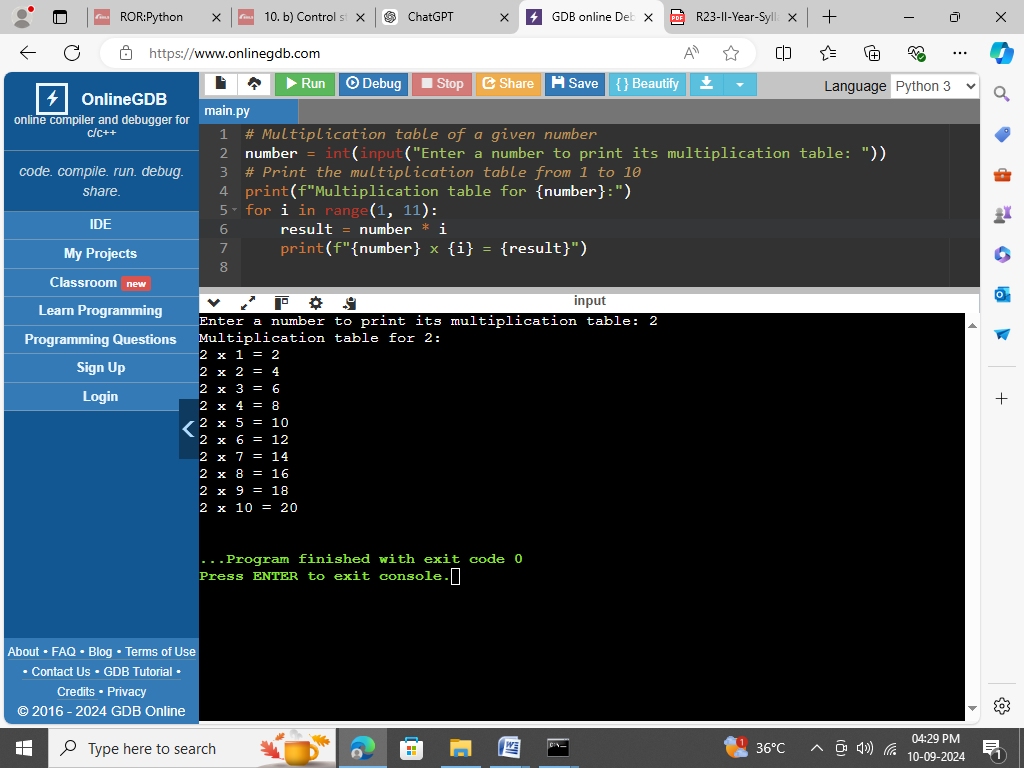
### Source Code: multiplication\_table.py

### 

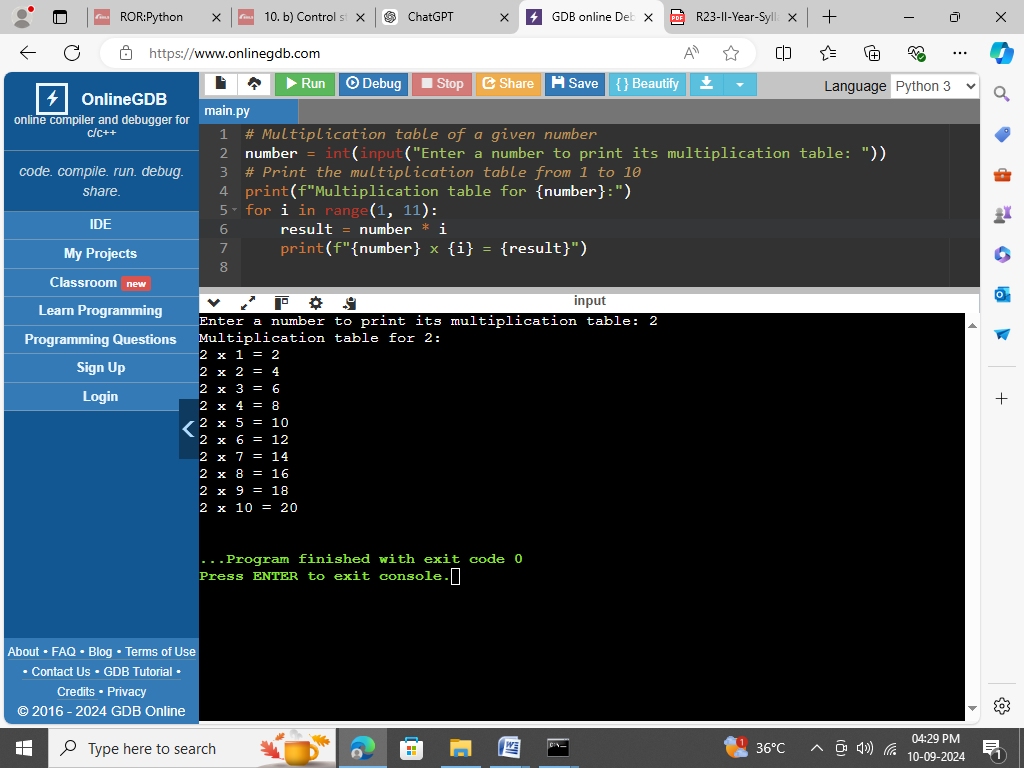
### Result: Successfully executed the multiplication table of a given number

### Run: D:\> python multiplication\_table.py

### Output:



### Screenshot:



# 7. Write a program to define a function with multiple return values.

<https://onlinegdb.com/z4xe6pQkQ>

### Source Code: multiple\_return\_values\_fun.py

### 

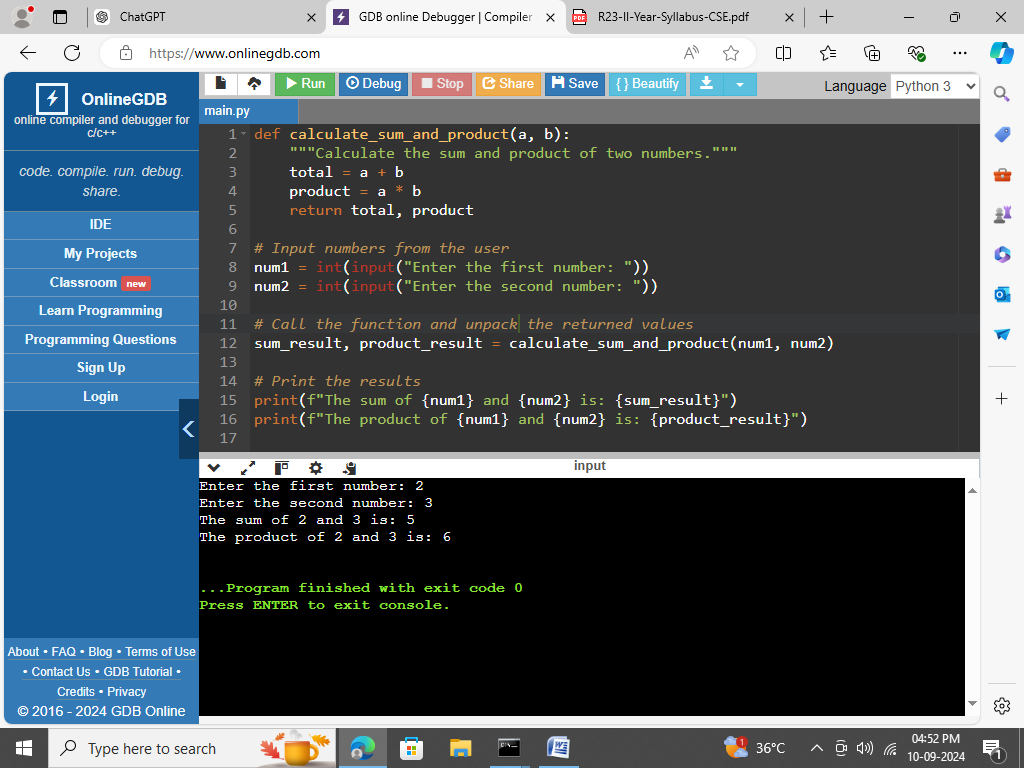
### Result: Successfully executed the define a function with multiple return values

### Run: D:\>python multiple\_return\_values\_fun.py

### Output:

### 

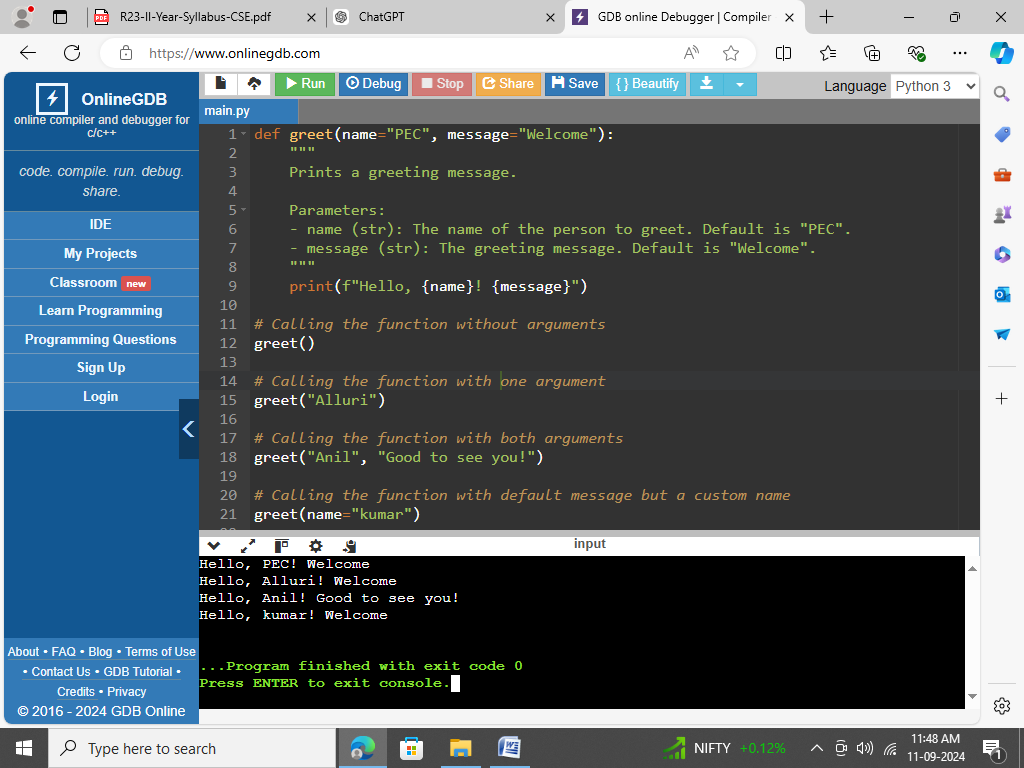
### Screenshot:



# Aim: Write a program to define a function using default arguments.

<https://onlinegdb.com/r8S-D5ONmX>

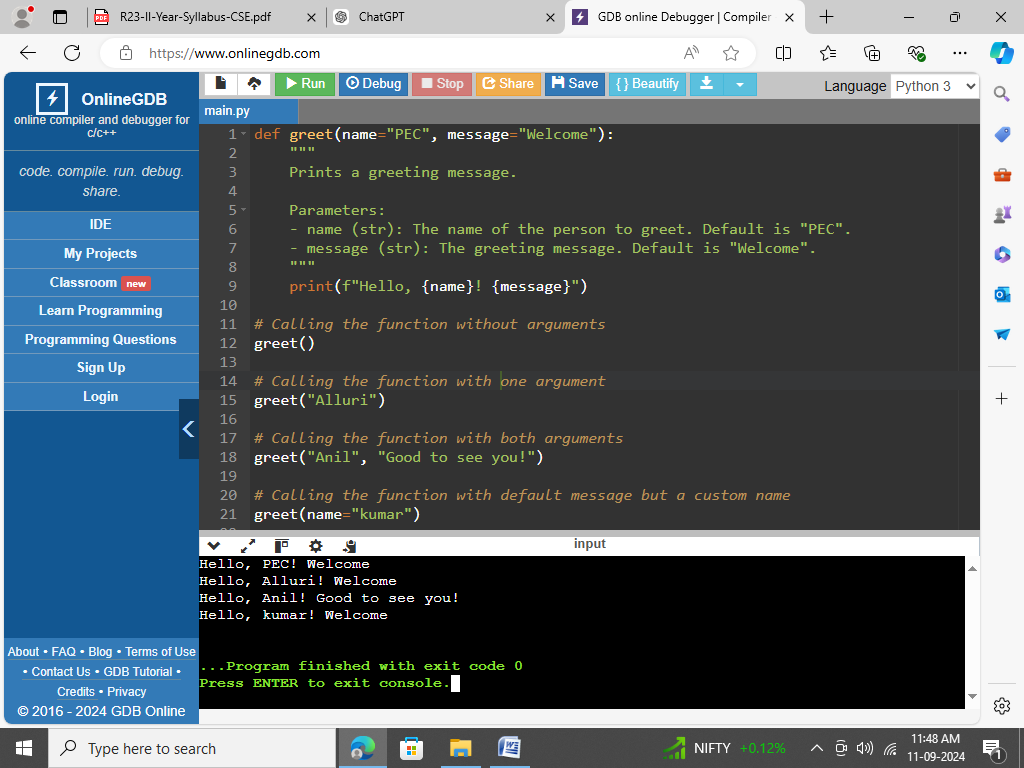
### Source Code: default\_args\_function.py



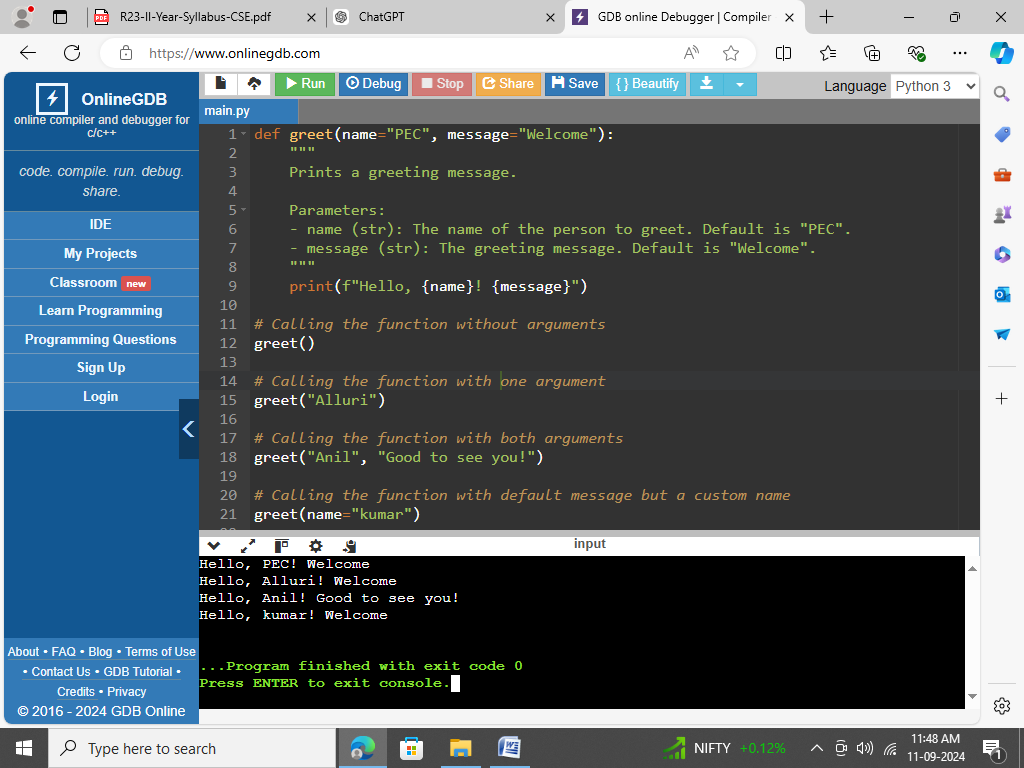
### Result: Successfully executed the defining a function using default arguments.

### Run: D:\> python Default\_Args\_Function.py

### Output:



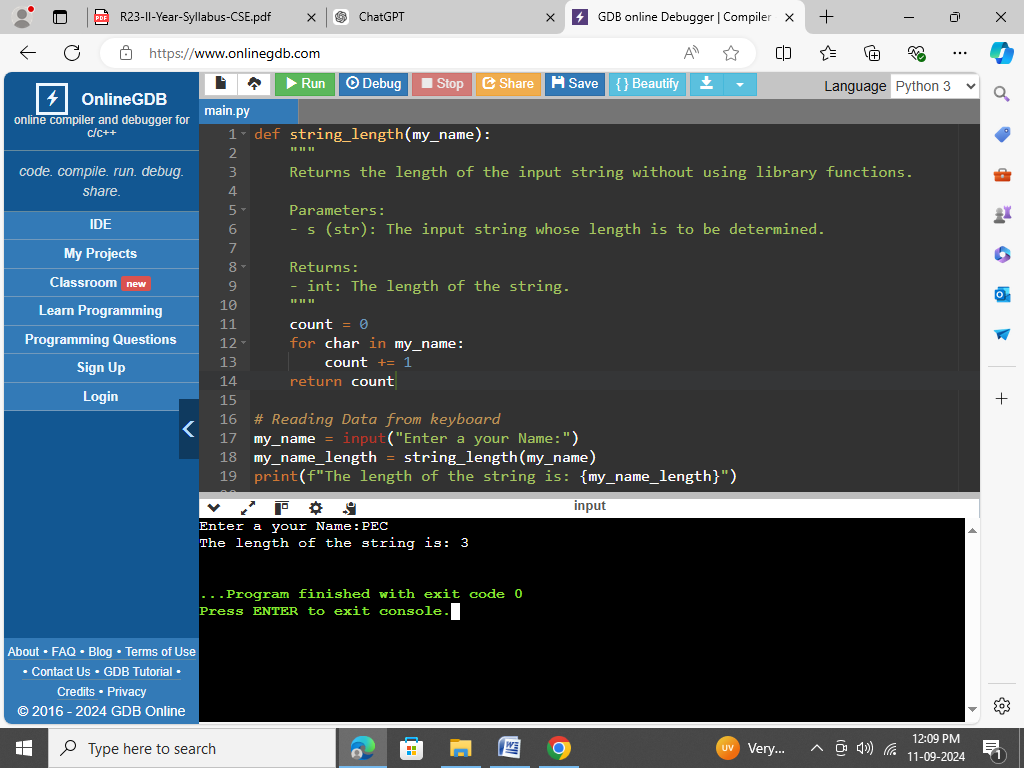
### Screenshot:



# Aim: Write a program to find the length of the string without using any library functions

<https://onlinegdb.com/-HNY_2mcqh>

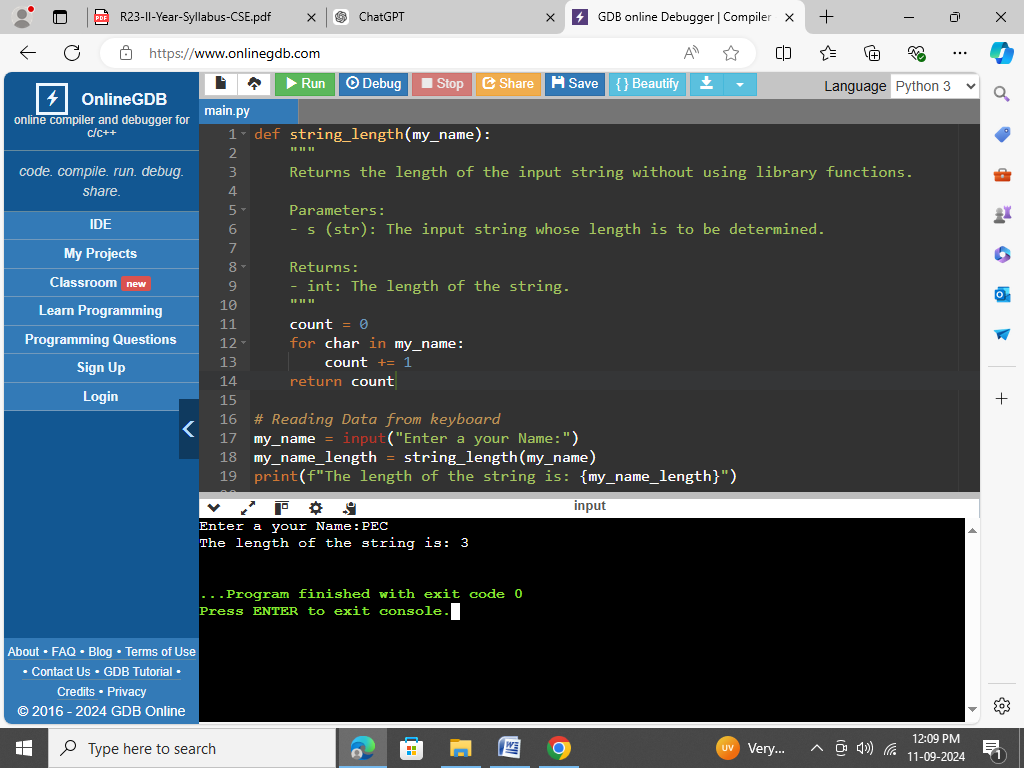
### Source Code: string\_length.py



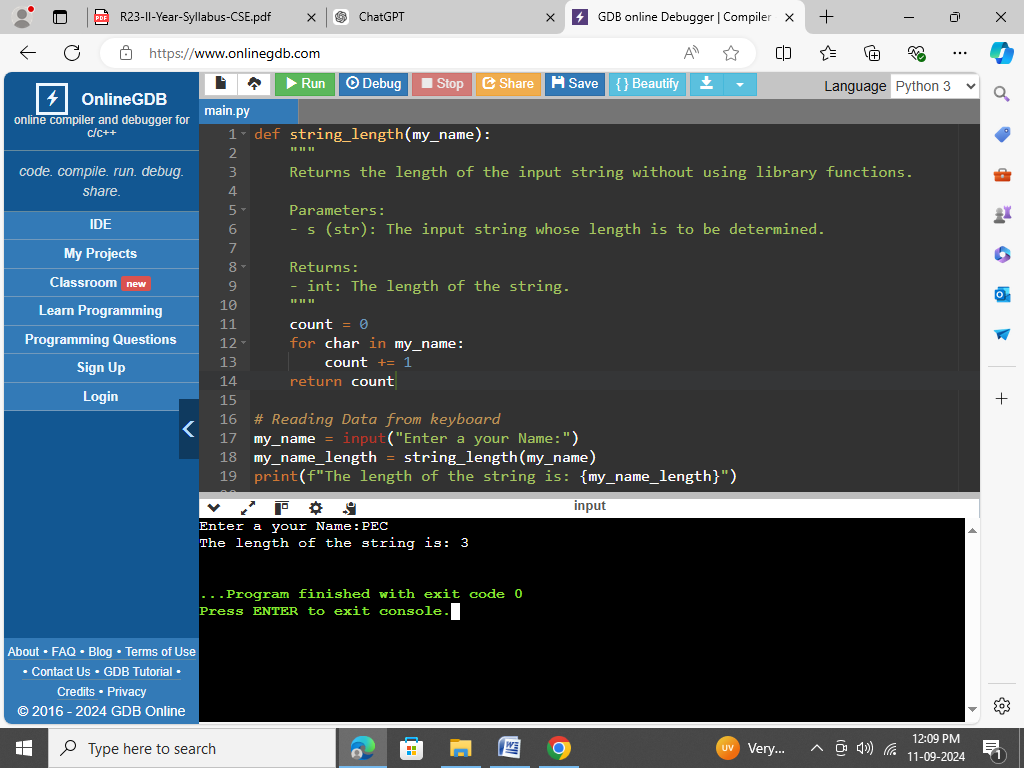
### Result: Successfully executed the length of the string without using any library functions.

### Run: D:\> python string\_length.py

### Output:



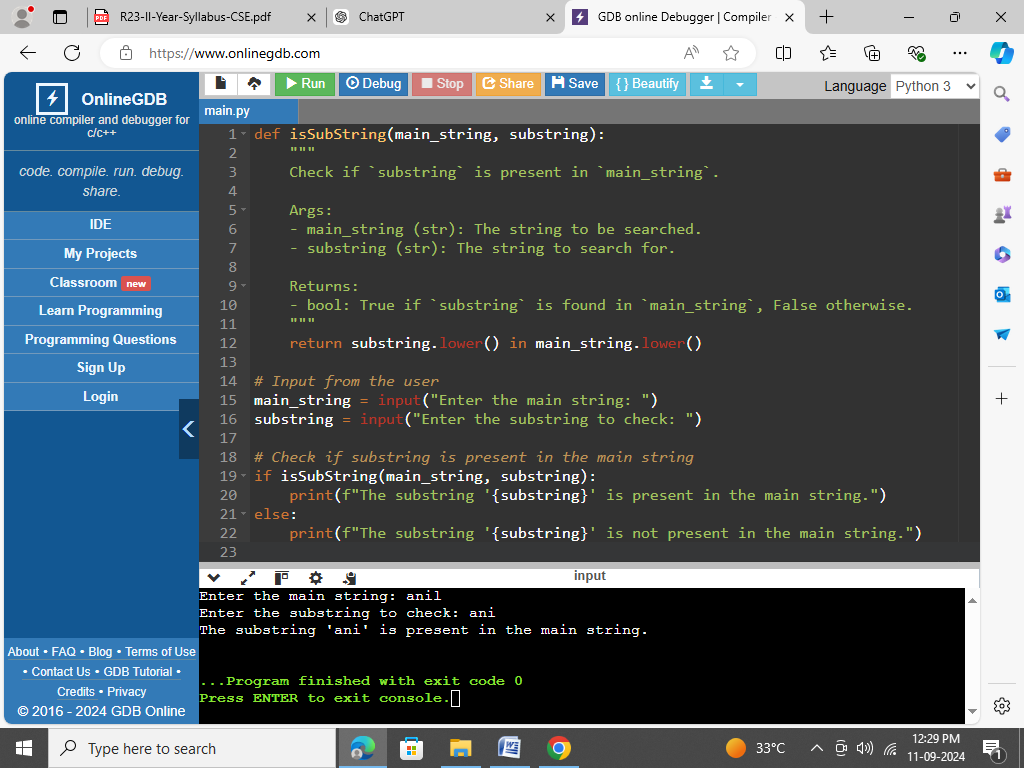
### Screenshot:



# Aim: Write a program to check if the substring is present in a given string or not.

<https://onlinegdb.com/6pq06Wfgwx>

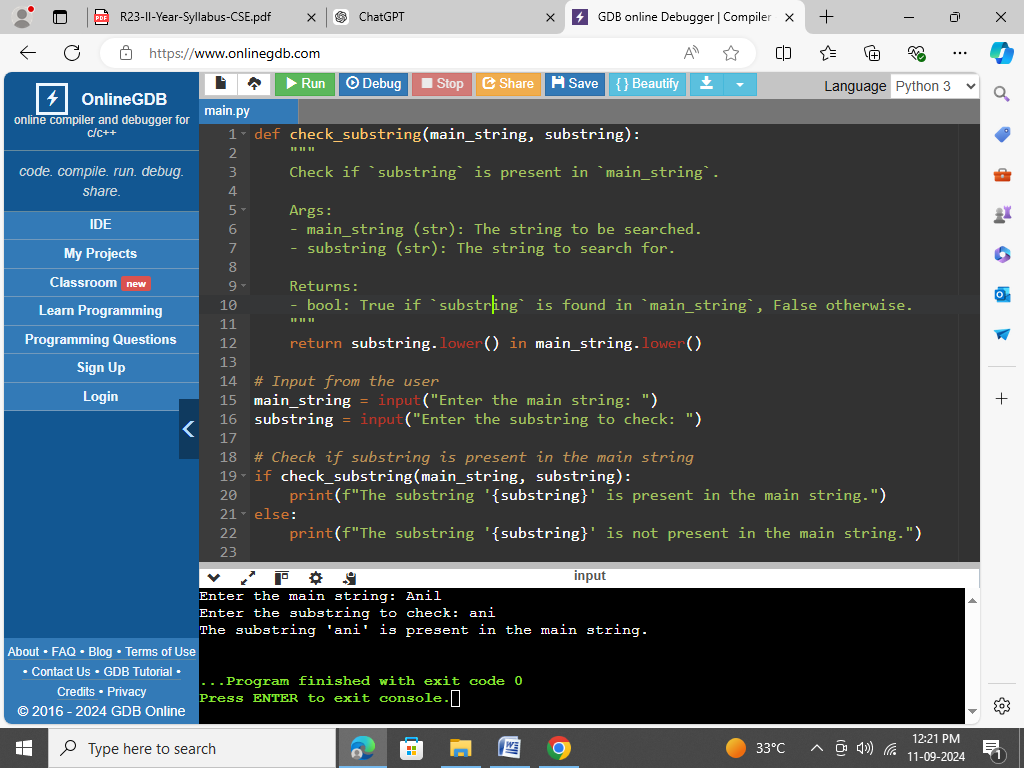
### Source Code: check\_substring.py



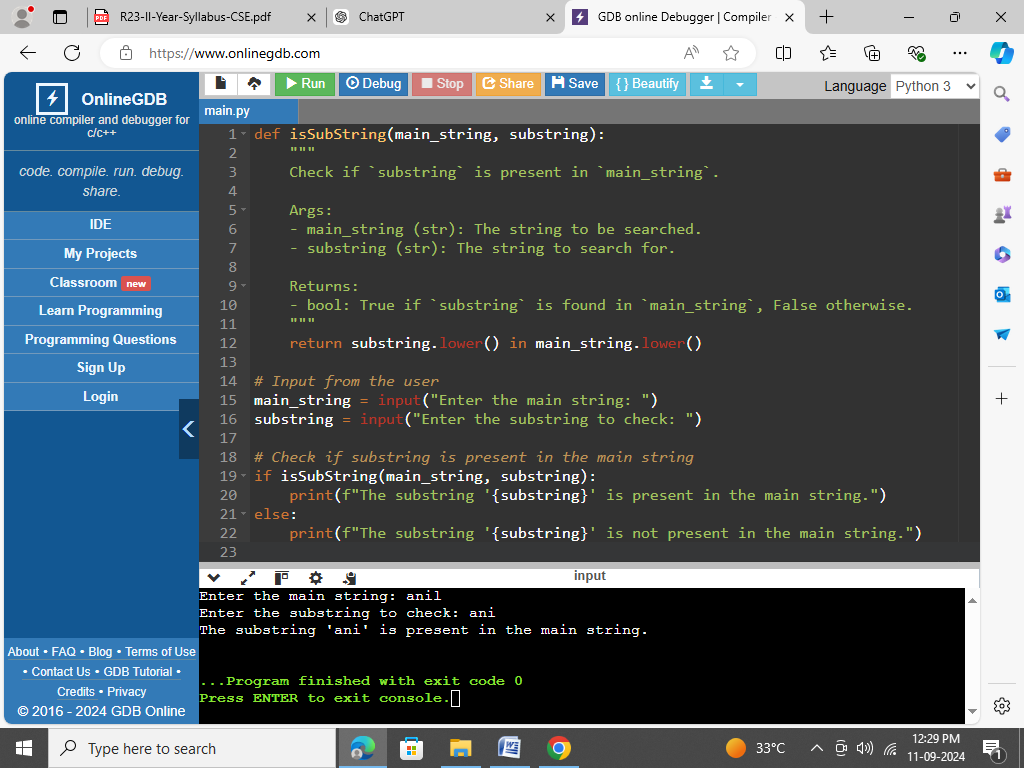
### Result: Successfully executed the program if the substring is present in a given string or not.

### Run: python check\_substring.py

### Output:



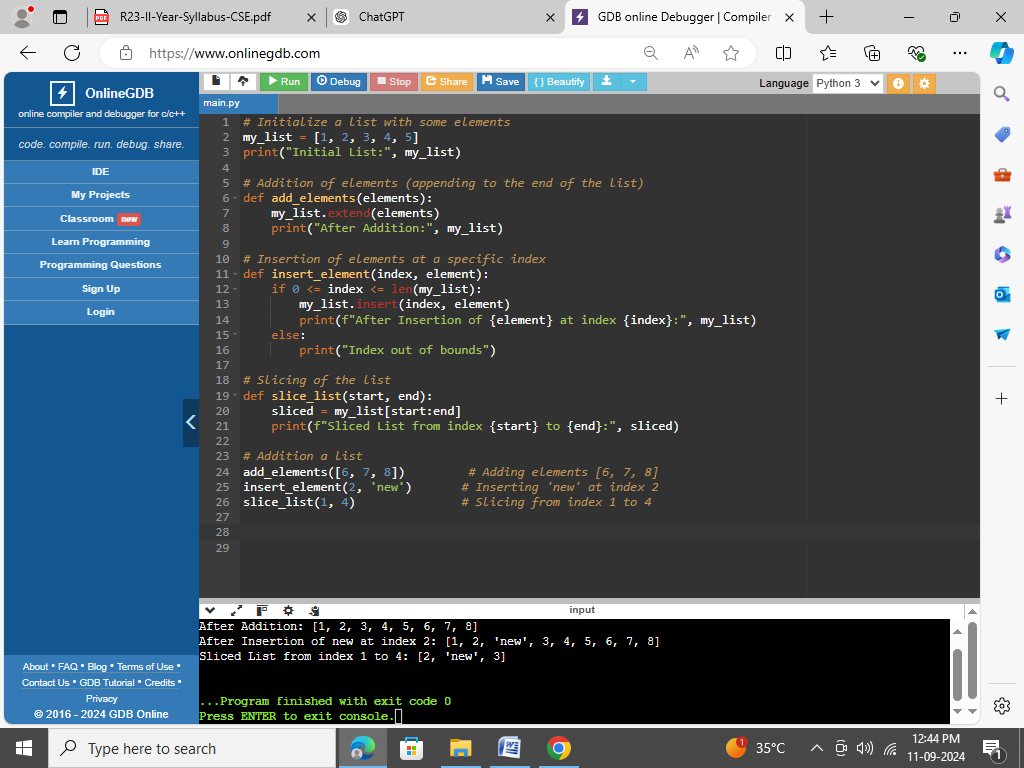
### Screenshot:



# Aim: Write a program to perform the given operations on a list: (i) addition (ii) Insertion (iii) slicing

<https://onlinegdb.com/ka2re0A8nA>

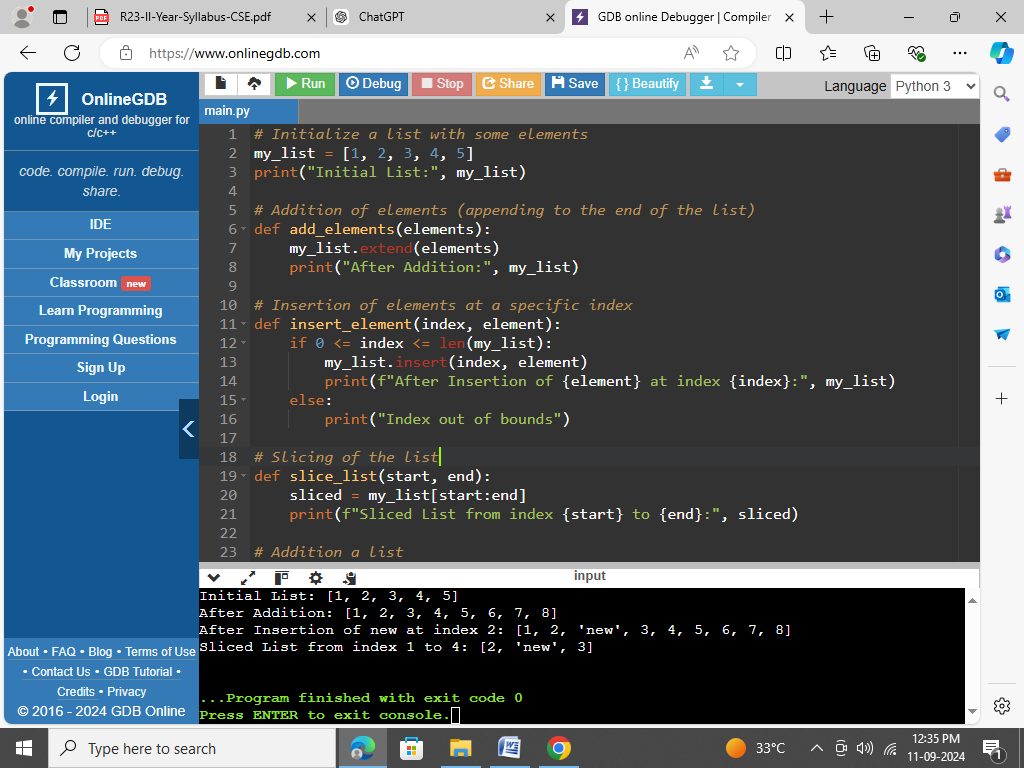
### Source Code: list\_operations.py



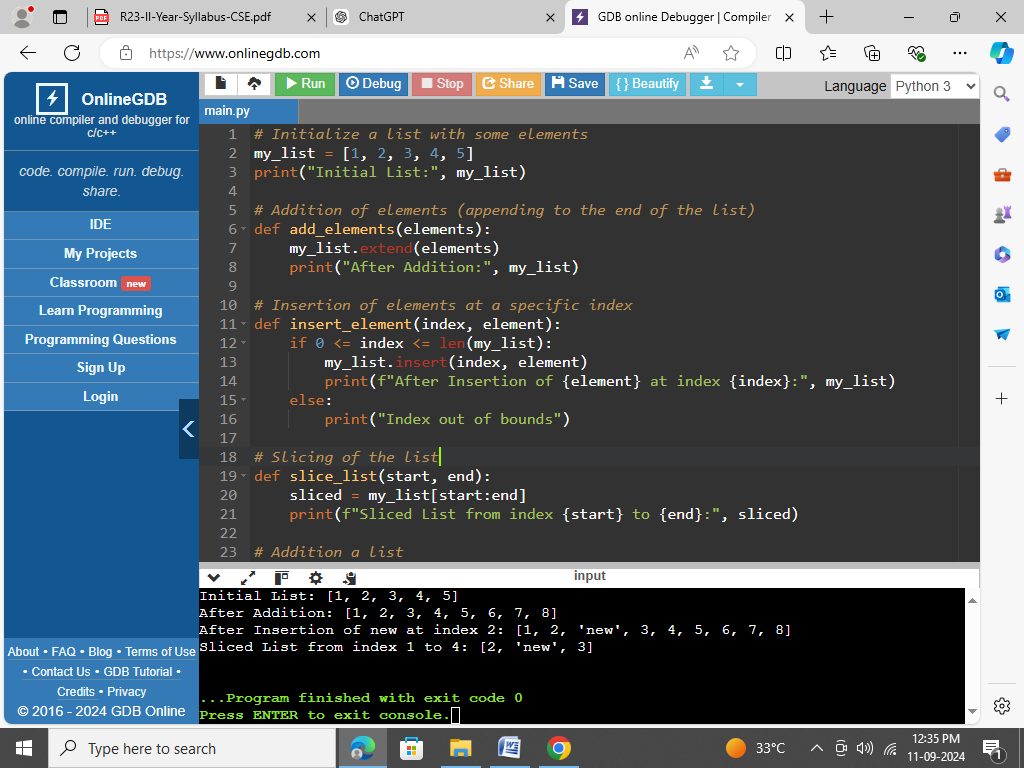
### Result: Successfully executed the list operation on addition, insertion and slicing.

### Run: D:\> python list\_operations.py

### Output:



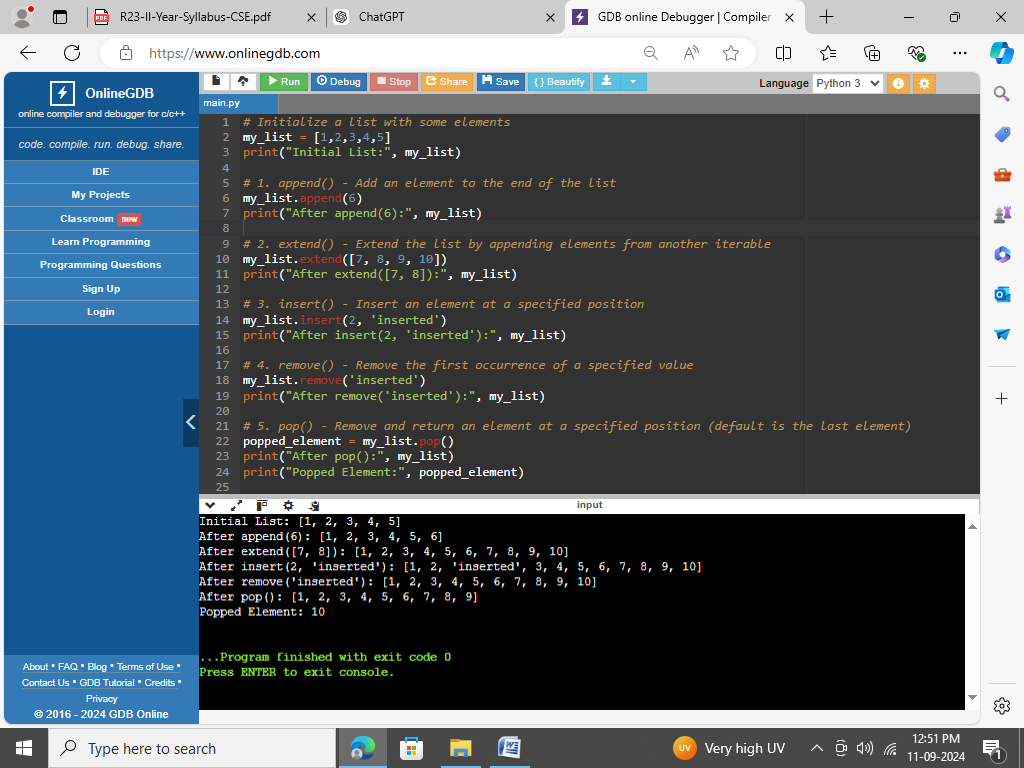
### Screenshot:



# Aim: Write a program to perform any 5 built-in functions by taking any list

<https://onlinegdb.com/Lwx8ENgVRn>

### Source Code: list\_5built\_in\_functions.py

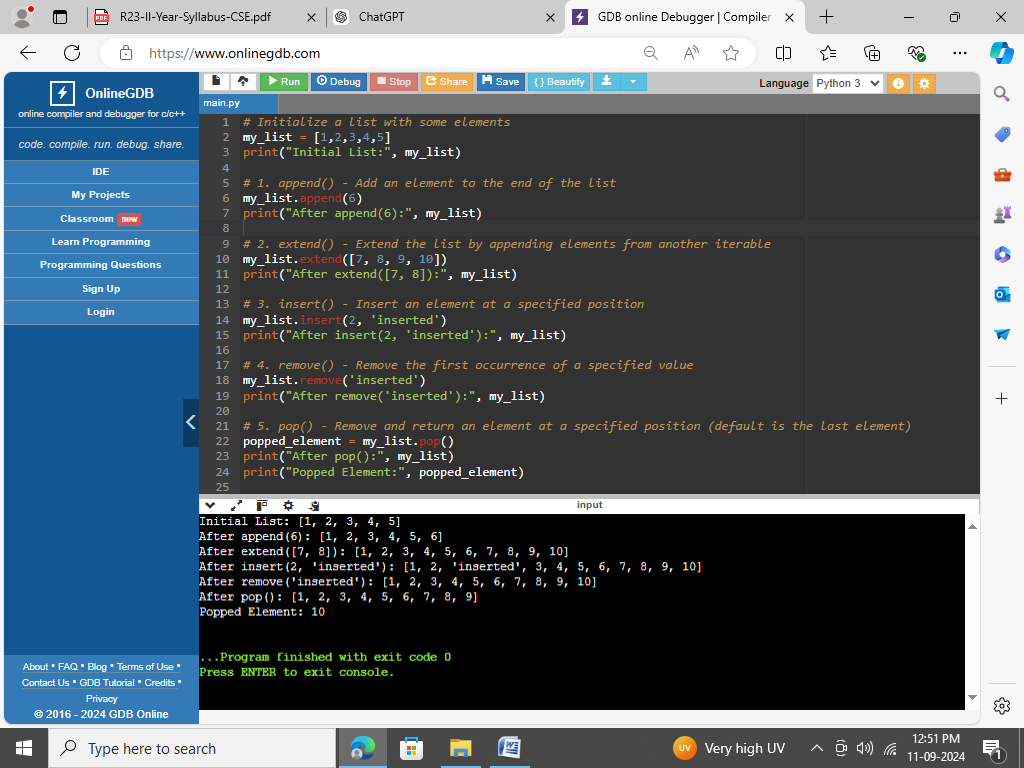


### Run or Execute: D:\> python list\_5built\_in\_functions.py

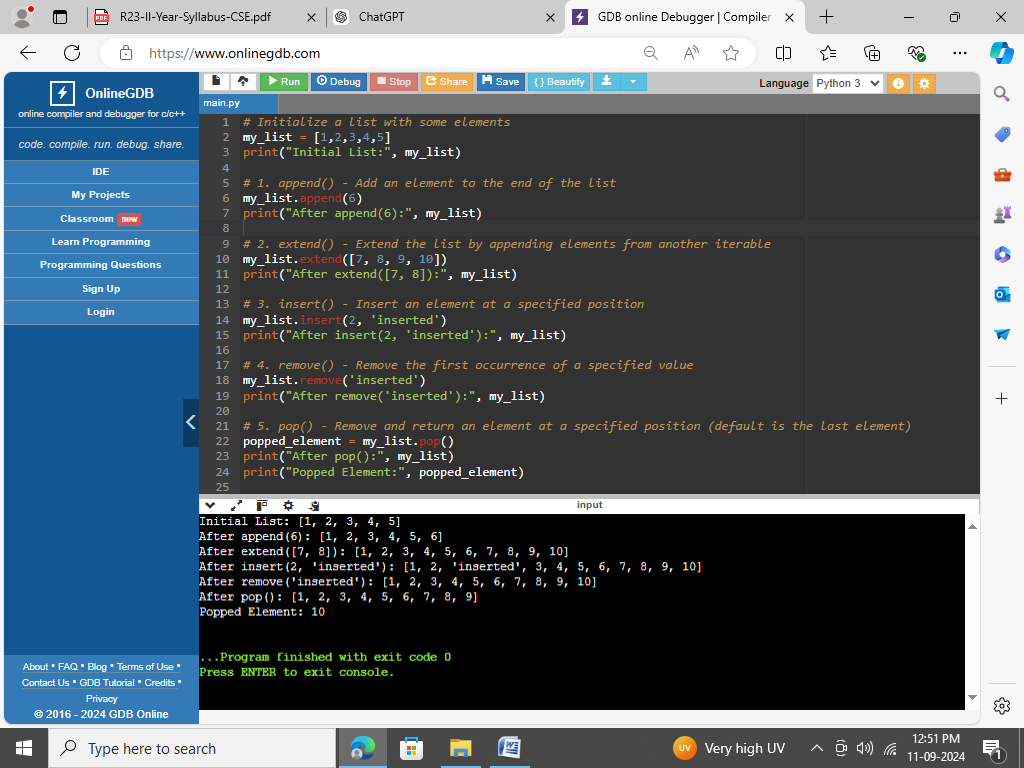
### 

### Result: Successfully executed the five built-in functions are append(v), extend(list), insert(i,v), remove(v), pop() of list.

### Output:

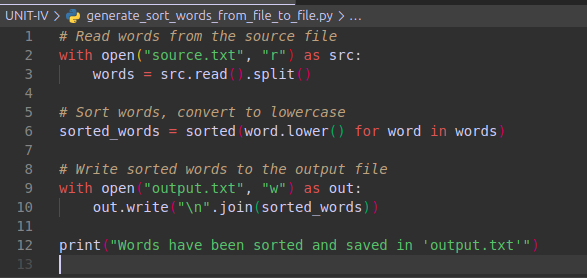


### Screenshot:



# Aim: Write a program to sort words in a file and put them in another file. The output file should have only lower-case words, so any upper-case words from source must be lowered.

### Source Code: generate\_sort\_words\_from\_file\_to\_file.py



### Run or Execute: D:\> python generate\_sort\_words\_from\_file\_to\_file.py

### 

### Result: Successfully generating a output of sorting words from input file to output file.

### Input and Output: source.txt Banana Apple orange Mango Cherry Grape apple banana Orange

Words have been sorted and saved in 'output.txt'

output.txt

apple

apple

banana

banana

cherry

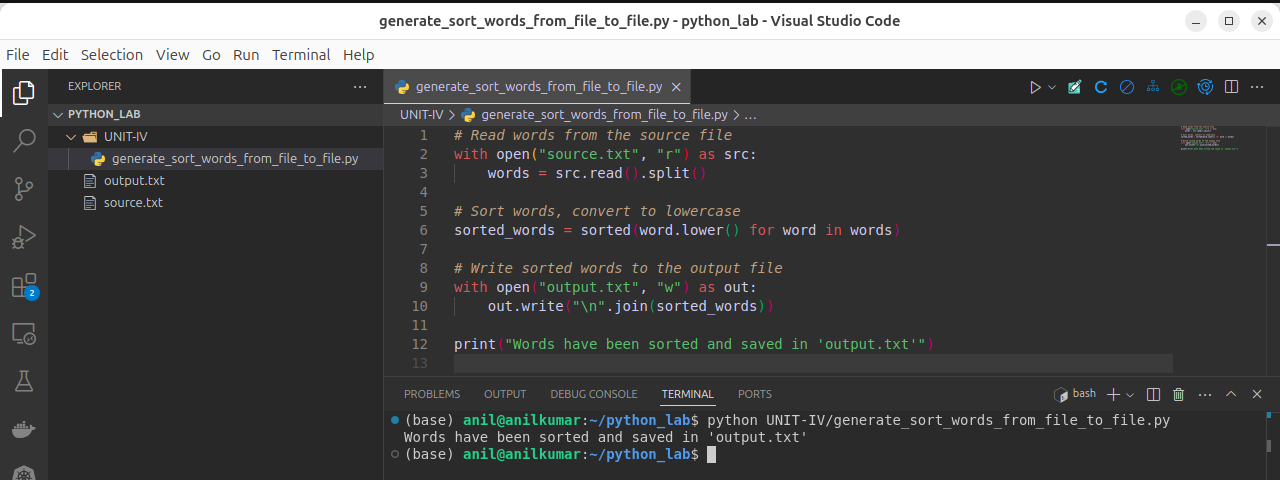
grape

mango

orange

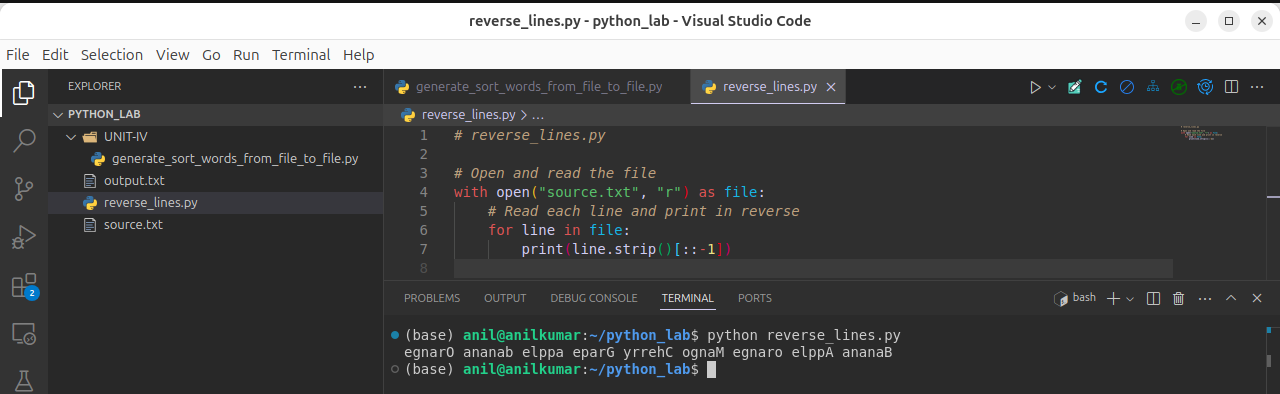
orange

### Screenshot:



# Aim: Python program to print each line of a file in reverse order.

### Source Code: reverse\_lines.py



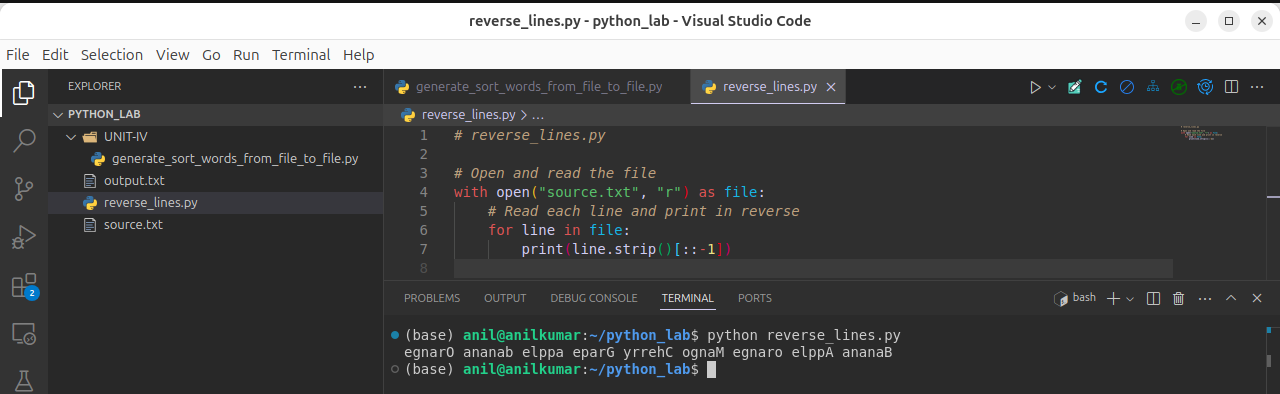
### Run or Execute: D:\> python reverse\_lines.py

### 

### Result: Successfully executed the each line of a file in reverse order

### Input and Output: egnarO ananab elppa eparG yrrehC ognaM egnaro elppA ananaB

### Screenshot:



# Aim: Python program to print each line of a file in reverse order.

### Source Code: count\_file\_content.py

### 

### Run or Execute: D:\> python count\_file\_content.py

### 

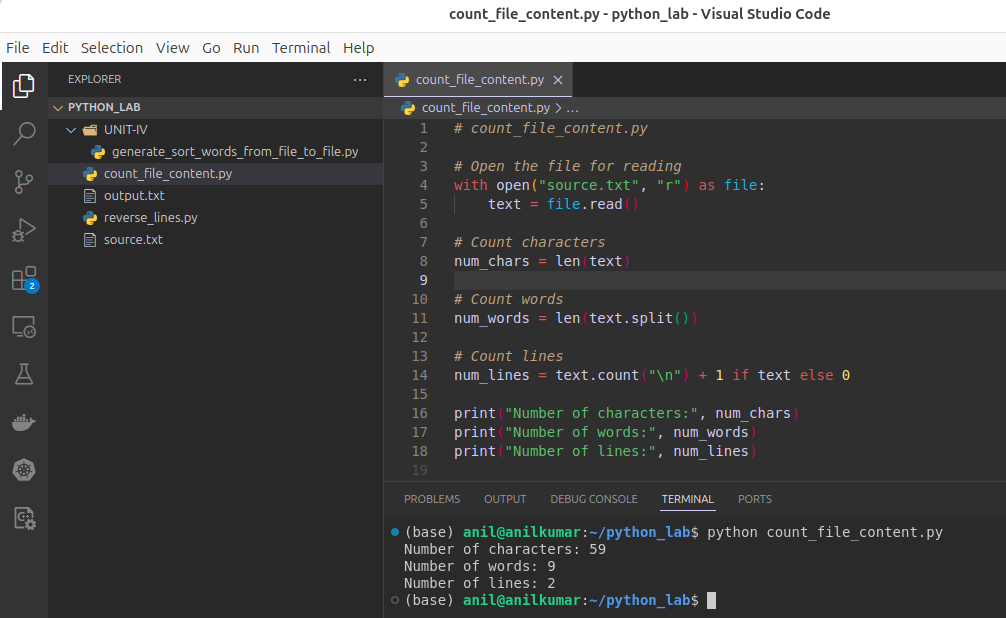
### Result: Successfully generating a output of sorting words from input file to output file.

### Input and Output: Number of characters: 59

### Number of words: 9

### Number of lines: 2

### Screenshot:

\*

# Aim: Python program to print each line of a file in reverse order.

### Source Code: count\_file\_content.py

### 

### Run or Execute: D:\> python count\_file\_content.py

### 

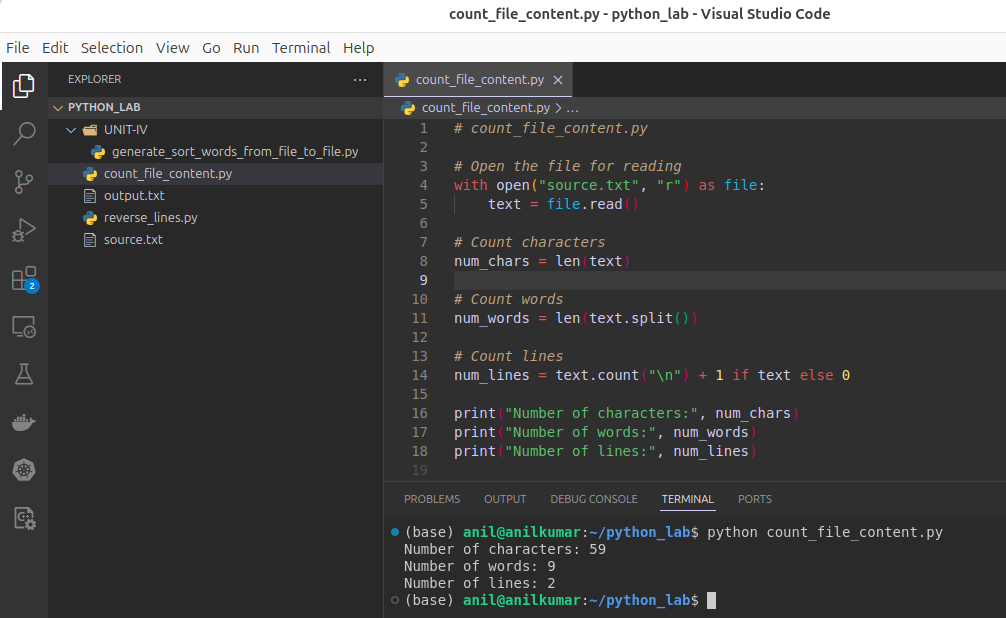
### Result: Successfully generating a output of sorting words from input file to output file.

### Input and Output: Number of characters: 59

### Number of words: 9

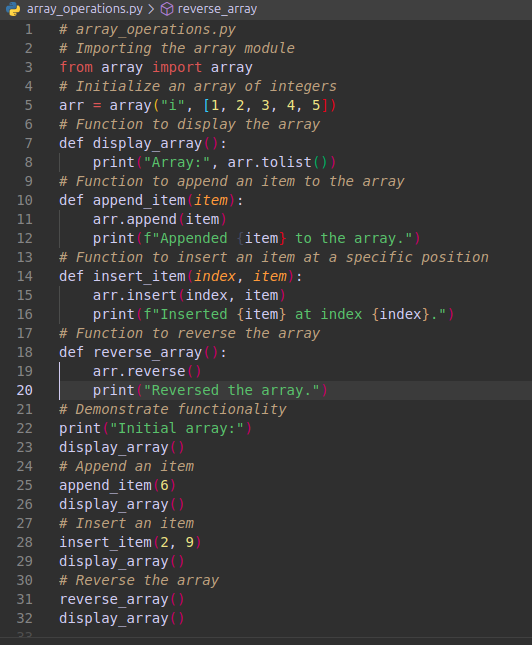
### Number of lines: 2

### Screenshot:



# Aim: . Write a program to create, display, append, insert and reverse the order of the items in the array.

### Source Code: array\_operations.py



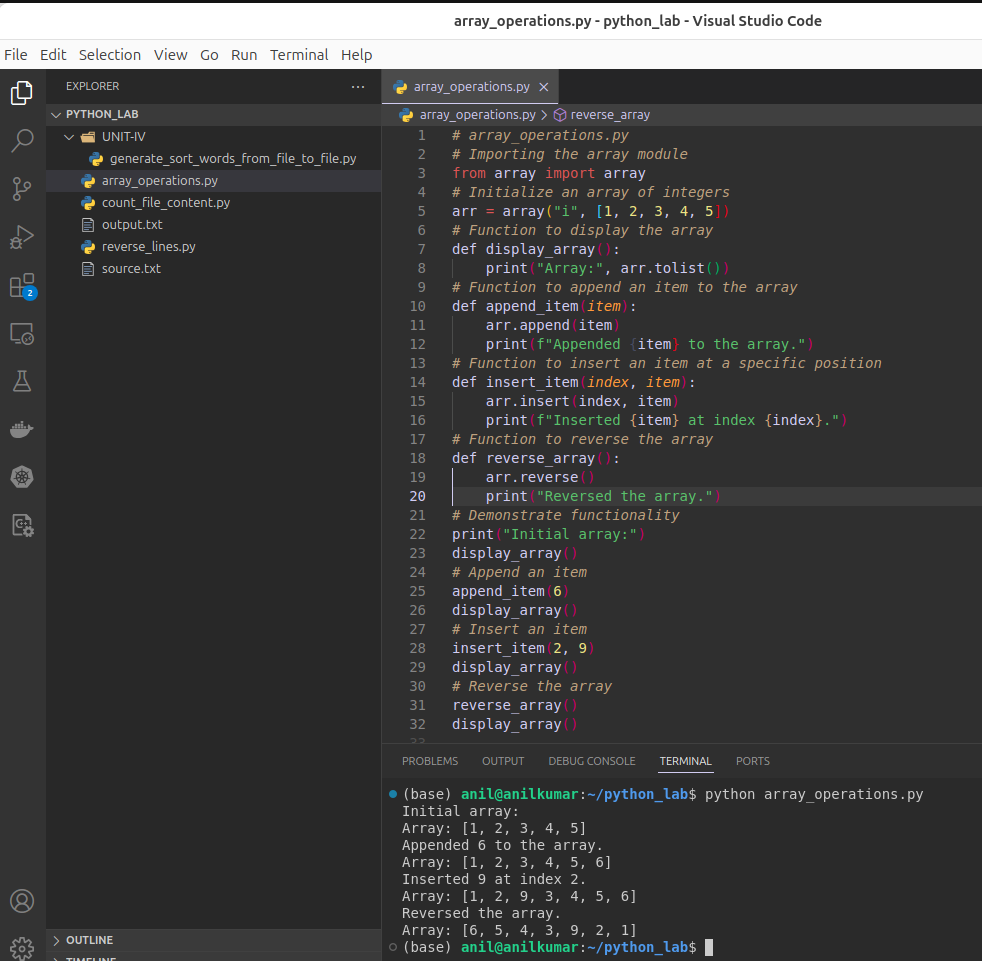
### Run or Execute: D:\> python array\_operations.py

### 

### Result: Successfully generating a output of sorting words from input file to output file.

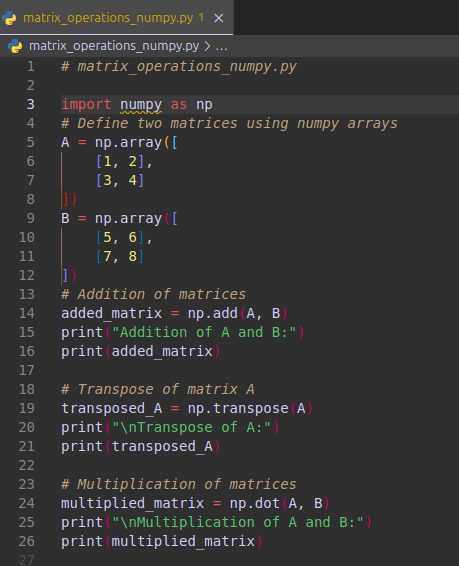
### Input and Output:

### Screenshot:

\*

# Aim: Write a program to add, transpose and multiply two matrices.

### Source Code: matrix\_operations\_numpy.py



### Run or Execute: D:\> python matrix\_operations\_numpy.py

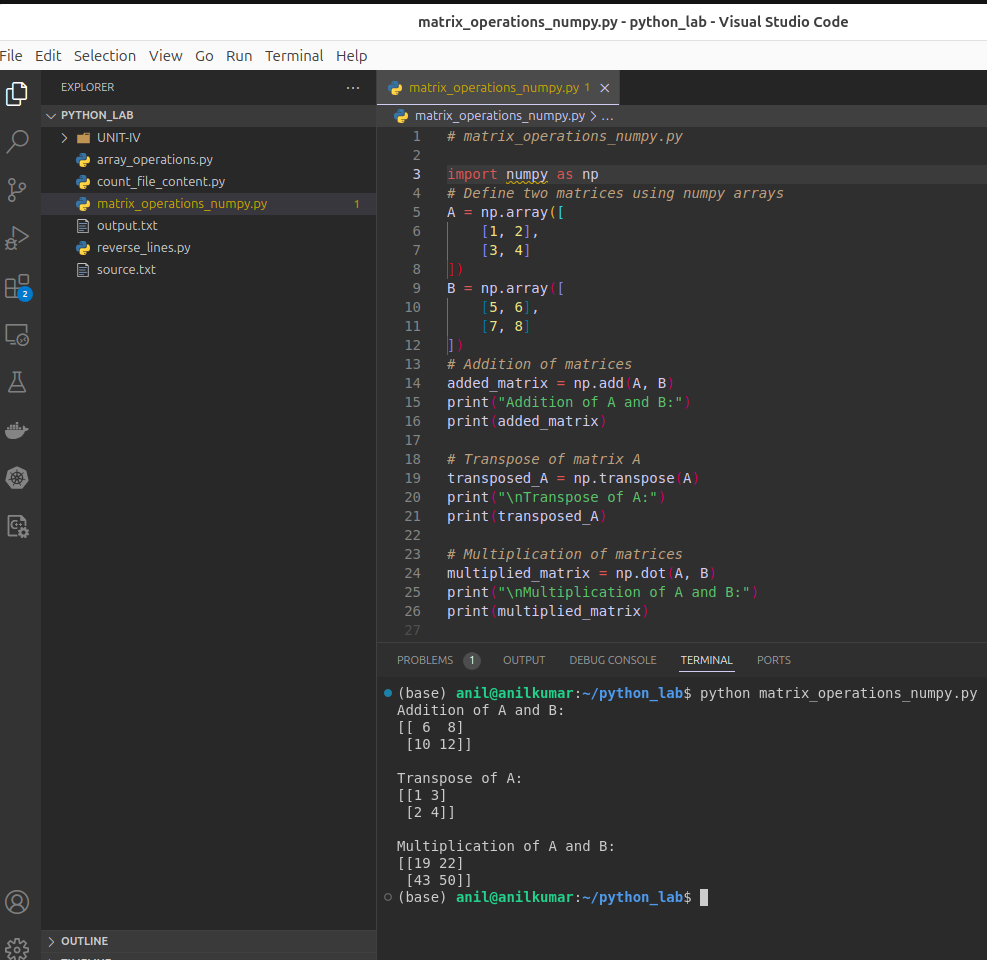
### 

### Result: Successfully generating a output of sorting words from input file to output file.

### Input and Output:

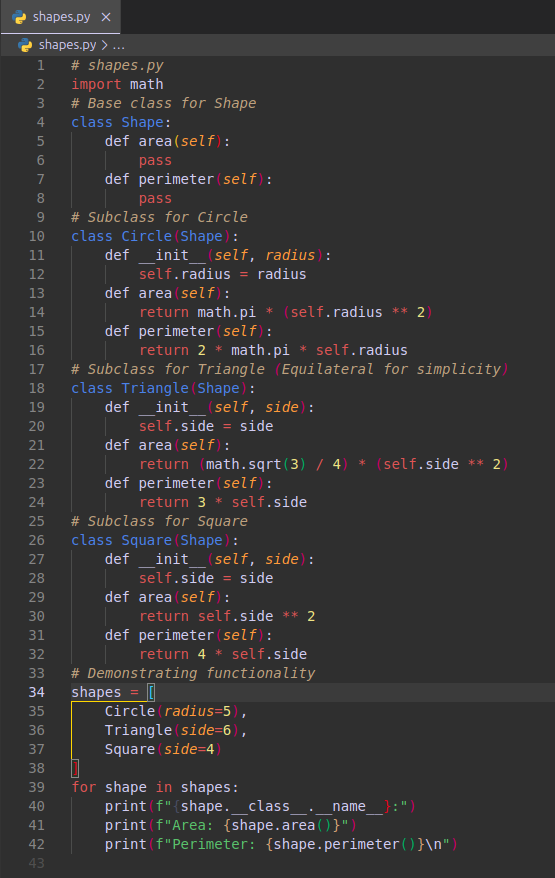
### 

### Screenshot:

\*

# Aim: Write a Python program to create a class that represents a shape. Include methods to calculate its area and perimeter. Implement subclasses for different shapes like circle, triangle, and square.

### Source Code: shapes.py



### Run or Execute: D:\> python shapes.py

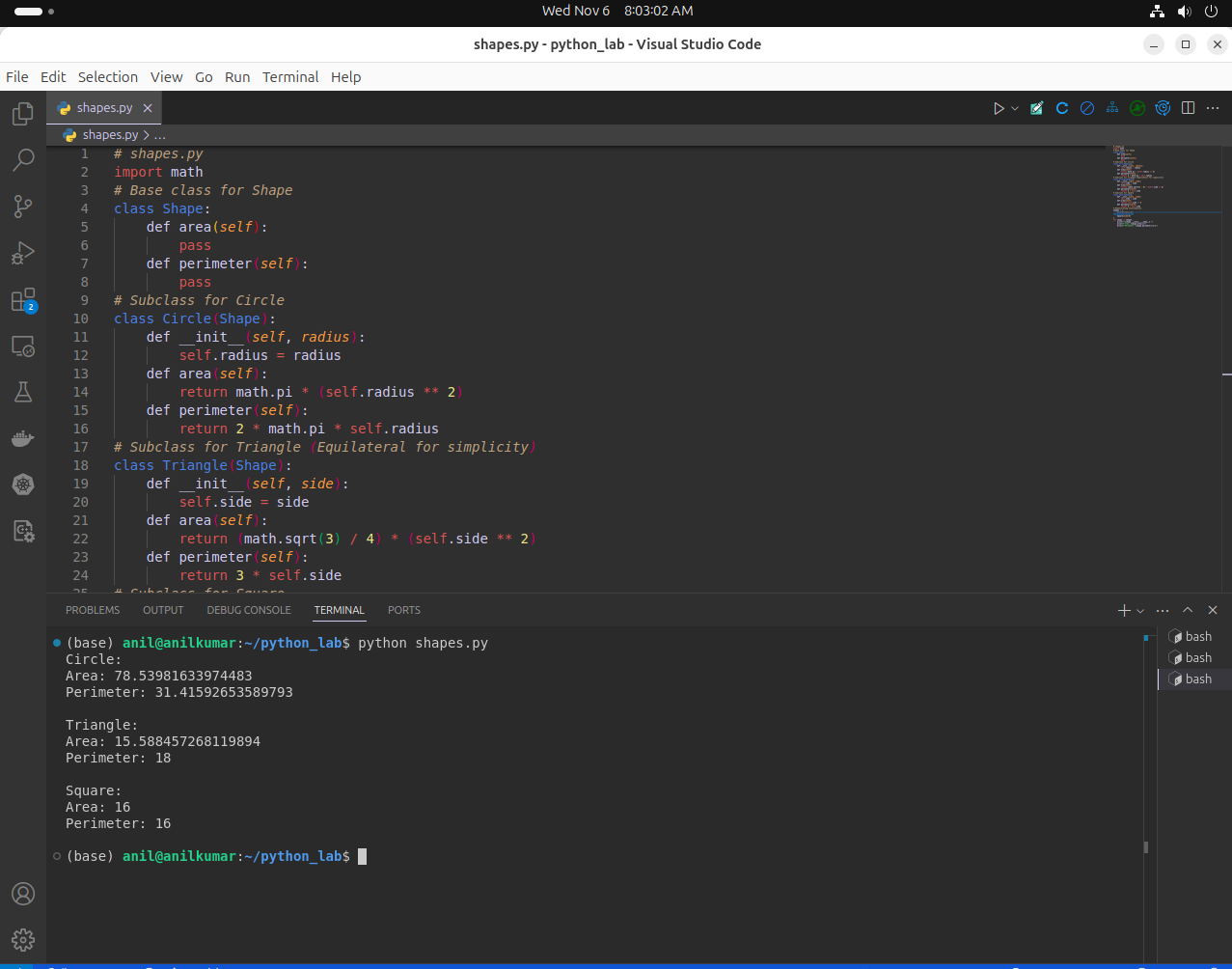
### 

### Result: Successfully generating a output of sorting words from input file to output file.

### Input and Output:

### 

### Screenshot:

\*