

Solving Equations

In this lesson, we will learn about solving different types of equations numerically and symbolically.

WE'LL COVER THE FOLLOWING ^

- Solving for a single variable
 - Single solution
 - Multiple solutions
 - Interval solution
 - Trigonometric equations
- Solving systems of equations
- Symbolic solution

SymPy has equation solvers that can handle linear and non-linear algebraic equations as well as linear and non-linear multivariate equations. Depending upon the input equation, the answer returned can be symbolic or numeric. The most commonly used one is `solve()`. It uses the following syntax:

```
solve(f(x), x)
```

The output type of `solve()` varies with input, sometimes it returns a Python list and sometimes a Python dictionary. Usually, for single variable equations, it returns a list and for multivariable equations, it returns a dictionary.

Let's explore different cases in the examples below:

Solving for a single variable

Single solution

Let's solve the following equation that has a single solution:

$$2x + 3 = 0$$

$$2x + 3 = 0$$

```
from sympy import *

def f(x):
    return 2*x + 3

x = Symbol('x')
sol = solve(f(x), x)
print(sol)
```



The `solve()` function returns a list with the only solution.

Multiple solutions

Let's solve the following equation, which has multiple solutions:

$$x^2 - x + 6 = 0$$

```
from sympy import *

def f(x):
    return x**2 - x - 6

x = Symbol('x')
sol = solve(f(x), x)
print(sol)
```



The `solve()` function returns a list with all the solutions. Let's solve an equation with complex roots:

$$x^2 + 4x + 5 = 0$$

```
from sympy import *

def f(x):
    return x**2 + 4*x + 5

x = Symbol('x')
sol = solve(f(x), x)
print(sol)
```

The `solve()` function returns a list with all the solutions, which are complex numbers.

Interval solution

Using SymPy, we can also find a solution for an interval. All we have to do is provide the interval alongside the equation

```
solve(f(x) > a, x)
```

where `> a` is the interval. Let's see an example of interval solution:

```
from sympy import *

def f(x):
    return x - 4

def g(x):
    return x**2 - 4

x = Symbol('x')
sol1 = solve(f(x) > 0, x)
print(sol1)
sol2 = solve(g(x) > 0, x)
print(sol2)
```

In the code above (lines 10 and 11), we have computed the values of x in the interval $f(x) > 0$ and $g(x) > 0$ respectively.

Trigonometric equations

Using the SymPy module, we can solve trigonometric equations as well. Let's solve the following equation in SymPy:

$$\tan^2(x) - 3 = 0$$

```
from sympy import *

def f(x):
    return tan(x)**2 - 3
```

```
x = Symbol('x')
sol = solve(f(x), x)
print(sol)
```



Solutions are in the range $-\pi \leq x < \pi$.

Solving systems of equations

SymPy `solve()` is not just limited to solving a single variable equation, it can also solve a system of multivariable equations. This system of equations is to be in a list as the first argument of `solve()` and the variables will be in a list as the second argument of `solve()`:

```
solve([eq1, eq2,...eqN], [x1, x2, x3,...xN])
```

Let's solve the following system of equations:

$$x + y - z - 4 = 0$$

$$x - 2y + 3z + 6 = 0$$

$$2x + 3y + z - 7 = 0$$


```
from sympy import *

x, y, z = symbols('x y z')
# defining equations
eq1 = x + y - z - 4
eq2 = x - 2*y + 3*z + 6
eq3 = 2*x + 3*y + z - 7

sol = solve([eq1, eq2, eq3], [x, y, z])

print(sol)
print(type(sol))
print("x =", sol[x])
print("y =", sol[y])
print("z =", sol[z])
```



 Notice that the type of the `sol` is a dictionary where each variable is a key to its corresponding value.

Symbolic solution

All the equations that we have solved can be solved with arbitrary coefficients as well. This technique is useful when optimizing systems. See the example below:

$$x + y - z - a = 0$$

$$x - 2y + 3z + b = 0$$

$$2x + 3y + z - c = 0$$

```
from sympy import *

x, y, z = symbols('x y z')
a, b, c = symbols('a b c')
# defining equations
eq1 = x + y - z - a
eq2 = x - 2*y + 3*z + b
eq3 = 2*x + 3*y + z - c

sol = solve([eq1, eq2, eq3], [x, y, z])

print(sol)

print("x =", sol[x])
print("y =", sol[y])
print("z =", sol[z])
```



The solution that we get from these equations is in the form of `a`, `b` and `c`.

In the next lesson, we will learn about solving ordinary differential equations.