Problem 1: Roots

- Compute and print both roots of the quadratic equation x2-5.86 x+ 8.5408.

In [2]:
```
import math
```

First we load "math" onto our enviornment

In [4]:
```
def Roots():
    print("This will find the solution to a quadratic")
    print("Please enter a,b,c variables for the quadratic equation")
    a,b,c = eval(input("Enter the coefficients(a,b,c):"))
    eq = math.sqrt(b * b - 4 * a * c)
    root1 = (-b + eq) / (2 * a)
    root2 = (-b - eq) / (2 * a)
    print()
    print("The roots are:",root1,"&",root2)
```

Then we creating a function that takes takes each variable (that has been input by the user) to calculate the root of the equation inside the function

In [5]:
```
Roots()
```

```
This will find the solution to a quadratic
Please enter a,b,c variables for the quadratic equation

The roots are: 3.1400000000000006 & 2.7199999999999998
```

The roots are correct but I feel like these numbers are hard to read with so many numbers after the decimal. Therefore I decided to edit my function to round the roots to two spots after the decimal point.

In [6]:
```
def Roots():
    print("This will find the solution to a quadratic")
    print("Please enter a,b,c variables for the quadratic equation")
    a,b,c = eval(input("Enter the coefficients(a,b,c):"))
    eq = math.sqrt(b * b - 4 * a * c)
    root1 = (-b + eq) / (2 * a)
    root2 = (-b - eq) / (2 * a)
    print()
    print("The roots are:",round(root1,2),"&",round(root2,2))
```

In [7]:
```
Roots()
```

```
This will find the solution to a quadratic
Please enter a,b,c variables for the quadratic equation

The roots are: 3.14 & 2.72
```

These results are now much cleaner and easier to interpret.

Problem 2: Reciprocals

- Use a for loop to print the decimal representations of 1/2, 1/3, ..., 1/10, one on each line.

In [9]:
```
def findDecimal():
    print("Finding the reciprocals:")
    for i in range (2,11):
        print("1/{i} is equal to", 1/i)
```

In [10]:
```
findDecimal()
```

```
Finding the reciprocals:
1/{i} is equal to 0.5
1/{i} is equal to 0.3333333333333333
1/{i} is equal to 0.25
1/{i} is equal to 0.2
1/{i} is equal to 0.16666666666666666
1/{i} is equal to 0.14285714285714285
1/{i} is equal to 0.125
1/{i} is equal to 0.1111111111111111
1/{i} is equal to 0.1
```

The function I created gives use the reciprocals but with varing values after the decimals. To make this a more uniform look I decided to edit the function to round the reciprocals to 3 spots after the deciaml

In [15]:
```
def findRDecimal():
    print("Finding the reciprocals:")
    for i in range (2,11):
        print("1/{i} is equal to", round(1/i,3))
```

In [16]:
```
findRDecimal()
```

```
Finding the reciprocals:
1/{i} is equal to 0.5
1/{i} is equal to 0.333
1/{i} is equal to 0.25
1/{i} is equal to 0.2
1/{i} is equal to 0.167
1/{i} is equal to 0.143
1/{i} is equal to 0.125
1/{i} is equal to 0.111
1/{i} is equal to 0.1
```

This give our answers more oof a uniform look.

In [ ]:
```

```