SymPy in Python

Symbolic Mathematics with SymPy

Symbolic Mathematics with SymPy allows us to work with algebraic expressions, solve equations symbolically, compute derivatives and integrals, and perform various other symbolic math operations in Python. SymPy is a powerful library for handling symbolic mathematics and provides functionalities that are similar to what you might find in a Computer Algebra System (CAS) like Mathematica or Maple.

1. **Introduction to SymPy**

SymPy is a Python library for symbolic computation. It allows you to define mathematical symbols, perform algebraic manipulations, and find analytical solutions to mathematical problems.

```
# Install SymPy if needed
# !pip install sympy

from sympy import symbols, Symbol
```

2. **Defining Symbols and Expressions**

Symbols are the building blocks of symbolic math in SymPy. You define variables and use them to create symbolic expressions.

```
from sympy import symbols

x, y, z = symbols('x y z')
expr = x**2 + 2*x*y + y**2  # Define a symbolic expression
```

- **Single Symbol**: `x = Symbol('x')`
- **Multiple Symbols**: `x, y, z = symbols('x y z')`
- **Subscripts**: Use `symbols('x1:4')` to create symbols `x1`, `x2`, `x3`.

3. **Simplifying Expressions**

SymPy can simplify expressions using algebraic rules to create a more compact form.

```
from sympy import simplify, expand, factor

expr = x**2 + 2*x + 1
simplified_expr = simplify(expr)          # x**2 + 2*x + 1 -> (x + 1)**2
expanded_expr = expand((x + y)**3)        # Expands to x**3 + 3*x**2*y + 3*x*y**2 +
y**3
factored_expr = factor(x**2 - y**2)       # Factorization -> (x + y)(x - y)
```

- **simplify()**: General simplification.
- **expand()**: Expands expressions.
- **factor()**: Factors expressions.

4. **Solving Equations**

SymPy can solve both algebraic and transcendental equations symbolically.

```
from sympy import Eq, solve

equation = Eq(x**2 - 4, 0)   # Defines equation x**2 - 4 = 0
solutions = solve(equation)  # Solves for x, returns [2, -2]
```

- **Eq()**: Defines an equation.
- **solve()**: Solves equations for variables.

Systems of Equations: SymPy can also solve multiple equations.

```
eq1 = Eq(x + y, 10)
eq2 = Eq(x - y, 4)
solutions = solve((eq1, eq2), (x, y))  # Solves for x and y
```

5. **Calculus with SymPy**

SymPy provides tools for symbolic differentiation and integration.

a. Differentiation

```
from sympy import diff

expr = x**3 + 3*x**2 + 5
derivative = diff(expr, x)  # Differentiate with respect to x
```

- **Higher-order Derivatives**: Use `diff(expr, x, n)` to differentiate `n` times.

  ```
  second_derivative = diff(expr, x, 2)  # Second derivative
  ```

b. Integration

```
from sympy import integrate
```

```
integral = integrate(expr, x)          # Indefinite integral
definite_integral = integrate(expr, (x, 0, 2))  # Definite integral from 0 to 2
```

- **Indefinite Integrals**: `integrate(expr, x)`
- **Definite Integrals**: `integrate(expr, (x, a, b))`

6. **Limits and Series Expansions**

a. Limits

Compute limits of expressions as a variable approaches a particular value.

```
from sympy import limit

limit_expr = limit((x**2 - 1) / (x - 1), x, 1)  # Calculates limit as x -> 1
```

b. Series Expansion

Find the Taylor or Maclaurin series expansion of an expression.

```
series_expr = expr.series(x, 0, 5)  # Expands expr around x=0 up to x^4
```

- **series()**: Expands around a point to a specified order.

7. **Linear Algebra Operations**

SymPy supports symbolic linear algebra, allowing you to work with matrices and solve systems of linear equations.

```
from sympy import Matrix

A = Matrix([[2, 1], [1, 3]])
B = Matrix([5, 10])

# Basic matrix operations
C = A + B                       # Matrix addition
product = A * B                 # Matrix multiplication
determinant = A.det()           # Determinant of matrix A
inverse = A.inv()               # Inverse of matrix A
solution = A.solve(B)           # Solves Ax = B for x
```

8. **Working with Polynomials**

Polynomials can be defined and manipulated using SymPy. This includes finding roots, expanding, factoring, and simplifying.

```
from sympy import Poly

poly = Poly(x**3 + 3*x**2 + 3*x + 1, x)
roots = poly.roots()            # Finds the roots of the polynomial
```

9. **Discrete Mathematics Functions**

SymPy supports combinatorial functions and sequences.

- **Factorials and Combinations**:

```
from sympy import factorial, binomial

factorial_5 = factorial(5)      # 5!
combinations = binomial(5, 3)   # "5 choose 3"
```

- **Summation and Products**:

```
from sympy import summation, Product

summation_expr = summation(x**2, (x, 1, 5))  # Sum of x^2 from x=1 to 5
product_expr = Product(x, (x, 1, 5)).doit()  # Product of x from 1 to 5
```

10. **Plotting with SymPy**

SymPy integrates with Matplotlib to enable symbolic plotting.

```
from sympy.plotting import plot

plot(x**2, (x, -10, 10))  # Plots x^2 from -10 to 10
```

Summary Table of Key Functions in SymPy

Concept	Function	Example	Description
Symbols	`symbols`, `Symbol`	`x, y = symbols('x y')`	Defines symbolic variables
Simplification	`simplify`, `expand`, `factor`	`simplify(x**2 + 2*x + 1)`	Simplifies expressions
Solving Equations	`Eq`, `solve`	`solve(Eq(x**2 - 4, 0))`	Solves equations
Differentiation	`diff`	`diff(x**3, x)`	Finds derivatives

Concept	Function	Example	Description
Integration	`integrate`	`integrate(x**2, (x, 0, 1))`	Finds integrals
Limits	`limit`	`limit((x**2 - 1)/(x - 1), x, 1)`	Calculates limits
Series Expansion	`series`	`series(sin(x), x, 0, 4)`	Finds series expansion
Linear Algebra	`Matrix`, `.det()`, `.inv()`	`Matrix([[2, 1], [1, 3]])`	Matrix operations
Polynomials	`Poly`, `.roots()`	`Poly(x**2 - 4)`	Works with polynomials
Discrete Math	`factorial`, `binomial`	`factorial(5)`, `binomial(5, 2)`	Factorials, combinations, summations
Plotting	`plot`	`plot(x**2, (x, -10, 10))`	Symbolic plotting