Mathematica in a Nutshell

Xiangdong Zeng

Department of Physics, Fudan University

Install

- ¥600 in Taobao
- https://tiebamma.github.io/InstallTutorial/

An advanced calculator

Use Python if you don’t like it (Use > at the beginning of a line):

In[ ]:=

```
[i for i in range(0, 30) if i % 5 == 0]
```

In[ ]:=

```
import matplotlib.pyplot as plt
import numpy as np

data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100

plt.scatter('a', 'b', c='c', s='d', data=data)
plt.xlabel('entry a')
plt.ylabel('entry b')
plt.show()
```

WolframAlpha computational intelligence

In[ ]:= integrate 1/(sqrt sin x) dx from 0 to pi

In[ ]:= plot 1/(sqrt sin x) from 0 to 2pi

Syntax

- Do whatever in your notebook (aka. 2D typesetting)
- Built-in functions start with a CAPITAL letter
- Brackets
  - [...] for function call
  - ⟦...⟧ for array/list index
  - {...} for list
  - (...) as in mathematics
  - (* comment *)
- Shift+Enter to run; Enter for new line
- F1 to find help
- Use semicolon (;) to hide output

Arithmetic, calculus & algebra

- Most useful function: Simplify[] & FullSimplify[]
- Expand[], Factor[], TrigExpand[], ...
- Solve[] & Reduce[]
- Derivative: D[] & Dt[]
- Integral: Integrate[]
  - Esc+int+Esc for $\int$
  - Esc+dd+Esc for $dⅆ$
- Differential equations: DSolve[]
- FourierTransform[]
- Try to use build-in functions (do NOT reinvent the wheel)

Numeric algorithms

- Exact and numeric things are totally different!
- N[]
- NSolve[], FindRoot[] & NDSolve[]

Save you from physics experiments

- Quantity[]
- UnitConvert[]
- Around[]

Visualization

- For expression: Plot[], Plot3D[], LogPlot[], ...
- For list of data: ListPlot[], ListPlot3D[], ListLogPlot[], ...
- Add a time axis: Manipulate[]
- Export[]

The language itself

- .nb vs .wl/.wls
- You may try other editor (e.g. VS Code)

Data structure (1): List

Basic

- {1,2,3,...} or List[1,2,3,...]
- Index begin with 1
- Use -1 to access the last item
- So, what is index 0?

Structure of expression

- list[[0]] gives the “Head”: List
- So called S-expression: Construct[]
- Meta-programming: “code is data structure”

Use list more efficiently

- Loop over a list:
  - Avoid using For[]
  - Map[] (/@)
  - Use built-in functions
- Thinking mathematically:
  - Inner & outer product
  - Transpose[]
  - Union[], RotateLeft[], ...

Data structure (2): Association

- Require version 10.0+
- Key-value list
- Like dict in Python, std::map in C++
- Map[] vs KeyMap[]
- Some kind of OOP

Functional programming

Explore Map[]

- What's the first arguments of Map[]?
- A "pure function"
  - f: X → Y, x ⟼ f(x)
  - Lambda expression: λ x . x
- A function actually has only one argument: Curry[]
  - F(x, y) = (f(x)) (y)
  - F: ℝ × ℤ → ℂ
  - f: ℝ → C(ℤ, ℂ)
  - f(x): ℤ → ℂ

Basic category theory

- Category (type): set, group, Integer
  - Object: element in a category
- Morphism (function): “function”, group homomorphism, (#+1)&
  - Object → object
  - Axioms: associativity & identity
- Functor: structure-preserving maps between categories
  - Object x in C ⇒ object F(x) in D
  - Morphism f: x → y over C ⇒ morphism F(f): F(x) → F(y) over D
  - E.g. Map: f (a function over Integer) ⇒ Map[f] (a function over List[Integer])

Other higher-order functions

- Apply[] (@@): replace head
- Nest[], NestWhile[] & FixedPoint[]

In[ ]:=
```
(*
  Newton's method:
   * (# + 2 / #) / 2 &: iteration function
   * 1.0: initial value
   * UnsameQ: condition to end iteration
   * 2: UnsameQ needs two arguments
*)
NestWhile[(# + 2 / #) / 2 &, 0.01, UnsameQ, 2]
```

- Parallelization: Map[] → ParallelMap[]

Pattern matching

Demo

```
qSort[{}] := {}
qSort[{x_, xs___}] := Join @@
  {qSort @ Select[{xs}, # ≤ x &], {x}, qSort @ Select[{xs}, # > x &]}
Echo /@ {#, qSort[#]} & @ RandomInteger[{0, 100}, 30];
```

```
checkboard = RandomInteger[1, {40, 100}];
update[1, 2] := 1; update[_, 3] := 1; update[_, _] := 0;
SetAttributes[update, Listable];
Dynamic[ArrayPlot[
  checkboard = update[
    checkboard,
    Plus @@ Map[
      RotateRight[checkboard, #]&,
      {{-1, -1}, {-1, 0}, {-1, 1}, {0, -1}, {0, 1}, {1, -1}, {1, 0}, {1, 1}}]
    ]
  ]]
```

Patterns

- Basic
  - _: any expression
  - __: 1+ expressions
  - ___: 0+ expression(s)
  - x_: name it as x for latter use
- Restrictions
  - _h: only match expr with Head h
  - _?p: only match expr when p gives True
- “Function define”
  - f[x_] := ...
  - Set[] (=) vs SetDelayed[] (:=)

Rules

- Rule[] (→) vs RuleDelayed[] (:→)
- Results of Solve[] & DSolve[] are rules

Debug

- Print[] & Echo[]
- Timing[] & AbsoluteTiming[]
- Write small functions for easier debugging

Example: COVID-19 in the US

Thank you!