Mathematica
(Wolfram Language)
Symbolic mathematics and computer algebra

Will Pazner
History

- Created by physicist Stephen Wolfram
- First version released in 1988
- Pioneered the “notebook interface” (long before Jupyter, etc.)
- Propriety/closed source (costs $$$$)
What is it?

- Mathematica is a computer algebra system
- Can be used for symbolic/mathematical computations and manipulations
- Has an extensive library of functionality
  - Calculus, abstract algebra, differential equations, visualization, graphics, statistics, graph computations, neural networks, etc.
- Also a “general purpose” programming language
Today’s lecture

- “Using Mathematica”
- Writing simple programs
- Learning the syntax of the language
- Using the built-in functionality to solve problems
- Next lecture: understanding the internal structure of the language
The notebook interface

- Very similar to Jupyter notebooks
- Enter code/expressions in a cell
- Shift-return (or enter on the numeric keypad) evaluates the expression
- The output is shown below the input cell

```
\textbf{MathematicaLectureNotebook.nb}
```

```
\textbf{MathematicaLectureNotebook.nb}
```

```math
\textbf{MathematicaLectureNotebook.nb}
```
Simple arithmetic

The usual operations work:

\[
\text{In}[3]:= 1 + 1 \\
\text{Out}[3]= 2
\]

**Note: division of integers gives rational numbers not floating point:**

\[
\text{In}[4]:= 2 / 3 \\
\text{Out}[4]= \frac{2}{3}
\]

Numbers with decimal points are interpreted as inexact (i.e. double precision floating point)

\[
\text{In}[5]:= 2. / 3. \\
\text{Out}[5]= 0.666667
\]

Group expressions using parentheses. Multiplication can be perform by adding a space (a gray x will appear):

\[
\text{In}[6]:= 2 \times 3 \\
\text{Out}[6]= 6
\]
\[
\text{In}[7]:= 2 \times 3 \\
\text{Out}[7]= 6
\]
More computations

There are a huge number of mathematical functions built in:

\[ 4! \]
\[ 24 \]

Sin, Cos, Log, Exp, Gamma, Erf

Note:
- All built-ins in Mathematica begin with a capital letter
- Even constants, e.g. Pi, E, and I
- This leaves all lowercase-beginning identifiers for user code
Function call syntax

Unlike in Julia, functions are called with square brackets.

Examples:

\[\text{Sin[Pi]}\]
\[\theta\]

\[\text{Exp[1]}\]
\[\text{e}\]

\[\text{Log[E]}\]
\[1\]

As in Julia, arguments are separated by commas:

\[\text{ArcTan[1, 1]}\]
\[\frac{\pi}{4}\]
Variables and expressions

Here we begin to see what makes Mathematica different

Unlike in Julia, variables may be used before they are assigned a value

Mathematica will manipulate the expression with variables held in an unevaluated form
In[13] := \[\text{\texttt{x + x}}\]

\[\text{ERROR: UndefVarError: x not defined}\
\text{Stacktrace:}\
\quad \text{[1] top-level scope at none:0}\
\]

Out[13]= \[\text{\texttt{2 \times}}\]

In[14] := \[\text{\texttt{x + x}}\]

Out[14]= \[\text{\texttt{2 \times}}\]
Mathematical expressions

Using variables in this way lets us manipulate mathematical objects such as equations and polynomials.

Note: using some shortcuts, you can use “fancy” notation in input cells (e.g. superscripts for powers, fractions for division, etc.)

```
In[15]:= a x^2 + b x + c
Out[15]= c + b x + a x^2

In[16]:= a x^2 + b x + c
Out[16]= c + b x + a x^2
```
Plotting

Expressions defining functions in one variable can be plotted:

```
In[17]:= Plot[x, {x, 0, 1}]
```

The syntax \{x, 0, 1\} defines a list (similar to an array in Julia). Lists can contain anything, and can be arbitrarily nested

```
In[18]:= {x, 0, 1}
Out[18]= {x, 0, 1}
```

We will see later how to access and manipulate lists. For now we will just use simple lists to define the axes.
More plotting

The plotting and visualization functionality of Mathematica is very advanced. You can read the built-in documentation to learn more. Some examples:

In[19]:= Plot[Evaluate@Table[BesselJ[n, x], {n, 4}], {x, 0, 10}, Filling -> Axis]

\[\text{Out[19]}=\]

In[20]:= Plot3D[Sin[x + y^2], {x, -3, 3}, {y, -2, 2}]

\[\text{Out[20]}=\]
Using Mathematica for calculus

Mathematica is what Wolfram Alpha is based on
(You may be familiar from Math 1A...)

There is very powerful built-in differentiation and integration of expressions:

\[
\frac{d}{dx} \sqrt{x} \tanh(\sin(x)) = \frac{x \cos(x) \text{sech}(\sin(x))^2 + \tanh(\sin(x))}{2 \sqrt{x}}
\]

\[
\int \sqrt{x} \tanh(\sin(x)) \text{sech}(\sin(x))^2 + \tanh(\sin(x)) dx = \frac{x \tanh(\sin(x))}{2}
\]

Integrate can be used to compute indefinite integrals

\[
\int_{0}^{\pi/2} \sqrt{x} \tanh(\sin(x)) \text{sech}(\sin(x))^2 + \tanh(\sin(x)) dx = \frac{\pi}{2} \tanh(1)
\]

Numerical (approximate results) can be obtained using the N function

\[
N[\sqrt{\pi/2} \tanh(1)]
\]

\[
N[\sqrt{\pi/2} \tanh(1), 30]
\]

If we want to be fancy, we can use mathematical notation in our code:

\[
N[\sqrt{\pi/2} \tanh(1), 30]
\]
In[27]:= Plot[Sqrt[x] Cos[x] Sech[Sin[x]]^2 + Tanh[Sin[x]]/2/Sqrt[x], {x, 0, 4 Pi}]
Creating and assigning variables

Variables are assigned using the equal sign = (Set)

Ihs = rhs

will evaluate the expression ‘rhs’ and assign the result to ‘lhs’. The right-hand side can be any Mathematica expression

```mathematica
In[28]:= x = Pi^2
Out[28]= 9.8696

In[29]:= N[x]
Out[29]= 9.8696

In[30]:= 2 * x^2 + x
```

Variables can be unset using Clear

```mathematica
In[31]:= Clear[x]

In[32]:= x
Out[32]=
```
Lists

As we saw, lists can be created using braces

\[
\text{ln[33]:= mylist = \{3, 8, 4\}}
\]
\[
\text{Out[33]= \{3, 8, 4\}}
\]

We can perform arithmetic operations on lists

\[
\text{ln[34]:= mylist + 1}}
\]
\[
\text{Out[34]= \{4, 9, 5\}}
\]
\[
\text{ln[35]:= mylist * 2}}
\]
\[
\text{Out[35]= \{6, 16, 8\}}
\]

The entries of a list are accessed using double-brackets (or the special notation \[
\text{〚\ldots〛}\]
)
The entries are indexed starting with one

\[
\text{ln[36]:= mylist[[1]]}}
\]
\[
\text{Out[36]= 3}}
\]
\[
\text{ln[37]:= mylist[[3]]}}
\]
\[
\text{Out[37]= 4}}
\]

We can also index a list using a list

\[
\text{ln[38]:= mylist[[\{1, 2\}]]}}
\]
\[
\text{Out[38]= \{3, 8\}}
\]

Lists can be mutated this way

\[
\text{ln[39]:= mylist[[3]] = Pi}}
\]
\[
\text{Out[39]= Pi}}
\]
\[
\text{ln[40]:= mylist}}
\]
\[
\text{Out[40]= \{3, 8, Pi\}}
\]
Recall

There are four kinds of bracketing in Mathematica:

- **(Parentheses)** for grouping mathematical expressions

```
ln[41]:= 2 + 3 * 4
Out[41]= 14
```

```
ln[42]:= (2 + 3) * 4
Out[42]= 20
```

- **[Square brackets]** for functions calls

```
ln[43]:= Sin[Pi]
Out[43]= 0
```

- **{Braces}** for lists

```
ln[44]:= {1, 2, 3}
Out[44]= {1, 2, 3}
```

- **[[Double brackets]]** for indexing

```
ln[45]:= {7, 8, 9} [[2]]
Out[45]= 8
```
Creating Lists

Simple ranges can be constructed using Range

\[ \text{Range}[10] \]
\[ \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \]

\[ \text{Range}[4, 10] \]
\[ \{4, 5, 6, 7, 8, 9, 10\} \]

\[ \text{Range}[4, 10, 2] \]
\[ \{4, 6, 8, 10\} \]

Much more flexible is Table, which allows for the creation of lists using expressions

\[ \text{Table}[i, \{i, 10\}] \]
\[ \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \]

\[ \text{Table}[i^2, \{i, 1\,\text{to}\,10\}] \]
\[ \{1, 4, 9, 16, 25, 36, 49, 64, 81, 100\} \]

\[ \text{Table}[i^2/i!, \{i, 1\,\text{to}\,10\}] \]
\[ \{1, \frac{2}{2}, \frac{3}{3}, \frac{4}{24}, \frac{5}{20}, \frac{6}{720}, \frac{1}{630}, \frac{1}{4480}, \frac{1}{36\,288}\} \]

Table can also be used to create matrices (which are represented as lists of lists)

\[ \text{mat} = \text{Table}[i+j, \{i, 1, 10\}, \{j, 1, 10\}] \]
\[ \{\{2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}, \{3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}, \{4, 5, 6, 7, 8, 9, 10, 11, 12, 13\}, \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}, \{6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}, \{7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}, \{8, 9, 10, 11, 12, 13, 14, 15, 16, 17\}, \{9, 10, 11, 12, 13, 14, 15, 16, 17, 18\}, \{10, 11, 12, 13, 14, 15, 16, 17, 18, 19\}\} \]

Matrices can be displayed nicely using MatrixForm

\[ \text{MatrixForm}[\text{mat}] \]
\[
\begin{array}{cccccccccccc}
2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\
5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \\
6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\
7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\
9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 \\
10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 \\
11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 \\
\end{array}
\]
Note: functions can also be evaluated using postfix notation with the // operator

\begin{verbatim}
In[64]:= mat // MatrixForm
Out[64]//MatrixForm=
\end{verbatim}
Interactivity

One of Mathematica's most powerful features is its interactivity
Consider plotting a set of parabolas

\[ y = (x - a)^2 \]

parameterized by the value \( a \)

We begin by choosing a "representative" set

\[
\text{In}[55]:= \text{parabolas} = \text{Table}[(x - a)^2, \{a, -2, 2, 1/2\}]
\]

\[
\text{Out}[55]= \{ (2 + x)^2, \left( \frac{3}{2} + x \right)^2, (1 + x)^2, \left( \frac{1}{2} + x \right)^2, x^2, \left( -\frac{1}{2} + x \right)^2, (-1 + x)^2, \left( -\frac{3}{2} + x \right)^2, (-2 + x)^2 \}
\]

Plotting these functions gives us some insight

\[
\text{In}[56]:= \text{Plot}[\text{parabolas}, \{x, -6, 6\}, \text{PlotRange} \to \{0, 60\}]
\]

\[
\text{Out}[56]=
\]

More useful is manipulating the parameter interactively
We can use manipulate to understand e.g. frequency and amplitude of waves

```mathematica
Manipulate[
  Plot[a * Sin[\[omega] * x], {x, 0, 2 \[Pi]}, PlotRange -> {-3, 3}],
  {a, 1, 3},
  {\[omega], 1, 10}
]
```

Out[58]=

![Diagram showing amplitude and frequency manipulation](image-url)
Algebraic manipulation and simplification

This is one of the most powerful features of Mathematica

\begin{verbatim}
In[59]:= Factor[1 + 2 x + x^2]
Out[59]= (1 + x)^2

In[60]:= Expand[(1 + x + 3 y)^4]
Out[60]= 1 + 4 x + 6 x^2 + 4 x^3 + x^4 + 12 y + 36 x y + 36 x^2 y + 12 x^3 y + 54 y^2 + 108 x y^2 + 54 x^2 y^2 + 108 y^3 + 108 x y^3 + 81 y^4

In particular, Simplify is one of the most useful single functions

In[61]:= Simplify[Sin[x]^2 + Cos[x]^2]
Out[61]= 1

Sometimes FullSimplify can simplify more complicated expressions (using an expanded set of rules)

In[62]:= Simplify[Gamma[x] Gamma[1 - x]]

In[63]:= FullSimplify[Gamma[x] Gamma[1 - x]]
Out[63]= \pi Csc[\pi x]

myexpr = (x - 1)^2 (2 + x) / ((1 + x) (x - 3)^2)

\begin{verbatim}
Out[64]= (-1 + x)^2 (2 + x)
         (-3 + x)^2 (1 + x)
Out[64]//Apart
\end{verbatim}

In[65]:= Apart[myexpr]
Out[65]= 1 + \frac{5}{(-3 + x)^2} + \frac{19}{4 (-3 + x)} + \frac{1}{4 (1 + x)}

The percent symbol can be used as a shortcut for the output of the last expression

In[66]:= Together[%]
Out[66]= \frac{2 - 3 x + x^3}{(-3 + x)^2 (1 + x)}

In[67]:= Factor[%]
Out[67]= \frac{(-1 + x)^2 (2 + x)}{(-3 + x)^2 (1 + x)}
\end{verbatim}

The percent symbol can be used as a shortcut for the output of the last expression.
\textbf{ln(68) := ExpandAll[\%]}\\
\textbf{Out[68]}\textbf{=} \frac{2}{9 + 3 x - 5 x^2 + x^3} \quad \frac{3 x}{9 + 3 x - 5 x^2 + x^3} \quad \frac{x^3}{9 + 3 x - 5 x^2 + x^3}
Substitution rules

Given an expression such as:

\[
\text{myexpr} = a \cdot x^2 + b \cdot x + c
\]

We can evaluate this expression with the variables replaced by numbers (or any other expressions) using substitution rules. The replacement operator is typed \texttt{/} and rules are typed ->

\[
\text{myexpr} / . \ x \rightarrow 3
\]

Multiple substitutions can be made at once using lists of rules.

\[
\text{myexpr} / . \ {x \rightarrow 3, a \rightarrow 1, b \rightarrow 2, c \rightarrow 0}
\]
More algebraic manipulations

Simplify tends not to make assumptions about the variables

\begin{verbatim}
ln[72]:= Simplify[Sqrt[x^2]]
Out[72]= \sqrt{x^2}

Assumptions can be provided as a second argument to Simplify:

ln[73]:= Simplify[Sqrt[x^2], x \geq 0]
Out[73]= x

Algebraic expressions can be analyzed programmatically:

ln[74]:= myexpr = Expand[(1 + 3 x + 4 y^2)^2]
Out[74]= 1 + 6 x + 9 x^2 + 8 y^2 + 24 x y^2 + 16 y^4

ln[75]:= Coefficient[myexpr, x]
Out[75]= 6 + 24 y^2

ln[76]:= Exponent[myexpr, y]
Out[76]= 4

ln[77]:= myexpr[[3]]
Out[77]= 9 x^2

ln[78]:= r = (1 + x) / (2 (2 - y))
Out[78]= \frac{1 + x}{2 (2 - y)}

ln[79]:= Numerator[r]
Out[79]= 1 + x

ln[80]:= Denominator[r]
Out[80]= 2 (2 - y)

ln[81]:= r = .
\end{verbatim}
Example: binomial theorem

Recall the binomial theorem:

\[(x + y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}\]

Mathematica immediately simplifies using this identity:

\[
\text{In}[82]:= \text{binom} = \text{Sum}[\text{Binomial}[n, k] x^k y^{n-k}, \{k, 0, n\}]
\]
\[
\text{Out}[82]= (x + y)^n
\]

We can evaluate for a specific \(n\) using a replacement rule

\[
\text{In}[83]:= \text{binom} /. n \rightarrow 10
\]
\[
\text{Out}[83]= (x + y)^{10}
\]

\[
\text{In}[84]:= \% // \text{Expand}
\]
\[
\text{Out}[84]= x^{10} + 10 x^9 y + 45 x^8 y^2 + 120 x^7 y^3 + 210 x^6 y^4 + 252 x^5 y^5 + 210 x^4 y^6 + 120 x^3 y^7 + 45 x^2 y^8 + 10 x y^9 + y^{10}
\]
More calculus and series

Mathematica can easily compute partial and repeated derivatives

\[
D[x^n, x]
\]
\[
x \cdot x^{-1+n}
\]

\[
D[x^n, \{x, 2\}]
\]
\[
(-1 + n) \cdot n \cdot x^{-2+n}
\]

\[
D[x^n, \{x, 5\}]
\]
\[
(-4 + n) (-3 + n) (-2 + n) (-1 + n) \cdot n \cdot x^{-5+n}
\]

\[
D[Sin[x] \cdot Cos[y], \{(x, y)\}]
\]
\[
\{Cos[x] \cdot Cos[y], -Sin[x] \cdot Sin[y]\}
\]

Finite and infinite sums can be computed using \texttt{Sum}

\[
\sum_{i=1}^{10} \frac{x^i}{i}
\]
\[
x + \frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{4} + \frac{x^5}{5} + \frac{x^6}{6} + \frac{x^7}{7} + \frac{x^8}{8} + \frac{x^9}{9} + \frac{x^{10}}{10}
\]

\[
\sum_{i=1}^{\text{Infinity}} \frac{1}{i^4}
\]
\[
\frac{\pi^4}{90}
\]

\[
\sum_{i=0}^{\text{Infinity}} \frac{1}{2^i}
\]
\[
2
\]

\[
\prod_{i=1}^{5} (x+i)
\]
\[
(1+x) (2+x) (3+x) (4+x) (5+x)
\]
Equations

Just like in Julia, equality is tested with ==

\[
\begin{align*}
\text{In}[93] &= \ x = 3 \\
\text{Out}[93] &= \ 3
\end{align*}
\]

\[
\begin{align*}
\text{In}[94] &= \ x = 4 \\
\text{Out}[94] &= \ \text{False}
\end{align*}
\]

\[
\begin{align*}
\text{In}[95] &= \ x = 3 \\
\text{Out}[95] &= \ \text{True}
\end{align*}
\]

\[
\text{In}[96] = \ \text{Clear}[x]
\]

Equations are themselves Mathematica expressions

\[
\begin{align*}
\text{In}[97] &= \ \text{myeqn} = a \ x^2 + b \ x + c = 0 \\
\text{Out}[97] &= \ c + b \ x + a \ x^2 = 0
\end{align*}
\]

Equations can be solved using Solve

\[
\begin{align*}
\text{In}[98] &= \ \text{Solve}[\text{myeqn}, x] \\
\text{Out}[98] &= \ \{ \{ x \to \frac{-b - \sqrt{b^2 - 4 a c}}{2 a} \}, \{ x \to \frac{-b + \sqrt{b^2 - 4 a c}}{2 a} \} \}
\end{align*}
\]

\[
\begin{align*}
\text{In}[99] &= \ \text{Solve}[x^2 + 2 \ x - 7 = 0, x] \\
\text{Out}[99] &= \ \{ \{ x \to -1 - \sqrt{2} \}, \{ x \to -1 + \sqrt{2} \} \}
\end{align*}
\]

\[
\begin{align*}
\text{In}[100] &= \ \text{N[\%]} \\
\text{Out}[100] &= \ \{ \{ x \to -3.82843 \}, \{ x \to 1.82843 \} \}
\end{align*}
\]

\[
\text{NSolve} \ (\text{for } \text{“numeric solve”}) \ \text{will look for numeric rather than symbolic solutions}
\]

\[
\begin{align*}
\text{In}[101] &= \ \text{NSolve}[x^2 + 2 \ x - 7 = 0, x] \\
\text{Out}[101] &= \ \{ \{ x \to -3.82843 \}, \{ x \to 1.82843 \} \}
\end{align*}
\]

For linear, quadratic, cubic, and quartic polynomials, Mathematica will always give a closed-form solution

\[
\begin{align*}
\text{In}[102] &= \ \text{Solve}[x^4 - 5 \ x^2 - 3 = 0, x] \\
\text{Out}[102] &= \ \{ \{ x \to \frac{-1}{2} \left( -5 + \sqrt{37} \right) \}, \{ x \to \frac{1}{2} \left( -5 + \sqrt{37} \right) \}, \{ x \to -\frac{1}{2} \left( 5 + \sqrt{37} \right) \}, \{ x \to \frac{1}{2} \left( 5 + \sqrt{37} \right) \} \}
\end{align*}
\]

Some equations do not admit closed-form solutions
In[103] := Solve[2 - 4 x + x^5 == 0, x]

Out[103] = 

\{x \rightarrow \text{Root}[2 - 4 \#1 + \#1^5 & , 1],
\{x \rightarrow \text{Root}[2 - 4 \#1 + \#1^5 & , 2],
\{x \rightarrow \text{Root}[2 - 4 \#1 + \#1^5 & , 3],
\{x \rightarrow \text{Root}[2 - 4 \#1 + \#1^5 & , 4],
\{x \rightarrow \text{Root}[2 - 4 \#1 + \#1^5 & , 5]\}\}

In[104] := NSolve[2 - 4 x + x^5 == 0, x]

Out[104] = 

\{x \rightarrow -1.51851,\{x \rightarrow -0.116792 - 1.43845 \text{\ i},
\{x \rightarrow -0.116792 + 1.43845 \text{\ i},\{x \rightarrow 0.508499,\{x \rightarrow 1.2436\}\}\}

You can also solve for simultaneous systems of equations

In[105] := Solve[{a x + y == 0, 2 x + (1 - a) y == 1}, {x, y}]

Out[105] = 

\{x \rightarrow \frac{1}{-2 + a - a^2}, y \rightarrow -\frac{a}{2 - a + a^2}\}
Differential equations

Mathematica can find analytical solutions to many differential equations, both initial value problems and boundary value problems.

The function DSolve looks for solutions to differential equations.

\[
\text{DSolve}[y'[x] = a y[x] + 1, y, x]
\]

We did not specify an initial condition, so the solution has a constant. Specifying the initial condition eliminates the constant.

\[
\text{DSolve}[\{y'[x] = a y[x] + 1, y[0] = 1\}, y, x]
\]

We can solve a boundary value problem:

\[
\text{DSolve}[\{y''[x] = -\sin[x], y[0] = 0, y[1] = 0\}, y, x]
\]
\textbf{In\textvisiblespace}[112]:= \textbf{Plot}[y[x] / . \text{\%}, \{x, 0, 1\}]

\textbf{Out\textvisiblespace}[112]=

\begin{figure}
\centering
\includegraphics[width=\textwidth]{MathematicaLectureNotebook.nb}
\end{figure}
Set and SetDelayed

Up until now, we have assigned expressions to variables using =, which is called Set
Whenever we write lhs = rhs, the rhs is evaluated immediately
Sometimes we don't want to evaluate the rhs immediately. Instead, we want to wait until we
encounter lhs somewhere else in our program. When we encounter lhs, it is replaced by rhs and
only then evaluated
This is denoted SetDelayed, and is written :=
For example, we will use Set for x, and SetDelayed for y

In[13]:= x = Random[]
y := Random[]

Out[13]= 0.714824

In[15]:= x

Out[15]= 0.714824

In[16]:= x

Out[16]= 0.714824

In[17]:= x

Out[17]= 0.714824

In[18]:= y

Out[18]= 0.894623

In[19]:= y

Out[19]= 0.902439

In[20]:= y

Out[20]= 0.0764528
Defining functions

Functions are just another type of expression in Mathematica. Their argument lists take the form of patterns. We will discuss patterns in more depth in the next lecture. For now, we will make use of one kind of pattern: \texttt{x\_} which matches any expression and calls it \texttt{x}.

For example:

\begin{verbatim}
In[121]:= addOne[\_] := x + 1
\end{verbatim}

This function can be called with any kind of argument. That argument will be called \texttt{x} in the body of the function, which gives a result by adding one.

\begin{verbatim}
In[122]:= addOne[2]
Out[122]= 3
\end{verbatim}

\begin{verbatim}
In[123]:= addOne[{3, 4, 5}]
Out[123]= {4, 5, 6}
\end{verbatim}

There are much more sophisticated patterns than just "match any expression." We will cover these later.

Suppose we want to recursively compute the factorial.

Base case: \texttt{fac(0) = 1}

Recursive definition: \texttt{fac(n) = n*fac(n-1)}

This is simple to define in Mathematica. We begin with the base case.

\begin{verbatim}
In[124]:= fac[\_] := 1
\end{verbatim}

Now, \texttt{fac[0]} will evaluate to 1.

\begin{verbatim}
In[125]:= fac[0]
Out[125]= 1
\end{verbatim}

We don't have a rule for the general case, so \texttt{fac[n]} will remain unevaluated:

\begin{verbatim}
In[126]:= fac[1]
Out[126]= fac[1]
\end{verbatim}

Defining the general case:

\begin{verbatim}
In[127]:= fac[n_] := n * fac[n - 1]
\end{verbatim}

\begin{verbatim}
In[128]:= fac[1]
Out[128]= 1
\end{verbatim}

\begin{verbatim}
In[129]:= fac[10]
Out[129]= 3628800
\end{verbatim}
Let's do a similar exercise with the Fibonacci sequence:

\[
\begin{align*}
F_0 &= 0 \\
F_1 &= 1 \\
F_n &= F_{n-1} + F_{n-2}
\end{align*}
\]

Of course, Mathematica also has this built in, so we can check our work...

\[
\begin{align*}
\text{In[135]:=} & \quad \text{Fibonacci[10]} \\
\text{Out[135]=} & \quad 55
\end{align*}
\]
Solving recurrence relations

Suppose we have the recurrence relation

\[
\begin{align*}
a_0 &= 2 \\
a_1 &= 3 \\
a_n &= 7 a_{n-1} - 10 a_{n-2}
\end{align*}
\]

We can easily create a function to recursively evaluate this sequence

\[
\begin{align*}
a[0] &= 2 \\
a[1] &= 3 \\
a[n_] &= 7 a[n-1] - 10 a[n-2]
\end{align*}
\]

What if we want to know a general (non-recursive formula)?
First, concentrate on the recursive definition \(a_n = 7 a_{n-1} - 10 a_{n-2}\) (ignoring boundary conditions)
We can solve this recurrence by making the ansatz that \(a_n\) is given by \(r^n\) for some \(r\)
In that case,

\[r^{n-2}(r^2 - 7r + 10) = 0\]

So, we look for solutions to this equation.

\[
\begin{align*}
\text{In}[138] := & \quad \text{Factor}\left[r^2 - 7r + 10\right] \\
\text{Out}[138] = & \quad (-5 + r)(-2 + r)
\end{align*}
\]

So, \(r = 5\) and \(r = 2\) satisfy this relationship.
The general solution will be given by \(a_n = c \cdot 5^n + d \cdot 2^n\)
We can easily solve the system of equations to find the values of \(c\) and \(d\).
But we can also let Mathematica help...

\[
\begin{align*}
\text{In}[141] := & \quad \text{expr} = c \cdot 5^{n} + d \cdot 2^{n} \\
& \quad \text{case0} = \text{expr} /. n \rightarrow 0 \\
& \quad \text{case1} = \text{expr} /. n \rightarrow 1 \\
\text{Out}[141] = & \quad 5^n c + 2^n d \\
\text{Out}[142] = & \quad c + d \\
\text{Out}[143] = & \quad 5 c + 2 d \\
\text{In}[144] := & \quad \text{soln} = \text{Solve}\left[\{\text{case0} = 2, \text{case1} = 3\}, \{c, d\}\right] \\
\text{Out}[144] = & \quad \left\{\left\{c \rightarrow -\frac{1}{3}, d \rightarrow \frac{7}{3}\right\}\right\}
\end{align*}
\]

Let’s check our formula
\[\text{In}[145]:= \text{expr} /\!. \text{soln}[1]\]

\[\text{Out}[145]= \frac{7 \times 2^n - 5^n}{3} - \frac{5^n}{3}\]

\[\text{In}[146]:= (\text{expr} /\!. \text{soln}[1] /\!. n \to 10) \text{\[\Rightarrow\] a[10]}\]

\[\text{Out}[146]= \text{True}\]

**ClearAll** deletes all the definitions we made for \(a\)

\[\text{In}[147]:= \text{ClearAll}[a]\]

**Mathematica** can also solve the recurrence directly using **RSolve**

\[\text{In}[148]:= \text{RSolve}\left\{\{a[0] \to 2, a[1] \to 3, a[n] \to 7 a[n - 1] - 10 a[n - 2]\}, a, n\right\}\]

\[\text{Out}[148]= \left\{\left\{a \to \text{Function}\left[n, \frac{1}{3} \left(7 \times 2^n - 5^n\right)\right]\right\}\right\}\]
More complicated functions and local variables

Sometimes we want to introduce local variables in a function for intermediate values in long calculations.

This can be achieved using Module.

\begin{verbatim}
In[149]:= \{x, y, z\}
Out[149]= \{0.714824, 0.255667, z\}

In[150]:= Module[\{x, y, z\},
         (*
         (This is a comment by the way)
         x, y, and z are declared as local variables
         *)
         x = 1;
y = x + 1;
z = y*2;
         Print[{x, y, z}];
]
\{1, 2, 4\}

In[151]:= Print[{x, y, z}]
\{0.714824, 0.0801186, z\}
\end{verbatim}

Mathematica also has other scoping mechanisms (i.e. for creating variables with local scope).

See e.g. Block, With.

For our uses, Module is sufficient.
Functional style programming

Mathematica is a multi-paradigm programming language

It supports both functional and imperative programming, but functional constructs are often very natural

For example, applying a function \( f \) to a list is known as mapping \( f \) over the list

\[
\text{In[152]} \quad \text{mylist} = \text{Table}[i^2, \{i, 1, 10\}]
\]

\[
\text{Out[152]} = \{1, 4, 9, 16, 25, 36, 49, 64, 81, 100\}
\]

\[
\text{In[153]} \quad \text{Map[Sqrt, mylist]}
\]

\[
\text{Out[153]} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}
\]

Map is considered so important in Mathematica it has its own syntax

\[
\text{In[154]} \quad \text{Sqrt}/\text{mylist}
\]

\[
\text{Out[154]} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}
\]

We can define more complicated functions if we want

\[
\text{In[155]} \quad f[x_] := \text{Sin[Sqrt[x] + x^{2/3}]}
\]

\[
\text{In[156]} \quad \text{Map[f, mylist]}
\]

\[
\text{Out[156]} = \{\text{Sin[2]}, \text{Sin[2 + 2 \times 2^{1/3}]}, \text{Sin[3 + 3 \times 3^{1/3}]}, \text{Sin[4 + 4 \times 2^{2/3}]}, \text{Sin[5 + 5 \times 5^{1/3}]}, \text{Sin[6 + 6 \times 6^{1/3}]}, \text{Sin[7 + 7 \times 7^{1/3}]}, \text{Sin[24]}, \text{Sin[9 + 9 \times 3^{2/3}]}, \text{Sin[10 + 10 \times 10^{1/3}]}\}
\]

Sometimes we want to create an anonymous function so that we don't need to name it

\[
\text{In[157]} \quad \text{Map[Sin[Sqrt[#] + #^{2/3}] & \text{mylist]}
\]

\[
\text{Out[157]} = \{\text{Sin[2]}, \text{Sin[2 + 2 \times 2^{1/3}]}, \text{Sin[3 + 3 \times 3^{1/3}]}, \text{Sin[4 + 4 \times 2^{2/3}]}, \text{Sin[5 + 5 \times 5^{1/3}]}, \text{Sin[6 + 6 \times 6^{1/3}]}, \text{Sin[7 + 7 \times 7^{1/3}]}, \text{Sin[24]}, \text{Sin[9 + 9 \times 3^{2/3}]}, \text{Sin[10 + 10 \times 10^{1/3}]}\}
\]

Here \# indicates the argument, and \& tells Mathematica that the preceding expression is a function

This notation can easily become hard to read, so be careful

\[
\text{In[158]} \quad \text{Sin[Sqrt[#] + #^{2/3}] &/\text{mylist]}
\]

\[
\text{Out[158]} = \{\text{Sin[2]}, \text{Sin[2 + 2 \times 2^{1/3}]}, \text{Sin[3 + 3 \times 3^{1/3}]}, \text{Sin[4 + 4 \times 2^{2/3}]}, \text{Sin[5 + 5 \times 5^{1/3}]}, \text{Sin[6 + 6 \times 6^{1/3}]}, \text{Sin[7 + 7 \times 7^{1/3}]}, \text{Sin[24]}, \text{Sin[9 + 9 \times 3^{2/3}]}, \text{Sin[10 + 10 \times 10^{1/3}]}\}
\]
More functional programming

Recall our list

In[159]:= mylist
Out[159]= \{1, 4, 9, 16, 25, 36, 49, 64, 81, 100\}

We can apply filters to the list using Select

In[160]:= Select[mylist, EvenQ]
Out[160]= \{4, 16, 36, 64, 100\}

We can combine this with anonymous functions

In[161]:= Select[mylist, \# > 20 \&]
Out[161]= \{25, 36, 49, 64, 81, 100\}

Sometimes we want to apply a function to a list for its side-effect rather than its return-value
For this we have Scan

In[162]:= Scan[Print[Sqrt[\#]] \&, mylist]

1
2
3
4
5
6
7
8
9
10

There are a ton more functional programming facilities in Mathematica
See e.g. Nest, Fold, Apply

In[163]:= mylist = RandomInteger[\{1, 50\}, 20]
Out[163]= \{49, 26, 21, 18, 45, 11, 32, 29, 38, 46, 9, 46, 44, 32, 5, 50, 45, 43, 30, 1\}

Can we get the sum of all of these entries?

In[164]:= Plus[mylist]
Out[164]= \{49, 26, 21, 18, 45, 11, 32, 29, 38, 46, 9, 46, 44, 32, 5, 50, 45, 43, 30, 1\}

In[165]:= Plus[\{1, 2, 3\}]
Out[165]= \{1, 2, 3\}
In[166]:= Plus[1, 2, 3]
Out[166]= 6

In[167]:= Apply[Plus, mylist]
Out[167]= 620

In[168]:= Total[mylist]
Out[168]= 620
Procedural control flow

Mathematica also has procedural control flow (perhaps more familiar from e.g. Julia)

```
In[169]:= x = RandomInteger[200]
    If[EvenQ[x],
        Print["x is even!"],
        Print["x is odd!"]
    ]

Out[169]= 108
    x is even!

    Which can be used for many if-else clauses

In[171]:= Which[
    Mod[x, 2] == 0, Print["x mod 2 == 0"],
    Mod[x, 3] == 0, Print["x mod 3 == 0"],
    Mod[x, 4] == 0, Print["x mod 4 == 0"],
    Mod[x, 5] == 0, Print["x mod 5 == 0"],
    Mod[x, 6] == 0, Print["x mod 6 == 0"],
    True, Print["I give up..."]
]

x mod 2 == 0

    The simplest loop is the Do loop, which repeats the contents \( n \) times

In[172]:= Do[
    x = RandomInteger[200];
    If[EvenQ[x],
        Print["x is even!"],
        Print["x is odd!"]
    ],
    10
]
x is odd!
x is odd!
x is even!
x is even!
x is even!
x is odd!
x is even!
x is odd!
x is odd!
x is even!

We also have traditional for loops

In[173]:= For[i = 0, i < 10, i++,
    Print["Iteration ", i];
    x = RandomInteger[200];
    If[EvenQ[x],
        Print["x is even!"],
        Print["x is odd!"]
    ]
]
Iteration 0
x is odd!
Iteration 1
x is even!
Iteration 2
x is odd!
Iteration 3
x is odd!
Iteration 4
x is odd!
Iteration 5
x is odd!
Iteration 6
x is odd!
Iteration 7
x is odd!
Iteration 8
x is odd!
Iteration 9
x is even!
**Application: merge sort**

Merge sort is a recursive algorithm

Given a list, we:
1. find a midpoint
2. split the list at its midpoint into a left list and right list
3. sort the left and right lists using merge sort (recursion)
4. merge the two lists

```mathematica
mergeSort[theList_] := Module[{midpt, left, right},
  If[Length[theList] > 2,
    midpt = Ceiling[Length[theList]/2];
    left = theList[[1 ;; midpt]];
    right = theList[[midpt + 1 ;;]];
    merge[mergeSort[left], mergeSort[right]],
    (* If there is only one item, the list is sorted *)
    theList]
]
```

```mathematica
merge[left_, right_] := Module[{ileft, iright},
  ileft = 1;
  iright = 1;
  Table[
    Which[
      ileft > Length[left], right[[iright++]],
      iright > Length[right], left[[ileft++]],
      left[[ileft]] < right[[iright]], left[[ileft++]],
      True, right[[iright++]]
    ],
    {Length[left] + Length[right]}
  ]
]
```

```mathematica
unsorted = Table[RandomInteger[200], 20]
```

```mathematica
{143, 73, 105, 97, 63, 46, 41, 30, 113, 159, 46, 55, 181, 29, 173, 170, 126, 136, 103, 165}
```

```mathematica
unsorted // mergeSort
```

```mathematica
{29, 30, 41, 46, 45, 55, 63, 73, 97, 103, 105, 113, 126, 136, 143, 159, 165, 170, 173, 181}
```

The runtime complexity of merge sort is given by the recurrence

\( t(n) = 2 \frac{t(n/2)}{} + O(n) \)

```mathematica
RSolve[t[n] == 2 t[n/2] + n, t, n]
```

```mathematica
{{t \to Function[n, \frac{1}{2} n C[1] + \frac{n \text{Log}[n]}{\text{Log}[2]}}
```
The runtime complexity is $n \log(n)$
**Functional-style merge sort**

In[179]:= functionalMergeSort[{}] := {}
functionalMergeSort[{x_}] := {x}
functionalMergeSort[theList_] := Module[{midpt = Ceiling[Length[theList]/2]},
  Apply[merge, 
    Map[functionalMergeSort, 
      {theList[1 ;; midpt], theList[midpt + 1 ;;]}, 1]
  ]
]

In[182]:= functionalMergeSort[unsorted]

Out[182]= {29, 30, 41, 46, 46, 55, 63, 73, 97, 103, 105, 113, 126, 136, 143, 159, 165, 170, 173, 181}
Finding all primes less than $n$

Algorithm known as the Sieve of Eratosthenes

Choose a number $n$

We will find all primes less than $n$

1. List all numbers 2 through $n$
2. Starting with, $p=2$, mark all multiples of $p$, beginning with $p^2$
   i.e. mark $p^2$, $p(p+1)$, $p(p+2)$, ... up to $n$
3. Repeat step 2 with $p$ the next unmarked number
4. Terminate when $p^2 > n$

```mathematica
sieve[n_] := Module[{numlist = Range[n]},
  Do[
    (* Choose $p$. Find the next unmarked number. Numbers are marked by setting them to zero. *)
    If[numlist[[p]] ≠ 0,
      Do[
        numlist[[p*j]] = 0,
        {j, p, n/p}
      ],
      {p, 2, Sqrt[n]}
    ];
    (* Return the unmarked numbers (and be sure to skip 1) *)
    Select[numlist, # > 1 &]
  ]

sieve[100]

{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97}

Let's check our work...

Select[Range[100], PrimeQ]

{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97}

We can also use a functional approach

primes[n_] := NestWhileList[NextPrime, 2, NextPrime[#] < n &]

primes[100]

{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97}
Graph algorithms

There is sophisticated built-in support for graphs and graph algorithms.

There are lots of built-in graphs obtainable using GraphData, e.g.

\[
\text{In}[188]:= \text{GraphData[]}
\]

\[
\begin{align*}
&\text{AGraph, \{Andrasfai, 4\}, \{Andrasfai, 5\}, \{Andrasfai, 6\}, \{Andrasfai, 7\}, \\
&\quad \{Andrasfai, 8\}, \{Andrasfai, 9\}, \{Andrasfai, 10\}, \ldots \text{6821} \ldots, \\
&\quad \text{ZeroTwoNonbipartite, \{7, 48\}}, \text{ZeroTwoNonbipartite, \{7, 49\}}, \\
&\quad \text{ZeroTwoNonbipartite, \{7, 50\}}, \text{ZeroTwoNonbipartite, \{7, 52\}}, \\
&\quad \text{ZeroTwoNonbipartite, \{7, 53\}}, \text{ZeroTwoNonbipartite, \{7, 54\}}, \\
&\quad \text{ZeroTwoNonbipartite, \{7, 55\}}, \text{ZeroTwoNonbipartite, \{7, 56\}}
\end{align*}
\]

\[
\text{In}[189]:= g = \text{GraphData["BananaTree", \{4, 8\}]}
\]

Let's perform a depth-first search on this tree.

We start with the root node (33) in our case, and visit each node in the graph by first visiting all the children of a given node before visiting any of its siblings.

\[
\text{In}[190]:= \text{Graph}[g, \text{VertexLabels} \to \text{Automatic}]
\]

We will need to find the neighbors of a given node:
We also don't want to visit our parent, so we will use Cases and Except to skip items.

```
In[192]:= AdjacencyList[g, rootNode]
Out[192]= {1, 9, 17, 25}

In[193]:= AdjacencyList[g, 1]
Out[193]= {8, 33}

In[194]:= Cases[AdjacencyList[g, 1], Except[33]]
Out[194]= {8}

In[195]:= depthFirst[graph_, node_, parent_] := Module[{adj},
   Print["Visiting ", node];
   adj = Cases[AdjacencyList[graph, node], Except[parent]];
   Scan[depthFirst[graph, #, node] &, adj]
   depthFirst[graph_, root_] := depthFirst[graph, root, root]

In[197]:= depthFirst[g, 33]
```
Visiting 33
Visiting 1
Visiting 8
Visiting 2
Visiting 3
Visiting 4
Visiting 5
Visiting 6
Visiting 7
Visiting 9
Visiting 16
Visiting 10
Visiting 11
Visiting 12
Visiting 13
Visiting 14
Visiting 15
Visiting 17
Visiting 24
Visiting 18
Visiting 19
Visiting 20
Visiting 21
Visiting 22
Visiting 23
Visiting 25
Visiting 32
Visiting 26
Visiting 27
Visiting 28
Visiting 29
Visiting 30
Visiting 31

In[198]:= DepthFirstScan[g, 33, {"PrevisitVertex" -> (Print["Visiting ", #] &)]]
Visiting 33
Visiting 1
Visiting 8
Visiting 2
Visiting 3
Visiting 4
Visiting 5
Visiting 6
Visiting 7
Visiting 9
Visiting 16
Visiting 10
Visiting 11
Visiting 12
Visiting 13
Visiting 14
Visiting 15
Visiting 17
Visiting 24
Visiting 18
Visiting 19
Visiting 20
Visiting 21
Visiting 22
Visiting 23
Visiting 25
Visiting 32
Visiting 26
Visiting 27
Visiting 28
Visiting 29
Visiting 30
Visiting 31

In[199]= BreadthFirstScan[g, 33, {"PrevisitVertex" -> (Print["Visiting ", #] &)]];
Visiting 33
Visiting 1
Visiting 9
Visiting 17
Visiting 25
Visiting 8
Visiting 16
Visiting 24
Visiting 32
Visiting 2
Visiting 3
Visiting 4
Visiting 5
Visiting 6
Visiting 7
Visiting 10
Visiting 11
Visiting 12
Visiting 13
Visiting 14
Visiting 15
Visiting 18
Visiting 19
Visiting 20
Visiting 21
Visiting 22
Visiting 23
Visiting 26
Visiting 27
Visiting 28
Visiting 29
Visiting 30
Visiting 31

\textbf{In[200]} \triangleright \texttt{FindShortestPath[g, 2, 21]}

\textbf{Out[200]} \triangleright \{2, 8, 1, 33, 17, 24, 21\}
\begin{verbatim}
In[201]:= g = Graph[PolyhedronData["Dodecahedron", "SkeletonGraph"], VertexLabels -> Automatic]

Out[201]=

In[202]:= path = FindHamiltonianPath[g]

Out[202]= {13, 18, 10, 15, 4, 20, 6, 2, 5, 19, 3, 7, 11, 12, 8, 16, 1, 14, 9, 17}

In[203]:= pg = PathGraph[path, VertexLabels -> None, EdgeLabels -> "Index"]

Out[203]=

In[205]:= edgelabels = MapIndexed[#1 -> #2[[1]] &, EdgeList[pg]]

Out[205]= {13 \rightarrow 18 \rightarrow 1, 18 \rightarrow 10 \rightarrow 2, 10 \rightarrow 15 \rightarrow 3, 15 \rightarrow 4 \rightarrow 4, 4 \rightarrow 20 \rightarrow 5, 20 \rightarrow 6 \rightarrow 6, 6 \rightarrow 2 \rightarrow 7, 2 \rightarrow 5 \rightarrow 8, 5 \rightarrow 19 \rightarrow 9, 19 \rightarrow 3 \rightarrow 10, 3 \rightarrow 7 \rightarrow 11, 7 \rightarrow 11 \rightarrow 12, 11 \rightarrow 12 \rightarrow 13, 12 \rightarrow 8 \rightarrow 14, 8 \rightarrow 16 \rightarrow 15, 16 \rightarrow 1 \rightarrow 16, 1 \rightarrow 14 \rightarrow 17, 14 \rightarrow 9 \rightarrow 18, 9 \rightarrow 17 \rightarrow 19}
\end{verbatim}
In[206]:= HighlightGraph[\text{g, pg, VertexLabels \rightarrow None, EdgeLabels \rightarrow edgelabels}]

Out[206]=

![Graph diagram with vertices labeled from 1 to 19 and edges labeled with numbers]