**OPERATORS:**

#arithmetic operator

num1 =int(input('Enter a first no:'))

num2 =int(input('Enter second no:'))

mod= num1 % num2

mul= num1 \* num2

add= num1 + num2

suc= num1 - num2

div = num1 / num2

exp = num1 \*\* num2

print(' mod of two no is:',mod)

print(' multiply of two no is:',mul)

print(' addition of two no is:',add)

print(' substraction of two no is:',suc)

print(' divition of two no is:',div)

print(' exponential of two no is:',exp)

#relational operator

print(num1>num2)

print(num1<num2)

print(num1!=num2)

print(num1>=num2)

print(num1<=num2)

print(num1==num2)

#logical operator

x=True

y=False

print('x and y is',x and y)

print('x or y is',x or y)

print('not x is',not x)

#bitwise operator

print('Bitwise and of x and y is',x&y)

print('Bitwise or of x and y is',x|y)

print('Bitwise not of x is',~x)

print('Bitwise xor of x and y is',x^y)

print('Bitwise 2 right shift of x is',x>>y)

print('Bitwise 2 left shift of x is',x<<y)

#assignment operator

x=4

print('the x value is ',x)

x+=4

print('additional assignment of x', x)

x-=4

print('Subtraction assignment of x', x)

x\*=4

print('multiplicational assignment of x',x)

x/=4

print('divitional assignment of x ',x)

x%=4

print('modules assignment of x ',x)

x=4

x//=4

print('floor divitional assignment of x',x)

x=4

x\*\*=4

print('exponential assignment of x',x)

#identity operators

print('~identity operator')

x=4

y=4

print(x is not y)

print(x is y)

#membership operator

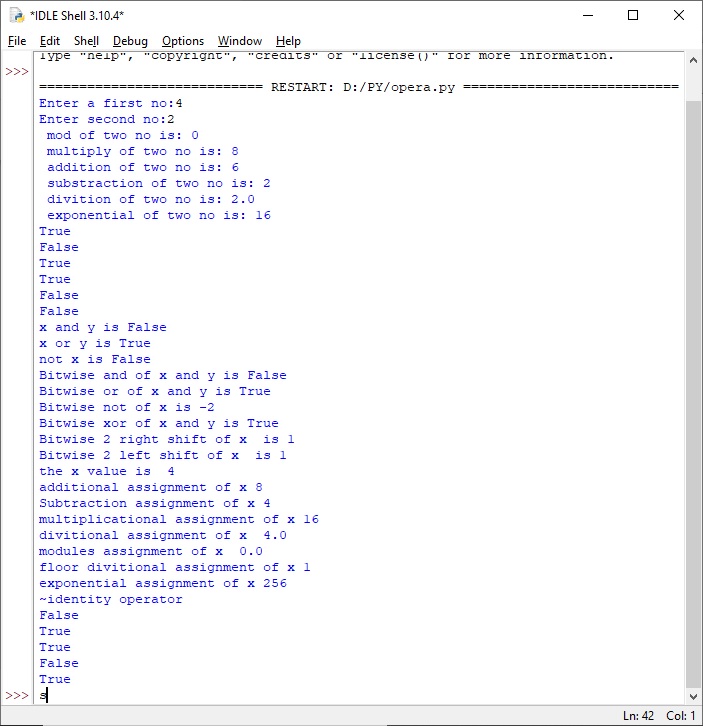
x='Boomer friend'

print('B' in x)

print('2' in x)

print('2' not in x)

**OUTPUT:**



**STRING MANIPULATION:**

word= "John Wick"

print(word)

#Accessing

word= "John Wick"

letter = word[0]

print(letter)

#length

print(len(word))

#finding char count

print (word.count('W'))

#finding char

print (word.find("J"))

#finding index

print (word.index("Wick"))

#Replacing

print (word.replace("John", "Kennady"))

#upperCase

print (word.upper())

#Reversing

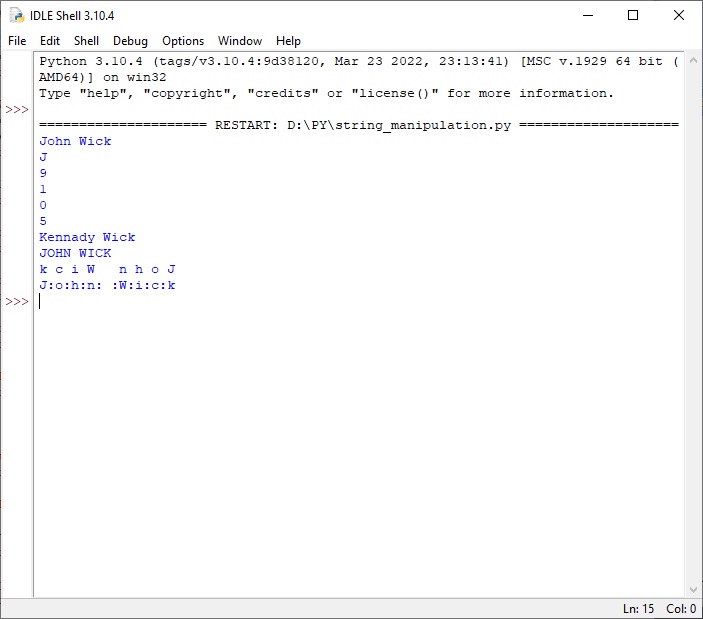
string = "John Wick"

print (' '.join(reversed(string)))

#join

print (":".join(word))

**OUTPUT:**



**QUADRATIC EQUATION:**

import math

def findRoots(a, b, c):

dis\_form = b \* b - 4 \* a \* c

sqrt\_val = math.sqrt(abs(dis\_form))

if dis\_form > 0:

print(" real and different roots ")

print((-b + sqrt\_val) / (2 \* a))

print((-b - sqrt\_val) / (2 \* a))

elif dis\_form == 0:

print(" real and same roots")

print(-b / (2 \* a))

else:

print("Complex Roots")

print(- b / (2 \* a), " + i", sqrt\_val)

print(- b / (2 \* a), " - i", sqrt\_val)

a = int(input('Enter a:'))

b = int(input('Enter b:'))

c = int(input('Enter c:'))

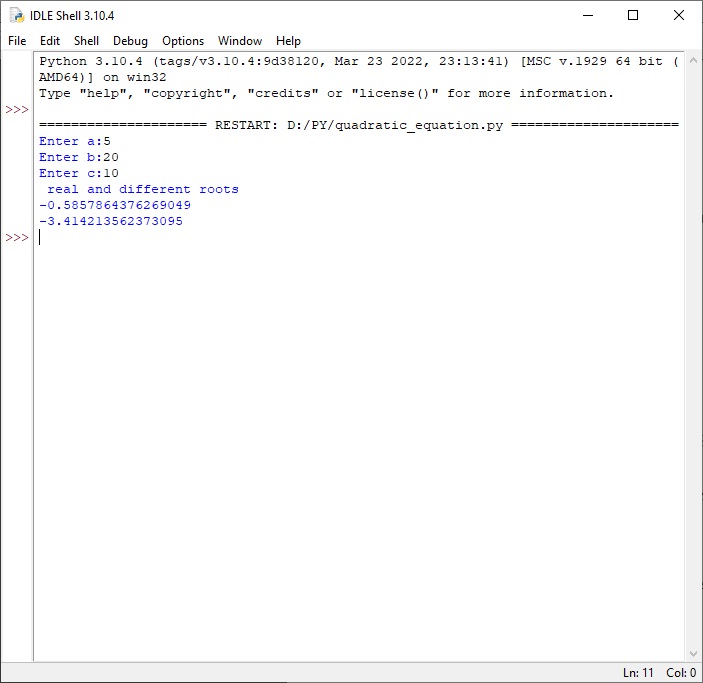
if a == 0:

print("Input correct quadratic equation")

else:

findRoots(a, b, c)

**OUTPUT:**



**BINARY SEARCH:**

def binary\_search(arr,low,high,x):

if high>=low:

mid=(high+low)//2

if arr[mid]==x:

return mid

elif arr[mid]>x:

return binary\_search(arr,low,mid-1,x)

else:

return binary\_search(arr,mid+1,high,x)

else:

return -1

arr=[2,31,42,11,48]

x=48

result=binary\_search(arr,0,len(arr)-1,x)

if result!=-1:

print('Element is present at index',str(result))

else:

print('Element is not present in array')

def binary\_search(arr,x):

low=0

high=len(arr)-1

mid=0

while low<=high:

mid=(high+low)//2

if arr[mid]<x:

low=mid+1

elif arr[mid]>x:

high=mid-1

else:

return mid

return -1

arr=[9,4,7,15,42,68,33]

x=7

result=binary\_search(arr,x)

if result!=-1:

print('Element is present at index',str(result))

else:

print('Element is not present in array')

**OUTPUT:**

