Introduction to Computer Programming

Week 8.2: Symbolic computation with SymPy

Bristol

What is symbolic computation?

Symbolic computation is about performing exact mathematical operations that go beyond the basics like addition and multiplication (think differentiation and integration)

The idea is to introduce a new type of variable, called a **symbol**, that behaves like an algebraic variable.

Symbols can be operated on without having a precise value assigned to them.

This is different from the variable types we have seen so far (e.g. int, float), which need a value assigned to them when they are created

```
x = 1.33
v = [1, 2, 3, 4]
```

SymPy

SymPy is a Python library for carrying out exact computations using symbolic computing.

Features of SymPy include:

- Solving algebraic equations (linear equations, polynomials, nonlinear equations)
- Differentiating and integrating functions
- Solving linear algebra problems (linear systems, determinants, eigenvalue problems)
- Solving differential equations
- And much more!

Getting started

To get started, let's load the SymPy library into Python:

In [1]:
```
from sympy import *
```

Some special variables

SymPy has exact representations of useful mathematical quantities

- `pi` represents π
- `E` represents Euler's number e
- `oo` (two o's) represents infinity ∞
- `I` represents the complex number $i = \sqrt{-1}$

In [2]:
```
sin(pi)
```
Out[2]: 0

In [3]:
```
log(E)
```
Out[3]: 1

In [4]:
```
1 / oo
```
Out[4]: 0

Defining variables as symbols

In order to make use of the capabilities of SymPy, we need to define variables as symbols.

This is done using the `symbols` function

In [5]:
```
x = symbols('x')
```

This creates a variable x of type symbol.

Even though we haven't assigned a value to x, we can still perform operations on it and use it to define new variables

In [6]:
```
y = x + 1
print(y)
```
```
x + 1
```

Defining mathematical functions

Once we define a symbol, we can use it to create mathematical functions.

Example: Define the function $y(x) = \sqrt{x}$

In [7]:
```
y = sqrt(x)
```

Values of x can be substituted into y using the `subs` method

Example: Substitute $x = 4$ into $y(x)$

In [8]:
```
y.subs(x, 4)
```
Out[8]: 2

Let's see what happens when we substitute $x = 8$ into the function $y = \sqrt{x}$

In [9]:
```
y.subs(x, 8)
```
Out[9]: $2\sqrt{2}$

The number $2\sqrt{2}$ is represented exactly as a symbol rather than being approximated by a float

The `evalf` method evaluates a symbolic expression as a floating point number

In [10]:
```
y_at_8 = y.subs(x, 8)
y_at_8.evalf()
```
Out[10]: 2.82842712474619

There are some other ways we can do this. The simplest is to substitute $x = 8.0$ into y, which automatically triggers the floating-point evaluation

In [11]:
```
y.subs(x, 8.0)
```
Out[11]: 2.82842712474619

The substitution and evaluation can be done at the same time using dictionaries

In [12]:
```
y.evalf(subs = {x:8})
```
Out[12]: 2.82842712474619

Differentiating functions

The `diff` function enables functions to be differentiated an arbitrary number of times

Example: Compute y' and y''' when $y = \sqrt{x}$

In [13]:
```
diff(y,x)
```
Out[13]: $\frac{1}{2\sqrt{x}}$

In [14]:
```
diff(y, x, 3)
```
Out[14]: $\frac{3}{8x^{\frac{5}{2}}}$

Example: Evaluate $y'''(3)$ when $y(x) = \cos(x)e^{3x}$

In [15]:
```
y = cos(x) * exp(3 * x)
y_ppp = diff(y, x, 3)
y_ppp.subs(x, 3.0)
```
Out[15]: -174127.050175619

Integrating functions

The `integrate` function computes the indefinite integral of a function (if it exists)

Example: Compute the indefinite integral of $y = x$

In [16]:
```
y = x
integrate(y, x)
```
Out[16]: $\frac{x^2}{2}$

Warning: SymPy does not add the constant of integration to indefinite integrals!

The `integrate` function can also handle definite integrals.

Example: Compute

$$\int_0^1 x^2\,dx$$

In [17]:
```
integrate(x**2, (x, 0, 1))
```
Out[17]: $\frac{1}{3}$

Example: Compute

$$\int_0^\infty \frac{\sin x}{x}\,dx$$

In [18]:
```
integrate(sin(x) / x, (x, 0, oo))
```
Out[18]: $\frac{\pi}{2}$

Solving an equation

The `solve` function solves algebraic equations of the form $F(x) = 0$

Example: Solve $x^2 = -4$

Solution: First we write this as $F(x) = x^2 + 4 = 0$

In [19]:
```
F = x**2 + 4
solve(F, x)
```
Out[19]: `[-2*I, 2*I]`

Example: Find the solutions to $x^3 = ax$

In [20]:
```
a, x = symbols('a x')
F = x**3 - a*x
solve(F, x)
```
Out[20]: `[0, -sqrt(a), sqrt(a)]`

Summary

SciPy performs **exact** mathematical calculations using* **symbolic computing**

- `symbols` is used to define algebraic variables
- `subs` and `evalf` are for substituting values and computing floating-point approximations
- `diff` and `integrate` compute derivatives and integrals
- `solve` solves algebraic equations

This is very useful for calculus homework!