Symbols and Complex Numbers

This lesson discusses symbolic variables and complex numbers.

WE'LL COVER THE FOLLOWING

Symbolic variables

Symbolic variables, called symbols, must be defined and assigned to Python variables before they can be used. This is typically done with the `Symbol` function:

```
x = Symbol('x' )
```

To assign multiple symbols in a single function call, we use the `symbols` function:

```
x, y, z = symbols('x y z')
```

creates three symbols representing variables named x, y, and z. In this particular instance, these symbols are all assigned to Python variables of the same name. However, the user is free to assign them to different Python variables, while representing the same symbol, such as `a, b, c = symbols('x y z')`.

> 💡 It is good practice to have the same names for variables and their associated symbols.

```
from sympy import *


x = Symbol('x')
y, z = symbols('y z')

print(x + y + z)
```

We can explicitly set the properties `positive`, `real`, `imaginary` and `complex` for the symbol. This helps us simplify the expressions.

```
x = Symbol('x', real = True, positive = True)
```

> 💡 Use the `help(Symbol)` command to find more properties.

Complex numbers

SymPy never performs simplifications on an expression unless their properties allow it.

For instance, the simplification $\sqrt{x^2} = x$ holds if x is nonnegative ($x \geq 0$), but it does not hold for a general complex x.

By default, SymPy performs all calculations assuming that symbols are complex. This makes it easier to solve mathematical problems regardless of whether a symbol is complex or not. Let's look at an example of this below:

```
from sympy import *

x = Symbol('x')
print(sqrt(x**2))
```

Intuitively the answer to this should have been x, but by assuming the most general case, that x is complex by default, SymPy avoids performing mathematically invalid operations.

The imaginary unit is denoted by `I` in SymPy.

```
from sympy import *
x = 2+5*I

print(x)
print(I**2)
```

Conversion to Python complex

The type of complex number generated from `I` is `sympy.core.add.Add`. We can use the built-in `complex` function to convert a SymPy complex to the regular Python complex type.

```
from sympy import *
x = 2+5*I

print("Type of y is", type(x))

y = complex(x) # converting x to Python complex type
print("Type of y is", type(y))
```

We will learn about numerical evaluation in the next lesson.