SophisticatedPrints2

More math towards Python programming language

Fırst Degree Functions

```
# -*- coding: utf-8 -*-
import sympy as sym
import math
from IPython.display import display, Math
sym.init_printing()
"""
use any supported function,  as expr(x)= body of the function, \n
whose independent variable is x

FIRST DEGREE FUNCTİONS



"""
x = sym.symbols("x")

expr = 23*x-9

expr = sym.simplify(expr)

a = sym.latex(expr)
```

```
b = sym.solve(expr, x)

display(Math('The Solution \\: of :{0:} = 0 \\: \\:is : x =  {1:0.4f}' . format(a, float(b[0]))))
```

Output in the Python3 notebook

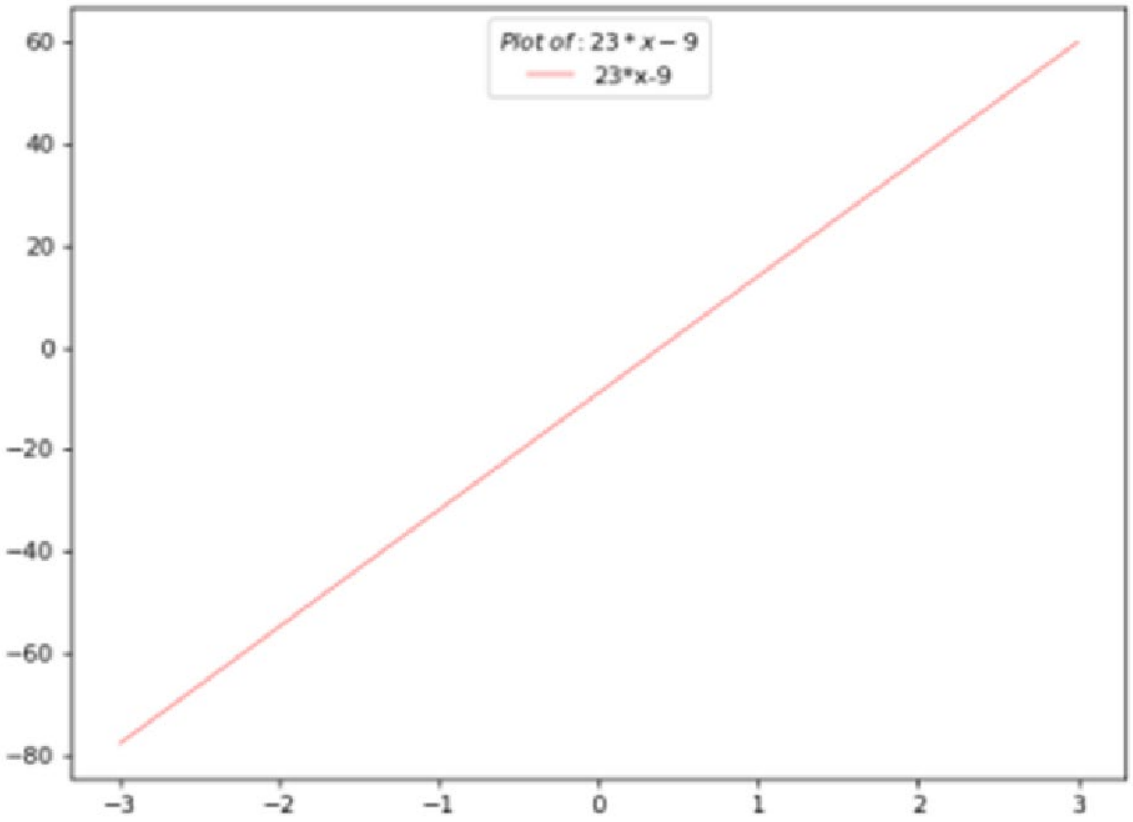
$$The Solution\ of : 23x - 9 = 0\ \ is : x = 0.3913$$

Simplest Plot Program with matplotlib in Python 3

```
from matplotlib.pyplot import *
from numpy.core.function_base import linspace

# Create a new figure of size 8x6 points, using 80 dots per inch

figure(figsize=(8,6), dpi=80)

# Data

x= linspace(-3 , 3 , 1000 , endpoint=True)

# Plot and Label

expr = 23*x-9

plot(x,expr, color="red", linewidth=0.5 ,
    linestyle="-", label='23*x-9')
# Legend
```

```
legend(loc="upper center" , title= '$Plot \\:of : 23*x-9$')

show()
```

Output of the Simplest Plot Program with Matplotlib in Python 3



Second Degree Functions

```
# -*- coding: utf-8 -*-

import sympy as sym
```

```
import math

from IPython.display import display, Math

sym.init_printing()

"""

use any supported function,  as expr(x)= body of the function, \n

whose independent variable is x


SECOND DEGREE FUNCTİONS


"""

x=sym.symbols('x')


expr = expr = x**2 -9


expr = sym.simplify(expr)


a = sym.latex(expr)


b = sym.solve(expr, x)


display(Math('The Solution \\: of :{0:} = 0 \\: \\:is : x =  {1:s} \\:,\\: {2:s}' . format(a,str(b[0]),str(b[1]))))


display(Math('The Solution \\: of :{0:} = 0 \\: \\:is : x =  {1:0.4f}\\:,\\: {2:0.4f}' .
format(a,float(b[0]),float(b[1]))))
```

Output in the Spider4

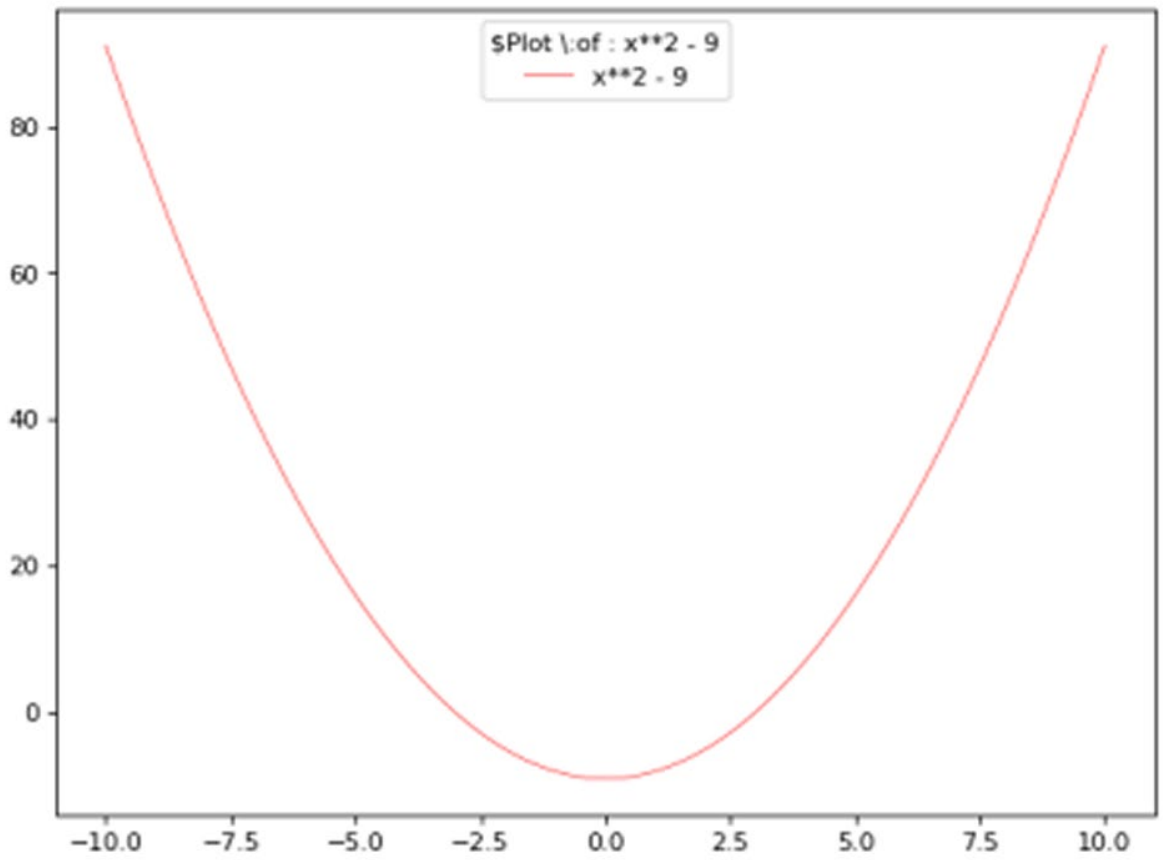
$$The Solution\ of : x^2 - 9 = 0\ \ is : x = -3\ ,\ 3$$

$$The Solution\ of : x^2 - 9 = 0\ \ is : x = -3.0000\ ,\ 3.0000$$

Output of the Simplest Plot Program with Matplotlib in Python 3

```
from matplotlib.pyplot import *
from numpy.core.function_base import linspace

# Create a new figure of size 8x6 points, using 80 dots per inch

figure(figsize=(8,6), dpi=80)

# Data

x= linspace(-10 , 10 , 1000 , endpoint=True)

# Plot and Label

expr = x**2 -9

plot(x,expr,color="red", linewidth=0.5 , linestyle="-", label='x**2 - 9')



# Legend
```

```
legend(loc="upper center" , title= '$Plot \\:of : x**2 - 9')

show()
```



Third Degree Functions

```
# -*- coding: utf-8 -*-
```

```
import sympy as sym

import math

from IPython.display import display, Math

sym.init_printing()

"""

use any supported function,  as expr(x)= body of the function, \n

whose independent variable is x


THIRD DEGREE FUNCTİONS


"""

x=sym.symbols('x')


expr = x**3-4*x + 12


expr = sym.simplify(expr)


a = sym.latex(expr)


b = sym.solve(expr, x)


display(Math('The Solution \\: of :{0:} = 0 \\: \\:is : x =  {1:s} \\:,\\: {2:s}\\:,\\: {3:s}' .
format(a,str(b[0]),str(b[1]),str(b[2]))))
```

Output in the Python3 notebook

$$The Solution\ of\ : x^3 - 4x + 12 = 0\ \ is\ : x = -4/((-1/2 - sqrt(3) * I/2) * (6 * sqrt(681) + 162) * *(1/3)) - (-1/2 - sqrt(3) * I/2) * (6 * sqrt(681) + 162) * *(1/3)/3\ , -(-1/2 + sqrt(3) * I/2) * (6 * sqrt(681) + 162) * *(1/3)/3 - 4/((-1/2 + sqrt(3) * I/2) * (6 * sqrt(681) + 162) * *(1/3))\ , -(6 * sqrt(681) + 162) * *(1/3)/3 - 4/(6 * sqrt(681) + 162) * *(1/3)$$

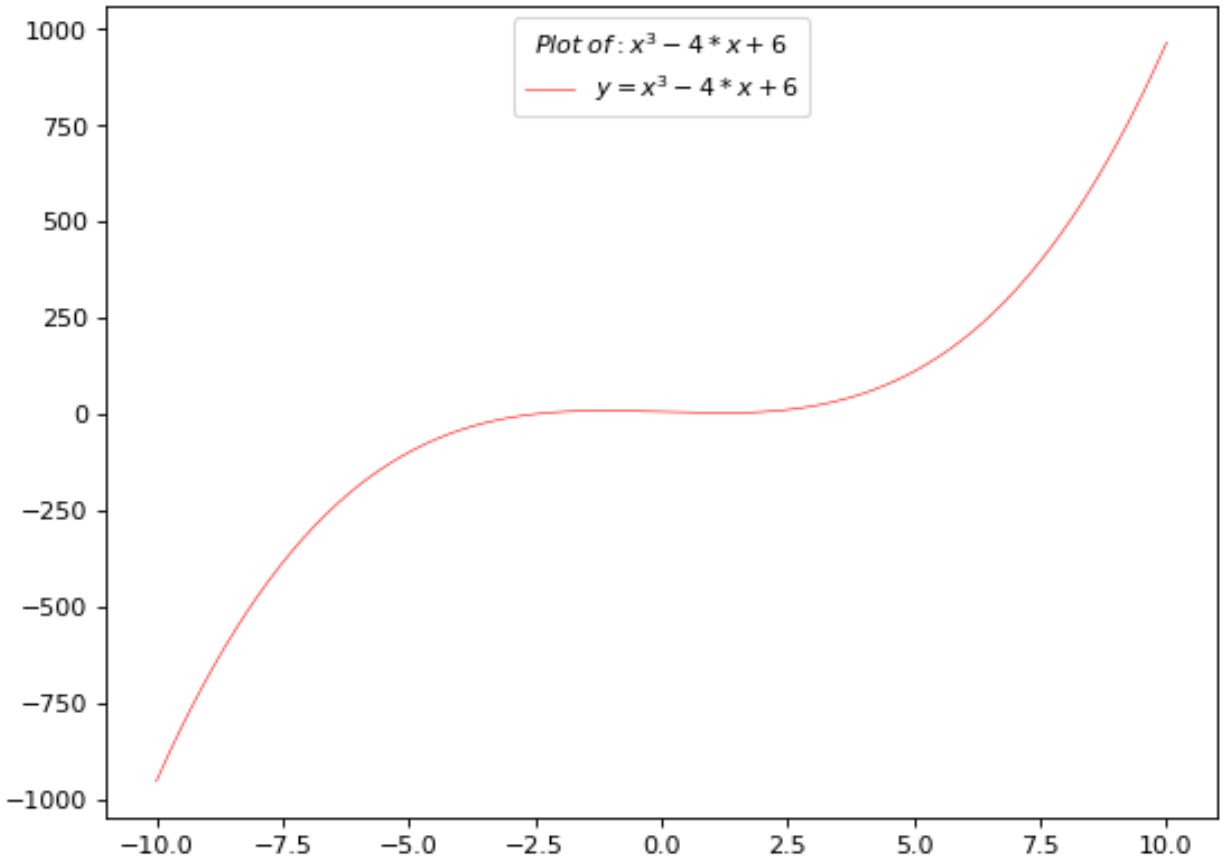
Numerical value of the only real root (third root)

```
expr = -(6*math.sqrt(681)+162)**(1/3)/3-4/(6*math.sqrt(681)+162)**(1/3)
```

```
print("{0:0.2f}".format(expr))
```

-2.86

Output of the Simplest Plot Program with Matplotlib in Python 3



Found root is justified by the plot. The only real root is at (-2.86 , 0)

Aplications

SolvingNew0.py

```
# -*- coding: utf-8 -*-

import math
```

```
import sympy as sym
from IPython.display import display, Math
sym.init_printing()
"""
use any supported function,  as expr(x)= body of the function, \n
whose independent variable is x

"""

x = sym.symbols("x")

expr = 3*x-2

expr = sym.simplify(expr)

a = sym.latex(expr)

b = sym.solve(expr, x)

display(Math('The Solution \\: of :{0:} = 0 \\: \\:is : x =  {1:0.4f}' . format(a,float(b[0]))))
```

Output in the Spider.4.py

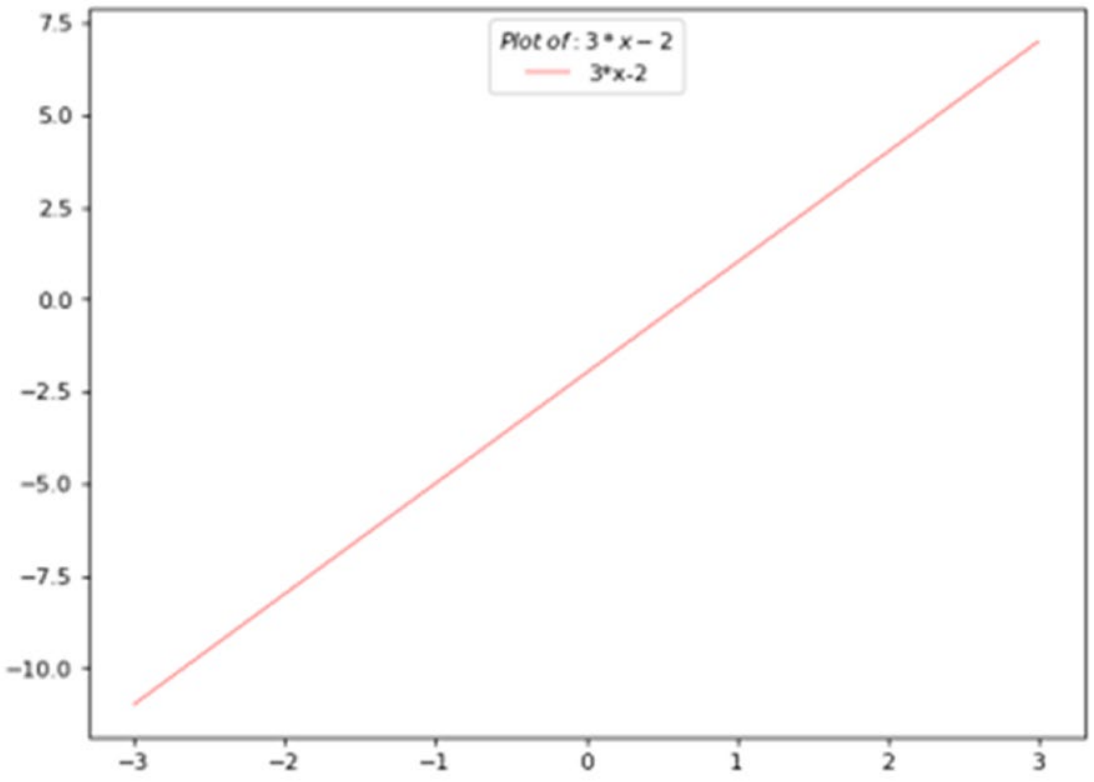
$$TheSolution\ of\ : 3x - 2 = 0\ \ is : x = 0.6667$$

Plot of this function: plotttingNew0.py

Example :

```
from matplotlib.pyplot import *
from numpy.core.function_base import linspace

# Create a new figure of size 8x6 points, using 80 dots per inch

figure(figsize=(8,6), dpi=80)

# Data

x= linspace(-3 , 3 , 1000 , endpoint=True)

# Plot and Label

plot(x,3*x-2, color="red", linewidth=0.5 ,
    linestyle="-", label='3*x-2')
# Legend

legend(loc="upper center" , title= '$Plot \\:of : 3*x-2$')


show()
```

Output in the Spider.4.py



Example :

```
from matplotlib.pyplot import *
from numpy.core.function_base import linspace
import sympy as sym

import math

# Create a new figure of size 8x6 points, using 80 dots per inch

figure(figsize=(8,6), dpi=80)
```

```
# Data


x= linspace(0,100 , 1000 , endpoint=True)


# Plot and Label


expr2 = sym.sqrt(7*sym.log(x)/sym.log(10)) - 0.3*x + 13


plot(x,sym.sqrt(7*sym.log(x)/sym.log(10)) - 0.3*x + 13, color="red", linewidth=0.5 , linestyle="-",
label='wait')

# Legend


#legend(loc="upper center" , title= '$Plot \\:of : expr')



show()
```

Output in the Spider.4.py

```
AttributeError: 'ImmutableDenseNDimArray' object has no attribute
'as_coefficient'

<Figure size 640x480 with 0 Axes>
```

However we got the solution by Mathcad Prime 4.0

$$\frac{\sqrt[2]{7\cdot\ln(x)}}{\ln(10)}-0.3\cdot x+13 \xrightarrow{simplify} 1.1490351948836456352\cdot\sqrt{\ln(x)}-0.3\cdot x+13.0$$

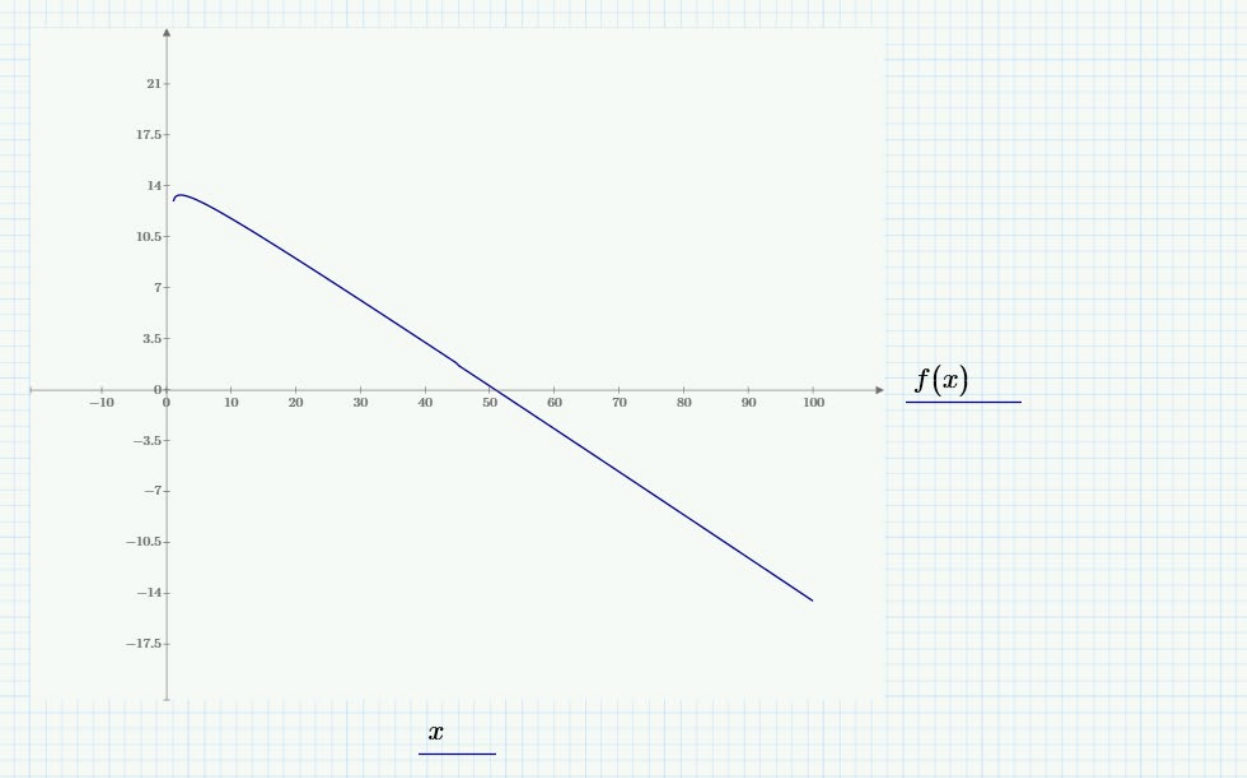
$$f(x):=1.1490351948836456352\cdot\sqrt{\ln(x)}-0.3\cdot x+13.0$$

$$f(x) \xrightarrow{solve,x} 50.926617686435427651$$

Plot in the Spider.4.py with matplotlib

```
# -*- coding: utf-8 -*-
from matplotlib.pyplot import *


from numpy.core.function_base import linspace

# Create a new figure of size 8x6 points, using 80 dots per inch

figure(figsize=(8 , 6) , dpi=80)

# Data

x = linspace(-10 , 10 , 1000 , endpoint = True)



# Plot and Label

plot(x sym.sqrt(7*(sym.log(x)/sym.log(10)) - 0.3*x + 13)
+ 6, color="red", linewidth=0.5 ,
    linestyle="-", label = "$y sym.sqrt(7*(sym.log(x)/sym.log(10)) - 0.3*x + 13)
$")

# Legend
legend(loc="upper center" , title= '$Plot\\: of : sym.sqrt(7*(sym.log(x)/sym.log(10)) - 0.3*x + 13
$')

show()
```

No plot may be produced may Mathplotlib, but, Mathcad prime 4.0 gave the plot :



The only real predicted is justified by the plot.

Example :

```
# -*- coding: utf-8 -*-
import sympy as sym
from IPython.display import display, Math
sym.init_printing()
```

```
q=sym.symbols("q")

eq = 2*q+3*q**2-5/q-4/q**3

display(Math(sym.latex(eq)))


display(Math(sym.latex(sym.simplify(eq))))

display(Math(sym.latex(sym.cancel(eq))))

display(Math(sym.latex(sym.solve(eq))))
```

$$3q^2 + 2q - \frac{5}{q} - \frac{4}{q^3}$$

$$3q^2 + 2q - \frac{5}{q} - \frac{4}{q^3}$$

$$\frac{3q^5 + 2q^4 - 5q^2 - 4}{q^3}$$

$$[\mathrm{RootOf}\left(3q^5 + 2q^4 - 5q^2 - 4, 0\right),\ \mathrm{CRootOf}\left(3q^5 + 2q^4 - 5q^2 - 4, 1\right),\ \mathrm{CRootOf}\left(3q^5 + 2q^4 - 5q^2 - 4, 2\right),\ \mathrm{CRootOf}\left(3q^5 + 2q^4 - 5q^2 - 4, 3\right),\ \mathrm{CRootOf}\left(3q^5 + 2q^4 - 5q^2 - 4, 4\right)]$$

The first line is the epression itself.

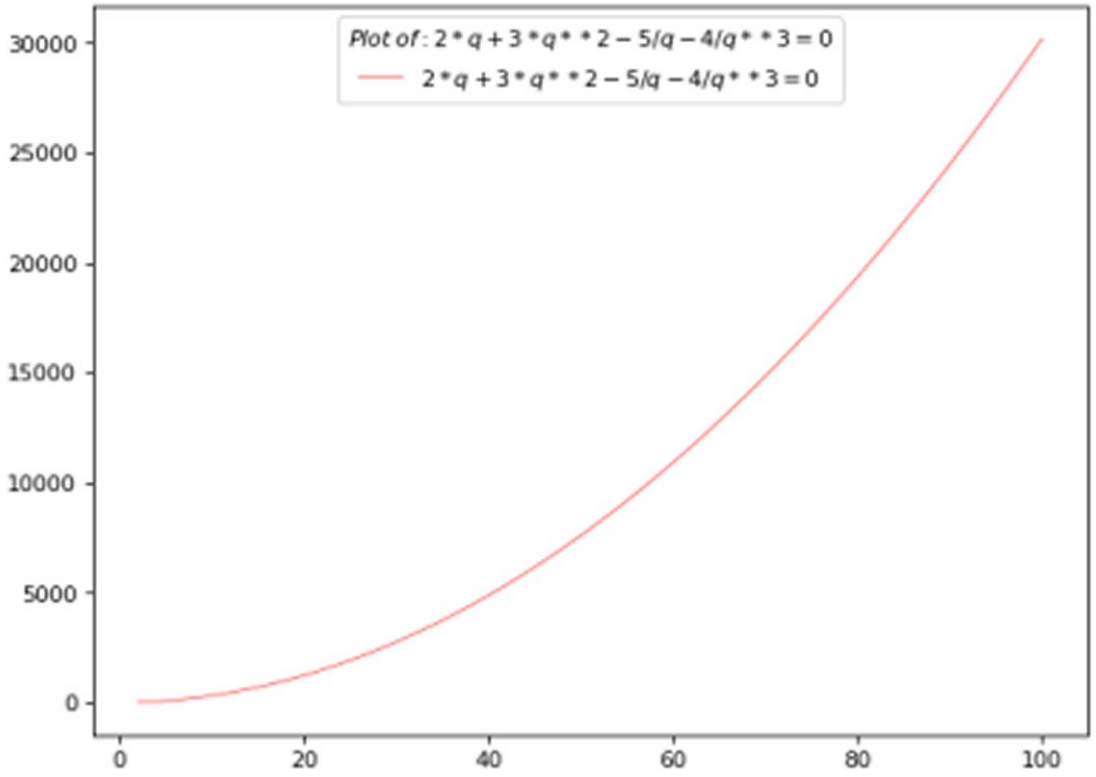
The second line is the the simplifed form of the expression. It seems that sympy found nothing to be simplifed.

The third line is an interesting construction. It finds the polinomial division which produced the original expression.

The fourth line is an attempt to finf the soluion for this function eq = 0 , found 5 roots, all of the are complex roots, the equation has no real solution.

Let us plot and see the actual real mapping.

```
from matplotlib.pyplot import *
from numpy.core.function_base import linspace
import sympy as sym

import math

# Create a new figure of size 8x6 points, using 80 dots per inch

figure(figsize=(8,6), dpi=80)

# Data

q= linspace(2,100 , 1000 , endpoint=True)

# Plot and Label

plot(q,2*q+3*q**2-5/q-4/q**3, color="red", linewidth=0.5 , linestyle="-", label='$2*q+3*q**2-5/q-4/q**3=0$')



# Legend

legend(loc="upper center" , title= '$Plot \\:of : 2*q+3*q**2-5/q-4/q**3=0$')


show()
```



The line does not cut the x axis, therefore it has no any real root.

Notice : With Python (Sympy and numpy) we get very limited support for function xeros, plotting etc..

Better support may be got from Mathcad, Matlab, Mathematica as paid programs.

From, Sage, SmathStudio, especially from Maxima and many others as free software.