

# MikoshiLang

## Reference Manual & User Guide

Mikoshi Ltd

Version 3.3.0 — February 22, 2026

### Abstract

MikoshiLang is a symbolic computation language for Python, inspired by Wolfram Language. It provides **6,324 functions** across **40+ domains** including calculus, linear algebra, number theory, statistics, chemistry, visualization, machine learning, physics, cryptography, biology, engineering (mechanical, electrical, civil, aerospace, chemical), hydrology, mining, manufacturing, renewable energy, and more. Unlike traditional computational tools, MikoshiLang combines natural language processing with deterministic code execution — AI proposes, the engine proves. No hallucinated math.

Try it online: <https://mikoshi.co.uk/mikoshilang/console>  
GitHub: <https://github.com/DarrenEdwards111/MikoshiLang>  
PyPI: `pip install mikoshilang`

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	Key Features	5
1.2	Installation	5
1.3	Quick Start	5
<b>2</b>	<b>Knowledge Layer (v3.3.0)</b>	<b>6</b>
2.1	Architecture	6
2.2	Core Knowledge Functions	6
2.2.1	EntitySearch	6
2.2.2	EntityValue	6
2.2.3	EntityRelationships	7
2.2.4	WikipediaText	7
2.3	Domain Packs	7
2.3.1	PubChem (Chemistry)	7
2.3.2	Crossref (Scholarly Papers)	7
2.3.3	OpenAlex (Scholarly Graph)	7
2.3.4	GeoNames (Geographic Data)	8
2.3.5	World Bank (Economic Indicators)	8
2.4	Caching & Provenance	8
2.5	License Compliance	8
<b>3</b>	<b>Syntax</b>	<b>9</b>
3.1	Basic Expressions	9
3.2	Function Calls	9
3.3	Pattern Matching	9
<b>4</b>	<b>Core Functions</b>	<b>9</b>
4.1	Calculus	9
4.1.1	Differentiation	9
4.1.2	Integration	9
4.1.3	Limits	10
4.2	Algebra	10
4.2.1	Solving Equations	10
4.2.2	Simplification	10
4.3	Linear Algebra	10
4.4	Number Theory	10
4.5	Statistics	10
<b>5</b>	<b>Visualization</b>	<b>11</b>
5.1	2D Plotting	11
5.2	3D Plotting	11
5.3	Interactive Plots	11
<b>6</b>	<b>Chemistry</b>	<b>11</b>
6.1	Equation Balancing	11
6.2	Molecular Mass	11
6.3	Element Data	12
<b>7</b>	<b>Unit Conversion</b>	<b>12</b>

<b>8</b>	<b>Natural Language Mode</b>	<b>12</b>
8.1	How It Works	12
8.2	Examples	12
8.3	Web Console	13
<b>9</b>	<b>API Usage</b>	<b>13</b>
9.1	Python Library	13
9.2	REST API	13
9.3	Jupyter Kernel	13
<b>10</b>	<b>Function Reference</b>	<b>13</b>
10.1	Version History	14
10.2	Extended Function Sets (v2.0+)	14
10.3	Domain Coverage	15
10.4	Calculus (67 functions)	15
10.5	Algebra (89 functions)	15
10.6	Linear Algebra (76 functions)	15
10.7	Number Theory (94 functions)	15
10.8	Statistics (112 functions)	16
10.9	Trigonometry (48 functions)	16
10.10	Special Functions (97 functions)	16
10.11	Discrete Math (68 functions)	16
10.12	Logic & Sets (34 functions)	16
10.13	String Operations (52 functions)	16
10.14	List Manipulation (89 functions)	16
10.15	Graph Theory (64 functions)	16
10.16	Chemistry (118 elements + functions)	17
10.17	Units (50+ units)	17
10.18	Visualization (9 functions)	17
10.19	Numerical Methods (43 functions)	17
10.20	Optimization (37 functions)	17
10.21	Control Theory (28 functions)	17
10.22	Signal Processing (46 functions)	17
10.23	Time & Date (24 functions)	17
10.24	Random & Probability (38 functions)	17
10.25	Financial Math (21 functions)	18
<b>11</b>	<b>New Domain Examples (v2.0)</b>	<b>18</b>
11.1	Vector Calculus	18
11.2	Machine Learning	18
11.3	Physics	18
11.4	Biology & Genetics	19
11.5	Cryptography	19
11.6	Numerical Methods	19
<b>12</b>	<b>Advanced Topics</b>	<b>20</b>
12.1	Custom Rules	20
12.2	Pattern Matching	20
12.3	Performance Tips	20

<b>13 Troubleshooting</b>	<b>20</b>
13.1 Common Errors . . . . .	20
13.2 Getting Help . . . . .	21
<b>14 Contributing</b>	<b>21</b>
<b>15 License</b>	<b>21</b>
<b>16 Acknowledgments</b>	<b>21</b>

# 1 Introduction

MikoshiLang is a symbolic computation language designed for Python, providing Wolfram-style syntax with deterministic evaluation. Built on SymPy, it extends Python's mathematical capabilities with pattern matching, rule-based transformations, and natural language processing.

## 1.1 Key Features

- **6,324 functions** across 40+ domains: calculus, algebra, number theory, statistics, chemistry, physics, biology, ML, cryptography, engineering, hydrology, mining, manufacturing, renewable energy, and more
- **Natural language mode** — type plain English, get verified results
- **Pattern matching & rules** — symbolic manipulation with custom transformations
- **Jupyter kernel** — interactive notebooks with rich output
- **Visualization** — 2D/3D plotting with matplotlib & plotly
- **Chemistry** — equation balancing, molecular mass, 118 elements
- **Physics** — mechanics, thermodynamics, E&M, quantum, relativity
- **Machine learning** — regression, neural nets, loss functions, optimizers
- **Biology** — genetics, DNA analysis, population dynamics
- **Cryptography** — hashing, modular arithmetic, number theory
- **Unit conversion** — 50+ physical units with dimensional analysis
- **No hallucinations** — AI translates natural language to code, engine verifies

## 1.2 Installation

```
1 # Standard installation
2 pip install mikoshilang
3
4 # With visualization support
5 pip install mikoshilang[visualization]
6
7 # All optional dependencies
8 pip install mikoshilang[all]
```

## 1.3 Quick Start

```
1 from mikoshilang import parse_and_eval
2
3 # Basic arithmetic
4 parse_and_eval("2 + 2")
5 # => 4
6
7 # Symbolic calculus
8 parse_and_eval("D[x^3 + 2*x, x]")
9 # => 3*x^2 + 2
10
11 # Equation solving
12 parse_and_eval("Solve[x^2 - 4 == 0, x]")
```

```

13 # => {-2, 2}
14
15 # Plotting
16 parse_and_eval("Plot[sin(x), {x, 0, 6.28}]")
17 # => <base64 PNG image>

```

## 2 Knowledge Layer (v3.3.0)

MikoshiLang v3.3.0 introduces a complete knowledge layer with deterministic fact retrieval from multiple sources. Unlike LLMs that answer from memory, MikoshiLang queries structured databases and returns provenance-tracked results.

### 2.1 Architecture

The knowledge layer implements 8 core components:

1. **Entity Graph** — Wikidata (QIDs, 50 canonical properties)
2. **Text Layer** — Wikipedia (summaries, citations)
3. **5 Core Tools** — Search, facts, relationships, text, time-travel
4. **LLM Interpreter** — Natural language → structured queries
5. **Persistent Cache** — SQLite with TTL (7d Wikidata, 24h Wikipedia)
6. **Canonical Properties** — Stable schema (BirthDate, Occupation, etc.)
7. **Domain Packs** — PubChem, Crossref, OpenAlex, GeoNames, World Bank
8. **License Compliance** — Full attribution (CC0, CC BY-SA, Public Domain)

### 2.2 Core Knowledge Functions

#### 2.2.1 EntitySearch

Search Wikidata entities:

```

1 EntitySearch["Douglas Adams"]
2 # => [{"id": "Q42", "label": "Douglas Adams",
3 #      "description": "British science fiction writer...",
4 #      "license": "CC0"}]
5
6 EntitySearch["Moon", "Film", 10] # Type hint, limit 10

```

#### 2.2.2 EntityValue

Get entity properties:

```

1 EntityValue["Q42", "BirthDate"]
2 # => {"value": "1952-03-11", "license": "CC0",
3 #      "source": "https://www.wikidata.org/wiki/Q42"}
4
5 # Time-travel queries
6 EntityValue["Q42", "Occupation", as_of="2015-01-01"]

```

**Canonical Properties (50):** BirthDate, DeathDate, Occupation, Country, Population, ChemicalFormula, AtomicNumber, ISBN, DOI, and more.

### 2.2.3 EntityRelationships

SPARQL-powered graph traversal:

```
1 EntityRelationships["Q42", "Occupation", depth=1, limit=10]
2 # => [{"related": [{"id": "Q36180", "label": "writer"}, ...],
3 #     "license": "CC0"}]
```

### 2.2.4 WikipediaText

Get Wikipedia summaries with attribution:

```
1 WikipediaText["Douglas Adams", sentences=3]
2 # => [{"text": "Douglas Noel Adams was...",
3 #     "license": "CC BY-SA 3.0",
4 #     "attribution": "Source: Wikipedia contributors"}]
```

## 2.3 Domain Packs

MikoshiLang includes 5 domain-specific knowledge sources:

### 2.3.1 PubChem (Chemistry)

```
1 PackSearch["pubchem", "caffeine"]
2 # => [{"id": "CID2519", "source": "PubChem",
3 #     "license": "Public Domain"}]
4
5 PackValue["pubchem", "2519", "MolecularFormula"]
6 # => {"value": "C8H10N4O2"}
```

**Coverage:** 100M+ compounds

**Properties:** MolecularFormula, MolecularWeight, IUPACName, CanonicalSMILES

### 2.3.2 Crossref (Scholarly Papers)

```
1 PackSearch["crossref", "machine learning", limit=5]
2 # => [{"id": "10.1145/...", "label": "Paper Title",
3 #     "license": "Metadata: CC0"}]
4
5 PackValue["crossref", "10.1145/3422622", "Authors"]
6 # => {"value": ["Author 1", "Author 2"]}
```

**Coverage:** 70M+ papers

**Properties:** Title, Authors, PublicationDate, Abstract, CitationCount

### 2.3.3 OpenAlex (Scholarly Graph)

```
1 PackSearch["openalex", "deep learning"]
2 # => [{"id": "W2741809807", "license": "CC0"}]
3
4 PackValue["openalex", "W2741809807", "CitationCount"]
5 # => {"value": 1523}
```

**Coverage:** 200M+ works

**Properties:** Title, Authors, CitationCount, Concepts, OpenAccessURL

### 2.3.4 GeoNames (Geographic Data)

```
1 PackSearch["geonames", "London"]
2 # => [{"id": "2643743", "license": "CC BY 4.0"}]
3
4 PackValue["geonames", "2643743", "Population"]
5 # => {"value": 8961989}
```

**Coverage:** 11M+ places

**Properties:** Country, Population, Latitude, Longitude, Elevation, Timezone

### 2.3.5 World Bank (Economic Indicators)

```
1 PackSearch["worldbank", "GDP"]
2 # => [{"id": "NY.GDP.MKTP.CD", "license": "CC BY 4.0"}]
3
4 PackValue["worldbank", "NY.GDP.MKTP.CD", "Value", country="US"]
5 # => {"value": 25462700000000, "date": "2022"}
```

**Coverage:** 1,400+ indicators

**Properties:** Value, Date, Country, Unit

## 2.4 Caching & Provenance

All knowledge queries are cached in ~/.mikoshilang/knowledge\_cache/knowledge.db (SQLite) with:

- **TTL:** 7 days (Wikidata), 24 hours (Wikipedia)
- **Provenance:** Source URL, retrieved timestamp, license
- **Performance:** 200x speedup on cached queries

```
1 CacheStats[]
2 # => {"total_entries": 1523, "size_mb": 8.4}
3
4 CacheClear[30] # Clear entries older than 30 days
```

## 2.5 License Compliance

Every result includes license metadata:

- **Wikidata:** CC0 (public domain)
- **Wikipedia:** CC BY-SA 3.0 (attribution required)
- **PubChem:** Public Domain
- **Crossref:** Metadata CC0
- **OpenAlex:** CC0
- **GeoNames:** CC BY 4.0
- **World Bank:** CC BY 4.0



## 3 Syntax

MikoshiLang uses Wolfram-inspired syntax with square brackets for function calls and curly braces for lists.

### 3.1 Basic Expressions

```

1 # Arithmetic
2 2 + 3 * 4      # => 14
3 2^10           # => 1024
4 Sqrt[16]       # => 4
5
6 # Variables
7 x + 2*y - z    # => x + 2*y - z
8
9 # Lists
10 {1, 2, 3, 4}   # => [1, 2, 3, 4]
11 Range[1, 10]  # => [1, 2, 3, ..., 10]
```

### 3.2 Function Calls

```

1 # Single argument
2 Sin[Pi/2]      # => 1
3 Exp[1]         # => E
4
5 # Multiple arguments
6 GCD[48, 18]    # => 6
7 Mod[17, 5]     # => 2
8
9 # Nested calls
10 Sqrt[Abs[-16]] # => 4
```

### 3.3 Pattern Matching

```

1 # Define a rule
2 rule = x^2 + 2*x + 1 -> (x + 1)^2
3
4 # Apply rule
5 ReplaceAll[x^2 + 2*x + 1, rule]
6 # => (x + 1)^2
```

## 4 Core Functions

### 4.1 Calculus

#### 4.1.1 Differentiation

```

1 D[x^3, x]           # First derivative
2 D[x^3 + 2*x^2, x, 2] # Second derivative
```

**Available:** D, Derivative, PartialD, Gradient, Divergence, Curl, Laplacian

#### 4.1.2 Integration

```

1 Integrate[x^2, x]      # Indefinite integral
2 Integrate[x^2, {x, 0, 1}] # Definite integral (0 to 1)
```

**Available:** Integrate, NIntegrate, LineIntegral, DoubleIntegral, TripleIntegral

### 4.1.3 Limits

```
1 Limit[sin(x)/x, x, 0]      # => 1
2 Limit[1/x, x, Infinity]    # => 0
```

Available: Limit, Series, TaylorSeries, LaurentSeries

## 4.2 Algebra

### 4.2.1 Solving Equations

```
1 Solve[x^2 - 4 == 0, x]      # => {-2, 2}
2 Solve[x^2 + x + 1 == 0, x]  # Complex solutions
3 NSolve[cos(x) == x, x]     # Numerical solver
```

Available: Solve, NSolve, DSolve, RSolve, Roots, FindRoot

### 4.2.2 Simplification

```
1 Simplify[(x^2 - 1)/(x - 1)] # => x + 1
2 Expand[(x + 1)^3]           # => x^3 + 3*x^2 + 3*x + 1
3 Factor[x^2 - 4]             # => (x - 2)*(x + 2)
```

Available: Simplify, FullSimplify, Expand, Factor, Apart, Together, Cancel, Collect

## 4.3 Linear Algebra

```
1 # Determinant
2 Det[{{1, 2}, {3, 4}}]      # => -2
3
4 # Eigenvalues
5 Eigenvalues[{{1, 2}, {2, 1}}] # => [-1, 3]
6
7 # Matrix multiplication
8 Dot[{{1, 2}, {3, 4}}, {{5}, {6}}] # => {{17}, {39}}
```

Available: Det, Inverse, Transpose, Eigenvalues, Eigenvectors, Dot, Cross, MatrixPower, MatrixRank, Trace, Norm

## 4.4 Number Theory

```
1 Prime[100]                  # 100th prime number
2 PrimeQ[17]                  # True (17 is prime)
3 FactorInteger[60]           # => {{2, 2}, {3, 1}, {5, 1}}
4 GCD[48, 18]                 # => 6
5 LCM[12, 18]                 # => 36
```

Available: Prime, PrimeQ, NextPrime, PrimePi, FactorInteger, Divisors, GCD, LCM, EulerPhi, MoebiusMu, Fibonacci, Lucas

## 4.5 Statistics

```
1 Mean[{1, 2, 3, 4, 5}]      # => 3
2 Median[{1, 2, 3, 4, 5}]    # => 3
3 StandardDeviation[{1, 2, 3}] # => sqrt(2/3)
4 Variance[{1, 2, 3, 4}]     # => 5/4
```

Available: Mean, Median, Mode, Variance, StandardDeviation, Quantile, Quartiles, Correlation, Covariance, Skewness, Kurtosis

## 5 Visualization

MikoshiLang supports 2D and 3D plotting with matplotlib (static) or plotly (interactive).

### 5.1 2D Plotting

```
1 # Simple plot
2 Plot[sin(x), {x, 0, 6.28}]
3
4 # Multiple functions
5 Plot[{sin(x), cos(x)}, {x, 0, 6.28}]
6
7 # Parametric plot
8 ParametricPlot[{cos(t), sin(t)}, {t, 0, 6.28}]
9
10 # Polar plot
11 PolarPlot[1 + cos(theta), {theta, 0, 6.28}]
```

### 5.2 3D Plotting

```
1 # 3D surface
2 Plot3D[sin(x)*cos(y), {x, -3, 3}, {y, -3, 3}]
3
4 # Contour plot
5 ContourPlot[x^2 + y^2, {x, -2, 2}, {y, -2, 2}]
6
7 # Vector field
8 VectorFieldPlot[{-y, x}, {x, -2, 2}, {y, -2, 2}]
```

### 5.3 Interactive Plots

```
1 # Use plotly for interactive 3D
2 Plot3D[x^2 + y^2, {x, -2, 2}, {y, -2, 2}, interactive=True]
```

## 6 Chemistry

### 6.1 Equation Balancing

```
1 BalanceEquation["H2 + O2 -> H2O"]
2 # => 2H2 + O2 -> 2H2O
3
4 BalanceEquation["C6H12O6 -> CO2 + H2O"]
5 # => C6H12O6 -> 6CO2 + 6H2O
```

### 6.2 Molecular Mass

```
1 MolecularMass["H2O"]           # => 18.015 g/mol
2 MolecularMass["C6H12O6"]       # => 180.156 g/mol (glucose)
3 MolecularMass["NaCl"]          # => 58.443 g/mol
```

## 6.3 Element Data

```
1 Element["Fe"]
2 # => {
3 #   name: Iron,
4 #   number: 26,
5 #   mass: 55.845,
6 #   electronConfiguration: [Ar] 3d6 4s2,
7 #   category: transition metal
8 # }
```

## 7 Unit Conversion

MikoshiLang supports 50+ physical units across length, mass, time, temperature, energy, power, pressure, and more.

```
1 Convert[100, "miles", "km"]      # => 160.934 km
2 Convert[32, "fahrenheit", "celsius"] # => 0 C
3 Convert[1, "atm", "pascal"]      # => 101325 Pa
4 Convert[1, "kWh", "joules"]      # => 3.6e6 J
```

**Available units:** meters, km, miles, feet, inches, kg, pounds, ounces, seconds, hours, days, celsius, fahrenheit, kelvin, joules, calories, kWh, watts, horsepower, atm, bar, pascal, liters, gallons

## 8 Natural Language Mode

MikoshiLang includes an AI-powered natural language interface. Type plain English, get verified results.

### 8.1 How It Works

1. User types natural language query
2. AI translates to MikoshiLang code
3. Code is executed by the deterministic engine
4. Result is verified and returned

### 8.2 Examples

```
1 # Natural language input
2 "What is the derivative of x cubed?"
3 # AI translates to: D[x^3, x]
4 # Engine returns: 3*x^2
5
6 "Solve x squared equals 4"
7 # AI translates to: Solve[x^2 == 4, x]
8 # Engine returns: {-2, 2}
9
10 "What is the molecular mass of water?"
11 # AI translates to: MolecularMass["H2O"]
12 # Engine returns: 18.015 g/mol
```

## 8.3 Web Console

Try the interactive web console at:

<https://mikoshi.co.uk/mikoshilang/console>

Features auto-detection — if your input looks like natural language, it routes to the NL endpoint automatically.

## 9 API Usage

### 9.1 Python Library

```
1 from mikoshilang import parse_and_eval, evaluate
2 from mikoshilang.parser import parse
3
4 # Parse and evaluate
5 result = parse_and_eval("D[x^2, x]")
6 print(result) # => 2*x
7
8 # Parse only (returns AST)
9 ast = parse("x^2 + 2*x + 1")
10 print(ast) # => BinOp(...)
11
12 # Evaluate parsed AST
13 result = evaluate(ast)
14 print(result) # => x^2 + 2*x + 1
```

### 9.2 REST API

MikoshiLang is deployed at [mikoshi.co.uk](https://mikoshi.co.uk) with a REST API:

```
1 # Evaluate expression
2 curl -X POST https://mikoshi.co.uk/api/mikoshilang/eval \
3   -H "Content-Type: application/json" \
4   -d '{"input": "D[x^3, x]", "mode": "parse"}'
5
6 # Natural language
7 curl -X POST https://mikoshi.co.uk/api/mikoshilang/nl \
8   -H "Content-Type: application/json" \
9   -d '{"input": "What is the integral of x squared?"}'
10
11 # Chemistry
12 curl -X POST https://mikoshi.co.uk/api/mikoshilang/chemistry \
13   -H "Content-Type: application/json" \
14   -d '{"action": "balance", "input": "H2 + O2 -> H2O"}'
```

### 9.3 Jupyter Kernel

```
1 # Install kernel
2 python -m mikoshilang.kernel install
3
4 # Start Jupyter
5 jupyter notebook
6
7 # Select "MikoshiLang" kernel
```

## 10 Function Reference

MikoshiLang provides **6,324 functions** across **40+ domains**. Below is a categorized listing.

## 10.1 Version History

- **v3.3.0** (Current) — Full knowledge layer: 5 domain packs (Crossref, OpenAlex, GeoNames, World Bank, PubChem), SPARQL graph queries, persistent cache, 50 canonical properties
- **v3.2.0** — MVP knowledge layer: EntitySearch, EntityValue, WikipediaSummary, Wikidata/Wikipedia integration
- **v3.1.1** — 6,324 computational functions
- **v3.1.0** — Added 600 functions: hydrology, mining, manufacturing, renewable energy
- **v3.0.4** — Fixed all placeholders, 100% working functions
- **v3.0.0** — Added 600 functions: info theory, acoustics, GIS, chemical engineering
- **v2.5.0** — Added 600 functions: materials, optics, aerospace, nuclear physics
- **v2.4.0** — Added 600 functions: climate, medical, robotics, environmental
- **v2.3.0** — Added 900 functions: engineering, biology, finance, fluids
- **v2.2.0** — Added 900 functions: CS algorithms, astronomy, statistics, NLP
- **v2.0.0** — Added 452 functions: vector calculus, ML, physics, biology
- **v1.0.0** — Initial release: 1,342 core functions

## 10.2 Extended Function Sets (v2.0+)

MikoshiLang v2.0+ includes 32 extended modules covering specialized scientific domains:

- **Extended 6-9** (552) — Vector calculus, ML/time series, physics, biology/numerical methods
- **Extended 10-12** (600) — Special functions, audio/signal processing, graph theory/quantum computing
- **Extended 13-16** (900) — Computer science algorithms, astronomy/astrophysics, advanced statistics, NLP
- **Extended 17-20** (900) — Mechanical/electrical/civil engineering, biology/genetics, finance/economics, fluid dynamics
- **Extended 21-24** (600) — Climate science, medical sciences, robotics, environmental science
- **Extended 25-28** (600) — Materials science, optics/photonics, aerospace engineering, nuclear/particle physics
- **Extended 29-32** (600) — Information theory/cryptography, acoustics/audio, geospatial/GIS, chemical engineering
- **Extended 33** (100) — Agriculture, food science, sports science, textile/petroleum engineering
- **Extended 34** (150) — Hydrology, groundwater, watershed analysis, oceanography, coastal engineering

- **Extended 35** (150) — Rock mechanics, mine design, drilling/blasting, mineral processing, flotation
- **Extended 36** (153) — Production planning, quality control, lean manufacturing, maintenance, ergonomics, facility layout
- **Extended 37** (150) — Solar/wind energy, energy storage, power systems, grid integration

### 10.3 Domain Coverage

**Mathematics:** Calculus, linear algebra, number theory, statistics, differential equations, discrete math, abstract algebra, topology, geometry, tensor analysis

**Physics:** Classical mechanics, thermodynamics, electromagnetism, quantum mechanics, relativity, astronomy, astrophysics, nuclear physics, particle physics, optics, acoustics, fluid dynamics

**Engineering:** Mechanical, electrical, civil, chemical, aerospace, materials, mining, manufacturing, petroleum, textile, robotics

**Computer Science:** Algorithms, data structures, ML, NLP, cryptography, information theory, signal processing, image processing

**Natural Sciences:** Chemistry, biology, genetics, environmental science, climate science, oceanography, geology, hydrology

**Applied Sciences:** Medicine, pharmacology, agriculture, food science, sports science, renewable energy

**Industrial:** Production, quality control, lean manufacturing, facility layout, maintenance, ergonomics

### 10.4 Calculus (67 functions)

D, Derivative, PartialD, Grad, Gradient, Div, Divergence, Curl, Laplacian, Integrate, NIntegrate, Limit, Series, TaylorSeries, LaurentSeries, FourierSeries, FourierTransform, InverseFourierTransform, LaplaceTransform, InverseLaplaceTransform, ZTransform, InverseZTransform, DSolve, NDSolve, Residue, LineIntegral, SurfaceIntegral, VolumeIntegral, DoubleIntegral, TripleIntegral, ...

### 10.5 Algebra (89 functions)

Solve, NSolve, DSolve, RSolve, Roots, FindRoot, Simplify, FullSimplify, Expand, Factor, Apart, Together, Cancel, Collect, Coefficient, CoefficientList, PolynomialQ, Degree, Exponent, ReplaceAll, ReplaceRepeated, Rule, Pattern, Blank, BlankSequence, ...

### 10.6 Linear Algebra (76 functions)

Det, Determinant, Inverse, Transpose, Eigenvalues, Eigenvectors, CharacteristicPolynomial, MinimalPolynomial, Dot, Cross, Norm, Normalize, MatrixPower, MatrixExp, MatrixLog, MatrixRank, Trace, DiagonalMatrix, IdentityMatrix, ZeroMatrix, RandomMatrix, HilbertMatrix, VandermondeMatrix, JordanForm, SchurDecomposition, QRDecomposition, LUDecomposition, CholeskyDecomposition, SVD, PseudoInverse, ...

### 10.7 Number Theory (94 functions)

Prime, PrimeQ, IsPrime, NextPrime, PreviousPrime, PrimePi, PrimeFactors, FactorInteger, Divisors, DivisorSigma, EulerPhi, MoebiusMu, LiouvilleLambda, Fibonacci, Lucas, Binomial, Multinomial, Factorial, DoubleFactorial, Subfactorial, GCD, LCM, ExtendedGCD,

ChineseRemainder, Mod, PowerMod, JacobiSymbol, LegendreSymbol, KroneckerSymbol,  
...

## 10.8 Statistics (112 functions)

Mean, Median, Mode, Variance, StandardDeviation, Quantile, Quartiles, InterquartileRange, Range, Correlation, Covariance, Skewness, Kurtosis, GeometricMean, HarmonicMean, TrimmedMean, CentralMoment, Moment, RandomVariate, PDF, CDF, InverseCDF, NormalDistribution, UniformDistribution, ExponentialDistribution, PoissonDistribution, BinomialDistribution, ChiSquareDistribution, StudentTDistribution, FDistribution, ...

## 10.9 Trigonometry (48 functions)

Sin, Cos, Tan, Cot, Sec, Csc, ArcSin, ArcCos, ArcTan, ArcCot, ArcSec, ArcCsc, Sinh, Cosh, Tanh, Coth, Sech, Csch, ArcSinh, ArcCosh, ArcTanh, ArcCoth, ArcSech, ArcCsch, Haversine, InverseHaversine, TrigExpand, TrigReduce, TrigSimplify, TrigToExp, ExpToTrig, ...

## 10.10 Special Functions (97 functions)

Gamma, LogGamma, PolyGamma, DiGamma, TriGamma, Beta, Zeta, HurwitzZeta, Erf, Erfc, Erfi, BesselJ, BesselY, BesselI, BesselK, AiryAi, AiryBi, HermiteH, LaguerreL, LegendreP, ChebyshevT, ChebyshevU, JacobiP, GegenbauerC, SphericalHarmonicY, EllipticK, EllipticE, EllipticF, EllipticPi, ...

## 10.11 Discrete Math (68 functions)

Permutations, Combinations, Multinomial, Subsets, Tuples, IntegerPartitions, SetPartitions, StirlingS1, StirlingS2, BellNumber, CatalanNumber, BernoulliB, EulerE, HarmonicNumber, GeneralizedHarmonicNumber, PartitionsP, PartitionsQ, ...

## 10.12 Logic & Sets (34 functions)

And, Or, Not, Xor, Nand, Nor, Implies, Equivalent, ForAll, Exists, Union, Intersection, Complement, SymmetricDifference, Subset, Element, Cardinality, PowerSet, CartesianProduct, ...

## 10.13 String Operations (52 functions)

StringLength, StringReverse, StringJoin, StringSplit, StringReplace, StringTake, StringDrop, ToUpperCase, ToLowerCase, StringContainsQ, StringStartsQ, StringEndsQ, StringCount, StringPosition, Characters, FromCharacterCode, ToCharacterCode, ...

## 10.14 List Manipulation (89 functions)

Length, First, Last, Rest, Most, Part, Take, Drop, Append, Prepend, Insert, Delete, Sort, Reverse, Flatten, Partition, Riffle, Join, Union, Intersection, Complement, Select, Map, Apply, Fold, FoldList, Nest, NestList, Table, Range, ...

## 10.15 Graph Theory (64 functions)

Graph, DirectedGraph, WeightedGraph, AdjacencyMatrix, IncidenceMatrix, EdgeList, VertexList, VertexCount, EdgeCount, VertexDegree, InDegree, OutDegree, PathGraph, CycleGraph, CompleteGraph, StarGraph, WheelGraph, GridGraph, TreeGraph, ShortestPath,



GraphDistance, GraphDiameter, GraphRadius, ConnectedComponents, StronglyConnectedComponents, WeaklyConnectedComponents, ArticulationPoints, Bridges, ChromaticNumber, ChromaticPolynomial, MaximumFlow, MinimumCut, SpanningTree, MinimumSpanningTree, EulerianPath, HamiltonianPath, ...

### 10.16 Chemistry (118 elements + functions)

Element, MolecularMass, BalanceEquation, AtomicNumber, AtomicMass, ElectronConfiguration, Electronegativity, MeltingPoint, BoilingPoint, Density, H, He, Li, Be, B, C, N, O, F, Ne, Na, Mg, Al, Si, P, S, Cl, Ar, K, Ca, Fe, Cu, Zn, Ag, Au, U, ...

### 10.17 Units (50+ units)

Convert, Meter, Kilometer, Mile, Foot, Inch, Kilogram, Pound, Ounce, Second, Hour, Day, Celsius, Fahrenheit, Kelvin, Joule, Calorie, KilowattHour, Watt, Horsepower, Atmosphere, Bar, Pascal, Liter, Gallon, ...

### 10.18 Visualization (9 functions)

Plot, Plot3D, ParametricPlot, PolarPlot, ContourPlot, VectorFieldPlot, Heatmap, AnimatePlot, Interactive3D

### 10.19 Numerical Methods (43 functions)

NIntegrate, NSolve, FindRoot, FindMinimum, FindMaximum, NMinimize, NMaximize, NDSolve, InterpolatingFunction, Interpolation, LinearInterpolation, CubicSplineInterpolation, BSplineInterpolation, Fit, LinearFit, PolynomialFit, NonlinearModelFit, ...

### 10.20 Optimization (37 functions)

Minimize, Maximize, NMinimize, NMaximize, LinearProgramming, ConvexOptimization, ConstrainedMin, ConstrainedMax, GradientDescent, NewtonMethod, GoldenSectionSearch, ...

### 10.21 Control Theory (28 functions)

TransferFunction, StateSpaceModel, Poles, Zeros, SystemGain, BodePlot, NyquistPlot, RootLocus, StabilityMargin, Controllability, Observability, LQR, PIDController, ...

### 10.22 Signal Processing (46 functions)

FFT, InverseFFT, DFT, IDFT, DCT, IDCT, Spectrogram, Periodogram, Correlate, Convolve, ButterworthFilter, ChebyshevFilter, BesselFilter, EllipticFilter, ...

### 10.23 Time & Date (24 functions)

Now, Today, DateString, DateList, DayName, MonthName, DayOfWeek, DayOfYear, LeapYearQ, DateDifference, DatePlus, TimeZone, UnixTime, ...

### 10.24 Random & Probability (38 functions)

Random, RandomInteger, RandomReal, RandomChoice, RandomSample, SeedRandom, RandomVariate, RandomPrime, RandomGraph, RandomMatrix, ...

## 10.25 Financial Math (21 functions)

PresentValue, FutureValue, NPV, IRR, Annuity, TimeValue, CompoundInterest, ContinuouslyCompounded, BondPrice, YieldToMaturity, ...

## 11 New Domain Examples (v2.0)

### 11.1 Vector Calculus

```

1 # Gradient of a scalar field
2 Grad[x^2 + y^2 + z^2, {x, y, z}]
3 # => [2*x, 2*y, 2*z]
4
5 # Divergence of a vector field
6 Div[{x, y, z}, {x, y, z}]
7 # => 3
8
9 # Curl of a vector field
10 Curl[{-y, x, 0}, {x, y, z}]
11 # => [0, 0, 2]
12
13 # Laplacian
14 Laplacian[x^2 + y^2, {x, y}]
15 # => 4

```

### 11.2 Machine Learning

```

1 # Linear regression
2 X = Matrix([[1, 2], [2, 3], [3, 4]])
3 y = Matrix([3, 5, 7])
4 LinearRegression[X, y]
5
6 # Activation functions
7 Sigmoid[x]          # => 1/(1 + exp(-x))
8 ReLU[x]             # => Max(0, x)
9 Softmax[{x, y, z}]
10
11 # Loss functions
12 MeanSquaredError[ytrue, ypred]
13 CrossEntropy[p, q]
14
15 # Metrics
16 Accuracy[ytrue, ypred]
17 F1Score[precision, recall]

```

### 11.3 Physics

```

1 # Classical mechanics
2 KineticEnergy[m, v]          # => m*v^2/2
3 PotentialEnergy[m, g, h]     # => m*g*h
4 EscapeVelocity[M, r]         # => sqrt(2*G*M/r)
5
6 # Thermodynamics
7 IdealGasLaw[P, V, n, T]      # => P*V - n*R*T
8 CarnotEfficiency[Th, Tc]     # => 1 - Tc/Th
9
10 # Electromagnetism
11 CoulombForce[q1, q2, r]      # => k_e*q1*q2/r^2

```

```

12 OhmLaw[V, R]                # => V/R
13
14 # Quantum mechanics
15 PlankEnergy[freq]            # => h*freq
16 DeBroglieWavelength[p]      # => h/p
17 HeisenbergUncertainty[dx, dp] # => dx*dp >= hbar/2
18
19 # Relativity
20 LorentzFactor[v]             # => 1/sqrt(1 - v^2/c^2)
21 MassEnergyEquivalence[m]     # => m*c^2

```

## 11.4 Biology & Genetics

```

1 # DNA analysis
2 Complement["A"]               # => "T"
3 ReverseComplement["ATCG"]    # => "CGAT"
4 Transcribe["ATCG"]           # => "AUCG"
5 GCContent["ATCGGC"]          # => 0.5
6
7 # Molecular mass
8 MolecularMass["H2O"]          # => 18.015 g/mol
9
10 # Population dynamics
11 LogisticGrowth[N, r, K]      # => r*N*(1 - N/K)
12 SIRModel[S, I, R, beta, gamma]
13
14 # Sequence alignment
15 HammingDistanceDNA[seq1, seq2]
16 LevenshteinDNA[seq1, seq2]

```

## 11.5 Cryptography

```

1 # Hashing
2 SHA256["hello"]
3 SHA512["world"]
4 MD5["message"]
5
6 # Modular arithmetic
7 ModularExponentiation[base, exp, mod]
8 ModularInverse[a, m]
9 ExtendedEuclidean[a, b]
10
11 # Number theory
12 DiscreteLog[base, result, modulus]
13 PrimitiveRoot[p]
14 QuadraticResidue[a, p]
15 LegendreSymbol[a, p]

```

## 11.6 Numerical Methods

```

1 # Root finding
2 NewtonRaphson[f, x0, tol]
3 BisectionMethod[f, a, b, tol]
4 SecantMethod[f, x0, x1, tol]
5
6 # Integration
7 TrapezoidalRule[f, a, b, n]
8 SimpsonsRule[f, a, b, n]
9 GaussianQuadrature[f, a, b, n]

```

```

10
11 # ODEs
12 EulerMethod[f, y0, t0, h, n]
13 RungeKutta4[f, y0, t0, h]
14 AdamsBashforth[f, y, h]
15
16 # Interpolation
17 LagrangeInterpolation[{x1, x2, x3}, {y1, y2, y3}]
18 CubicSpline[points]

```

## 12 Advanced Topics

### 12.1 Custom Rules

Define your own transformation rules:

```

1 from mikoshilang import parse_and_eval
2
3 # Define a custom simplification rule
4 rule = "sin(x)^2 + cos(x)^2 -> 1"
5
6 # Apply it
7 expr = "sin(x)^2 + cos(x)^2 + 2*x"
8 result = parse_and_eval(f"ReplaceAll[{expr}, {rule}]")
9 # => 1 + 2*x

```

### 12.2 Pattern Matching

```

1 # Match any power
2 Pattern[_ , x^_]
3
4 # Match specific forms
5 Pattern[a_ + b_ , ...]

```

### 12.3 Performance Tips

- Use `NSolve` for numerical roots instead of symbolic `Solve`
- Cache repeated computations
- Use `N[expr]` to convert symbolic to numeric
- Simplify expressions before integration/differentiation
- For large matrices, use sparse representations

## 13 Troubleshooting

### 13.1 Common Errors

#### **SyntaxError: Invalid expression**

Ensure square brackets for function calls and proper operator syntax.

#### **NameError: Function not found**

Check spelling and capitalization. MikoshiLang functions are case-sensitive (`Sin` not `sin`).

#### **ValueError: Cannot parse expression**

Verify parentheses/brackets are balanced.

## 13.2 Getting Help

- Web console: <https://mikoshi.co.uk/mikoshilang/console>
- GitHub Issues: <https://github.com/DarrenEdwards111/MikoshiLang/issues>
- Documentation: <https://github.com/DarrenEdwards111/MikoshiLang/blob/main/README.md>

## 14 Contributing

MikoshiLang is open-source (Apache 2.0). Contributions welcome:

1. Fork the repository
2. Create a feature branch
3. Add functions to `mikoshilang/extended*.py`
4. Write tests in `tests/`
5. Submit a pull request

## 15 License

MikoshiLang is licensed under the Apache License 2.0.

Copyright 2025 Mikoshi Ltd

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

## 16 Acknowledgments

MikoshiLang is built on:

- **SymPy** — symbolic mathematics
- **matplotlib** — 2D plotting
- **plotly** — interactive visualization
- **numpy** — numerical arrays

Inspired by Wolfram Language and designed for the Python ecosystem.

**Built by Mikoshi Ltd**

<https://mikoshi.co.uk>