

# A Glimpse To Mathematica [Wolfram Language]

**Feras Awad**

February 20, 2024





**This book is prepared for the students at Philadelphia University in Jordan who are taking Math 372, a course in Computer Aided Mathematics. Topics in Set Theory, Number Theory, Calculus, Linear Algebra and Statistics are covered after introducing a brief introduction about Mathematica. The entire document was written in LaTeX, implemented for Windows using the MiKTeX distribution. As for the text editor of my choice, I fancy TeXstudio. All the commands were implemented using Mathematica 13.1.**

Contents

<b>Before Starting</b>	<b>3</b>
<b>1 Introduction</b>	<b>4</b>
1.1 What Is the Wolfram Language?	4
1.2 Wolfram Cloud	4
1.3 What is <i>Mathematica</i> ?	4
1.4 The Structure of <i>Mathematica</i>	4
1.5 Common Kinds of Interfaces to <i>Mathematica</i>	5
1.6 Notebook Interfaces	5
1.7 Editing Cells and Text	6
1.8 Palettes	6
1.9 Important Resources	7
<b>2 Mathematica as a Calculator</b>	<b>8</b>
2.1 Commands for Basic Arithmetic	8
2.2 Precedence	8
2.3 Built-in Constants	8
2.4 Built-in Functions	9
2.5 Numerical and Scientific Notations	10
2.6 Prefix and Postfix Forms for Built-in Functions	10
2.7 <i>Mathematica</i> Help	11
2.8 Exercises	12
<b>3 Variables and Functions</b>	<b>13</b>
3.1 Rules for Names	13
3.2 Immediate Assignment	13
3.3 Functions	14
3.4 Transformation (Substitution) Rule	15
3.5 Anonymous Functions	15
3.6 Functions with Conditions	16
3.7 Recursion	16
3.8 Exercises	17
3.9 Project	18
<b>4 Lists</b>	<b>19</b>
4.1 What is a List?	19
4.2 Functions Producing Lists	19
4.3 Displaying Lists	20
4.4 Working with Elements of a List	20
4.5 Pseudorandom Numbers	21
4.6 Useful Functions	22
4.7 Listable Functions	22
4.8 Nested Loops	23
4.9 Vectors	24
4.10 Matrices	25
4.11 Special Types of Matrices	26
4.12 Basic Matrix Operations	26
4.13 Exercises	28
4.14 Project	30
<b>5 Logic and Set Theory</b>	<b>31</b>
5.1 Being Logical	31
5.2 Truth Tables	31
5.3 Element “ $\in$ ”	32
5.4 Handling Sets	32
5.5 Quantifiers	32
5.6 Exercises	33
5.7 Project	34
<b>6 Number Theory</b>	<b>35</b>
6.1 Primes	35
6.2 Integer Factorization	36
6.3 Digits in Numbers	36
6.4 Fibonacci Sequence	37
6.5 Number Theoretic Functions	37
6.6 Selecting from Lists	38
6.7 Exercises	39
6.8 Project	40

<b>7</b>	<b>Computer Algebra and Solving Equations</b>	<b>42</b>
7.1	Working with Polynomials and Powers . . . . .	42
7.2	Working with Rational Functions . . . . .	43
7.3	Working with Transcendental Functions . . . . .	43
7.4	Equations and Their Solutions . . . . .	44
7.5	Inequalities . . . . .	47
7.6	Exercises . . . . .	47
<b>8</b>	<b>Single Variable Calculus</b>	<b>49</b>
8.1	Limits . . . . .	49
8.2	Differentiation . . . . .	50
8.3	Implicit Differentiation . . . . .	51
8.4	Maximum and Minimum . . . . .	51
8.5	Integration . . . . .	52
8.6	Sequences . . . . .	53
8.7	Series . . . . .	54
8.8	Taylor Polynomials . . . . .	55
8.9	Exercises . . . . .	55
<b>9</b>	<b>Graphics in Mathematica</b>	<b>58</b>
9.1	Making Graphs . . . . .	58
9.2	Plotting Curves . . . . .	60
9.3	Making Graphs in Space . . . . .	63
9.4	Surfaces in Cylindrical and Spherical Coordinates . . . . .	64
9.5	Changing Coordinate Systems . . . . .	66
9.6	Level Curves and Level Surfaces . . . . .	68
9.7	Parametric Curves and Surfaces in Space . . . . .	68
9.8	Visualizing Data . . . . .	69
9.9	Advanced Graphics . . . . .	72
9.10	Region Integrals Measures . . . . .	78
9.11	Graphs and Networks . . . . .	79
9.12	Exercises . . . . .	81
	<b>References</b>	<b>83</b>
	<b>Index</b>	<b>84</b>

## Before Starting

This book is meant to be an active companion during the process of learning how to use Mathematica. The main body of the text will certainly provide insights into how Mathematica works, but the examples should be retyped as a starting point for individual exploration. Each chapter contains discussion, tips, and a description of Mathematica functionality, along with actual examples that serve as starting points. Each chapter ends with additional exercises to emphasize comprehension, which can be used as an assignment to students or simply to work through on your own.

No matter what format this book is viewed in, it is recommended that readers have Mathematica on the desktop or Mathematica Online immediately accessible to type the examples and work through the exercises. It is recommended that as readers work through the book, they save a new file for each chapter in Wolfram Notebook format (.nb), either locally or in the Wolfram Cloud, for future reference.

All new Mathematica students should work through chapters one through twelve (at least) to obtain the necessary basis of how to use Mathematica for solving mathematical problems in different mathematical subjects. These chapters will be of value to intermediate Mathematica users by filling in gaps in knowledge that can result from using Mathematica only for a narrowly defined set of tasks, or by broadening the horizons of users who may have learned Mathematica from an older version.

There is a lot more to Wolfram Language, like dealing with the vast majority of options of commands and using new commands for topics and areas that we have missed, than we have been able to cover in this book.

If you have understood what is in this book, and can do its exercises, then you can now consider yourself a Wolfram Language programmer! There will always be more you can learn, but you are ready to start using what you know to do real programming.

As a mathematician, there will probably be something you want to solve or program every day. With a traditional computer language it would take too long to actually do it. But with the Wolfram Language, and with all its built-in knowledge and automation, anyone who knows the language can write very useful programs even in a matter of minutes. The first step in creating a program for something is to see how to think about the thing in computational terms. It might be something where computers have long been used. It might be something that is only now conceivable for computers as a result of the Wolfram Language. Whatever it is, try to imagine a Wolfram Language function for doing it. What input would the function get? What output would it generate? What might the function be called? Do not at first think about how you would write the code. Just think about what the function should do. And only after you have understood that, start writing the code.

# 1 Introduction

## 1.1 What Is the Wolfram Language?

The Wolfram Language is a computer language. It gives you a way to communicate with computers, in particular so you can tell them what to do. There are many computer languages, such as C++, Java, Python and JavaScript. The Wolfram Language is unique in that it's knowledge based. That means that it already knows a lot, so you have to tell it much less to get it to do things you want.

It's designed to make it as easy as possible to describe what you want, making use of huge amounts of knowledge that are built into the language. And the crucial thing is that when you use the Wolfram Language to ask for something, the computer immediately knows what you mean, and then can actually do what you want.

You can make things that are visual, textual, interactive or whatever. You can do analyses or figure things out. You can create apps and programs and websites. You can take a very wide variety of ideas and implement them-on your computer, on the web, on a phone, on tiny embedded devices and more.

## 1.2 Wolfram Cloud

The Wolfram Open Cloud gives anyone a way to do “casual” programming whenever they want-with access to all the core computation, interface, deployment and knowledge capabilities of the Wolfram Language. If you want to get more serious-about computation, deployments or storage-you'll need to have an actual subscription for Wolfram Cloud. With the introductory Wolfram Cloud plan, you get 200 MB of cloud storage and temporary file storage and deployments (each file and deployment expires after 60 days). For more information, visit [support.wolfram.com](https://support.wolfram.com).

To start using your Wolfram Cloud product: in a browser, go to [wolframcloud.com](https://wolframcloud.com). Its homepage appears.



To create a new account, click sign up for free. Enter your email address, first name, last name, and password, and agree to the terms of service and privacy policy.

Create a Wolfram ID

☐ I agree to the Terms of Service and the retention of my personal data as described in the Privacy Policy.

Create Wolfram ID

To access your Wolfram Cloud product, click Sign in if you have an account. Its login screen appears. Enter your Wolfram ID and password, then select Sign in. The product's starting page appears. Enjoy with Mathematica

WOLFRAM CLOUD

Sign In with Your Wolfram ID

☒ Remember me

Sign In

Forgot your password?

Don't have a Wolfram ID? Create one.

## 1.3 What is Mathematica?

*Mathematica* is a tool for technical computing based on the Wolfram Language. It contains an extensive knowledge base for working with a very broad range of tasks, including solving equations, programming, importing and exporting data, visualizing functions and data, and much more.

Although *Mathematica* is a very large and powerful system, you can get up and running with it in just a few minutes and become fairly proficient by learning some basic concepts about how it is organized, the syntax of its commands, and how to get help when you are stuck. This course will guide you through these first steps and provide you with the foundation you need to incorporate *Mathematica* into your work and/or teaching.

Calculations can often be written in several different styles, with advantages and disadvantages in each scenario. This book focuses on conventions and shortcuts in the Wolfram Language to make calculations shorter, clearer, or easier to understand.

## 1.4 The Structure of Mathematica

The basic parts of the *Mathematica* system:

1. *Mathematica* kernel the part that actually performs computations.
2. *Mathematica* front end the part that handles interaction with the user.

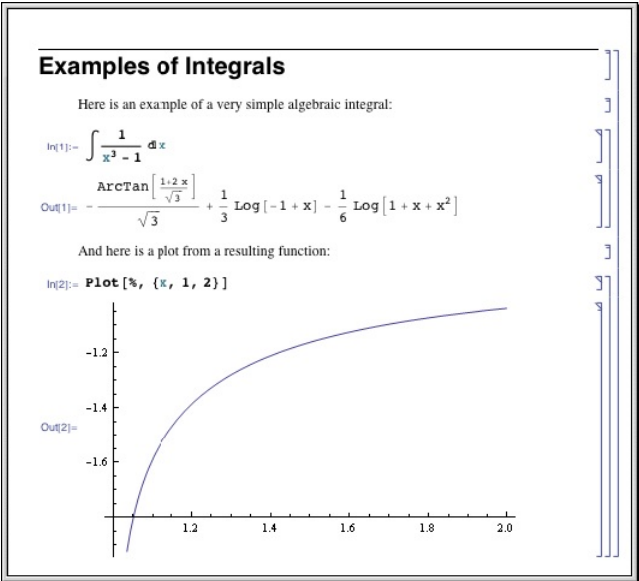
The most common way to work on *Mathematica* is to use interactive documents known as notebooks. Notebooks mix *Mathematica* input and output with text, graphics, palettes, and other material. You can use notebooks either for doing ongoing computations, or as a means of presenting or publishing your results. You should realize that notebooks are part of the “front end” to *Mathematica*. The *Mathematica* kernel which actually performs computations may be run either on the same computer as the front end, or on another computer connected via some kind of network or line. In most cases, the kernel is not even started until you actually do a calculation with *Mathematica*.

1.5 Common Kinds of Interfaces to Mathematica

- 1. Notebook interface you interact with *Mathematica* by creating interactive documents.
- 2. Text-based interface you interact with your computer primarily by typing text on the keyboard. You may be able to start *Mathematica* with a text-based interface by double-clicking on a *Mathematica* Kernel icon.
- 3. MathLink interface communication with other programs. An important aspect of *Mathematica* is that it can interact not only with human users but also with other programs. This is achieved primarily through MathLink, which is a standardized protocol for two-way communication between external programs and the *Mathematica* kernel.

1.6 Notebook Interfaces

If you use your computer via a purely graphical interface, you will typically double-click the *Mathematica* icon to start *Mathematica*. In a “notebook” interface, you interact with *Mathematica* by creating interactive documents. The notebook front end includes many menus and graphical tools for creating and reading notebook documents and for sending and receiving material from the *Mathematica* kernel. A notebook mixing text, graphics, and *Mathematica* input and output.



When *Mathematica* is first started, it displays an empty notebook with a blinking cursor. You can start typing right away. *Mathematica* by default will interpret your text as input. You enter *Mathematica* input into the notebook, then press **Shift+Enter** together to make *Mathematica* process your input. If your keyboard has a numeric keypad, you can use its **Enter** key instead of **Shift+Enter**. After you send *Mathematica* input from your notebook, *Mathematica* will label your input with **In[n] :=**. It labels the corresponding output **Out[n] =**. Labels are added automatically.



The output is placed below the input. By default, input/output pairs are grouped using rectangular cell brackets displayed in the right margin. In addition to the standard textual input, *Mathematica* supports the use of generalized, non-textual input such as graphics and user interface controls, freely mixed with textual input. To exit *Mathematica*, you typically choose the **Exit** menu item in the notebook interface.

**Important Notes** When you input a command in *Mathematica*, make sure you do the following:

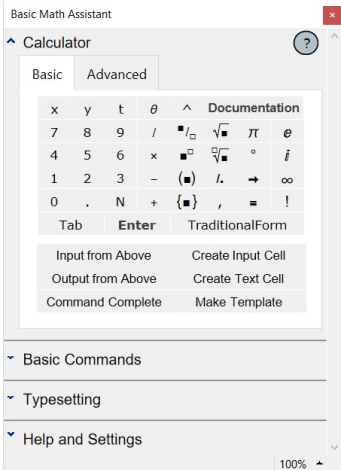
- 1. Use upper and lower case characters exactly as we do. *Mathematica* is very “case sensitive”. If you use the wrong capitalization, you may not get the desired result.
- 2. Use exactly the type of brackets we show. There are three types of brackets: square brackets [ ], parentheses ( ), and curly braces { }. Each has its own meaning in *Mathematica*. If you use the wrong one, *Mathematica* may not do what you expect.
- 3. Your output might appear in a slightly different from ours in some examples. We explain why in the discussion of each example.
- 4. If your *Mathematica* notebook contains a lot of output, especially if graphical output is involved, it can become very large when saved to your disk. Consider choosing **Delete All Output** from the **Cell** menu before saving your notebook to disk. Your saved file will then be much smaller, and it will be easier to transmit electronically to others using *Mathematica*.
- 5. You can stop *Mathematica* in the middle of a computation by choosing **Abort Evaluation** under the **Evaluation** menu.

1.7 Editing Cells and Text

- 1. **Start a new cell in a notebook:**  
Move the mouse to the new location which is between or outside existing cells. Wait for the cursor to change to the horizontal insertion shape. Click the mouse. A horizontal line will appear between cells. Start typing.
- 2. **Delete a cell:**  
Click the cell bracket to select the cell. The bracket will be highlighted. Choose either the **Cut** or **Clear** command from the **Edit** menu, or hit the **Delete** key.
- 3. **Make a copy of a cell in a new location:**  
Click the cell bracket to select the cell. The bracket will be highlighted. Choose the **Copy** command from the **Edit** menu. Move the mouse to the new location. Wait for the cursor to change to the horizontal insertion shape. Click the mouse. A horizontal line will appear between cells. Choose the **Paste** command from the **Edit** menu.
- 4. **Move a cell to a new location:**  
Save procedures as above, except use the **Cut** command instead of the **Copy** command.
- 5. **Cut, copy, or paste the text of a cell within the same or another cell:**  
Handle this the same way that you manipulate text in any word processor. (Use the mouse to select, and then use one of the Cut, Copy or Paste commands.)
- 6. **Change the font, size, or style of an entire cell:**  
Click the cell bracket to select the cell. From the Format menu select the appropriate font, size, and style.
- 7. **Change the font, size, or style of some (or all) of the text within the cell:**
  - (a) Select the text with the mouse.
  - (b) From the **Format** menu select the appropriate font, size, and style.
- 8. **Change the default font, size, or style of all the cells of a given type in a notebook:**
  - (a) Start with the menu selection **Format**→**Edit Stylesheet** to see the Style Definitions in use with your notebook.
  - (b) Select a cell style, such as Input, from the pull-down menu.
  - (c) It appears in the stylesheet window. Click on its grouping bracket.
  - (d) Make a format change such as **Format**→**Size**→ 16. All Input cells in your notebook now appear in 16 point type.

1.8 Palettes

Palettes are specially-prepared types of notebooks that provide graphical shortcuts for entering commands and expressions for those who like visual menus. You open one of them by using the Palettes Menu. There are several predefined palettes available, depending on the specific version of *Mathematica* you have. For example, The **Basic Math Assistant** palette lets you enter expressions involving integrals, roots, and fractions in a more pleasing, *Mathematical* way. It also provides buttons to enter Greek symbols and some special characters directly.



Many of the buttons on the Basic Math Assistant palette, as well as the other Assistant palettes available from the Palettes menu, provide command templates when they are clicked. For example, navigating to the 2D tab of the Basic Commands section and clicking the Plot button yields the following.

```
Plot[ function , { var , min , max } ]
```



Such a template provides the appropriate syntax for the command name and only requires the user to enter the remaining arguments before evaluating the command. The arguments can be entered with the keyboard (and Tab can be used to jump between the placeholders) or by clicking buttons in the palette.

You can use the `CreatePalette` command to construct your own custom palette, which is handy if you find yourself doing the same operations and typesetting constructions over and over again.

## 1.9 Important Resources

**Stephen Wolfram Book Online** Complete text, with full runnable examples and automatically graded exercises.

[wolfram.com/language/elementary-introduction/2nd-ed/](http://wolfram.com/language/elementary-introduction/2nd-ed/)

**Wolfram Language Home Page** Broad collection of resources about the Wolfram Language.

[wolfram.com/language/](http://wolfram.com/language/)

**Wolfram Documentation Center** Documentation on all functions in the Wolfram Language, with extensive examples.

[reference.wolfram.com/language/](http://reference.wolfram.com/language/)

**Wolfram Programming Lab** Online and desktop access to the Wolfram Language, with educational Explorations.

[wolfram.com/programming-lab/](http://wolfram.com/programming-lab/)

**Wolfram Challenges** Dynamic collection of online Wolfram Language programming challenges.

[challenges.wolfram.com](http://challenges.wolfram.com)

## 2 Mathematica as a Calculator

### 2.1 Commands for Basic Arithmetic

*Mathematica* works much like a calculator for basic arithmetic. Just use the +, −, \*, and / keys on the keyboard for addition, subtraction, multiplication, and division. As an alternative to typing \*, you can multiply two numbers by leaving a space between them (the × symbol will automatically be inserted when you leave a space between two numbers). You can raise a number to a power using the ^ key. Use the dot . to type a decimal point.

**Example 2.1.** Calculate:

1)  $\frac{25.5}{5}$

In[ ]: 25.5 / 5

Out[ ]: 5.1

2)  $4 + 2^5$

In[ ]: 4 + 2^5

Out[ ]: 36

3)  $\frac{23}{5} - \frac{3}{5} + 5(2^3)$

In[ ]: 23/5 - 3/5 + 5 \* 2^3

Out[ ]: 44

Did you notice that when you entered the expressions in Example (2.1), *Mathematica* was actively coloring parts of your input as you typed? *Mathematica* uses this coloring aid scheme to tell the user whether the input is complete and syntactically correct.

### 2.2 Precedence

*Mathematica* follows the laws of precedence of multiplication over addition and so on, just as you do by hand. Precedence of common operators is generally defined so that higher-level operations are performed first. For simple expressions, operations are typically ordered from highest to lowest in the order:

1. Parentheses.
2. Factorial.
3. Exponentiation, from right to left.
4. Multiplication and Division, from left to right.
5. Addition and Subtraction, from left to right.

**Note** Use parentheses ( ) to group terms in expressions. Do not use square brackets [ ] or curly braces { }, they mean something different.

**Example 2.2.** In *Mathematica*, the expression  $\frac{3 \times 5}{4 - 2} - \frac{2}{1 + 5}$  is entered as follows.

In[ ]: (3 \* 5) / (4 - 2) - 2 / (1 + 5)

Out[ ]:  $\frac{43}{6}$

Note that, *Mathematica* normally gives you an exact (symbolic) value for every expression.

### 2.3 Built-in Constants

The *Mathematical* constants used most often are already built into *Mathematica*. The table below shows a few of these. Notice that all names begin with a capital letter.

$\pi$	Ratio of a circle's circumference to its diameter	Pi
e	Natural exponential	E
i	Imaginary number	I
°	Degree to radian conversion multiplier	Degree
$\infty$	Positive infinity	Infinity

Once again, you will see some coloring aids as you type the names of built-in constants. When entering Degree, for example, the D will be colored black (it means something on its own), then the characters will be colored blue until you type the final e, at which point the entire name will be colored black. That's because none of the names De, Deg, Degr, or Degree is known. Also, use parentheses in expressions to clarify what you mean. This helps avoid mistakes. For example, you might think that  $E^{2\pi}$  means  $e^{2\pi}$ , but it doesn't! It is actually  $e^2 \times \pi$  because the exponentiation is done before the multiplication of 2 and  $\pi$ . To get  $e^{2\pi}$ , you should write  $E^{(2\pi)}$ .

**Note** It is possible to enter each of these constants, or many other symbols, directly from the keyboard, as well. You can type

<code>Esc</code> + <code>p</code> + <code>i</code> + <code>Esc</code>	for	$\pi$
<code>Esc</code> + <code>e</code> + <code>e</code> + <code>Esc</code>	for	$e$
<code>Esc</code> + <code>i</code> + <code>n</code> + <code>t</code> + <code>Esc</code>	for	$\int$
<code>Ctrl</code> + <code>/</code>	for	$\frac{\blacksquare}{\blacksquare}$
<code>Ctrl</code> + <code>6</code>	for	$\blacksquare^{\blacksquare}$
<code>Ctrl</code> + <code>2</code>	for	$\sqrt{\blacksquare}$
<code>Ctrl</code> + <code>2</code> + <code>5</code>	for	$\sqrt[\blacksquare]{\blacksquare}$

2.4 Built-in Functions

*Mathematica* has many built-in functions. These are the ones you will probably use the most.

Natural logarithm	$\ln x$	<code>Log[x]</code>
Logarithm to base a	$\log_a x$	<code>Log[a,x]</code>
Exponential	$e^x$	<code>Exp[x]</code>
Absolute value	$ x $	<code>Abs[x]</code> , <code>RealAbs[x]</code>
Square root	$\sqrt{x}$	<code>Sqrt[x]</code>
Trigonometric	$\sin x, \cos x, \dots$	<code>Sin[x]</code> , <code>Cos[x]</code> , ...
Inverse trigonometric	$\sin^{-1} x, \cos^{-1} x, \dots$	<code>ArcSin[x]</code> , ...
Hyperbolic	$\sinh x, \cosh x, \dots$	<code>Sinh[x]</code> , <code>Cosh[x]</code> , ...
Inverse hyperbolic	$\sinh^{-1} x, \cosh^{-1} x, \dots$	<code>ArcSinh[x]</code> , ...

Notes

- 1. The names of *Mathematica*'s built-in functions begin with an upper-case letter, and each uses square brackets for the argument(s) of the function.
- 2. *Mathematica* uses radian measure for trigonometric functions.

**Example 2.3.** Evaluate:

1) $\sin\left(\frac{2\pi}{3}\right)$	4) $\sqrt{2} + \sqrt{8}$
<code>In[ ]: Sin[2 Pi / 3]</code>	<code>In[ ]: Sqrt[2] + Sqrt[8]</code>
<code>Out[ ]: <math>\frac{\sqrt{3}}{2}</math></code>	<code>Out[ ]: <math>3\sqrt{2}</math></code>
2) $\log_4(1024)$	5) $e^{3\ln 5}$
<code>In[ ]: Log[4, 1024]</code>	<code>In[ ]: Exp[3 Log[5]]</code>
<code>Out[ ]: 5</code>	<code>Out[ ]: 125</code>
3) $\cos(120^\circ)$	6) $\tan^{-1}(-\infty)$
<code>In[ ]: Cos[120 Degree]</code>	<code>In[ ]: ArcTan[-Infinity]</code>
<code>Out[ ]: <math>-\frac{1}{2}</math></code>	<code>Out[ ]: <math>-\frac{\pi}{2}</math></code>

To evaluate the real-valued cube root of a real number  $x$ , use the `CubeRoot[x]` function. We evaluate the value of  $\sqrt[3]{-2.46038}$  as follows.

```
In[ ]: CubeRoot[-2.46038]
Out[ ]: -1.35
```

In general, to evaluate the real-valued  $n^{\text{th}}$  root of a real number  $x$ , use the built-in function `Surd[x,n]`.

**Example 2.4.** Evaluate  $\sqrt[8]{\frac{390625}{6561}}$ .

```
In[ ]: Surd[390625/6561, 8]
Out[ ]:  $\frac{5}{3}$ 
```

Notes

- 1. Log2[x] gives the base-2 logarithm of x. Also, Log10[x] gives the base-10 logarithm of x.
- 2. You can add a comment to any expression by enclosing it within the symbol pairs (\* and \*). For example:

```
In[]: 27*3      (* This multiplies 27 and 3. *)
Out[]: 81
```

*Mathematica* does not try to evaluate the phrase “This multiplies 27 and 3.” when it evaluates your input. The phrase is for your use only. *Mathematica* reminds you of this by coloring comments gray.

- 3. To do a computation, but do not print its output, you have to end your command by semicolon ;.

2.5 Numerical and Scientific Notations

*Mathematica* normally gives you an exact (symbolic) value for every expression, for example

```
In[]: (3 + 9) * (4 - 8) / 1247 * 67
Out[]: -3216/1247
```

You can force *Mathematica* to give you an answer that looks like the decimal answer you would get on a calculator by using the function N with square brackets around an expression. If we attempt to give a result with n–digit precision for an expression (expr) we use N[expr,n]. For example,

```
In[]: N[(3 + 9)*(4 - 8)/1247*67]
Out[]: -2.57899
```

**Example 2.5.** Find the 45–digits precision of  $\pi$ .

```
In[]: N[Pi, 45]
Out[]: 3.14159265358979323846264338327950288419716940
```

**Example 2.6.** Find the numerical value of  $\log_2\left(\frac{\sqrt{2} + \sqrt[7]{123}}{1 + \sin\left(\frac{\pi}{13}\right)}\right)$  using 16–digits of precision.

```
In[]: N[Log2[(Sqrt[2] + Surd[123, 7])/(1 + Sin[Pi/13])], 16]
Out[]: 1.457204671921001
```

*Mathematica* uses standard scientific notation to display results when the numbers either get very large or very small. For example,

```
In[]: N[1234567890]
Out[]: 1.23457 × 109

In[]: N[1234567890,2]
Out[]: 1.23 × 109

In[]: N[0.000003492836]
Out[]: 3.49284 × 10−6
```

**Note** The command IntegerPart[x] gives the integer part of x while FractionalPart[x] gives the fractional part of x. For example

```
In[]: IntegerPart[N[Pi, 10]]
Out[]: 3

In[]: FractionalPart[N[Pi, 10]]
Out[]: 0.141592654
```

2.6 Prefix and Postfix Forms for Built-in Functions

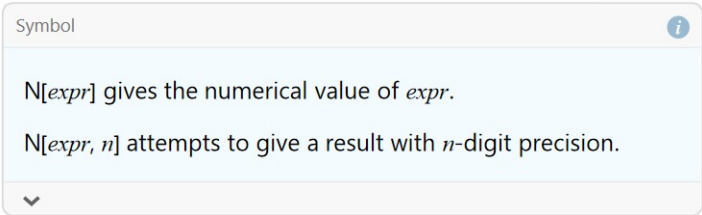
There are three ways to write expressions in *Mathematica*.

f[x,y]	Standard form for f[x,y]	N[Sqrt[2]]	1.41421
f@x	Prefix form for f[x]	N@Sqrt[2]	1.41421
x//f	Postfix form for f[x]	Sqrt[2]//N	1.41421

You should notice that // has very low precedence. If you put //f at the end of any expression containing arithmetic or logical operators, the f is applied to the whole expression. So, for example, x+y//f means f[x+y], not x+f[y]. While the prefix form @ has a much higher precedence. f@x+y is equivalent to f[x]+y, not f[x+y]. You can write f[x+y] in prefix form as f@(x+y).

## 2.7 Mathematica Help

In order to get a quick description of a command, use `?Command`. For example, to get quick info about the function `N[]` just type `?N`.



If you need more explanation use `??Command`. If you need even more, use *Mathematica's* Document Center in the Help Menu and search for the command. *Mathematica* comes with an excellent Help which contains explanations and many nice examples demonstrating how to use each of the functions available in this software.

**Using the Input Assistant** The Input Assistant helps you automatically complete code, discover functions and options, and reduce oversights and typographical errors while coding.

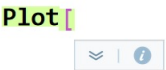
As you type, a list of possible functions and variables (both system- and user-defined) is displayed after a user-defined delay. The list is refined automatically as you type additional characters.



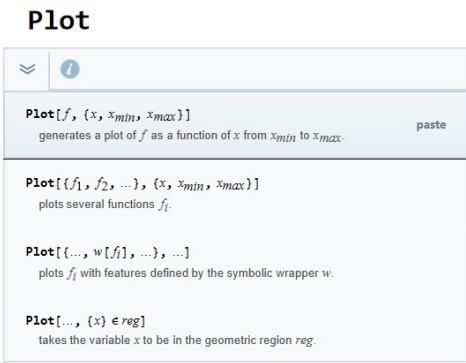
Completions can be inserted using the keyboard or the mouse. Select a completion with the mouse pointer or arrow keys. Press `Enter` or `Tab`, or click to insert the completion.

Access documentation for a function or variable by clicking the document icon ⓘ next to the function name. The documentation will open in a new window.

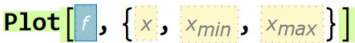
**Function Templates** describe common formatting for specified functions. Function templates can be accessed via code completion. If an inserted function has associated templates, a new button will display after inserting the completion.



Press `Tab` or `Enter`, or click the displayed down arrow icon to access a list of templates for the current function. Alternatively, pressing `Ctrl`+`Shift`+`K` after fully typing a function name will access the template list.



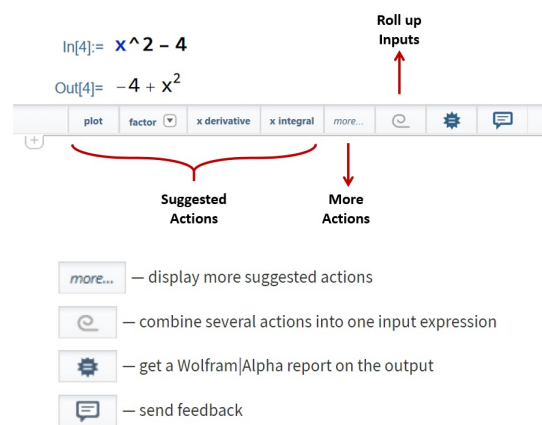
Inserted function templates are fully editable text. Variables that require completion are represented by a yellow placeholder. The currently selected placeholder is highlighted in blue and you can use the `Tab` key to advance to the next placeholder. Typing any character will replace the placeholder with that character. You must replace each placeholder to successfully evaluate your input.



**Note** The Input Assistant feature set has many other components such as:

- 1. **Option templates:** View and insert options related to your current function.
- 2. **Dynamic highlighting:** Highlight code as you type to more easily identify which part of the code you are working in.
- 3. **String completion:** Autocomplete string arguments inside functions.
- 4. **Color chooser:** Choose a specific RGB color from an autocomplete interface.

**Wolfram Predictive Interface** Once you finish a computation, the Suggestions Bar provides immediate access to possible next steps optimized for your results. Clicking suggested actions will perform the action on the output.



2.8 Exercises

- 1. Compute a 20 decimal place approximation to e, the base of the natural logarithm.  
*Ans.* 2.7182818284590452354

- 2. Calculate

(a) 
$$\left| \frac{\log_2(128) + \sqrt[6]{16777216}}{\cos(\sin^{-1}(\frac{1}{2}))} \right|$$

(b) 
$$\tan\left(\frac{\pi}{5}\right)$$

*Ans.* (a)  $\frac{46}{\sqrt{3}}$     (b)  $\sqrt{5 - 2\sqrt{5}}$

- 3. Compute the common logarithm (base 10) of 2<sup>5</sup>. What is its numerical approximation?  
*Ans.* 1.50515

- 4. What happens if you try to subtract ∞ from ∞? What happens if you compute  $\frac{1}{0}$ ?  
*Ans.* Indeterminate, ComplexInfinity

- 5. Can *Mathematica* calculate the expression 2<sup>9941</sup> − 1 easily?  
*Ans.* Yes

6. Show that  $\sqrt{\sqrt[3]{64}\left(2^2 + \left(\frac{1}{2}\right)^2\right)} - 1 = 4$ .

- 7. Add parentheses to 4 − 2 \* 3 + 4 to make 14.

- 8. Try to find the numerical value of  $\sin^2\left(\frac{\pi}{5}\right)$ .  
*Ans.* 0.345492

9. Use the help to read about the command `FunctionExpand`, and use it to prove that  $\sin(24^\circ) = \frac{1}{8}\sqrt{3}\left(\sqrt{5} + 1\right) - \frac{1}{4}\sqrt{\frac{1}{2}\left(5 - \sqrt{5}\right)}$ .

- 10. Find the rational number with smallest denominator that approximates the value of π with error (tolerance) 0.001.  
**Hint:** see the command `Rationalize`.  
*Ans.*  $\frac{201}{64}$

### 3 Variables and Functions

#### 3.1 Rules for Names

You are free to make up the names of the variables you use, as long as you use only letters and numbers and obey these rules:

- 1. Names cannot begin with a number.
- 2. Names cannot contain special characters like @, -, \_, &, %, \$,!,...etc.
- 3. You cannot use names that conflict with the names that are predefined in *Mathematica*. For example, you cannot name one of your own variables Sin.

Notice that because of the first rule above, *Mathematica* will automatically “do the right thing” with an expression such as 2x. Since this is not an acceptable name, *Mathematica* will interpret it as “2” times “x”. All of the following are examples of legitimate names that you can use for variables:

a, m, pI, A, area, perimeter, ABBA, good4you, classOf2016

*Mathematica* distinguishes uppercase and lowercase characters. For example, the names Batman, batman, and batMan are all different. One convention we will use throughout the remainder of the text is that all of the variable names we define will begin with a lowercase letter. You will know that the names belong to us, since all of *Mathematica*’s predefined names start with an uppercase letter.

#### 3.2 Immediate Assignment

You can define a name by assigning any value to it. You can then use the value whenever you want later in a computation. You do this using the equal sign =, which is the symbol for immediate assignment. For example:

```
In[]: a = 3
Out[]: 3
```

The name “a” can also be called a symbol or a variable. Once you have assigned a value to a variable, you can recall it by using it directly in an expression:

```
In[]: N@Sqrt[2/a]
Out[]: 0.816497
```

You may want to know how *Mathematica* keeps track of all the symbols and variables that you define. Say we assign the name mySum to the sum of x and 3 times y.

```
In[]: mySum = x + 3 y
Out[]: x + 3y
```

In this case, *Mathematica* simply repeated our definition because x and y do not yet have an associated value. Now suppose we assign the value 2 to x and the value 5 to y, and then reevaluate mySum:

```
In[]: x = 2; y = 5; mySum
Out[]: 17
```

When *Mathematica* reevaluated mySum this time, it replaced x and y by their respective values, and simplified the resulting expression.

**Note** *Mathematica* will interpret spaces between letters as multiplication, but it will not put a  $\times$  sign on the screen. If you are not looking carefully, you may not notice the difference between the expression x y and xy. The former means x times y, while the latter is the name of a single variable called xy.

**Clearing Symbols** You tell *Mathematica* to forget about the assignment using the Clear command:

```
In[]: Clear[a]
Out[]: a
```

You can also Clear assignments for many names at the same time in one statement Clear[mySum, x, y, X, Y, A, B]. You can use the command

```
In[]: Clear["Global' *"]
```

to clear all the variable and function names you have created so far in your *Mathematica* session.

### 3.3 Functions

*Mathematica* has many built-in functions such as `N`, `Sqrt`, `Sin` and `Log`. You can add your own functions as well.

**Defining Functions** To define a function  $f(x)$  in *Mathematica*, you use the syntax:

$$f[x_] := \text{formula in terms of the variable } x$$

This syntax may look a little awkward, but you should notice:

- The underscore character immediately following the variable on the left, `x_`, tells *Mathematica* that  $x$  is the variable of the function.
- The colon-equal sign `:=` is a delayed assignment command. It behaves differently from `=`. The basic difference between these forms is when the expression at `rhs` is evaluated. `lhs=rhs` is an *immediate assignment*, in which `rhs` is evaluated at the time when the assignment is made. `lhs:=rhs`, on the other hand, is a *delayed assignment*, in which `rhs` is not evaluated when the assignment is made, but is instead evaluated each time the value of `lhs` is requested.
- After you type the `x_`, *Mathematica* visually identifies occurrences of the variable name in the formula by setting them in italic type. On some systems, the variable will also appear in a different color, like the green color.

**Example 3.1.** Define the function  $f(x) = \frac{x^2 + 4}{x - 1}$  in *Mathematica*, and then evaluate the value of  $f(3)$ ,  $f(-1.2)$ , and  $f(1)$ .

```
In[]: f[x_] := (x^2 + 4)/(x - 1)
In[]: f[3]
Out[]: 13/2
In[]: f[-1.2]
Out[]: -2.47273
In[]: f[1]
Out[]: ComplexInfinity
```

**Functions of Several Variables** Functions with more than one variable are defined using a similar syntax, as illustrated in the following examples.

**Example 3.2.** Define a function that computes the average speed of a moving object passes a distance 40 (km) in a time 34 (min).

```
In[]: speed[distance_, time_] := distance / time
In[]: N@speed[40, 34]
Out[]: 1.17647 (* 1.17647 kilometers per minute. *)
In[]: N@speed[40, 34/60]
Out[]: 70.5882 (* 70.5882 kilometers per hour. *)
```

**Example 3.3.** Heron's formula states that the area of a triangle whose sides have lengths  $a$ ,  $b$ , and  $c$  is  $A = \sqrt{s(s-a)(s-b)(s-c)}$  where  $s$  is the semi-perimeter of the triangle; that is,  $s = \frac{a+b+c}{2}$ . Let  $\triangle ABC$  be the triangle with sides  $a = 4$ ,  $b = 13$  and  $c = 15$ , find its area.

```
In[]: TriangleArea[a_, b_, c_] := (
    s = (a + b + c)/2;
    Sqrt[s (s - a) (s - b) (s - c)]
)
In[]: TriangleArea[4, 13, 15]
Out[]: 24
```

**Piecewise Functions** A new command named `Piecewise` has been introduced in *Mathematica* 6 specifically for dealing with functions of split definition. Its syntax has either of these two forms, each of which pairs together values and conditions:

```
Piecewise[{ {val1, cond1} , {val2, cond2} , ...}]
Piecewise[{ {val1, cond1} , {val2, cond2} , ...}, default value]
```



**Example 3.4.** Let  $g(x) = \begin{cases} 1-x & : 0 < x \leq 2 \\ x \ln x & : 2 < x \leq 5 \\ e^x & : x \leq 0 \text{ or } x > 5 \end{cases}$ . Find the values of  $g(1)$ ,  $g(5)$ , and  $g(-4)$ .

```
In[]: g[x_]:=Piecewise[{{1-x, 0<x<=2}, {x Log[x], 2<x<=5}}, Exp[x]]
In[]: g[1]
Out[]: 0
In[]: g[5]
Out[]: 5 ln 5
In[]: g[-4]
Out[]: e-4
```

Use `[Esc]+[p]+[w]+[Esc]` to enter `{` and `[Ctrl]+[,]` and then `[Ctrl]+[Enter]` for each additional piecewise case.

### 3.4 Transformation (Substitution) Rule

You can substitute values into any symbolic expression without having to assign values to the variables explicitly. The substitution symbol `/.` is made using the slash and period symbols, with no space in between. It is used in the form:

**expression /. {list of transformations using `->`, separated by commas}**

For example, to substitute  $x = 2$  and  $y = 5$  into the expression  $x^2 - 2xy$ :

```
In[]: x^2 - 2*x*y /. {x->2, y->5}
Out[]: -16
```

The arrow symbol `->` is formed by entering the minus sign and greater than sign together, with no spaces in between. (As soon as you finish typing these two signs, *Mathematica* will automatically change them to a very spiffy looking  $\rightarrow$ ) The arrow symbol `->` represents a transformation rule. We may use it many times throughout the lectures. The primary advantage of using the transformation command is that the value you substitute into a variable is temporary and is not assigned to the variable. It is not remembered by *Mathematica*.

**Example 3.5.** The roots of a quadratic function  $f(x) = ax^2 + bx + c$  are given by the quadratic formula

$$r_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \text{ and } r_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Let  $f(x) = x^2 + x - 6$ . Find the larger root of  $f$ .

```
In[]: (-b + Sqrt[b^2 - 4 a c])/(2 a) /. {a -> 1, b -> 1, c -> -6}
Out[]: 2
```

**Note** You can use `/.` to replace heads of expressions:

```
In[]: Sin[x] /. Sin -> Cos
Out[]: cos(x)
```

### 3.5 Anonymous Functions

Sometimes we need to “define a function as we go” and use it on the spot. *Mathematica* enables us to define a function without giving it a name (nor any reference to any specific variables) use it, and then move on! These functions are called anonymous or pure functions. Obviously if we need to use a specific function frequently, then the best way is to give it a name and define it as we did before. Here is an anonymous function equivalent to  $f(x) = x^2 + 4$ .

```
In[]: (#^2+4)&
```

The expression `(#^2+4)&` defines a nameless function. As usual we can plug in data in place of `#`. The symbol `&` determines where the definition of the function is completed. For example, we find the value of  $f(x) = x^2 + 4$  when  $x = 5$  using the pure function form as follows

```
In[]: (#^2+4)&[5]
Out[]: 29
```

Also, the value of  $g(x) = \sqrt{x} \sin(x)$  when  $x = \frac{\pi^2}{16}$  in numerical form is

```
In[]: (Sqrt[#] Sin[#])&[Pi^2/16] //N
Out[]: 0.454328
```

Anonymous functions can handle several variables. Here is an example of an anonymous function for  $f(x, y) = \sqrt{x^2 + y^2}$ .

```
In[]: Sqrt[#1^2+#2^2]&[3,4]
Out[]: 5
```

As you might guess, #1 and #2 refer to the first and second variables in the function.

### 3.6 Functions with Conditions

Everything in *Mathematica* is an expression and each expression has a pattern. One can search for a specific pattern and change it to another pattern. This is called pattern matching programming.

The Wolfram Language provides a general mechanism for specifying constraints on patterns. All you need to do is to put `;/condition` at the end of a pattern to signify that it applies only when the specified condition is True. For example, this gives a definition for `f` that applies only when its argument `x` is positive:

```
In[]: f[x_ /; x > 0] := x^2
In[]: f[5]
Out[]: 25
In[]: f[-4]
Out[]: f[-4]
```

In general, you can put `;/condition` at the end of any `:=` definition to tell the Wolfram Language that the definition applies only when the specified condition holds.

```
In[]: f2[x_] := x^2 /; x > 0
In[]: f2[5]
Out[]: 25
In[]: f2[-4]
Out[]: f2[-4]
```

There is a collection of functions built into the Wolfram Language for testing the properties of expressions. Some of these functions are: `Positive`, `Negative`, `NonPositive` and `NonNegative`. Using the function `Positive`, and the question mark "?" as a pattern test, we can define:

```
In[]: f3[x_?Positive] := x^2
In[]: f3[5]
Out[]: 25
In[]: f3[-4]
Out[]: f3[-4]
```

Note that the condition `expr_;/test` can be shortened to `expr_?test`

**Types of Numbers** Four underlying types of numbers are built into the Wolfram System: *Integer*, *Rational*, *Real* and *Complex*. For example, the following function applies only if its argument is a positive integer.

```
In[]: f4[x_Integer?Positive] := x^2
In[]: f4[5]
Out[]: 25
In[]: f4[1/5]
Out[]: f4[1/5]
In[]: f4[-4]
Out[]: f4[-4]
```

There is another collection of functions built into the Wolfram Language for testing the properties of expressions. The most useful of these functions are: `EvenQ[x]`, `OddQ[x]` and `PrimeQ[x]`.

**Example 3.6.** Define the Collatz function  $f(x) = \begin{cases} \frac{x}{2} & : x \text{ is even} \\ 3x + 1 & : x \text{ is odd} \end{cases}$ .

```
In[]: f[x_Integer?EvenQ] := x/2
      f[x_Integer?OddQ] := 3 x + 1
```

### 3.7 Recursion

Many important and classical problems in mathematics and computer science are defined, or have solutions in terms of recursive definitions like the factorial function. A function is defined using recursion if in its definition, it makes *calls to itself*. The classic example is the factorial function which can be defined recursively as follows.

```
In[]: f[0] = 1; f[1] = 1;
In[]: f[n_Integer?Positive] := n f[n - 1]
In[]: f[5]
Out[]: 120
```

**Fibonacci Numbers** Recursive definitions of *Mathematical* quantities were used by mathematicians for centuries before computers even existed. One famous example is the definition of a special sequence of numbers first studied in the Middle Ages by the Italian mathematician *Leonardo Fibonacci*.  
The Fibonacci numbers are generated as follows: start with two 1s, then add them to generate the third number in the sequence; and generally, each new number in the sequence is created by adding the previous two numbers you have written down.

1

$F_1$

1

$F_2$

2

$F_3$

3

$F_4$

5

$F_5$

8

$F_6$

13

$F_7$

21

$F_8$

...

The simplest way to define these numbers is with recursion.

$$F(1) = 1$$

$$F(2) = 1$$

$$F(3) = F(n - 1) + F(n - 2), \quad \text{for } n > 2$$

In this form, we can translate the definition directly into Mathematica.

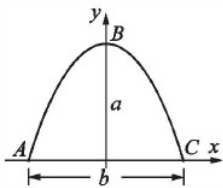
```
In[]: f[1] = 1; f[2] = 1;
In[]: f[n_Integer] := f[n - 1] + f[n - 2] /; n > 2
In[]: f[7]
Out[]: 13
In[]: f[15]
Out[]: 610
```

3.8 Exercises

1. The arc length of a segment of a parabola ABC of an ellipse with semi-minor axes a and b is given approximately by:

$$L_{ABC} = \frac{1}{2} \sqrt{b^2 + 16a^2} + \frac{b^2}{8a} \ln \left[ \frac{4a + \sqrt{b^2 + 16a^2}}{b} \right]$$

Determine  $L_{ABC}$  if  $a = 11$  cm and  $b = 9$  cm.



Ans. 24.5637

2. Define the two functions  $f(x) = x^2 - 1$  and  $g(x) = x^3$ .

(a) Evaluate:  $f(g(x))$ ,  $g(2f(3) - 13)$ ,  $f(f(f(0.5)))$ .

Ans.  $x^6 - 1$  , 27 , -0.808594

(b) What will the following functions do?

```
Compose[f, g, x]
ComposeList[{g, f}, x]
```

3. In the triangle shown  $a = 5.3$  cm,  $\gamma = 32^\circ$ , and  $b = 6$  cm. Define  $\alpha$ ,  $\beta$ , and  $c$  as variables, and then:

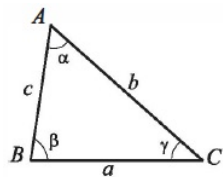
(a) Calculate the length  $c$  by using the Law of Cosines:

$$c^2 = a^2 + b^2 - 2ab \cos \gamma.$$

Ans. 3.18656

(b) Calculate the angles  $\alpha$  and  $\beta$  (in degrees) using the Law of Cosines.

Ans.  $\alpha = 61.8095^\circ$  ,  $\beta = 86.1905^\circ$



4. Define the function  $g(x) = \begin{cases} x \sin(\pi x) & : -\pi \leq x \leq \pi \\ 1 - x & : \text{otherwise} \end{cases}$ , then evaluate the value of  $g(-5)$ ,  $g(6)$ ,  $g(\pi)$ , and  $g(1.8)$ .  
*Ans. 6, -5,  $\pi \sin(\pi^2)$ , -1.05801*

5. For each of the following sequences of numbers, see if you can deduce the pattern and write a Mathematica function to compute the general term.

- (a)    2    3    6    18    108    1944    209952    ...  
          $a_1$   $a_2$   $a_3$   $a_4$   $a_5$      $a_6$      $a_7$     ...
- (b)    0    1    -1    2    -3    5    -8    13    -21    ...  
          $b_1$   $b_2$   $b_3$   $b_4$   $b_5$   $b_6$   $b_7$   $b_8$   $b_9$     ...
- (c)    0    1    2    3    6    11    20    37    ...  
          $c_1$   $c_2$   $c_3$   $c_4$   $c_5$   $c_6$   $c_7$   $c_8$     ...

3.9   Project

1. One way to speed up the computation of the Fibonacci numbers, is to use a different algorithm. A much more efficient algorithm is based on the following identities.

$$\begin{aligned} F_1 &= 1 \\ F_2 &= 1 \\ F_{2n} &= 2F_{n-1}F_n + F_n^2, \quad \text{for } n \geq 1 \\ F_{2n+1} &= F_{n+1}^2 + F_n^2, \quad \text{for } n \geq 1 \end{aligned}$$

Program a Fibonacci number generating function using these identities.

2. **ReplaceRepeated:** A transformation (substitution) rule (`/.`) is applied only once to each part of an expression. For example, the product of `x` and `y` below is replaced by the sum of `x` and `y`, but this is only done for the first such occurrence that matches.

```
In[ ]: a b c d /. x_ y_ -> x + y
Out[ ]: a + bcd
```

In order to apply one or more transformation rules repeatedly to an expression until the expression no longer changes, `ReplaceRepeated` ( `//.` ) is used. Using `ReplaceRepeated`, the rule is applied repeatedly until the expression no longer changes. For example,

```
In[ ]: a b c d //. x_ y_ -> x + y
Out[ ]: a + b + c + d
```

3. Write rules for a function `log` (note lowercase) that encapsulate the following identities:
- (a)  $\log(ab) = \log(a) + \log(b)$   
(b)  $\log(a/b) = \log(a) - \log(b)$   
(c)  $\log(a^n) = n \log(a)$
4. Read about the command `Input`, then write a code that asks the user to enter the temperature in Fahrenheit (`F`) and print its corresponding Celsius (`C`) value using the command `Print`. The formula of conversion is  $C = \frac{F - 32}{1.8}$ .

## 4   Lists

### 4.1   What is a List?

A list in *Mathematica* is an expression whose elements are separated by commas and enclosed in { curly braces }. The elements of a list can be of any type, and need not be all of the same type. For example:

```
{2, 5, 7, 10, -3, -25}
{"good", 17, {23, 89}}
```

Lists are important structures in *Mathematica*. Many of *Mathematica*'s inputs and outputs are expressed using lists.

Lists respect order, so that {1,2} is not the same as {2,1} in *Mathematica*. Also, a list can contain a copy of the same object several times, so {1,2,1} is defer from {1,2}.

One of the nice features of *Mathematica* is that it often allows us to perform operations simultaneously on each element of a list with a natural syntax, especially when the elements of the list are numerical.

**Example 4.1.** For the list of the first five positive integers do the following.

- 1. add 6 to each element.

```
In[]: {1, 2, 3, 4, 5} + 6
Out[]: {7,8,9,10,11}
```

- 2. multiply each element by -3.

```
In[]: -3 * {1, 2, 3, 4, 5}
Out[]: {-3,-6,-9,-12,-15}
```

- 3. rise each element to the power 5.

```
In[]: {1, 2, 3, 4, 5}^5
Out[]: {1,32,243,1024,3125}
```

- 4. divide 120 by each element.

```
In[]: 120 / {1, 2, 3, 4, 5}
Out[]: {120,60,40,30,24}
```

If two lists have the same number of elements, we can add, subtract, multiply and divide them together element by element. For example

```
In[]: {1, 2, 3} + {3, 1, 4}
Out[]: {4,3,7}
In[]: {3, 4}^ {2, 3}
Out[]: {9,64}
In[]: {1, 2, 3} * {3, 1, 4}
Out[]: {3,2,12}
```

### 4.2   Functions Producing Lists

*Mathematica* provides us with commands for which the output is a list. These commands have a nature of repetition and replace loops in procedural programming.

#### Range

```
Range[imax]
generates the list {1,2,...,imax}

Range[imin,imax]
generates the list {imin,...,imax}

Range[imin,imax,di]
uses step di
```

#### Table

```
Table[expr,{i, imax}]
generates a list of the values of expr when i runs from 1 to imax

Table[expr,{i, imin, imax}]
starts with i = imin

Table[expr,{i, imin, imax, di}]
uses step di

Table[expr,{i, {i1, i2, ...}}]
uses the successive values i1, i2,...
```

`Table[expr, n]`  
generates a list of `n` copies of `expr`.

**Example 4.2.** Produce the list of

1. the first ten integers

```
In[ ]: Range[10]
In[ ]: Table[i,{i, 10}]
Out[ ]: {1,2,3,4,5,6,7,8,9,10}
```

2. the first 10 even integers

```
In[ ]: Range[2,20,2]
In[ ]: 2*Range[10]
In[ ]: Table[2i,{i, 10}]
Out[ ]: {2,4,6,8,10,12,14,16,18,20}
```

3. the reciprocal of the first 10 odd integers

```
In[ ]: 1/Range[1,20,2]
In[ ]: Table[1/(2 i - 1), {i, 10}]
Out[ ]: {1, 1/3, 1/5, 1/7, 1/9, 1/11, 1/13, 1/15, 1/17, 1/19}
```

4.3   Displaying Lists

The default output form of a list, like its input form, uses the curly brace notation.

```
{1,2,3}
```

Several formatting functions are available for displaying lists in different forms. For example, `TableForm` is useful for displaying nested lists (multi-dimensional data) in a simple rectangular array.

```
In[ ]: TableForm[{1,2,3}]
Out[ ]: 1
        2
        3

In[ ]: {{ "A", "B", "C" }, {1,2,3}} // TableForm
Out[ ]: A  B  C
        1  2  3
```

Another useful function for displaying nested lists is `Grid`. It contains numerous options specifically for formatting tabular data.

```
In[ ]: Grid[{{ "A", "B", "C" }, {1,2,3}}]
Out[ ]: A  B  C
        1  2  3
```

4.4   Working with Elements of a List

*Mathematica* lets you work with the elements of a list directly. You can do this using either the `Part` command, or - as a shortcut - the double square brackets `[[ ]]` notation. For example, suppose we're given the following definition of a list:

```
In[ ]: mylist = {"good",17,{23,89}};
```

Then

```
In[ ]: mylist[[2]]
Out[ ]: 17

In[ ]: Part[mylist,3]
Out[ ]: {23,89}

In[ ]: mylist[[3,2]]
Out[ ]: 89
```

**Example 4.3.** Consider the list of the 26-English letters from "a" to "z". You can use the command `CharacterRange` to generate such list.

```
In[ ]: letters = CharacterRange["a", "z"]
Out[ ]: {a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z}
```

1. Obtain the second element from the end of the list `letters`.

```
In[]: letters[[-2]]
Out[]: y
```

2. Pick out the 3rd, 10th, and the last characters in the list `letters`.

```
In[]: letters[{{3, 10, -1}}]
Out[]: {c,j,z}
```

3. Find all the characters from the 11th character to the 21th character in the list `letters`.

```
In[]: letters[[11 ;; 21]]
Out[]: {k,l,m,n,o,p,q,r,s,t,u}
```

4.5 Pseudorandom Numbers

The Wolfram Language has several different random number functions to generate random numbers in various ranges, domains, and distributions. Three functions for generating pseudorandom numbers that are distributed uniformly over a range of values: `RandomReal`, `RandomInteger`, `RandomComplex`. In this section, we are interested in *random integers*.

<code>RandomInteger[]</code>	pseudorandomly gives 0 or 1.
<code>RandomInteger[imax]</code>	gives a pseudorandom integer in the range $\{0,\dots,imax\}$ .
<code>RandomInteger[{imin,imax}]</code>	gives a pseudorandom integer in the range $\{imin, imax\}$ .
<code>RandomInteger[{imin,imax},n]</code>	gives a list of $n$ pseudorandom integers in the range $\{imin, imax\}$ .

**Example 4.4.** Pick four random integers between  $-3$  and  $9$ .

```
In[]: RandomInteger[{-3, 9}, 4]
Out[]: {3,-2,7,5}
```

Each time you ask, you get another random numbers:

```
In[]: RandomInteger[{-3, 9}, 4]
Out[]: {-1,0,-2,8}
```

**Note** The sequences that you get from `RandomInteger` are in fact produced by applying a definite *Mathematical* algorithm, starting from a particular “seed”. If you want to make sure that you always get the same sequence of pseudorandom numbers, you can explicitly give a seed for the pseudorandom generator, using `SeedRandom[s]` with the integer  $s$ .

If you reseed the pseudorandom generator with the same seed, you get the same sequence of pseudorandom numbers:

```
In[]: SeedRandom[143]; RandomInteger[100, 5]
Out[]: {14,22,92,70,91}
```

Putting both the `Seed` and the `RandomInteger` commands in the same input cell (or the same line of code) resulting the same sequence of generated random numbers permanently. But, what if the `Seed` command is in an input cell, while the `RandomInteger` is in the following input cell? In this case, the sequence of random numbers will be changed each time you execute the line that contains the random command. The results of this series of executions repeat themselves each time you open the file or re-execute the line that contains the `Seed`.

**Random Samples** Additional functions are available for generating random samples from lists, with or without replacement. For example, `RandomChoice` selects elements from a list with replacement. That can be a list of numbers or any arbitrary expressions.

```
In[]: RandomChoice[{"A", "B", "C"}, 5]
Out[]: {B,A,B,C,A}
```

`RandomSample`, on the other hand, selects without replacement and so its output is limited by the size of the list from which you are selecting. For example, this generates 6 random numbers from the first 10 integers.

```
In[]: RandomSample[Range[10], 6]
Out[]: {2,1,10,6,5,8}
```

Weights can be assigned in both `RandomChoice` and `RandomSample`. This chooses ten 0's and 1's, with a 25% chance of a 0 being chosen and a 75% chance of a 1.

```
In[]: RandomChoice[{0.25, 0.75} -> {0, 1}, 10]
Out[]: {0,1,1,0,1,1,0,1,1,1}
```

## 4.6 Useful Functions

We will list many of the commands we additionally use to manipulate lists and their elements. All of the following examples will be demonstrated using the list `newlist` generated by 25 random integers between 0 and 10 using `seed = 5`:

```
In[ ]: SeedRandom[5]; newlist = RandomInteger[10, 25]
Out[ ]: {0, 0, 3, 3, 4, 5, 1, 9, 1, 0, 10, 5, 1, 8, 0, 5, 9, 5, 3, 7, 3, 5, 8, 6, 7}
```

- `Length` reports the number of elements in a list.

```
In[ ]: Length[newlist]
Out[ ]: 25
```

- `Sort` will arrange the elements of the list in the natural increasing order of the elements.

```
In[ ]: newlist//Sort
Out[ ]: {0, 0, 0, 0, 1, 1, 1, 3, 3, 3, 3, 4, 5, 5, 5, 5, 5, 6, 7, 7, 8, 8, 9, 9, 10}
```

- `Reverse` will reverse the order of the elements in a list.

```
In[ ]: Reverse@Sort@newlist
Out[ ]: {10, 9, 9, 8, 8, 7, 7, 6, 5, 5, 5, 5, 5, 4, 3, 3, 3, 3, 1, 1, 1, 0, 0, 0, 0}
```

- `Count` will show how many times a particular item appears in a list.

```
In[ ]: Count[newlist, 3]
Out[ ]: 4
```

- `Total` gives the total (sum) of the elements in list.

```
In[ ]: Total@newlist
Out[ ]: 108
```

## 4.7 Listable Functions

There are times when we would like to apply a function to all the elements of a list. Suppose `f` is a function and `{a,b,c}` is a list. We want to be able to “push” the function `f` inside the list and get `{f[a],f[b],f[c]}`. Many of *Mathematica*’s built-in functions have the property that they simply “go inside” a list. A function with this property is said to be **listable**. For example, All the arithmetic functions are listable, also `Sqrt` is a listable function.

```
In[ ]: Sqrt@{1, 4, 9}
Out[ ]: {1, 2, 3}
```

Should all operations and commands be listable ? The answer is clearly “**no**.” The simplest example of a command that should not be listable is the `Reverse` command.

```
In[ ]: Reverse[{1, 2}, {9, 10}]
Out[ ]: {{9, 10}, {1, 2}}
```

Indeed, if `Reverse` were listable, the output above would instead have been treated as `{{2, 1}, {10, 9}}`. The `Reverse` command should not be listable because it acts on the structure of the list as a whole. It is not designed to act individually on each element of a list. This observation brings us to the `Map` command.

**Map** The `Map` command can be used to force listability of any command/function and allow it to act on each element of a list. This command has the form:

```
Map[ function name , list ]
```

As an example,

```
In[ ]: Map[Reverse, {{1, 2}, {9, 10}}]
Out[ ]: {{2, 1}, {10, 9}}
```

The equivalent shorthand to apply a function to a list is `/@` as follows

```
In[ ]: Reverse/@{{1, 2}, {9, 10}}
Out[ ]: {{2, 1}, {10, 9}}
```

**Example 4.5.** Consider the list

```
newlist = {0, 0, 3, 3, 4, 5, 1, 9, 1, 0, 10, 5, 1, 8, 0, 5, 9, 5, 3, 7, 3, 5, 8, 6, 7}.
```

Suppose we want to count how many times the numbers 6 and 3 appear in the list, we will get unexpected answer using `Count`:



```
In[]: Count[newlist, {6, 3}]
Out[]: 0
```

This is because Count searches inside the list newlist for an expression like {6, 3} which does not exist. The correct way to do the job is as follows:

```
In[]: Count[newlist, #] & /@ {6, 3}
Out[]: {4, 1}
```

## 4.8 Nested Loops

Let  $f(x)$  be a function defined on a variable  $x$ . There are times when one needs to apply the function  $f$  to itself several times, i.e.,  $f(\dots f(f(x)) \dots)$ . *Mathematica* provides a command to do exactly this:

```
Nest[f, expr, n]

gives an expression with f applied n times to expr.
```

For example,

```
In[]: Nest[f, x, 4]
Out[]: f(f(f(f(x))))
```

If one wants to keep track of each step, the command NestList is available

```
In[]: NestList[f, x, 4]
Out[]: {x, f(x), f(f(x)), f(f(f(x))), f(f(f(f(x))))}
```

Here is a nice example:

```
In[]: NestList[Sqrt[6 + #] &, Sqrt[6], 3]
Out[]: {Sqrt[6], Sqrt[Sqrt[6] + 6], Sqrt[Sqrt[Sqrt[6] + 6] + 6], Sqrt[Sqrt[Sqrt[Sqrt[6] + 6] + 6] + 6]}
```

Sometimes we want to start with expr, then repeatedly applies a function  $f$  until applying test to the result no longer yields True. This is can be done using NestWhile or NestWhileList. For example, we want to keep dividing an integer by 2 until the result is no longer an even number:

```
In[]: NestWhile[#/2 &, 123456, EvenQ]
Out[]: 1929
In[]: NestWhileList[#/2 &, 123456, EvenQ]
Out[]: {123456, 61728, 30864, 15432, 7716, 3858, 1929}
```

**Example 4.6.** Recall that the Collatz function, for any integer  $n$ , returns  $3n+1$  for odd  $n$ , and  $n/2$  for even  $n$ . We investigated the Collatz function earlier in Example 3.6 of Chapter 3. The Collatz Conjecture is the statement that, for any initial positive integer  $n$ , the iterates of the Collatz function always reach the cycle 4, 2, 1, ... Create the function CollzSeq[n] that lists the iterates of the auxiliary Collatz function.

```
In[]: colz[n_Integer?EvenQ] := n/2;
In[]: colz[n_Integer?OddQ] := 3 n + 1;
In[]: CollzSeq[n_Integer?Positive] := NestWhileList[colz, n, #>1 &]
Out[]: {7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1}
```

**Example 4.7.** A point  $p$  is a *fixed point* of a function  $f$  if  $p$  belongs to both the domain and the codomain of  $f$ , and  $f(p) = p$ . For example,  $p = 2$  is a fixed point of  $f(x) = x^2 - 3x + 4$ , because  $f(2) = 2$ .

1. Create the function fixedpoint[p0,n] that approximates the fixed point giving the results of applying  $f$  repeatedly  $n$ -times, starting with  $p_0$ .

```
In[]: f[x_] := x^2 - 3 x + 4;
In[]: fixedpoint[p0_, n_] := Nest[f, p0, n]
In[]: fixedpoint[1, 3]
Out[]: 2
```

2. Modify on the fixedpoint[p0,n] to generate a list giving the results of applying  $f$  repeatedly until successive results are within an error err.

```
In[]: f[x_] := x^2 - 3 x + 4;
In[]: fixedpoint[p0_, err_] := NestWhileList[f, p0, Abs[#1-#2]>err&, 2]
In[]: fixedpoint[1.15, 0.01]
Out[]: {1.15, 1.8725, 1.88876, 1.90113, 1.91091}
```

## 4.9 Vectors

Vectors in *Mathematica* are represented by lists. There is no need to specify if a particular vector is a row vector or column vector; this makes things easier for the user and keeps the focus on the result instead of *Mathematical* bookkeeping.

Vectors can be constructed explicitly or programmatically. To define a vector manually, just create a list. For example, the vectors  $\mathbf{v} = (2, 3)$  and  $\mathbf{u} = (-1, 1, 2)$  are entered as

```
In[]: v = {2, 3}
Out[]: {2, 3}
In[]: u = {-1, 1, 2}
Out[]: {-1, 1, 2}
```

To define a vector programmatically, use `Table`.

```
In[]: Table[i^2, {i, 1, 10}]
Out[]: {1, 4, 9, 16, 25, 36, 49, 64, 81, 100}
```

There is also a function, `Array`, that constructs a vector from a function. `Table` can do this as well, but it requires specification for the iterator, while `Array` assumes the iterator and only requires its bound. For example,

```
In[]: f[x_] := x^2;
In[]: Array[f, 10]
Out[]: {1, 4, 9, 16, 25, 36, 49, 64, 81, 100}
```

or, you may use the pure function as follows.

```
In[]: Array[#^2 &, 10]
Out[]: {1, 4, 9, 16, 25, 36, 49, 64, 81, 100}
```

*Mathematical* operations like addition  $+$ , subtraction  $-$ , multiplication  $*$ , division  $/$ , and exponentiation  $^$  can operate on vectors of the same length. The operations are applied element-wise, so the operation is performed on the first two elements of the vector, then the operation is performed on the second two elements of the vector, and so on.

**Example 4.8.** Let  $\mathbf{u} = (2, 3, 7)$  and  $\mathbf{v} = (7, 1, 5)$ . Find  $5\mathbf{u} - 2\mathbf{v}$

```
In[]: u = {2, 3, 7}; v = {7, 1, 5};
In[]: 5 u - 2 v
Out[]: {-4, 13, 25}
```

**The Dot Product** You compute the dot product of two vectors using the `Dot` command:

```
In[]: Dot[{1, 2, 3}, {-4, 7, 0}]
Out[]: 10
```

Sometimes new users expect the multiplication of two vectors to give the dot product instead of performing element-wise multiplication.

**The Cross Product** To compute a cross product of two vectors in  $\mathbb{R}^3$  (also known as the vector product), you use the `Cross` command.

```
In[]: Cross[{1, 2, 3}, {-4, 7, 0}]
Out[]: {-21, -12, 15}
```

**Length and Angle of Vectors** The length, or the norm, of the vector  $\mathbf{u} = (a_1, a_2, \dots, a_n)$  is

$$\|\mathbf{u}\| = \sqrt{a_1^2 + a_2^2 + \dots + a_n^2}$$

In *Mathematica*, we use the command `Norm` to find the vector length. For example,

```
In[]: Norm[{1, 2, 1}]
Out[]:  $\sqrt{6}$ 
```

Also the angle between two vectors  $\mathbf{u}$  and  $\mathbf{v}$  can be found using the command `VectorAngle`. For example,

```
In[]: VectorAngle[{1, 1}, {1, 0}]
Out[]:  $\frac{\pi}{4}$ 
In[]: VectorAngle[{1, 1, 0}, {1, 0, 2}]
Out[]:  $\cos^{-1}\left(\frac{1}{\sqrt{10}}\right)$ 
```

### 4.10   Matrices

Like vectors, matrices can be comprised of any sort of expression: symbols, numbers, strings, and images, and even mixtures thereof. Matrices are constructed by creating nested lists. Small matrices can be manually entered by typing, and the simplest method is to create a nested list in one-dimensional format using list notation by entering each row as a list, beginning with the first row. That is, a matrix is entered in the form:

$$\{ \text{list of row 1} , \text{list of row 2} , \dots \}$$

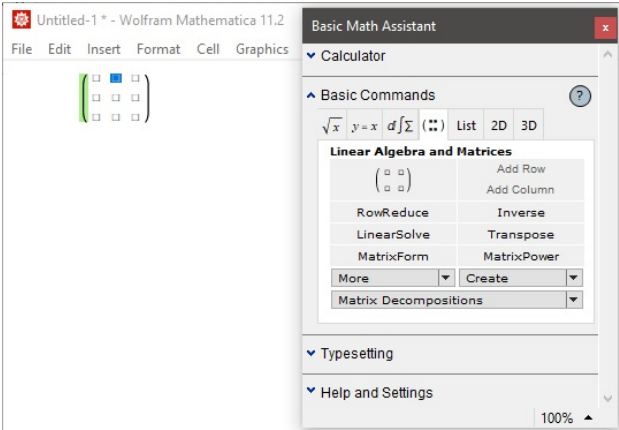
For example, the matrix  $\begin{bmatrix} 3 & -4 & 7 \\ -1 & 0 & 5 \end{bmatrix}$  is entered in *Mathematica* as follows:

```
In[]: {{3, -4, 7}, {-1, 0, 5}}
```

We can use `MatrixForm` to display the result in two-dimensional format.

```
In[]: {{3, -4, 7}, {-1, 0, 5}} // MatrixForm
Out[]:  $\begin{pmatrix} 3 & -4 & 7 \\ -1 & 0 & 5 \end{pmatrix}$ 
```

Matrices can also be entered with palettes like the Basic Math Assistant. The Basic Commands section of the palette has a tab for matrix commands, including a button that will paste an empty  $2 \times 2$  matrix into a notebook. There are buttons to add rows and columns to newly created matrices, providing users with an interactive way to construct a template for a larger matrix. For example, clicking the matrix button to create a  $2 \times 2$  matrix template and then clicking the Add Row and Add Column buttons once each will create a blank  $3 \times 3$  matrix template as seen in the following figure.



Programmatic creation of matrices can be accomplished using functions like `Table` and `Array` as shown in the following examples.

**Example 4.9.** Define a  $3 \times 4$  matrix  $(a_{ij})$  where  $a_{ij} = i + j$ .

```
In[]: Table[i + j, {i, 3}, {j, 4}] // MatrixForm
Out[]:  $\begin{pmatrix} 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \end{pmatrix}$ 

In[]: Array[#1 + #2 &, {3, 4}] // MatrixForm
Out[]:  $\begin{pmatrix} 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \end{pmatrix}$ 
```

**Example 4.10.** Write a code that generates the matrix  $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix}$ .

```
In[]: Table[j + 5 (i - 1), {i, 1, 5}, {j, 1, 5}]
In[]: Array[#2 + 5 (#1 - 1) &, {5, 5}]
In[]: Partition[Range[25], 5]
```

Note that the command `Partition[list, n]` partitions a list into nonoverlapping sublists of length `n`.

### 4.11 Special Types of Matrices

There are special commands in *Mathematica* for matrices with special structures like identity matrices, constant matrices, and the triangular matrices (upper, lower, and diagonal).

1. The command `ConstantArray[c, n]` generates a list (vector) of  $n$  copies of the element  $c$ . While `ConstantArray[c, {m, n}]` generates an  $m \times n$  array of nested lists containing copies of the element  $c$ . For example,

```
In[]: ConstantArray[3, 10]
Out[]: {3, 3, 3, 3, 3, 3, 3, 3, 3, 3}
In[]: ConstantArray[10, {2, 3}] // MatrixForm
Out[]: 
$$\begin{pmatrix} 10 & 10 & 10 \\ 10 & 10 & 10 \end{pmatrix}$$

```

2. The identity matrix, or a unit matrix, of size  $n$  is the  $n \times n$  square matrix with ones on the main diagonal and zeros elsewhere. `IdentityMatrix[n]` gives the  $n \times n$  identity matrix. For example,

```
In[]: IdentityMatrix[2] // MatrixForm
Out[]: 
$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

```

3. A diagonal matrix is a matrix (usually a square matrix) in which the off-diagonal elements are all zero. The main diagonal entries themselves may or may not be zero. The command `DiagonalMatrix[list]` gives a matrix with the elements of `list` on the leading diagonal, and 0 elsewhere. For example, the diagonal matrix of the first 4 positive even integers is

```
In[]: DiagonalMatrix[2 Range[4]] // MatrixForm
Out[]: 
$$\begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 8 \end{pmatrix}$$

```

4. A triangular matrix is a special kind of square matrix. A square matrix is called lower triangular if all the entries above the main diagonal are zero. Similarly, a square matrix is called upper triangular if all the entries below the main diagonal are zero. A triangular matrix is one that is either lower triangular or upper triangular. A matrix that is both upper and lower triangular is called a diagonal matrix.

**Example 4.11.** Construct the  $3 \times 3$  upper triangular matrix whose elements follow the rule  $a_{ij} = \begin{cases} 2i - j & : i \leq j \\ 0 & : i > j \end{cases}$ .

```
In[]: a[i_, j_] := 2i - j /; i <= j
In[]: a[i_, j_] := 0
In[]: Table[a[i, j], {i, 3}, {j, 3}] // MatrixForm
Out[]: 
$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 1 \\ 0 & 0 & 3 \end{pmatrix}$$

```

To generate the lower triangular matrix, just change the inequality signs in  $a_{ij}$ .

```
In[]: a[i_, j_] := 2i - j /; i >= j
In[]: a[i_, j_] := 0
In[]: Table[a[i, j], {i, 3}, {j, 3}] // MatrixForm
Out[]: 
$$\begin{pmatrix} 1 & 0 & 0 \\ 3 & 2 & 0 \\ 5 & 4 & 3 \end{pmatrix}$$

```

### 4.12 Basic Matrix Operations

*Mathematica* can do addition, subtraction, scalar multiplication, and matrix multiplication with the operators  $+$ ,  $-$ ,  $*$ , and  $.$ , respectively, in addition to many other matrix computations like the determinant, the transpose, and the inverse. To see how *Mathematica* operates with these computations, let

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \quad F = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \quad G = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

and define them in *Mathematica* as a two-dimensional lists as follows. Care must be taken, however, not to use `//MatrixForm` in the definition of the matrix. This command is for display purposes only.

```

In[]: A = {{1, 2}, {3, 6}};
In[]: B = {{5, 6}, {7, 8}};
In[]: F = {{1, 2}, {3, 4}, {5, 6}};
In[]: G = {{1, 2, 3}, {4, 5, 6}}, {7, 8, 9}};

```

Then,

1. In[]: A + B

```
Out[]:  $\begin{pmatrix} 6 & 8 \\ 10 & 14 \end{pmatrix}$ 
```

Addition, here, is defined since A and B have the same size. If we add the matrix A to G then the result is undefined since they are different in size.

```
In[]: A + G
```

```
Out[]: Objects of unequal length and cannot be combined
```

2. In[]: 3 B - 2 A

```
Out[]:  $\begin{pmatrix} 13 & 14 \\ 15 & 12 \end{pmatrix}$ 
```

3. In[]: A B // MatrixForm

```
Out[]:  $\begin{pmatrix} 5 & 12 \\ 21 & 48 \end{pmatrix}$ 
```

Be careful ! This command evaluates the product as element by element product, not the ordinary (dot) product you have learned in Linear Algebra class. To perform the ordinary product, we use the command Dot as follows.

```
In[]: Dot[A, B] // MatrixForm
```

```
Out[]:  $\begin{pmatrix} 19 & 22 \\ 57 & 66 \end{pmatrix}$ 
```

Note that the dot product above is defined since the number of columns of the first matrix A equals the number of rows in the second matrix B. This case is not satisfied when try to evaluate  $A \cdot F$  for example.

```
In[]: Dot[A, F] // MatrixForm
```

```
Out[]: {{1,2},{3,6}} and {{1,2},{3,4},{5,6}} have incompatible shapes.
```

What if you like to find  $A^5$ ? If we use the command  $A^5$ , this will rise each element of A to the power 5 as follows.

```
In[]: A^5
```

```
Out[]:  $\begin{pmatrix} 1 & 32 \\ 243 & 7776 \end{pmatrix}$ 
```

The command `MatrixPower[A,n]` gives the  $n^{\text{th}}$  matrix power of the matrix A by the mean of ordinary product.

```
In[]: MatrixPower[A, 5]
```

```
Out[]:  $\begin{pmatrix} 2401 & 4802 \\ 7203 & 14406 \end{pmatrix}$ 
```

4. The transpose of an  $m \times n$  matrix C is another matrix of size  $n \times m$  denoted by  $C^T$  and created by writing the rows of C as columns of  $C^T$  in the same order. For example,  $G^T$  is

```
In[]: Transpose[G] // MatrixForm
```

```
Out[]:  $\begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$ 
```

5. The determinant is a useful value that can be computed from the elements of a square matrix. It can be computed in *Mathematica* with the function Det. For example,

```
In[]: Det /@ {A, B, G}
```

```
Out[]: {0, -2, 0}
```

6. An  $n \times n$  square matrix A is called invertible (also nonsingular) if there exists an  $n \times n$  square matrix B such that  $AB = BA = I_n$  where  $I_n$  denotes the  $n \times n$  identity matrix and the multiplication used is ordinary matrix multiplication. If this is the case, then the matrix B is uniquely determined by A and

is called the inverse of  $A$ , denoted by  $A^{-1}$ . The inverse can be computed in *Mathematica* with the function `Inverse`. For example,

```
In[ ]: Inverse[B] // MatrixForm
Out[ ]:  $\begin{pmatrix} -4 & 3 \\ \frac{7}{2} & -\frac{5}{2} \end{pmatrix}$ 
```

A square matrix that is not invertible is called singular. A square matrix is singular if and only if its determinant is 0. We have seen that  $\text{Det}[G] = 0$ , so that  $G$  is singular matrix.

```
In[ ]: Inverse[G]
Out[ ]: Matrix {{1,2,3},{4,5,6},{7,8,9}} is singular
```

7. The trace of an  $n \times n$  square matrix  $A$  is defined to be the sum of the elements on the main diagonal. The command `Tr[C]` finds the trace of the square matrix  $C$ .

```
In[ ]: Tr /@ {A, B, G}
Out[ ]: {7, 13, 15}
```

8. The solution of the eigenvalue problem is one of the major areas for matrix computations. For an  $n \times n$  matrix  $C$ , the eigenvalues are the  $n$  roots of its characteristic polynomial  $p(\lambda) = \det(\lambda I_n - C)$ . For each eigenvalue  $\lambda$  there exists at least one corresponding vector  $x$  that satisfies  $Cx = \lambda x$  and it is called an eigenvector. *Mathematica* has various functions for computing eigenvalues and eigenvectors.

The command `Eigenvalues[C]` gives a list of the eigenvalues of the square matrix  $C$ . Also, the command `Eigenvectors[C]` gives a list of the eigenvectors of the square matrix  $C$ . You can find a list of the eigenvalues and eigenvectors of the square matrix  $C$  in one command, which is `Eigensystem[C]`. The characteristic polynomial for the matrix  $C$  with respect to the variable  $x$  can be found using `CharacteristicPolynomial[C, x]`. Note that the characteristic polynomial has the determinant and the trace of the matrix as coefficients.

```
In[ ]: Eigenvalues[B]
Out[ ]:  $\left\{ \frac{1}{2}(\sqrt{177} + 13), \frac{1}{2}(13 - \sqrt{177}) \right\}$ 
In[ ]: Eigenvectors[B]
Out[ ]:  $\begin{pmatrix} \frac{1}{14}(\sqrt{177} - 3) & 1 \\ \frac{1}{14}(-\sqrt{177} - 3) & 1 \end{pmatrix}$ 
In[ ]: Eigensystem[B]
Out[ ]:  $\left( \begin{array}{cc} \frac{1}{2}(\sqrt{177} + 13) & \frac{1}{2}(13 - \sqrt{177}) \\ \left\{ \frac{1}{14}(\sqrt{177} - 3), 1 \right\} & \left\{ \frac{1}{14}(-\sqrt{177} - 3), 1 \right\} \end{array} \right)$ 
In[ ]: CharacteristicPolynomial[B, x]
Out[ ]:  $x^2 - 13x - 2$ 
```

### 4.13 Exercises

- Produce the following lists:
  - The square of the reciprocal of the first 12 integers
  - The tuples that contains  $n, n^2, n^3$  from  $n = 1$  to  $n = 6$
- Sort the list in exercise (1a) in ascending order.
- Find the total of the elements in the list in exercise (1a), and write your answer in numerical form using 50 decimal digits.  
**Ans.** 1.5649766384209024901665594306286946979587672228365
- Find the value of  $\sin x$  when  $x = \left\{ \pi, \frac{\pi}{2}, \frac{\pi}{3}, \frac{\pi}{4}, \frac{\pi}{5}, \frac{\pi}{6} \right\}$ .  
**Ans.**  $\left\{ 0, 1, \frac{\sqrt{3}}{2}, \frac{1}{\sqrt{2}}, \sqrt{\frac{5}{8} - \frac{\sqrt{5}}{8}}, \frac{1}{2} \right\}$
- For the list  $\{ \{1,2\}, \{3,4\}, \{5,6\}, \{7,8\}, \{9,10\} \}$ 
  - reverse its elements,
  - reverse the interior sub-lists only,
  - reverse the list and each of its interior elements.
- Which of the following functions is listable?
  - `Length`
  - `Total`

(c) Piecewise functions

7. How to generate a list such as follows using *Mathematica*?

$$\{\{1\}, \{1, 2\}, \{1, 2, 3\}, \{1, 2, 3, 4\}, \{1, 2, 3, 4, 5\}\}$$

8. Generate list of 30 integers between  $-5$  and  $5$  using seed value equals 372.

- Count the number of 0's.

**Ans.** 4

- Find a way to count all those elements of the list which are not 1's. Read more about `Except`.

**Ans.** 27

9. When Newton's method is used to compute  $\sqrt{R}$ ;  $R > 0$  (by solving the equation  $x^2 = R$ ), the sequence of iterates is defined by

$$x_{n+1} = \frac{1}{2} \left( x_n + \frac{R}{x_n} \right)$$

Write a code to generate a list giving the results of applying Newton's formula repeatedly to approximate  $\sqrt{2}$  until successive results are within an error 0.0001 with at most 10-steps.

**Ans.** {0.5, 2.25, 1.56944, 1.42189, 1.41423, 1.41421}

10. Use the built-in functions `FixedPoint` and `FixedPointList` to resolve the parts in Example 4.7.

11. Consider the  $5 \times 5$  tridiagonal matrix  $A = \begin{bmatrix} 2 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix}$

(a) Use `ToeplitzMatrix` to write a code that generates such matrix.

(b) Find the determinant of the matrix  $A$ .

**Ans.** 6

(c) Find the matrix  $A^{-2}$

**Ans.**  $\begin{bmatrix} 55/36 & -20/9 & 9/4 & -16/9 & 35/36 \\ -20/9 & 34/9 & -4 & 29/9 & -16/9 \\ 9/4 & -4 & 19/4 & -4 & 9/4 \\ -16/9 & 29/9 & -4 & 34/9 & -20/9 \\ 35/36 & -16/9 & 9/4 & -20/9 & 55/36 \end{bmatrix}$

(d) A symmetric matrix is a square matrix that is equal to its transpose. Show that  $A$  is symmetric.

12. The Hilbert matrix is a square matrix whose element  $a_{ij}$  is  $\frac{1}{i+j-1}$ . Construct the Hilbert matrix of order 5.

13. Find a unit vector having the same direction as  $v = (1, -2, 2, -3)$ .

**Hint:** see the command `Normalize`.

**Ans.**  $(1/3\sqrt{2}, -\sqrt{2}/3, \sqrt{2}/3, -1/\sqrt{2})$

14. The vector projection of a vector  $a$  on (or onto) a nonzero vector  $b$  (also known as the vector component of  $a$  in the direction of  $b$ ) is the orthogonal projection of  $a$  onto a straight line parallel to  $b$ . It is a vector parallel to  $b$ , defined as  $a \cdot \frac{b}{|b|}$  where the operator  $\cdot$  denotes a dot product, and  $|b|$  is the length of  $b$ . Find the projection of the vector  $a = (1, 2, 3)$  onto the vector  $b = (1, -2, 5)$ .

**Hint:** see the command `Projection`.

**Ans.**  $(2/5, -4/5, 2)$

15. Show that for any  $2 \times 2$  matrix  $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ , where  $a, b, c, d \in \mathbb{R}$ , the following statement is hold:

$$p_A(x) = x^2 - \text{tr}(A)x + \det(A),$$

where  $p_A(x)$  is the characteristic polynomial of the matrix  $A$ .

16. The Cayley-Hamilton theorem, says that every square matrix satisfies its characteristic equation.

Verify the Cayley-Hamilton theorem for the matrix of  $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ .

17. Consider the following data:

$$\text{data} = \{13, 15, 15, 8, 16, 20, 28, 19, 18, 15, 21, 23, 30, 17, 10, 16, 15, 16, 20, 15\}$$

Use *Mathematica* documentation to find for data the: Mean, Median, Variance, and StandardDeviation.

**Ans.**  $35/2, 16, 529/19, 23/\sqrt{19}$

18. Consider the following data:  $\text{data} = \{\{3, 4\}, \{6, 6\}, \{-1, 5\}, \{2, 9\}, \{4, 0\}, \{1, 3\}\}$ . Find the regression line equation that best fits the data.

**Hint:** see the command `Fit`

**Ans.**  $4.79661 - 0.118644x$

### 4.14 Project

Write a code that compresses a list by dividing it into runs of occurrences of a single element, and returns a list of the runs, each represented as a pair containing the element and the length of its run. So the following list,

```
{9, 9, 9, 9, 9, 1, 5, 5, 5, 1, 1, 1, 1, 2, 2}
```

should produce the following encoding.

```
{{9, 5}, {1, 1}, {5, 3}, {1, 4}, {2, 2}}
```

The following are some helpful functions in your code:

- **Split:** splits *list* into sublists consisting of runs of identical elements.
- **Tally:** tallies the elements in *list*, listing all distinct elements together with their multiplicities.
- **First:** gives the first element in *list*.



## 5 Logic and Set Theory

### 5.1 Being Logical

In *Mathematical* logic, statements can have a value of `True` or `False`. These are called *Boolean expressions*. This helps us to “make a decision” and write programs based on the value of a statement. We have seen `==`, which compares the left-hand side with the right-hand side. For example

```
In[]: 3^2+4^2==5^2
Out[]: True
In[]: 3-4==1
Out[]: False
```

Other logical relations are

- `x != y` which is `True` if and only if `x` and `y` have different values.
- `x < y` which is `True` if and only if `x` is numerically less than `y`.
- `x > y` which is `True` if and only if `x` is numerically greater than `y`.
- `x <= y` which is `True` if and only if `x` is numerically less or equal to `y`.
- `x >= y` which is `True` if and only if `x` is numerically greater or equal to `y`.

For example,

```
In[]: 3^2+4^2>=5^2
Out[]: True
In[]: Sqrt[49]!=7
Out[]: False
In[]: E > Pi
Out[]: False
```

**Note** In logical expressions, `!` means the negation ( $\sim$ ). If we type in *Mathematica* `3!` this means the factorial of 3 which equals 6. We can express the factorial function of an `expr` using the *Mathematica* built-in function `Factorial[expr]`.

```
In[]: 3! != Factorial[3]
Out[]: False
```

### 5.2 Truth Tables

One can combine logical statements with the usual Boolean operations `And` ( $\wedge$ ), `Or` ( $\vee$ ) or the equivalent shortcuts `&&`, `||`. In general, for two statements `A` and `B`, the statement `A & B` is false if one of `A` or `B` is false and `A | B` is true if one of them is true. In order to produce all possible combinations of true and false cases, we use the command `BooleanTable` as the following example shows.

```
In[]: BooleanTable[{p, q, p&q}] /.{True->T, False->F} //Grid
      T T T
Out[]: T F F
      F T F
      F F F

In[]: BooleanTable[{p, q, p||q}] /.{True->T, False->F} //Grid
      T T T
Out[]: T F T
      F T T
      F F F
```

In logic, a tautology is a formula that is true in every possible interpretation. `TautologyQ[bf]` gives `True` if all combinations of values of variables make the Boolean function `bf` yield `True`.

```
In[]: BooleanTable[(a||b) || (!a && !b)]
Out[]: {True, True, True, True}
In[]: TautologyQ[(a||b) || (!a && !b)]
Out[]: True
```

### 5.3 Element “ $\in$ ”

In *Mathematica*, for a variable, one can specify certain domains. This means that the variable takes its values from a specific type of data. The domains available are Algebraics, Booleans, Complexes, Integers, Primes, Rationals and Reals. One of the fundamental theorems in number theory is to show that  $\pi$  is not a rational number, i.e., is not of the form  $m/n$ , where  $m$  and  $n \neq 0$  are integers. Look at the following examples:

```
In[]: Element[Pi, Rationals]
Out[]: False
In[]: Element[Sqrt[25], Integers]
Out[]: True
In[]: IntegerQ[Sqrt[25]]
Out[]: True
In[]: Element[131, Primes]
Out[]: True
In[]: PrimeQ[131]
Out[]: True
```

### 5.4 Handling Sets

Now it has been agreed among mathematicians that any mathematics starts by considering sets, i.e., collections of objects. As we mentioned, the difference between *Mathematical* sets and lists in *Mathematica* is that lists respect order and repetition, which is to say one can have several copies of one object in a list. Sets are not sensitive about repeated objects, e.g., the set  $\{a, b\}$  is the same as the set  $\{a, b, b, a\}$ . There is no concept of sets in *Mathematica* and if necessary one considers a list as a set. If one wants to get rid of duplication in a list, one can use

```
In[]: DeleteDuplicates[{a, b, b, a}]
Out[]: {a,b}
```

Considering two sets, the natural operations between them are union and intersection. *Mathematica* provides *Union* to collect all elements from different lists into one list (after removing all the duplication) and *Intersection* for collecting common elements (again discarding repeated elements). The following examples show how these commands work.

```
In[]: U = {1, 2, 3, 4, 5, 7};
In[]: A = {1, 4, 7, 3};
In[]: B = {5, 4, 3, 2}
In[]: Union[A,B]
Out[]: {1,2,3,4,5,7}
In[]: Intersection[A,B]
Out[]: {3,4}
In[]: Complement[U,A]
Out[]: {2,5}
```

The command *Complement*[U,A] will give the elements of the universal set U which are not in the set A.

In mathematics, the power set (or powerset) of any set S, written  $\mathcal{P}(S)$ , is the set of all subsets of S, including the empty set and S itself. For example, the powerset of the set  $\{1, 3, 2\}$  is

```
In[]: Subsets[{1, 3, 2}]
Out[]: {{}, {1}, {3}, {2}, {1, 3}, {1, 2}, {3, 2}, {1, 3, 2}}
```

To check up a set is a subset of a powerset, we use the function *SubsetQ*[A,B], which yields True if B is a subset of A, and False otherwise.

```
In[]: SubsetQ[{1, 2, 3}, {3, 1}]
Out[]: True
```

In mathematics, two sets are said to be disjoint if they have no element in common. For example,

```
In[]: DisjointQ[{2, 4, 6}, {1, 3, 5}]
Out[]: True
```

### 5.5 Quantifiers

In a statement like  $x^4 + x^2 > 0$ , *Mathematica* treats the variable  $x$  as having a definite, though unspecified, value. Sometimes, however, it is useful to be able to make statements about whole collections of possible values for  $x$ . You can do this using quantifiers  $\forall$  *ForAll* and  $\exists$  *Exists*. In most cases, the quantifiers can be simplified by the *Resolve* function. The command *Resolve*[*expr*] attempts to resolve *expr* into a form that eliminates *ForAll* and *Exists* quantifiers. For example, the following is true.

```
In[]: Resolve@Exists[x, x^2 + x^4 > 0]
Out[]: True
```

But, the following is false since  $x = 0$  does not satisfy the statement.

```
In[]: Resolve@ForAll[x, x^2 + x^4 > 0]
Out[]: False
```

**Example 5.1.** Decide whether the statement is True or False.

1. There exists a natural number  $n$  such that  $n^2 > 2^n$ .

```
In[]: Resolve[Exists[n, n^2 > 2^n]]
Out[]: True
```

2. For all natural numbers  $n$ , if  $n \geq 5$ , then  $n^2 < 2^n$ .

```
In[]: Resolve[ForAll[n, n>=5, n^2 < 2^n], Integers]
Out[]: True
```

3. The equation  $x^2 + x = 1$  has a negative real number solution.

```
In[]: Resolve[Exists[x, x<0, x^2+x==1], Reals]
Out[]: True
```

4. For all  $x \in \mathbb{R}$  and  $x < 1$ ,  $x^3 - x \leq 0$ .

```
In[]: Resolve[ForAll[x, x<1, x^3-x<=0], Reals]
Out[]: False
```

5. There exists boolean statements  $p$  and  $q$  such that  $(p \vee q) \wedge \sim q$  is true.

```
In[]: Resolve[Exists[{p,q}, (p||q) && !q], Booleans]
Out[]: True
```

6. Every integer is either even or odd.

$\forall n \in \mathbb{Z}, \exists m \in \mathbb{Z}$  such that  $n = 2m$  or  $n = 2m + 1$

```
In[]: Resolve[ForAll[n, Exists[m, n==2 m || n==2m+1]], Integers]
Out[]: True
```

## 5.6 Exercises

1. Construct the truth table for the statement  $p \vee (q \wedge r)$ .
2. Show that the statement  $(p \wedge q) \vee (\sim p \vee \sim q)$  is a tautology.
3. A *contradiction* is a formula that is false in every possible interpretation. Show that the statement  $(p \wedge \sim q) \Leftrightarrow (p \Rightarrow q)$  is a contradiction.  
**Hint:** use *Mathematica*'s help to read about the commands *Equivalent* ( $\Leftrightarrow$ ) and *Implies* ( $\Rightarrow$ ).
4. Let  $U = \{1, 2, 3, 4, 5, 6, 7\}$ ,  $A = \{2, 5, 7\}$ , and  $B = \{3, 4, 6, 7\}$ .
  - (a) Determine whether  $A$ , and the complement of  $B$  are disjoint or not.
  - (b) Evaluate  $A \cup B$ ,  $B \cap A$ , and  $\mathcal{P}(A) \cap \mathcal{P}(B)$ .
  - (c) Show that  $A \subseteq A \cup B$  and  $A \cap B \subseteq A$ .
5. Let  $\mathbf{F}$  be the set that contains 10 elements. Read more about the command *Subsets* to answer the following questions:
  - (a) How many subsets of  $\mathbf{F}$  are there contains at most 7 elements?  
*Ans. 968*
  - (b) How many subsets of  $\mathbf{F}$  are there contains at exactly 7 elements?  
*Ans. 120*
  - (c) How many subsets of  $\mathbf{F}$  are there contains between 3 and 7 elements?  
*Ans. 912*
6. Which of the following are true?
  - (a)  $\exists x \in \mathbb{R}$  such that  $3^x = x^2$ .
  - (b)  $\exists x \in \mathbb{R}$  such that  $3^x = x$ .
  - (c)  $\forall x \in \mathbb{R}$ ,  $x^2 + 4x + 5 \geq 0$ .
  - (d)  $\forall x \in \mathbb{N}$ ,  $\exists y \in \mathbb{N}$  such that  $x = 3y$  or  $x = 3y + 1$  or  $x = 3y + 2$ .
  - (e)  $-\frac{1}{2} \leq \frac{(x+y)(1-xy)}{(1+x^2)(1+y^2)} \leq \frac{1}{2}$  for all  $x, y \in \mathbb{R}$ .

5.7 Project

Create a function `TruthTable[expr, vars]` that takes a logical expression such as  $p \wedge q$  and outputs a truth table similar to the following.

<i>a</i>	<i>b</i>	$a \wedge b$
T	T	T
T	F	F
F	T	F
F	F	F

You can create a list of truth values using `Tuples`. For example,

```
Tuples[{True, False}, 2]
{{True, True}, {True, False}, {False, True}, {False, False}}
```

You will also find it helpful to consider threading rules over the tuples using `MapThread` or `Thread`.

## 6 Number Theory

### 6.1 Primes

A prime number (or a prime) is a natural number greater than 1 that has no positive divisors other than 1 and itself. A natural number greater than 1 that is not a prime number is called a composite number. For example, 5 is prime because 1 and 5 are its only positive integer factors, whereas 6 is composite because it has the divisors 2 and 3 in addition to 1 and 6. Either the built-in function `PrimeQ` or `CompositeQ` will help you to determine whether an integer is prime or composite.

- `PrimeQ[expr]` yields `True` if `expr` is a prime number, and yields `False` otherwise.
- `CompositeQ[expr]` yields `False` if `expr` is a prime number, and yields `True` otherwise.

For example, the number 13 is prime while  $10^{100} + 1$  is composite since

```
In[]: PrimeQ[13]
Out[]: True
In[]: CompositeQ[13]
Out[]: False
In[]: PrimeQ[10^100+1]
Out[]: False
In[]: CompositeQ[10^100+1]
Out[]: True
```

**Example 6.1.** Prove that for every integer  $n > 2$ , the number  $2^{2^n-1} + 1$  is not a prime number.

```
In[]: Resolve[ForAll[n, n>2, !PrimeQ[2^(2^n-1)+1]], Integers]
Out[]: True
```

The first few primes are 2, 3, 5, 7, 11, 13,  $\dots$ . The first prime is 2, the second is 3, the sixth is 13, and so on. In *Mathematica*, `Prime[n]` gives the  $n^{\text{th}}$  prime number. For example,

```
In[]: Prime[1]
Out[]: 2
In[]: Prime[6]
Out[]: 13
In[]: Prime[2016]
Out[]: 17519
```

`Prime[n]` is listable function, that is it can find the primes at many orders or positions. For example,

```
In[]: Prime[{1,6,2016}]
Out[]: {2,13,17519}
```

**Example 6.2.** Find the first 10 primes.

```
In[]: Prime[Range[10]]
Out[]: {2,3,5,7,11,13,17,19,23,29}
```

To find a prime number above a certain value  $n$ , you may use `NextPrime[n]`. If you would the  $k^{\text{th}}$  prime above  $n$  you should use `NextPrime[n,k]`. For example,

```
In[]: NextPrime[4]
Out[]: 5
In[]: NextPrime[4,3]
Out[]: 11
```

If  $k$  is negative in `NextPrime[n,k]`, then the result is the  $k^{\text{th}}$  prime previous to  $n$ .

```
In[]: NextPrime[88,-5]
Out[]: 67
```

The number of primes less than or equal to  $x$  is evaluated using `PrimePi[x]`. For example, there are 25 primes less than or equal to 100 since

```
In[]: PrimePi[100]
Out[]: 25
```

**Example 6.3.** How many 3-digit prime numbers are there?

```
In[]: PrimePi[999] - PrimePi[100]
Out[]: 143
```

## 6.2 Integer Factorization

In number theory, the prime factors of a positive integer are the prime numbers that divide that integer exactly. The prime factorization of a positive integer is a list of the integer's prime factors, together with their multiplicities; the process of determining these factors is called integer factorization. The fundamental theorem of arithmetic says that every positive integer has a single unique prime factorization. To shorten prime factorizations, factors are often expressed in powers (multiplicities). For example,  $360 = 2^3 \times 3^2 \times 5$ . In *Mathematica*, we use

```
In[]: FactorInteger[360]
Out[]: {{2, 3}, {3, 2}, {5, 1}}
```

The list of all integers that divides a number  $n$  can be evaluated using `Divisors[n]`. For example,

```
In[]: Divisors[10]
Out[]: {1, 2, 5, 10}
In[]: Divisors[13]
Out[]: {1, 13}
```

**Example 6.4.** How many divisors are there for  $13!$ .

```
In[]: Length@Divisors[13!]
Out[]: 1584
```

**Example 6.5.** How many integers are there divide both 1545 and 1230.

```
In[]: Intersection[Divisors[1545], Divisors[1230]] // Length
Out[]: 4
```

Sometimes, you need to know the highest power  $k$  of  $b$  such that  $b^k$  divides  $n$ . To do this, use the *Mathematica* function `IntegerExponent[n,b]`. For example, the highest power of 2 that divides 360 is 3.

```
In[]: IntegerExponent[360,2]
Out[]: 3
```

**Example 6.6.** With how many zeros will  $130!$  end ?

```
In[]: IntegerExponent[130!,10]
Out[]: 32
```

The command `IntegerLength[n]` in *Mathematica* gives the number of digits in the base 10 representation of the integer  $n$ . Also, `DigitCount[n,b]` gives a list of the numbers of 1, 2, ...,  $b-1$ , 0 digits in the base- $b$  representation of  $n$ .

**Example 6.7.** How many digits are there for  $130!$  ?

```
In[]: IntegerLength[130!]
Out[]: 220
```

and they are distributed as follows:

```
14 1's, 21 2's, 14 3's, 23 4's, 24 5's,
16 6's, 20 7's, 15 8's, 22 9's, 51 0's,
```

since

```
In[]: DigitCount[130!]
Out[]: {14, 21, 14, 23, 24, 16, 20, 15, 22, 51}
```

**Example 6.8.** Find only the number of 0's, 5's and 9's do appear in  $16837^{1921}$ .

```
In[]: DigitCount[16837^1921, 10, #] & /@ {0, 5, 9}
Out[]: {822, 790, 821}
```

## 6.3 Digits in Numbers

In this section we learn how to convert between numbers and lists. To obtain a list of the decimal digits in the integer  $n$ , you may use `IntegerDigits[n]`. To construct an integer from the list of its decimal digits use `FromDigits[list]`.

**Example 6.9.** Reverse the integer 365435296161.

```
In[]: FromDigits[Reverse[IntegerDigits[365435296161]]]
Out[]: 161692534563
In[]: IntegerReverse[365435296161]
Out[]: 161692534563
```

**Example 6.10.** Find the sum of the digits of  $1980!$ .

```
In[]: Total@IntegerDigits[1980!]
Out[]: 23418
```

## 6.4 Fibonacci Sequence

As we have seen in Chapter 3, the sequence  $1, 1, 2, 3, 5, 8, 13, 21, \dots$  is called Fibonacci sequence. By definition, the first two numbers in the Fibonacci sequence are 1 and 1, and each subsequent number is the sum of the previous two. The *Mathematica* function `Fibonacci[n]` gives the  $n^{\text{th}}$  Fibonacci number.

```
In[]: Fibonacci[7]
Out[]: 13
```

`Fibonacci[n]` is listable function, so you can find, for example, the first 14 Fibonacci numbers as follows.

```
In[]: Fibonacci[Range[14]]
Out[]: {1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377}
```

**Example 6.11.** Find the sum of the first 100 Fibonacci numbers.

```
In[]: Total@Fibonacci[Range[100]]
Out[]: 927372692193078999175
```

**Example 6.12.** Determine whether the  $13^{\text{th}}$  Fibonacci number is prime.

```
In[]: PrimeQ[Fibonacci[13]]
Out[]: True
```

## 6.5 Number Theoretic Functions

In mathematics, the remainder is the amount “left over” after performing some computation. In arithmetic, the remainder is the integer “left over” after dividing one integer by another to produce an integer quotient (integer division). In computing, the modulo operation finds the remainder after division of one number by another (sometimes called modulus).

Given two positive numbers,  $a$  (the dividend) and  $n$  (the divisor),  $a$  modulo  $n$  (abbreviated as  $a \bmod n$ ) is the remainder of the Euclidean division of  $a$  by  $n$ . For instance, the expression “ $5 \bmod 2$ ” would evaluate to 1 because 5 divided by 2 leaves a quotient of 2 and a remainder of 1, while “ $9 \bmod 3$ ” would evaluate to 0 because the division of 9 by 3 has a quotient of 3 and leaves a remainder of 0; there is nothing to subtract from 9 after multiplying 3 times 3.

**Quotient and Remainder** In *Mathematica*, if we divide  $a$  by  $n$ , then the quotient is evaluated using `Quotient[a, n]` and the remainder (modulus) is `Mod[a, n]`. For example,

```
In[]: Quotient[5, 2]
Out[]: 2
In[]: Mod[5, 2]
Out[]: 1
In[]: QuotientRemainder[5, 2]
Out[]: {2, 1}
In[]: QuotientRemainder[9, 3]
Out[]: {3, 0}
```

If the remainder is 0 when we divide  $a$  by  $n$ , then we say that  $a$  is divisible by  $n$ . To test whether  $m$  is divisible by  $n$  you may use `Divisible[m, n]` which yields `True` if  $m$  is divisible by  $n$ , and yields `False` if it is not. For example,

```
In[]: Divisible[5, 2]
Out[]: False
In[]: Divisible[130!, 1210]
Out[]: True
In[]: Mod[130!, 1210]
Out[]: 0
```

**GCD and LCM** In mathematics, the greatest common divisor (gcd) of two or more integers, when at least one of them is not zero, is the largest positive integer that divides the numbers without a remainder. For example, the GCD of 8 and 12 is 4.

```
In[]: GCD[8, 12]
Out[]: 4
```

The least common multiple of two or more integers is the smallest positive integer that is divisible by all of them. Since division of integers by zero is undefined, this definition has meaning only if the integers are different from zero. For example, the LCM of 8 and 12 is 24.

```
In[]: LCM[8, 12]
Out[]: 24
```

**Example 6.13.** Find the GCD and LCM for the numbers 2145, 1716, 9918.

```
In[]: GCD[2145, 1716, 9918]
Out[]: 3
In[]: LCM[2145, 1716, 9918]
Out[]: 14182740
```

In number theory, two integers  $a$  and  $b$  are said to be relatively prime, mutually prime, or coprime if the only positive integer that evenly divides both of them is 1. That is, the only common positive factor of the two numbers is 1. This is equivalent to their greatest common divisor being 1. For example, 8, 9, and 11 are relatively prime since

```
In[]: CoprimeQ[8, 9, 11]
Out[]: True
```

The function `CoprimeQ[n1, n2, ...]` yields `True` if all pairs of the  $n_i$  are relatively prime, and yields `False` otherwise. For example, 6, 9, and 11 are not relatively prime since 3 divides 6 and 9.

```
In[]: CoprimeQ[6, 9, 11]
Out[]: False
```

## 6.6 Selecting from Lists

So far we have been able to create a list of data by using functions producing lists. The next step is to be able to choose, from a list, certain data which fit a specific description. This can be achieved using the command `Select`. The structure of `Select` is as follows.

```
Select[list, crit]
picks out all elements ei of list for which crit[ei] is True.

Select[list, crit, n]
picks out the first n elements for which crit[ei] is True.
```

**Example 6.14.** From the first 25 Fibonacci numbers, find the even numbers? find the prime numbers?

```
In[]: Select[Fibonacci[Range[25]], EvenQ]
Out[]: {2, 8, 34, 144, 610, 2584, 10946, 46368}
In[]: Select[Fibonacci[Range[25]], PrimeQ[#] &]
Out[]: {2, 3, 5, 13, 89, 233, 1597, 28657}
```

**Example 6.15.** How many 3-digits integer has at least 30 divisors?

```
In[]: Length@Select[Range[100, 999], Length[Divisors[#]] >= 30 &]
Out[]: 2
```

**Example 6.16.** An integer  $p$  is called prime-palindromic if it is prime and the number when we reverse the digits of  $p$  is also prime (for example 941). How many prime-palindromic numbers are there up to 1000.

```
In[]: list = Prime[Range[PrimePi[1000]]];
In[]: crit[x_] := PrimeQ[IntegerReverse[x]];
In[]: Select[list, crit] // Length
Out[]: 56
```

**Example 6.17.** Notice that  $12^2 = 144$  and  $21^2 = 441$ , i.e., the numbers and their squares are reverses of each other. Find all the numbers up to 100 with this property.

```
In[]: Select[Range[100], IntegerReverse[#]^2 == IntegerReverse[#^2] &]
Out[]: {1, 2, 3, 10, 11, 12, 13, 20, 21, 22, 30, 31, 100}
```

**Example 6.18. Twin primes** are pairs of primes of the form  $(p, q)$  such that  $q=p+2$ . For example, (3,5) and (41,43) are twin primes. How many twin primes are there up to 429?

```
In[]: Select[Prime[Range[PrimePi[429]]], PrimeQ[#+2] &] //Length
Out[]: 22
```

**Example 6.19.** A number is called a Harshad number if it is divisible by the sum of its digits (e.g., 12 is Harshad as it is divisible by  $1+2=3$ ). Find all 2-digit Harshad numbers.



```

In[]: Select[Range[10,99], Divisible[#,Total@IntegerDigits[#]]&]
In[]: Select[Range[10,99], Mod[#,Total@IntegerDigits[#]]==0&]
Out[]: {10, 12, 18, 20, 21, 24, 27, 30, 36, 40, 42, 45, 48, 50, 54,
        60, 63, 70, 72, 80, 81, 84, 90}

```

## 6.7 Exercises

- Find the 12<sup>th</sup> prime number?

*Ans. 37*

- Determine whether  $2^5 - 1$  is prime or composite? What is the prime number next to  $2^5 - 1$ ? What is the 5th prime number next to  $2^5 - 1$ ? What is the 5th prime number previous to  $2^5 - 1$ ?

*Ans. Prime, 37, 53, 13*

- How many primes are there less than 1000? How many 6-digit primes are there?

*Ans. 168, 68906*

- Factor the integer 12345. Write all its divisors. How many are there?

*Ans.  $3 \times 5 \times 823$ , {1, 3, 5, 15, 823, 2469, 4115, 12345}, 8*

- Find the least common multiple and the greatest common divisor of the integers 16, 24, 524.

*Ans. 4, 6288*

- Find the quotient and remainder when  $13!$  is divided by 2256.

*Ans. 2760204, 576*

- How many positive integers are there divides both  $12!$  and the 36<sup>th</sup> Fibonacci number, but not  $2^{100} - 1$ .

*Ans. 18*

- Find the sum of the first 100 primes. Find the sum of the first 123 even integers.

*Ans. 24133, 15252*

- Show that among the first 500 Fibonacci numbers, 18 of them are prime. Which of the first 10 Fibonacci number is not prime?

- We call *repunits* the numbers that contain only the digit 1 in their writing, namely numbers of the form  $111 \dots 11$ .

- We know that 11 is a prime repunit number. One wonders what is the next prime repunit. Find all such numbers up to 500 digits of ones.

- Find the smallest repunit divisible by 19.

*Ans. 111111111111111111*

- The sum of two positive integers is 5432 and their least common multiple is 223020. Find the numbers.

*Ans. 1652, 3780*

- Find the smallest positive multiple of 99999 that contains no 9's amongst its digits.

*Ans. 11112*

- A Dudeney number is a positive integer that is a perfect cube such that the sum of its decimal digits is equal to the cube root of the number. For example,

$$4913 = 17^3 \quad \text{where} \quad 17 = 4 + 9 + 1 + 3$$

Find all 5 digits Dudeney numbers.

*Ans. 17576, 19683*

- A **palindromic number** is a number that remains the same when its digits are reversed. The command `PalindromeQ[n]` returns True if the integer `n` is identical to `IntegerReverse[n]`, and False otherwise. From the first 100 primes, how many of them is palindromic?

- Show that: All 4-digits palindromic numbers are composite. **Hint:** You may use the command `AllTrue`.

- Show that: All 4-digits palindromic numbers are divisible by 11.

- A number is perfect if it is equal to the sum of its proper divisors. For example,  $6 = 1 + 2 + 3$  is perfect, while  $18 \neq 1 + 2 + 3 + 6 + 9$  is not. You may use the command `PerfectNumberQ` to test whether a number is perfect or not. For example, `PerfectNumberQ[6]` returns True, while `PerfectNumberQ[18]` returns False.

- (a) Write a program to find all the perfect numbers up to 10000.
- (b) Use the command `PerfectNumber` to find the first 7 perfect numbers.
16. A **partition** of a positive integer  $n$ , also called an **integer partition**, is a way of writing  $n$  as a sum of positive integers. Two sums that differ only in the order of their summands are considered the same partition. For example, 4 can be partitioned in five distinct ways:

$$4 = 4$$
$$= 3 + 1$$
$$= 2 + 2$$
$$= 2 + 1 + 1$$
$$= 1 + 1 + 1 + 1$$

Use the command `IntegerPartitions` to answer the following questions.

- (a) Give a list of all possible ways to partition the integer 5 into smaller integers ?
- (b) Partitions 5 into at most 3 integers ?
- (c) Partitions 5 into exactly 3 integers ?
- (d) Find all partitions of 5 that involve only 1 and 3 ?
- (e) How many ways are there to make change for 156 JOD with the paper coins 1, 5, 10, 20, and 50 with exactly 10 paper coins?
17. In mathematics, a **Mersenne prime** is a prime number of the form  $M_n = 2^n - 1$  for some integer  $n$ . In order for  $M_n$  to be prime,  $n$  must itself be prime. This is true since for composite  $n$  with factors  $a$  and  $b$ ,  $n = ab$ . Therefore,  $2^n - 1$  can be written as  $2^{ab} - 1$ , which always has the factors  $(2^a - 1)$  and  $(2^b - 1)$ . The first five Mersenne primes are  $M_n = 3, 7, 31, 127, 8191$  corresponding to  $n = 2, 3, 5, 7, 13$ .
- (a) In *Mathematica*, the command `MersennePrimeExponentQ[n]` returns `True` if  $n$  is a Mersenne prime exponent, and `False` otherwise. Using this command, find all Mersenne prime exponents smaller than 10000.
- (b) The command `MersennePrimeExponent[n]` gives the  $n^{\text{th}}$  Mersenne prime exponent. What is the  $40^{\text{th}}$  Mersenne prime exponent? How many digits are there in the  $40^{\text{th}}$  Mersenne prime number?
- Ans. 20996011, 8*

18. **Goldbach's conjecture** is one of the oldest and best-known unsolved problems in number theory and all of mathematics. It states:

**Every even integer greater than 2 can be expressed as the sum of two primes.**

The expression of a given even number as a sum of two primes is called a Goldbach partition of that number. The following are examples of Goldbach partitions for some even numbers:

$$6=3 + 3$$
$$8=3 + 5$$
$$10=3 + 7 = 5 + 5$$

Write a code using *Mathematica* to find all possible Goldbach partitions given any even integer  $n > 2$ , then use it to find all Goldbach partitions for  $n = 430$ .  
**Hint:** Is is useful to use the command `IntegerPartitions`.

19. How does the `Apply` command function in *Mathematica*, and how can it be utilized, alongside other commands, to express the prime factorization of a number such as 12, as  $2^2 \cdot 3^1$ ?

6.8 Project

1. In mathematics, the **look-and-say** sequence is the sequence of integers beginning as follows:

$$1, 11, 21, 1211, 111221, 312211, 13112221, 1113213211, \dots$$

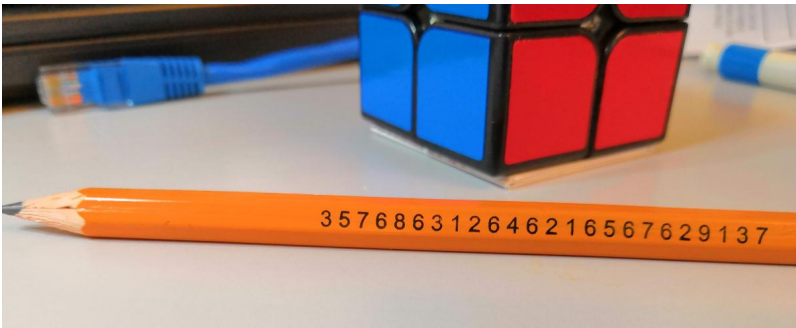
To generate a member of the sequence from the previous member, read off the digits of the previous member, counting the number of digits in groups of the same digit. For example:

1	is read off as	"one 1" or 11.
11	is read off as	"two 1s" or 21.
21	is read off as	"one 2, one 1" or 1211.
1211	is read off as	"one 1, one 2, two 1s" or 111221.
111221	is read off as	"three 1s, two 2s, one 1" or 312211.

The look-and-say sequence was analyzed by *John Conway* after he was introduced to it by one of his students at a party. Write a *Mathematica* code to generate the look-and-say sequence given a starting digit  $d$  from 0 to 9 with a length  $n$ . Project 4.14 in Chapter 4 is helpful for your code.

2. A PRIME PENCIL: TRUNCATABLE PRIMES

In number theory, a left-truncatable prime is a prime number which contains no 0, and if the leading (“left”) digit is successively removed, then all resulting numbers are prime. For example, 9137, since 9137, 137, 37 and 7 are all prime. Write a Mathematica code to find the *largest* left-truncatable prime.



The idea of *recursion* in Section 3.7 from Chapter 3 is helpful in your code. In addition, you may need the following built-in command: `Prepend`.

## 7 Computer Algebra and Solving Equations

The Wolfram Language has a variety of commands for algebraic manipulation operations like expansion and factoring of polynomials, addition of fractions with unlike denominators, and collection of terms with like variables. These commands have names that describe exactly what they do, making it intuitive for new users to find the right commands for their needs.

### 7.1 Working with Polynomials and Powers

One of the abilities of Mathematica is to handle symbolic computations, i.e., Mathematica can comfortably work with symbols. For example, the `Expand` command does exactly what its name says it does:

```
In[]: Expand [(x-2)(x-3)(x+1)^2]
Out[]: x^4 - 3x^3 - 3x^2 + 7x + 6
```

Mathematica can also do the inverse of this task, namely factorize an expression:

```
In[]: Factor[x^2 + 2 x + 1]
Out[]: (1 + x)^2
```

While expansion of an algebraic expression is a simple and routine procedure, the factorization of algebraic expressions is often quite challenging. My favorite example is this one.

```
In[]: Factor[x^10 + x^5 + 1]
Out[]: (1 + x + x^2) (1 - x + x^3 - x^4 + x^5 - x^7 + x^8)
```

`Factor` only works completely if all the roots are rational.

```
In[]: Factor[x^2 - 2]
Out[]: -2 + x^2
```

`Factor` will not use complex numbers in its answer  $x^2 + 1$  unless one of the coefficients is complex, so it won't write  $x^2 + 1 = (x - i)(x + i)$ .

```
In[]: Factor[x^2 + 1]
Out[]: 1 + x^2
```

Setting `Factor[poly, Extension->All]` will extend the domain of coefficients to include any irrational or complex numbers.

```
In[]: Factor[x^2 + 2, Extension -> All]
Out[]: (x - i√2) (x + i√2)
In[]: Factor[2 + 2 Sqrt[2] x + x^2, Extension -> All]
Out[]: (x + √2)^2
```

`Expand` and `Factor` do a nice job even when you use numerical coefficients.

```
In[]: Expand[(x - 1.25) (2 x - 0.5)]
Out[]: 0.625 - 3.x + 2x^2
In[]: Factor[0.625 - 3 x + 2 x^2]
Out[]: 2.(-1.25 + x)(-0.25 + x)
```

The `Expand` and `Factor` commands also work for polynomials with more than one variable.

```
In[]: Expand[(2 x - 5 y + 3 z)^2]
Out[]: 4x^2 - 20xy + 25y^2 + 12xz - 30yz + 9z^2
In[]: Factor[4 x^2 - 20 x y + 25 y^2 + 12 x z - 30 y z + 9 z^2]
Out[]: (2x - 5y + 3z)^2
```

The `Simplify` command produces an expression that is the shortest to write out. More often than not, this will agree with what you think of as “simplest”. For example,  $x^2 - 2x + 1 = (x - 1)^2$  is factored when you `Simplify` it:

```
In[]: Simplify[x^2 - 2 x + 1]
Out[]: (-1 + x)^2
```

However, if we ask Mathematica to simplify  $x^3 + 2x^2 - 2x - 1$ ,

```
In[]: Simplify[x^3 + 2 x^2 - 2 x - 1]
Out[]: -1 - 2x + 2x^2 + x^3
```

Mathematica returns the expression, even though  $x^3 + 2x^2 - 2x - 1$  can be factored as  $(x - 1)(x^2 + 3x + 1)$ . Mathematica thinks that writing this cubic polynomial directly with 4 terms is simpler than factoring it into two expressions with a total of 5 terms. The `Expand` and `Simplify` commands can't help much when expressions have fractional powers (i.e. roots). For example, you might think that  $\sqrt{x^2} = (x^2)^{1/2} = x$ , but Mathematica doesn't agree:

```
In[]: Simplify[Sqrt[x^2]]
Out[]:  $\sqrt{x^2}$ 
```

The reason for the apparent failure should be clear:  $\sqrt{x^2} = x$  only if  $x \geq 0$ . Sometimes the reason that Mathematica does not simplify expressions is that it treats  $x$  as a complex number. The `Simplify` command supports an `Assumptions` option with which we can force Mathematica to treat  $x$ , say, as a nonnegative real number, or simply a real number. Here's what happens with the example above when including appropriate assumptions:

```
In[]: Simplify[Sqrt[x^2], Assumptions -> x >= 0]
Out[]: x

In[]: Simplify[Sqrt[x^2], Assumptions -> Element[x, Reals]]
Out[]: Abs[x]

In[]: Simplify[(x^6)^(1/3), Assumptions -> Element[x, Reals]]
Out[]: x^2
```

### 7.2 Working with Rational Functions

A rational function is an expression of the form  $\frac{P(x)}{Q(x)}$ , where  $P$  and  $Q$  are polynomials of a single variable  $x$ . The following are three common algebraic operations involving rational functions. Combining terms over a common denominator can be achieved with the `Together` command. For example, to combine  $\frac{2}{3x+1} + \frac{5x}{x+2}$ :

```
In[]: Together[2/(3 x + 1) + 5 x/(x + 2)]
Out[]:  $\frac{4 + 7x + 15x^2}{(2 + x)(1 + 3x)}$ 
```

Splitting up a rational function into its partial fraction decomposition can be done with the `Apart` command. For example, to decompose  $\frac{11x^2 - 17x}{(x - 1)^2(2x + 1)}$

```
In[]: Apart[(11 x^2 - 17 x)/((x - 1)^2 (2 x + 1))]
Out[]:  $-\frac{2}{(-1 + x)^2} + \frac{3}{-1 + x} + \frac{5}{1 + 2x}$ 
```

The `Apart` command also does long division. The quotient  $\frac{x^5 - 2x^2 + 6x + 1}{x^2 + x + 1}$  is found with:

```
In[]: Apart[(x^5 - 2 x^2 + 6 x + 1)/(x^2 + x + 1)]
Out[]:  $-1 - x^2 + x^3 + \frac{2 + 7x}{1 + x + x^2}$ 
```

### 7.3 Working with Transcendental Functions

The `Simplify` command has reasonable success working many basic identities involving the trigonometric and hyperbolic functions, such as

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

In[]: Simplify[Sin[x/2] Cos[x/2]]
Out[]:  $\frac{\text{Sin}[x]}{2}$ 

In[]: Simplify[Cosh[x]^2 - Sinh[x]^2]
Out[]: 1
```

The `TrigExpand` and `TrigReduce` commands provide alternate methods of working with trigonometric functions, performing the jobs suggested by their names as follows.

<code>TrigExpand[expr]</code>	Splits up sums and integer multiples that appear in arguments of trigonometric functions, and then expands out products of trigonometric functions into sums of powers, using trigonometric identities when possible.
-------------------------------	---

For example:

```
In[]: TrigExpand[Sin[2 x]]
Out[]: 2Cos[x]Sin[x]
In[]: TrigExpand[Cos[x + y]]
Out[]: Cos[x]Cos[y] - Sin[x]Sin[y]
In[]: TrigExpand[Sin[2 x] Cos[3 x]]
Out[]: -Sin[x]/2 + 5/2 Cos[x]^4 Sin[x] - 5Cos[x]^2 Sin[x]^3 + Sin[x]^5/2
```

TrigReduce[ <i>expr</i> ]	Given a trigonometric polynomial, TrigReduce typically yields a linear expression involving trigonometric functions with more complicated arguments.
---------------------------	--

For example:

```
In[]: TrigReduce[2 Cos[x]^2]
Out[]: 1 + Cos[2x]
In[]: TrigReduce[2 Sin[x] Cos[y]]
Out[]: Sin[x - y] + Sin[x + y]
In[]: TrigReduce[Sin[2 x] Cos[3 x]]
Out[]: 1/2 (-Sin[x] + Sin[5x])
```

**The FullSimplify Command** You may (and we usually do) prefer to use FullSimplify when working with transcendental functions. Think of this command as a “full-strength” version of Simplify. It uses a wider variety of expression simplification possibilities than Simplify (although it could take a long time with complicated expressions).

**Example 7.1.** Find the exact value of  $\sqrt{2} + \sqrt{3} - \sqrt{2\sqrt{6} + 5}$ .

```
In[]: FullSimplify[Sqrt[2] + Sqrt[3] - Sqrt[5+2Sqrt[6]]]
Out[]: 0
```

7.4 Equations and Their Solutions

Many commands are available for equation solving, from solving over specific domains to returning general solutions to finding roots of equations. For new users, becoming familiar with just a few of the most common commands is sufficient to solve many different types of problems.

Solving equations and finding roots for different types of equations and relations are one of the main endeavors of mathematics. For polynomials with one variable of the form  $a_nx^n + a_{n-1}x^{n-1} + \cdots + a_1x + a_0$ , it has been proved that there is no formula for finding the roots when  $n \geq 5$  (in fact, when  $n = 3$  or  $n = 4$ , the formulas are not that pretty!). This forces us to find numerical ways to estimate the roots of the equations.

Using Wolfram Mathematica we have several commands at our disposal. There are different kinds of equations and they require different commands to find the roots.

**The Solve Command for an Equation** Mathematica's Solve command will solve an equation for an unknown variable. You use it in the form:

```
Solve[ an equation , variable(s) to solve for ]
```

For example,

```
In[]: Solve[2 x + 5 == 9, x]
Out[]: {{x -> 2}}
```

Notice that:

- Equations in Mathematica are written using the double-equal sign “==”.
- The output of Solve will always have an outer layer of curly braces { }.
- Inside the outer layer of curly braces, you’ll see substitution rules with the familiar “→” syntax.

You can check that x=2 is the correct solution to the above equation:

```
In[]: 2 x + 5 == 9 /. {x -> 2}
Out[]: True
```

This means that after the substitution  $x=2$ , it is True that the left-hand side of the equation equals the right-hand side.

It may be useful to start by counting the number of roots before calculating these roots, if any. The command `CountRoots[poly, x]` gives the number of real roots of the polynomial `poly` in `x`.

```
In[]: CountRoots[6 x^3 + 36 x^2 - 78 x - 252, x]
Out[]: 3
```

To count complex roots of a polynomial you have to specify a rectangular interval for the polynomial. For example,

```
In[]: CountRoots[x^2 + 5, {x, -3 I, 3 I}]
Out[]: 2
```

Here are a few more examples involving the `Solve` command.

**Example 7.2.** Equations can have more than one solution. We can see numerical approximations of the solutions using the `N` command.

```
In[]: Solve[x^2 - 3 x + 1 == 0, x]
Out[]: {{x -> 1/2(3 - sqrt(5))}, {x -> 1/2(3 + sqrt(5))}}
In[]: NSolve[x^2 - 3 x + 1 == 0, x]
Out[]: {{x -> 0.381966}, {x -> 2.61803}}
```

**Example 7.3.** Here, two of the solutions are complex, with `i` standing for the familiar  $\sqrt{-1}$ .

```
In[]: Solve[x^3 + x^2 == -3 x, x]
Out[]: {{x -> 0}, {x -> 1/2(-1 - i sqrt(11))}, {x -> 1/2(-1 + i sqrt(11))}}
```

**Example 7.4.** If the equation involves other variables, Mathematica will treat them as constants (or parameters, depending on your use).

```
In[]: Solve[y^2 - a*y == 2 a, y]
Out[]: {{y -> 1/2(a - sqrt(a) sqrt(8 + a))}, {y -> 1/2(a + sqrt(a) sqrt(8 + a))}}
```

**Note** The `Solve` command works very well for equations involving polynomials. However, it doesn't have much success with trigonometric, exponential, logarithmic, or hyperbolic functions.

**The Solve Command for a System of Equations** `Solve` can also be used to solve a system of equations. In general, it is used in this form:

```
Solve[{ eqn 1, eqn 2, ...}, { var 1, var 2, ...}]
```

**Example 7.5.** Solve the system  $\begin{cases} 3x + 8y = 5 \\ 5x + 2y = 7 \end{cases}$ .

```
In[]: Solve[{3 x + 8 y == 5, 5 x + 2 y == 7}, {x, y}]
Out[]: {{x -> 23/17, y -> 2/17}}
```

**Example 7.6.** Solve the system  $\begin{cases} 2xy + y^2 = -4 \\ x + y = 2 \end{cases}$ .

```
In[]: Solve[{2 x y + y^2 == -4, x + y == 2}, {x, y}]
Out[]: {{x -> 2 sqrt(2), y -> 2(1 - sqrt(2))}, {x -> -2 sqrt(2), y -> 2(1 + sqrt(2))}}
```

Two sets of solutions are obtained.

**Example 7.7.** Solve the system  $\begin{cases} x + y = 0 \\ x + y = 1 \end{cases}$

```
In[]: Solve[{x + y == 0, x + y == 1}, {x, y}]
Out[]: {}
```

There are no solutions to this system of equations.

**Example 7.8.** Solve the system  $\begin{cases} 2x - y + z = 2 \\ x + y + z = 3 \end{cases}$

```
In[]: Solve[{2 x - y + z == 2, x + y + z == 3}, {x, y, z}]
Out[]: {{y -> 1/2 + x/2, z -> 5/2 - 3x/2}}
```

The above command produced the warning message `Solve::svars: Equations may not give solutions for all solve variables.` >> It alerts you to the fact that you supplied only two equations, so one of the three variables (here, it's  $x$ ) will be treated as a constant (or parameter) in the solution.

**Equations and Numerical Solutions** There are some equations, however, for which it is mathematically impossible to find explicit formulas for the solutions. The Wolfram Language uses `Root` objects to represent the solutions in this case.

```
In[]: Solve[2 - 4 x + x^5 == 0, x]
Out[]: {{x -> -1.52...}, {x -> 0.508...}, {x -> 1.24...},
        {x -> -0.117... - 1.44... i}, {x -> -0.117... + 1.44... i}}
```

Even though you cannot get explicit formulas, you can still evaluate the solutions numerically.

```
In[]: Solve[2 - 4 x + x^5 == 0, x] // N
Out[]: {{x -> -1.51851}, {x -> 0.508499}, {x -> 1.2436},
        {x -> -0.116792 - 1.43845I}, {x -> -0.116792 + 1.43845I}}
```

Or, you can use `NSolve`. The `NSolve` command uses efficient numerical techniques to approximate roots of polynomials and a few other simple functions. It has the same syntax as `Solve`.

```
In[]: NSolve[2 - 4 x + x^5 == 0, x]
Out[]: {{x -> -1.51851}, {x -> 0.508499}, {x -> 1.2436},
        {x -> -0.116792 - 1.43845I}, {x -> -0.116792 + 1.43845I}}
```

`NSolve` doesn't work at all with exponential, logarithmic, trigonometric, or hyperbolic functions. You should solve these types of equations using `FindRoot`, and giving a starting value for  $x$ . `FindRoot` has the form:

```
FindRoot[equation, {variable, estimation of a sol.}]
```

The estimate of a solution is just that a value that you think is close to or somehow near the point where you expect to find a solution. `FindRoot` then iteratively searches for a solution starting from that estimate.

**Example 7.9.** Find a root of  $e^x + \sin x$  near  $x = 0$ .

```
In[]: FindRoot[Sin[x] + Exp[x], {x, 0}]
Out[]: {x -> -0.588533}
```

Note that `FindRoot[lhs == rhs, {x,  $x_0$ ,  $x_1$ }]` searches for a solution using  $x_0$  and  $x_1$  as the first two values of  $x$ , avoiding the use of derivatives. If you specify only one starting value of  $x$ , `FindRoot` searches for a solution using Newton methods. If you specify two starting values, `FindRoot` uses a variant of the secant method.

**Example 7.10.** Find the positive root of  $\cos x = x^2$  using both Newton and Secant methods.

```
In[]: FindRoot[Cos[x] == x^2, {x, 3}]
Out[]: {x -> 0.824132}
In[]: FindRoot[Cos[x] == x^2, {x, 3, 5}]
Out[]: {x -> 0.824132}
```

Also, `FindRoot[lhs == rhs, {x,  $x_{\text{start}}$ ,  $x_{\text{min}}$ ,  $x_{\text{max}}$ }]` searches for a solution, stopping the search if  $x$  ever gets outside the range  $x_{\text{min}}$  to  $x_{\text{max}}$ .

**Example 7.11.** The equation  $\tan x = 8 - 17x^2$  has two real roots in the interval  $[-1, 1]$ . Find these two roots.

```
In[]: FindRoot[Tan[x] == 8 - 17 x^2, {x, 0.5, -1, 1}]
Out[]: {x -> 0.652415}
In[]: FindRoot[Tan[x] == 8 - 17 x^2, {x, -0.75, -1, 1}]
Out[]: {x -> -0.722824}
```



To solve a system of two or more equations in the unknowns  $x$ ,  $y$ , etc., we use  $x = x_0$ ,  $y = y_0$ ,  $\dots$  as initial estimates of a solution.

**Example 7.12.** Solve the system 
$$\begin{cases} -x^3 + y^2 = 5 \\ 3 \cos x - x + y = 4 \end{cases}$$

The above equations have a simultaneous solution that's close to  $x = 1$  and  $y = 2$ . We can pinpoint it with:

```
In[]: FindRoot[{y^2-x^3==5, y-x+3Cos[x]==4}, {x,1}, {y,1}]
Out[]: {x -> 0.663687, y -> 2.30051}
```

**The Command Reduce** In addition to being able to solve purely algebraic equations, Mathematica can also solve some equations involving other functions like  $\sin x = a$ . It is important to realize that an equation such as  $\sin x = a$  actually has an infinite number of possible solutions, in this case differing by multiples of  $2\pi$ . However, Solve by default returns just one solution, but prints a message telling you that other solutions may exist. You can use Reduce to get more information.

```
Reduce[expr, vars]
reduces the statement expr by solving equations or
inequalities for vars and eliminating quantifiers
```

For example,

```
In[]: Reduce[a x + b == 0, x]
Out[]: (b == 0 && a == 0) || (a != 0 && x == -b/a)

In[]: Reduce[Sin[x] == 1, x]
Out[]: C[1] ∈ Integers && x == π/2 + 2πC[1]

In[]: Reduce[x^4 - 1 == 0, x]
Out[]: x == -1 || x == -I || x == I || x == 1
```

## 7.5 Inequalities

Just as the equation  $x^2 + 3x = 2$  asserts that  $x^2 + 3x$  is equal to 2, so also the inequality  $x^2 + 3x > 2$  asserts that  $x^2 + 3x$  is greater than 2. In Mathematica, Reduce works not only on equations, but also on inequalities. For example, the following pair of inequalities reduces to a single inequality.

```
In[]: Reduce[{0 < x < 2, 1 < x < 4}, x]
Out[]: 1 < x < 2
```

These inequalities can never simultaneously be satisfied.

```
In[]: Reduce[{x < 1, x > 3}, x]
Out[]: False
```

This inequality yields three distinct intervals.

```
In[]: Reduce[(x - 1) (x - 2) (x - 3) (x - 4) > 0, x]
Out[]: x < 1 || 2 < x < 3 || x > 4
```

The ends of the intervals are at roots and poles.

```
In[]: Reduce[1 < (x^2 + 3 x)/(x + 1) < 2, x]
Out[]: -1 - √2 < x < -2 || -1 + √2 < x < 1
```

The following inequality allows only finitely many intervals.

```
In[]: Reduce[{Sin[x] > 0, 0 <= x <= 2 Pi}, x]
Out[]: 0 < x < π
```

## 7.6 Exercises

1. Expand the function  $(1 + x)^2(1 - x)^3$ .

**Ans.**  $-x^5 + x^4 + 2x^3 - 2x^2 - x + 1$

2. Factorize the polynomials

(a)  $(1 + x)^3 + (1 - x)^3$

**Ans.**  $2(3x^2 + 1)$

(b)  $x^2 + 5$

**Ans.**  $(x - i\sqrt{5})(x + i\sqrt{5})$

3. Write  $\frac{1}{x-1} + \frac{x}{x^2+1}$  as single fraction.  
*Ans.*  $\frac{2x^2 - x + 1}{(x-1)(x^2+1)}$
4. Decompose  $\frac{x}{x^2+6x+5}$  into sum of partial fractions.  
*Ans.*  $\frac{5}{4(x+5)} - \frac{1}{4(x+1)}$
5. Show that:

(a)  $\frac{(1+\sqrt{5})^{10} - (1-\sqrt{5})^{10}}{1024\sqrt{5}} = 55$

(b)  $\tan\left(\frac{3\pi}{11}\right) + 4\sin\left(\frac{2\pi}{11}\right) = \sqrt{11}$

(c)  $\frac{e^{\tanh^{-1}x} - e^{-\tanh^{-1}x}}{e^{\tanh^{-1}x} + e^{-\tanh^{-1}x}} = x$
6. Find all the coefficients of  $(2+3x)^4$ . Hint: see CoefficientList.  
*Ans.* 16, 96, 216, 216, 81
7. What is the coefficient of  $xy^3$  when we expand  $(x+y)^4$  ? What is the coefficient of  $y^2$  ? Hint: see the command Coefficient.  
*Ans.* 4,  $6x^2$ ,  $6x^2$
8. Find the exact value of the number  $\sqrt{3+\sqrt{8}} - \sqrt{2}$ .  
*Ans.* 1
9. Compare TrigExpand and TrigReduce on the expression  $\sin^2 x + \tan^2 x$ .
10. Find the quotient and remainder of the polynomial division  $\frac{x^2+x+1}{2x+1}$ .  
**Hint:** see PolynomialQuotient and PolynomialRemainder.  
*Ans.*  $\frac{x}{2} + \frac{1}{4}, \frac{3}{4}$
11. Solve each of the following.

(a)  $x^2 - 3x + 1 = 0$

(b)  $x^5 - 3x + 1 = 0$

(c)  $x + y + z = 3$  ,  $2x^2 + 3y^2 - 6z^2 = -1$  and  $3z^2 = 9x^2 - 6y^3$

(d)  $\tan^3 x + e^{2x} = 10$  over the interval  $[1, 5]$   
*Ans.* 1.87452

(e)  $x^2 \geq 1$
12. 

(a) Solve the equation  $|x-1| = 3$  over real numbers.

(b) Find integer solution(s) for the equation  $(x^2-3x+1)^{x+1} = 1$ .  
*Ans.* -1, 0, 1, 3
13. Solve the equation  $\sin^2 x - 3\sin x + 2 = 0$  for  $\sin x$
14. **The Taxicab Number.** The famous number theorist G. H. Hardy (1887-1947) was visiting S. Ramanujan (1887-1920) when Ramanujan was ill in a hospital. Hardy said that the number of his taxicab is 1729, seemed to him to be a dull number. Ramanujan responded that, on the contrary, 1729 is the sum of two cubes in two ways. Prove Ramanujan claim. Use the command Solve then use the new function PowersRepresentations.  
*Ans.* 1, 12 or 9, 10
15. The built-in function FindInstance[expr, vars] finds an instance of vars that makes the statement expr be True. Use Mathematica's help to read more about this function, and then use it to solve exercise (14).
16. For what truth values of the statements a, b, c, and d will the following proposition is true:  $(\sim a \vee b) \wedge (c \wedge \sim d)$ .  
**Hint:** again, you may use FindInstance.  
*Ans.*

a	b	c	d
T	T	T	F
F	T	T	F
F	F	T	F

## 8 Single Variable Calculus

Two important machineries in calculus are differentiation and integration, and both use the concept of limit. We assume the reader is familiar with calculus as all the material in these notes.

### 8.1 Limits

Mathematica provides the command `Limit` for exploring the limit of a function. If  $f$  is a function of a single variable, Mathematica evaluates  $\lim_{x \rightarrow a} f(x)$  with the following syntax:

```
Limit[ function , variable -> a ]
```

For example,  $\lim_{x \rightarrow 0} \frac{\sin x}{x} = 1$  as shown below.

```
In[]: Limit[Sin[x]/x, x -> 0]
Out[]: 1
```

The limit of the function  $\frac{|x|}{x}$  when  $x \rightarrow 0$  does not exist.

```
In[]: Limit[RealAbs[x]/x, x -> 0]
Out[]: Indeterminate
```

Be careful! The function  $\frac{|x|}{x}$  has a different limiting value at  $x = 0$ , depending on whether you approach from above or below. We know that the right-hand limit  $\lim_{x \rightarrow 0^+} \frac{|x|}{x} = 1$  and the left-hand limit  $\lim_{x \rightarrow 0^-} \frac{|x|}{x} = -1$ .

Therefore  $\lim_{x \rightarrow 0} \frac{|x|}{x}$  does not exist. To calculate a left-hand  $\lim_{x \rightarrow a^-} f(x)$ , you have to specify the `Direction` option in the `Limit` command. To find  $\lim_{x \rightarrow 0^-} \frac{|x|}{x}$ , you use:

```
In[]: Limit[RealAbs[x]/x, x -> 0, Direction -> "FromBelow"]
Out[]: -1
```

Similarly, to find the right-hand limit  $\lim_{x \rightarrow 0^+} \frac{|x|}{x}$ , you can use:

```
In[]: Limit[RealAbs[x]/x, x -> 0, Direction -> "FromAbove"]
Out[]: 1
```

**Note:** To evaluate the limit from both real directions we use `Direction -> Reals` or `Direction -> "TwoSided"`.

The following examples show some sample limit computations. Mathematica can compute most limits, even those that involve infinite limits, or limits at infinity:

1.  $\lim_{x \rightarrow 0} \frac{e^x - x - 1}{x}$

```
In[]: Limit[(Exp[x] - x - 1)/x, x -> 0]
Out[]: 0
```

2.  $\lim_{x \rightarrow 3} \frac{1 - x}{(x - 3)^2}$

```
In[]: Limit[(1 - x)/(x - 3)^2, x -> 3]
Out[]: -∞
```

3.  $\lim_{x \rightarrow \infty} \frac{x}{\sqrt{x^2 + 1}}$

```
In[]: Limit[x/Sqrt[x^2 + 1], x -> Infinity]
Out[]: 1
```

4.  $\lim_{x \rightarrow -\infty} \frac{x}{\sqrt{x^2 + 1}}$

```
In[]: Limit[x/Sqrt[x^2 + 1], x -> -Infinity]
Out[]: -1
```

5.  $\lim_{x \rightarrow 0} \frac{3x - \sin(3x)}{4x - \tan(4x)}$

```
In[]: Limit[(3 x - Sin[3 x])/(4 x - Tan[4 x]), x -> 0]
Out[]: -27/128
```

6.  $\lim_{x \rightarrow \infty} \sin x$

```
In[]: Limit[Sin[x], x -> Infinity]
Out[]: Indeterminate
```

7.  $\lim_{x \rightarrow 1} \frac{\ln x}{1 - x}$

```
In[]: Limit[Log[x]/(1 - x), x -> 1]
Out[]: -1
```

## 8.2 Differentiation

If  $f$  is a function of a single variable, Mathematica will understand the symbol  $f'$  as the derivative of  $f$ , where the prime is entered using the single-quote character. For example,

```
In[]: f[x_] := x^4
In[]: f'[x]
Out[]: 4x^3
```

Higher-order derivatives follow with the usual notation:

```
In[]: {f''[x], f'''[a]}
Out[]: {12x^2, 24a}
```

Also, you can use the differential operator  $D$  for computing derivatives.

```
D[ function , variable]
```

For example,

```
In[]: f[x_] := x^4
In[]: D[f[x], x]
Out[]: 4x^3
```

To calculate a second derivative, you can use  $D$  either of these short-hand formats:

```
In[]: D[f[x], x, x]
Out[]: 12x^2
In[]: D[f[x], {x, 2}]
Out[]: 12x^2
```

Similarly, for the third derivative of  $f$  we can use either of the following:

```
In[]: D[f[x], x, x, x]
Out[]: 24x
In[]: D[f[x], {x, 3}]
Out[]: 24x
```

**Example 8.1.** If  $g(t) = t^2 + t^3 + \ln t$ , find  $g''(1)$ .

```
In[]: g[t_] := t^2+t^3+Log[t]
In[]: g''[1]
Out[]: 7
In[]: D[g[t], {t, 2}] /. {t -> 1}
Out[]: 7
```

**Example 8.2.** Find the equation of the tangent line for the function  $g(t)$ , given in the previous example, at  $t = 1$ . Remember that the equation of tangent is given by  $y = g(1) + g'(1)(t - 1)$ . So,

```
In[]: Simplify[g[1] + g'[1] (t - 1)]
Out[]: -4 + 6t
```

**Example 8.3.** Evaluate  $\frac{d}{dx} \left[ x \sin(3\pi e^x) \right] \Big|_{x=\ln 2}$ .

```
In[]: D[x Sin[3 Pi Exp[x]], x] /. x -> Log[2]
Out[]: 6πLog[2]
```

**Example 8.4.** Find all number(s)  $c$  that satisfy the conclusion of the Mean-Value Theorem for  $f(x) = x^2 - 3x$  on  $\left[0, \frac{7}{2}\right]$ . Note that  $f$  is continuous on  $\left[0, \frac{7}{2}\right]$  and differentiable on  $\left(0, \frac{7}{2}\right)$  since it is a polynomial. So,  $c \in \left(0, \frac{7}{2}\right)$  exists and satisfies  $f'(c) = \frac{f\left(\frac{7}{2}\right) - f(0)}{\frac{7}{2} - 0}$ .

```
In[]: f[x_] := x^2 - 3 x
In[]: Solve[{f'[c]==(f[7/2]-f[0])/(7/2-0) && 0<c<7/2}, c]
Out[]: {{c -> 7/4}}
```

### 8.3 Implicit Differentiation

In calculus, an implicit equation is an equation where the dependent variable is not explicitly expressed in terms of the independent variable. This means that an implicit equation is a statement that two or more variables are related to each other, but the relationship is not necessarily expressed as a function of one variable with respect to the others. For example, the circle centered at the origin with radius 5 represents an implicit curve whose equation is  $x^2 + y^2 = 25$ . It is implicit because  $y$  is not explicitly solved for in terms of  $x$ , and vice versa.

Implicit equations often require special techniques, such as implicit differentiation, to find derivatives or solve related calculus problems. In *Mathematica*, the command

```
ImplicitD[eqn,y,x]
gives the derivative of y, assuming that the variable y represents an implicit function defined by the equation eqn.
```

**Example 8.5.** Find  $y'$  if  $x + y = \sin^{-1} y$ .

```
In[]: ImplicitD[x + y == ArcSin[y], y, x]
Out[]: -\frac{\sqrt{1-y^2}}{\sqrt{1-y^2}-1}
```

To find the higher-order implicit derivative  $\frac{d^ny}{dx^n}$ , one may use `ImplicitD[eqn,y,{x,n}]`.

**Example 8.6.** Find  $y''$  if  $x^2 + y^2 = 25$ .

```
In[]: ImplicitD[x^2 + y^2 == 25, y, {x, 2}]
Out[]: \frac{-x^2 - y^2}{y^3}
```

### 8.4 Maximum and Minimum

In *Mathematica*, `Maximize[f,x]` try to find the absolute maximum value of  $f$  with respect to  $x$  on its natural domain or a given domain. While `Minimize[f,x]` try to find the absolute minimum value of  $f$  with respect to  $x$ .

```
In[]: Maximize[-2 x^2 - 3 x + 5, x]
Out[]: {{49/8, {x -> -3/4}}}
In[]: Maximize[{Sin[x], 0 <= x <= 2 Pi}, x]
Out[]: {1, {x -> Pi/2}}
In[]: Minimize[{Cos[x], -3 Pi/4 <= x <= 5 Pi/6}, x]
Out[]: {-\frac{\sqrt{3}}{2}, {x -> 5 Pi/6}}
```

Finding local maximum and minimum values of  $f$  is available in *Mathematica* using the commands `FindMinimum` and `FindMaximum` as follows:

```
In[]: FindMinimum[x^3-x, x]
Out[]: {-0.3849, {x -> 0.57735}}
In[]: FindMaximum[x^3-x, {x,-4}]
Out[]: {0.3849, {x -> -0.57735}}
```

## 8.5 Integration

$F(x)$  is an antiderivative of  $f(x)$  if  $F'(x) = f(x)$ . The symbol  $\int f(x)dx$  means “find all antiderivatives of  $f(x)$ .” Because all antiderivatives of a given function differ by a constant, we usually write  $\int f(x)dx = F(x) + C$  where  $C$  represents an arbitrary constant. You can use the `Integrate` command to compute  $\int f(x)dx$ . It has the form:

```
Integrate[f[x], x]
```

You specify a function or an expression to integrate, as well as the variable in which the integration is to take place. For example, to compute  $\int x^3 dx$ , use this syntax:

```
In[]: Integrate[x^3, x]
Out[]:  $\frac{x^4}{4}$ 
```

Mathematica can integrate almost every integral that can be done using standard integration methods (e.g., substitution, integration by parts, partial fractions). Here are some typical integrations:

1.  $\int x \ln x \, dx$

```
In[]: Integrate[x Log[x], x]
Out[]:  $-\frac{x^2}{4} + \frac{1}{2}x^2\text{Log}[x]$ 
```

2.  $\int \frac{1}{1-x^2} \, dx$

```
In[]: Integrate[1/(1 - x^2), x]
Out[]:  $-\frac{1}{2}\text{Log}[1-x] + \frac{1}{2}\text{Log}[1+x]$ 
```

3.  $\int \frac{e^{\sqrt{x}}}{\sqrt{x}} \, dx$

```
In[]: Integrate[Exp[Sqrt[x]]/Sqrt[x], x]
Out[]:  $2e^{\sqrt{x}}$ 
```

4.  $\int \sqrt{4-9x^2} \, dx$

```
In[]: Integrate[Sqrt[4 - 9 x^2], x]
Out[]:  $\frac{1}{2}x\sqrt{4-9x^2} + \frac{2}{3}\sin^{-1}\left(\frac{3x}{2}\right)$ 
```

5.  $\int \frac{x}{1+x^4} \, dx$

```
In[]: Integrate[x/(1 + x^4), x]
Out[]:  $\frac{\tan^{-1}(x^2)}{2}$ 
```

6.  $\int \frac{xe^x}{(1+x)^2} \, dx$

```
In[]: Integrate[x Exp[x]/(1 + x)^2, x]
Out[]:  $\frac{e^x}{1+x}$ 
```

7.  $\int \cos(\sin(y^2)) \, dy$

```
In[]: Integrate[Cos[Sin[y^2]], y]
Out[]:  $\int \cos(\sin(y^2)) \, dy$ 
```

As you see in the last of the examples above, the integrand has no closed-form anti-derivative. So, Mathematica will return your input unevaluated whenever it can't handle an integral. This can mean either that it's not possible to find an antiderivative in closed form or that Mathematica hasn't yet been programmed to do the integral.

**Definite Integral** A definite integral  $\int_a^b f(x)dx$  is computed in Mathematica with this form of the `Integrate` command:

```
Integrate[f[x], {x,a,b}]
```

Mathematica will try to find an antiderivative first, then evaluate it at the endpoints and subtract (according to the Fundamental Theorem of Calculus). Here are some examples:

1.  $\int_{-1}^2 x^2 dx$   

```
In[]: Integrate[x^2, {x, -1, 2}]
```

```
Out[]: 3
```
2.  $\int_0^\infty \frac{1}{4+y^2} dy$   

```
In[]: Integrate[1/(4 + y^2), {y, 0, Infinity}]
```

```
Out[]:  $\frac{\pi}{4}$ 
```
3.  $\int_0^1 \sqrt{\sin(t^2)} dt$   

```
In[]: Integrate[Sqrt[Sin[t^2]], {t, 0, 1}]
```

```
Out[]:  $\int_0^1 \sqrt{\sin(t^2)} dt$ 
```

In the last example, Mathematica can't evaluate an antiderivative at the endpoints. The `NIntegrate` command finds a numerical approximation for a definite integral  $\int_a^b f(x)dx$  with the following syntax:

```
NIntegrate[f[x], {x,a,b}]
```

Notice that its syntax is the same as the `Integrate` command. For example,

```
In[]: NIntegrate[x^2, {x, -1, 2}]
```

```
Out[]: 3
```

```
In[]: NIntegrate[Sqrt[Sin[t^2]], {t, 0, 1}]
```

```
Out[]: 0.486177
```

The `NIntegrate` command doesn't attempt to find a symbolic antiderivative, so it's quick and it works with almost all integrands, including  $\int_0^1 \sqrt{\sin(t^2)} dt$  for which `Integrate` failed (as you saw above).

## 8.6 Sequences

A **sequence** is a function whose domain is the set of positive integers. The terms of the sequence  $\{a_n\}$  are  $a_1, a_2, a_3, \dots$ . The term  $a_n$  is called the general term of the sequence. For example, the sequence  $\frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{5}{6}, \frac{6}{7}, \dots$  has the general term  $a_n = \frac{n}{n+1}$ . In Mathematica, we attempt to find a simple function that yields the sequence  $\{a_n\}$  when given successive integer arguments by using the command `FindSequenceFunction`.

```
In[]: FindSequenceFunction[{1/2, 2/3, 3/4, 4/5}, n]
```

```
Out[]:  $\frac{n}{n+1}$ 
```

A sequence  $\{a_n\}$  is said to converge to the limit  $\ell$  if  $\lim_{n \rightarrow \infty} a_n = \ell$ . A sequence that does not converge to some finite limit is said to diverge. The command `DiscreteLimit` gives the limit  $\lim_{n \rightarrow \infty} f_n$  for the sequence  $f_n$  as  $n$  tends to infinity over the integers.

**Example 8.7.** Determine whether the sequence converges or diverges.

1.  $\frac{\sqrt{9n^2 + 4n - 7}}{2n + 1}$   

```
In[]: DiscreteLimit[Sqrt[9n^2+4n-7]/(2n+1), n -> Infinity]
```

```
Out[]:  $\frac{3}{2}$ 
```
2.  $\frac{n^n}{n!}$   

```
In[]: DiscreteLimit[n^n/n!, n -> Infinity]
```

```
Out[]:  $\infty$ 
```

## 8.7 Series

You can use the Sum command to add up a finite number of terms of an indexed expression. To find  $\sum_{n=a}^b$  for an expression, you type:

```
Sum[ expression , {n,a,b}]
```

For example, the value of  $\sum_{n=1}^{20} n^2$  is

```
In[]: Sum[ n^2 , {n,1,20}]
Out[]: 2870
```

The Sum command has moderate success even with certain symbolic summations. For example,  $\sum_{n=0}^k r^n$  is a partial sum for a geometric series:

```
In[]: Sum[ r^n , {n,0,k}]
Out[]:  $\frac{-1 + r^{1+k}}{-1 + r}$ 
```

**Example 8.8.** Show that  $1 + 2 + 3 + \cdots + k = \frac{k(k+1)}{2}$ .

```
In[]: Sum[ n , {n,1,k}]
Out[]:  $\frac{1}{2}k(k+1)$ 
```

If you want an approximate value for a summation, use NSum instead. It has the same syntax as Sum. For example,

```
In[]: NSum[Sin[n], {n, 0, 500}]
Out[]: 1.4903
```

Both the Sum and NSum commands allow you to specify an infinite range, providing the ability to evaluate certain infinite series. Each is also pretty good at recognizing series that do not converge. In particular, Sum can symbolically reproduce almost all standard series computations found in Calculus books and tables. For example,

1.  $\sum_{n=0}^{\infty} (ar^n)$  (\* Geometric Series \*)  

```
In[]: Sum[a r^n, {n, 0, Infinity}]
Out[]:  $\frac{a}{1-r}$ 
```
2.  $\sum_{n=1}^{\infty} \frac{1}{n}$  (\* Divergent Series \*)  

```
In[]: Sum[1/n, {n, 1, Infinity}]
Out[]: Sum::div: Sum does not converge.
```
3.  $\sum_{n=1}^{\infty} \frac{(-1)^n}{n}$  (\* Alternating Series \*)  

```
In[]: Sum[(-1)^n/n, {n, 1, Infinity}]
Out[]: -Log[2]
```
4.  $\sum_{n=1}^{\infty} \frac{1}{n^2}$  (\* Convergent p-Series \*)  

```
In[]: Sum[1/n^2, {n, 1, Infinity}]
Out[]:  $\frac{\pi^2}{6}$ 
```
5.  $\sum_{n=0}^{\infty} \frac{2^n}{n!}$  (\* Taylor Series of  $e^x$  \*)  

```
In[]: Sum[2^n/n!, {n, 0, Infinity}]
Out[]: e^2
```
6.  $\sum_{n=0}^{\infty} \sin(e^{-n})$  (\* Unevaluated \*)  

```
In[]: Sum[Sin[Exp[-n]], {n, 0, Infinity}]
Out[]:  $\sum_{n=0}^{\infty} \sin(e^{-n})$ 
```



Notice that `Sum` returns its input unevaluated if it doesn't have a known simplification or result for it, or if it is known to not converge (see the last example above). If you do not receive an error message telling you that a series does not converge then you can investigate using `NSum`.

```
In[]: NSum[Sin[Exp[-n]], {n, 0, Infinity}]
Out[]: 1.41477
```

**Example 8.9.** Find the sum  $\frac{1}{1} + \frac{1}{1+2} + \frac{1}{1+2+3} + \cdots + \frac{1}{1+2+3+\cdots+99}$ .

```
In[]: Sum[1/Sum[j, {j, 1, i}], {i, 1, 99}]
Out[]: 99/50
```

The command `SumConvergence` gives conditions for the sum  $\sum_n f_n$  to be convergent. For example, the

$p$ -series  $\sum_n \frac{1}{n^p}$  converges for all real numbers  $p > 1$ .

```
In[]: Assuming[Element[p, Reals], SumConvergence[1/n^p, n]]
Out[]: p > 1
```

**Example 8.10.** Find the interval of convergence for the real power series  $\sum_n \frac{x^n}{n 3^n}$ .

```
In[]: SumConvergence[(x^n)/(n 3^n), n, Assumptions->Element[x, Reals]]
Out[]: -3 <= x < 3
```

## 8.8 Taylor Polynomials

Recall that if a function  $f$  satisfies certain reasonable conditions, then it can be approximated by a polynomial  $p_n(x)$  of degree  $n$  near a point  $x = a$  defined by:

$$p_n(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n$$

A composition of the `Normal` and `Series` commands as follows will more easily produce the Taylor polynomial of degree  $n$  for a function about a point  $x=a$ :

```
Normal[Series[function, {x, a, n}]]
```

For example,

```
In[]: Normal[Series[Exp[x], {x, 0, 6}]]
Out[]: 1 + x + x^2/2 + x^3/6 + x^4/24 + x^5/120 + x^6/720
```

The following command will give an approximate value of  $e^3$  using the Taylor polynomial of degree 6 for  $e^x$  about  $x = 0$ :

```
In[]: Normal[Series[Exp[x], {x, 0, 6}]] /. {x -> 3} // N
Out[]: 19.4125
```

The `Series` command by itself actually gives a Taylor polynomial together with a remainder term which shows its order using the notation  $O[x]$ .

```
In[]: Series[Exp[x], {x, 0, 6}]
Out[]: 1 + x + x^2/2 + x^3/6 + x^4/24 + x^5/120 + x^6/720 + O[x]^7
```

Applying `Normal` to this result removes the remainder term and give the Taylor polynomial.

## 8.9 Exercises

1. Evaluate each of the following limits:

- $\lim_{x \rightarrow 3} \frac{x^2 - 9}{x - 3}$ .
- $\lim_{x \rightarrow \infty} \frac{3x - \sin(3x)}{4x - \tan(4x)}$ .
- $\lim_{x \rightarrow \infty} \frac{\tan^{-1} x}{e^x}$ .

(d)  $\lim_{x \rightarrow 1} \frac{1-x}{|x^2+x-2|}.$

(e)  $\lim_{x \rightarrow \infty} \left(1 + \frac{a}{x}\right)^{bx}$

2. Find the domain and range of each of the following functions.

**Hint:** use the commands `FunctionDomain` and `FunctionRange`

(a)  $f(x) = \frac{1}{x-x^2}.$

**Ans.**  $x < 0 \vee 0 < x < 1 \vee x > 1, y < 0 \vee y \geq 4$

(b)  $\sqrt{|x|}.$

**Ans.**  $\mathbb{R}, y \geq 0$

(c)  $f(x) = 2 \ln x$  and  $g(x) = \ln(x^2).$

3. Let  $f(x) = \begin{cases} 2x^2 & x < 1 \\ -3x + 5 & x \geq 1 \end{cases}$ . Find:  $f'(2)$  and  $f'(1)$ .

**Ans.**  $-3, \text{does not exist}$

4. Let  $f(x) = x^3 + e^x + \sin x$ . Find:  $f'(x)$  and  $f''(0)$ .

5. Find  $\frac{d^{87}}{dx^{87}} [x \sin x]$  at  $x = \frac{\pi}{2}$ .

**Ans.**  $-87$

6. Given that  $f(x) = x^3 - 3x^2 + 3$ ;  $x \in [-2, 4]$ . Find for  $f$ :

- Critical numbers.
- Intervals of increasing and decreasing.
- Absolute maximum and absolute minimum values.
- Intervals of concavity.
- Inflection points.

7. Evaluate each of the following integrals:

(a)  $\int x^2 \sqrt{1+x^2} \, dx$

(b)  $\int \frac{x+1}{x^2-9} \, dx$

(c)  $\int \frac{e^{1/x}}{x^2} \, dx$

(d)  $\int x^2 \cos x \, dx$

(e)  $\int_1^4 \frac{x^2+1}{\sqrt{x}} \, dx$

(f)  $\int_0^\infty e^{-x^2} \, dx$

(g)  $\int_0^{\pi/2} \sqrt{\sin x} \, dx$

8. Find the values of  $A$  such that  $\lim_{x \rightarrow A} \left( \frac{1}{x-3} - \frac{6}{x^2-9} \right) = \lim_{x \rightarrow A} \frac{x^2-x-12}{36x-144}.$

**Ans.**  $-9$  or  $3$

9. Assume  $n \geq 1$ , find a formula for the  $n^{\text{th}}$  order derivative of each of the following functions.

- $f(x) = xe^x.$
- $g(x) = \sin x$
- $h(x) = x \cos x$

10. Show that  $(1+2+\dots+n)^2 = 1^3 + 2^3 + \dots + n^3.$

11. Evaluate the following:

(a)  $\sum_{n=0}^{\infty} (-1)^n$

(b)  $\sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$

12. Find Taylor polynomial of degree 10 for  $f(x) = \sqrt{x}$  about  $x = 1$  and use it to approximate the value of  $\sqrt{1.5}.$

13. Determine whether the series converges or diverges.

$$1 + \frac{1}{2} + \frac{1}{3} - \frac{1}{4} - \frac{1}{5} - \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} - \dots$$

*Ans.*  $\frac{1}{9} (2\pi\sqrt{3} + \log(8))$

14. Find  $\sum_{n=0}^{\infty} \frac{1}{F_{2^n}}$  where  $F_m$  is the  $m^{\text{th}}$  term of Fibonacci sequence.

*Ans.*  $\frac{1}{2} (7 - \sqrt{5})$

15. Find the interval of convergence of the power series  $\sum_n \frac{(x-3)^n}{\sqrt{n}}$ .

16. Find the general term of each of the following sequences.

(a)  $-\frac{1}{2}, \frac{1}{8}, -\frac{1}{24}, \frac{1}{64}, -\frac{1}{160}, \frac{1}{384}, \dots$

(b)  $2, 3, 5, 7, 11, 13, \dots$

(c)  $1, 0, -1, 0, 1, 0, -1, 0, \dots$

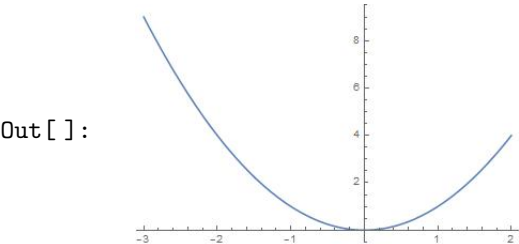
17. Determine whether the sequence  $-\frac{1}{5}, \frac{1}{25}, -\frac{1}{125}, \frac{1}{625}, \dots$  converges or diverges.

## 9 Graphics in Mathematica

### 9.1 Making Graphs

There are many ways that you can plot a two-dimensional picture in Mathematica. In this chapter, we will concentrate on drawing the graph of a function. You draw the graph of a function  $y = f(x)$  with the `Plot` command. To see the graph of  $f(x) = x^2$  over the interval  $[-3, 2]$ , type

```
In[]: Plot[x^2, {x, -3, 2}]
```

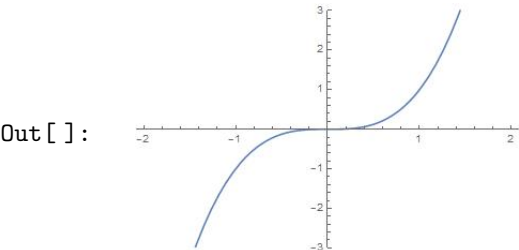


In general, to plot a function of  $x$  over an interval  $x \in [a, b]$ , you type:

```
Plot[f(x), {x, a, b}]
```

When you use the `Plot` command, you usually don not need to specify the vertical range of the graph. Mathematica automatically adjusts the vertical range to show you the full range of (vertical) values covered by the plot. If you wish, you can explicitly control the horizontal and vertical range by adding a `PlotRange` specification in the `Plot` command. For example:

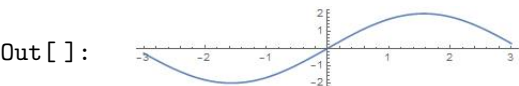
```
In[]: Plot[x^3, {x, -2, 2}, PlotRange -> {-3, 3}]
```



Here you see only the portion of the graph with  $y$ -values between  $-3$  and  $3$ . Thus, using `Plot` with the `PlotRange` option is similar to setting up a view window on a graphing calculator. You can specify a range of  $x$ -values as well, by using the option `PlotRange-> {{Xmin, Xmax} , {Ymin, Ymax}}`.

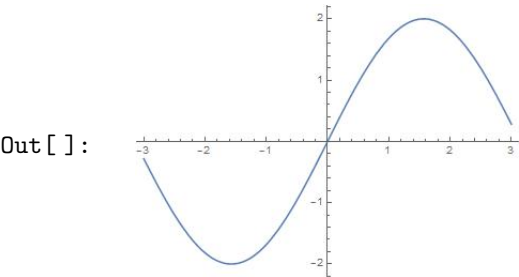
**Proportion and Aspect Ratio** When you are working in a Mathematica notebook, you can resize a picture by first clicking on it and then dragging along one of its comers or edges with the mouse. By default, no matter how big or small a picture you create, the aspect ratio of the picture (the ratio of its height to width) will remain the same. The default aspect ratio is very close to the relative proportions of a standard credit card. You can change the aspect ratio for a `Plot` command by directly specifying a value for its `AspectRatio`. For example, the height of the following picture is about one-fifth its width:

```
In[]: Plot[2 Sin[x], {x, -3, 3}, AspectRatio -> 1/5]
```

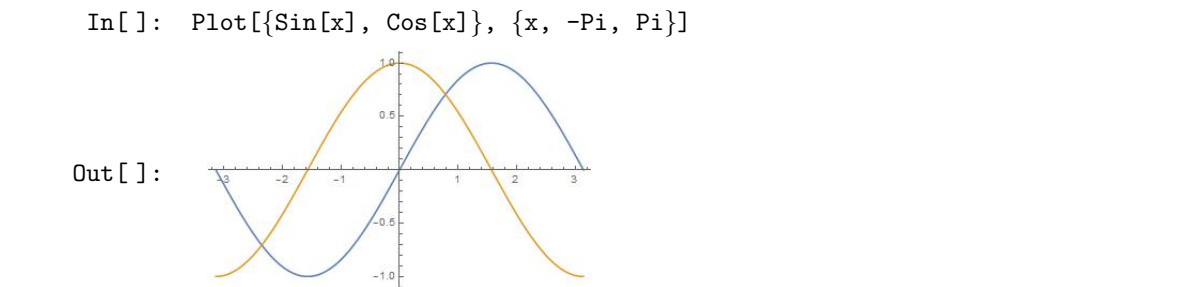


To make the units on the  $x$ - and  $y$ -axes have the same length, we set `AspectRatio` to `Automatic`.

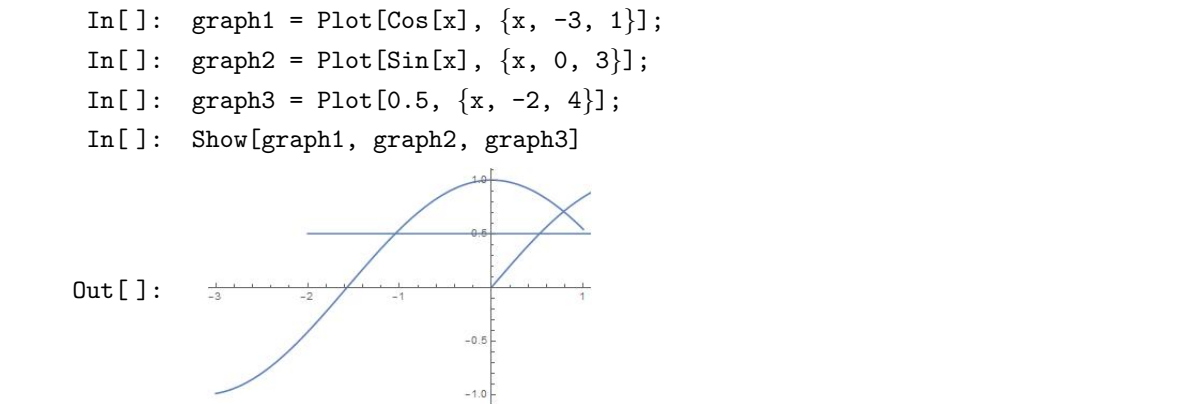
```
In[]: Plot[2 Sin[x], {x, -3, 3}, AspectRatio -> Automatic]
```



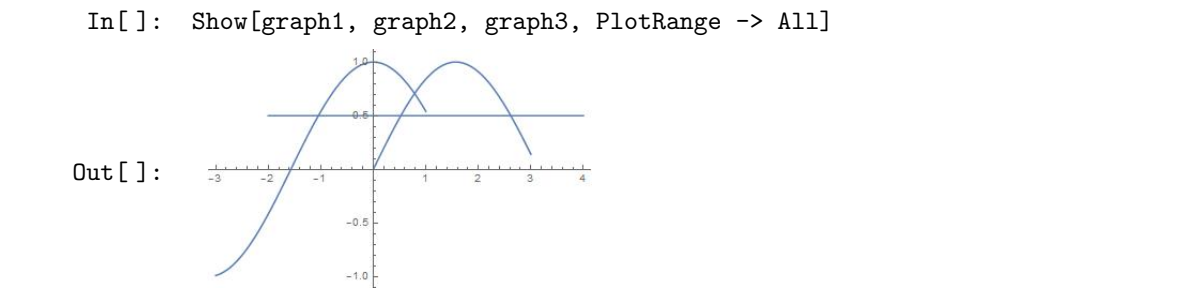
**Plotting Multiple Expressions** You can plot several functions in the same picture by listing all the functions separated by commas, enclosing them in curly braces. For example:



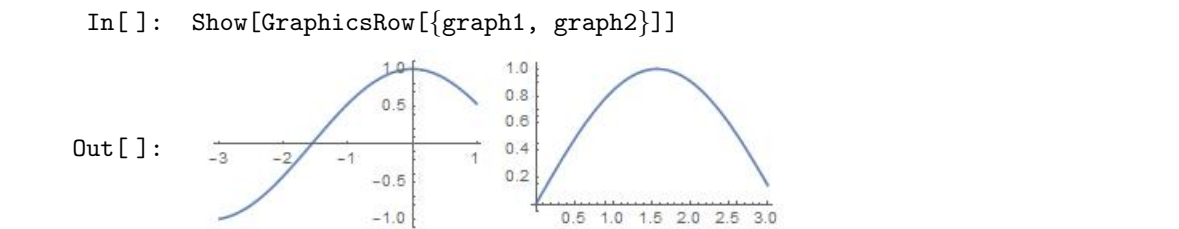
Several graphics produced independently can be combined into a single picture by using the `Show` command. First, name each of the graphics as they are produced. Then, use the `Show` command to combine them into a single output.



Notice that the combined graphic above is shown only over the range  $[-3, 1]$ , which is the range of the first graphic listed in the `Show` command. This is a default behavior of `Show`, it arranges its output according to the options attached to the first graphic. We can change this behavior by specifying `PlotRange -> All` in the `Show` command. The result is:

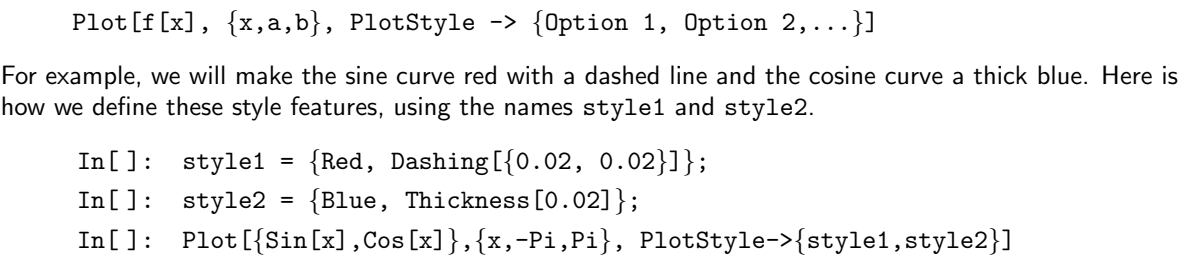


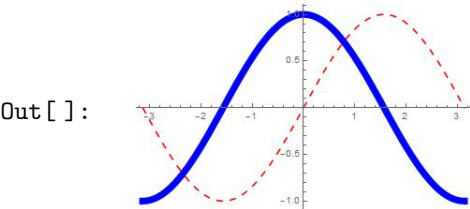
We can use the `GraphicsRow` command to ask Mathematica to show several plots side by side:



The `GraphicsRow` command has several options that can be specified. For example, one (`Alignment`) allows for adjusting the alignment of the graphics, while another (`Dividers`) specifies whether lines should be drawn between and/or above and below the graphics. Mathematica also has a `GraphicsGrid` command that allows you to arrange graphics in a two-dimensional display.

**The `PlotStyle` Option** You can add more properties to a graph by adjusting the color, thickness, opacity level, and dashing pattern of the plot. This can be done by specifying one or multiple `PlotStyle` options for the `Plot` command. It has the syntax:





`Thickness[0.02]` means that lines will be drawn to be about 2% of the width of the graphic, while `Dashing[{0.02,0 .04}]` means that the curve will be drawn solid for about 2% of the width of the graphic, then 2% will be omitted, and so on.

9.2   Plotting Curves

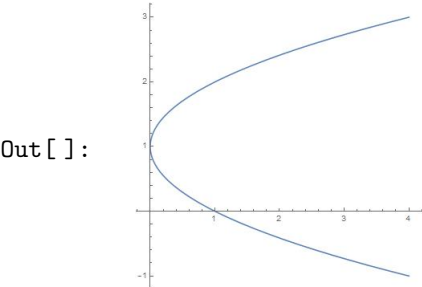
In the previous section you saw how to use `Plot` to draw curves that are graphs of functions. But not all curves are the graphs of functions.

**Parametric Plots** A two-dimensional (2D) parametric curve is usually defined by an ordered pair of coordinate functions  $(x(t),y(t))$ , with the parameter  $t$  allowed to vary over some set. You use the `ParametricPlot` command to draw such a curve over a fixed interval  $[a,b]$ . The command has the form:

```
ParametricPlot[{x[t],y[t]}, {t,a,b}]
```

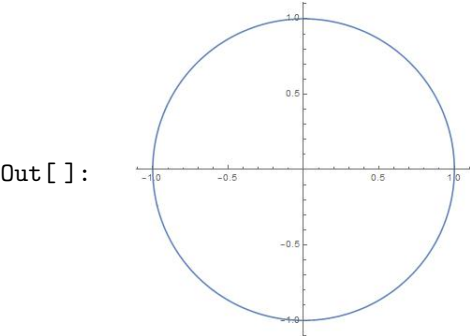
For example, to see the curve  $(t^2,t + 1)$  for  $t \in [-2,2]$ , we use:

```
In [ ]: ParametricPlot[{t^2, t + 1}, {t, -2, 2}]
```



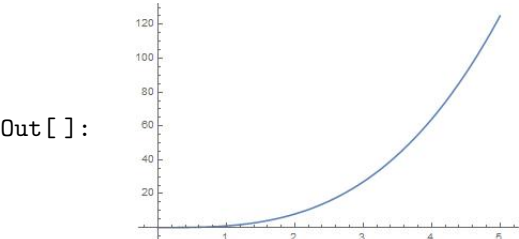
As with the `Plot` command, Mathematica will always, by default, show you a large enough picture to see all values on the curve. If necessary, you can specify a given `PlotRange` for `ParametricPlot`. Also, unlike the `Plot` command, `ParametricPlot` uses the option `AspectRatio -> Automatic` by default. Thus, curves which are circles will truly appear as circles:

```
In [ ]: ParametricPlot[{Cos[t], Sin[t]}, {t, 0, 2 Pi}]
```



However, setting `AspectRatio -> Automatic` is not always the right thing to do, as you will see in drawing the curve  $(t,t^3)$  where  $t \in [0,5]$  (try it by your self). You can draw a better picture with the following command

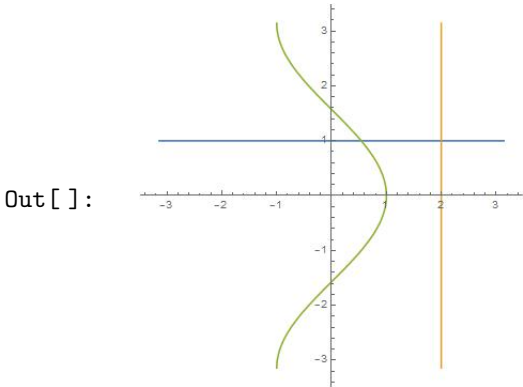
```
In [ ]: ParametricPlot[{t,t^3}, {t,0,5}, AspectRatio->1/GoldenRatio]
```



We suggest you redraw the graphic either by setting `AspectRatio->1` which gives a square graphic, or by setting `AspectRatio->1/GoldenRatio` which gives the standard proportion graphic you see with most other commands.

**Multiple Curves** The `ParametricPlot` command lets you plot several curves together, just like the `Plot` command. To do this, you include the formulas for the curves in a list. For example, to draw the horizontal line  $(t, 1)$ , the vertical line  $(2, t)$ , and the curve  $(\cos t, t)$  for  $t \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ , we use the command

```
In[]: ParametricPlot[{t,1}, {2,t}, {Cos[t],t}, {t,-Pi,Pi}]
```

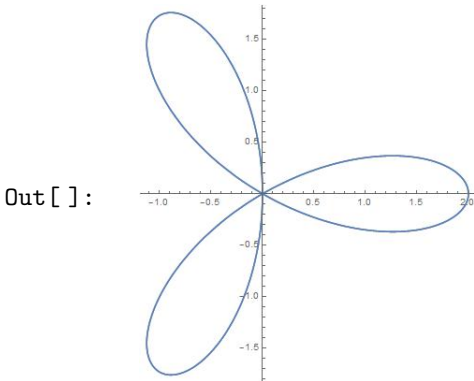


**Plotting in Polar Coordinates** If a curve is given in polar coordinates  $(r, \theta)$ , where  $r$  is the radius,  $\theta$  is the angle in radians, and  $r = f(\theta)$  for  $\theta \in [\theta_1, \theta_2]$ , then we can draw it with the `PolarPlot` command using the syntax:

```
PolarPlot[f[theta], {theta, theta1, theta2}]
```

Notice that this form is very similar to that of the `Plot` command you already know. For example, to plot the three-leaf rose  $r = 2 \cos(3\theta)$ , use:

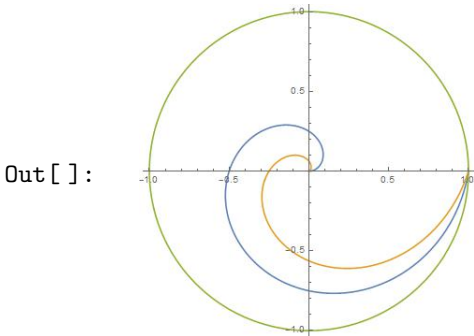
```
In[]: PolarPlot[2 Cos[3 theta], {theta, 0, 2Pi}]
```



The default setting for `PolarPlot` is `AspectRatio -> Automatic`. As we discussed in the case of `ParametricPlot`, you may want to directly specify the value of the option `AspectRatio` for a `PolarPlot` command that does not display well.

Just as with the `Plot` and `ParametricPlot` commands, you can plot more than one polar curve at the same time by entering all of them using a list. For example, you can plot the spirals  $r = \frac{\theta}{2\pi}$ ,  $r = \left(\frac{\theta}{2\pi}\right)^2$ , and the circle  $r = 1$  all in the same picture with:

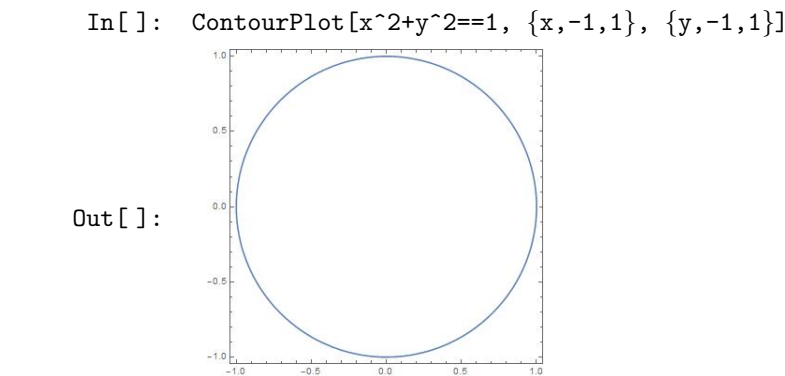
```
In[]: PolarPlot[{theta/(2Pi), (theta/(2Pi))^2, 1}, {theta, 0, 2Pi}]
```



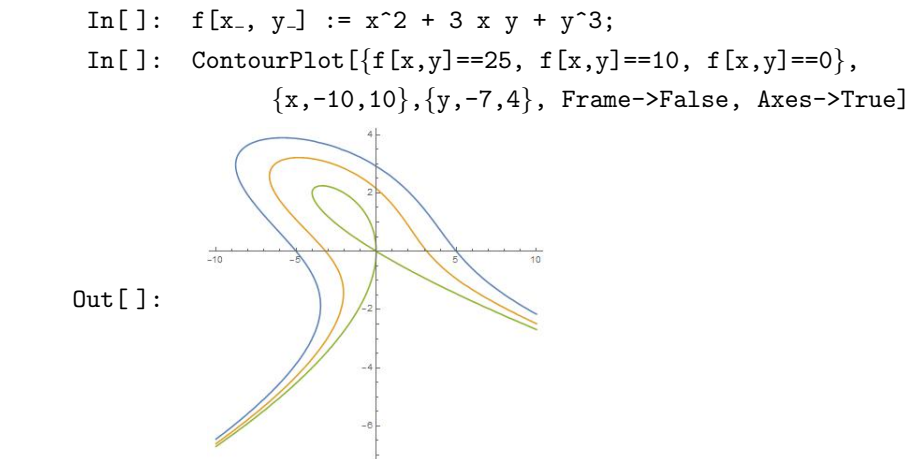
**Plotting Graphs of Implicit Equations** When a curve is defined by an equation in two variables, you can draw the curve with the `ContourPlot` command whose full functionality we will discuss more carefully later. For now, we will use `ContourPlot` to draw just that portion of the curve defined by an equation in, say, the two variables  $x$  and  $y$ , which lies within a rectangle  $x \in [a, b]$  and  $y \in [c, d]$  with the syntax:

```
ContourPlot[ equation of x and y, {x,a,b}, {y,c,d}]
```

For example, to see the unit circle  $x^2 + y^2 = 1$  which we know lies within the rectangle  $[-1, 1] \times [-1, 1]$ , we use:

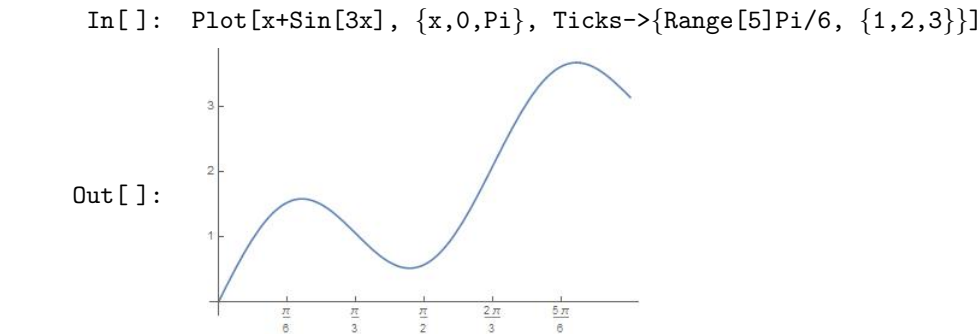


You can plot several curves defined by equations in a single command. The syntax is exactly the same as it is for the Plot, ParametricPlot, and PolarPlot commands. For example, we will plot the portions of the curves  $x^2 + 3xy + y^3 = 25$ ,  $x^2 + 3xy + y^3 = 10$ , and  $x^2 + 3xy + y^3 = 0$  which lie within the rectangle  $-10 \leq x \leq 10$  and  $-7 \leq y \leq 4$ . We will remove the frame, and draw the axes within the picture itself.

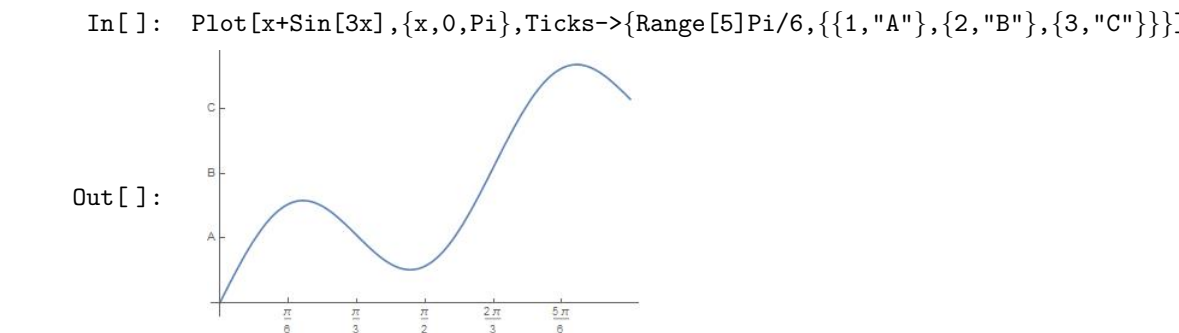


An interesting feature of the default output from ContourPlot is that if you position the mouse over any of the three curves shown in the graphic above, you will see the equation shown as a tool tip on the mouse.

**Tick Marks** Each of the commands Plot, ParametricPlot, PolarPlot, and ContourPlot allows you to control the placement of tick marks on each of the axes using the Ticks option. This is done as we do in the following example. Suppose we draw the graph of  $x + \sin(3x)$ , and we wish to have tick marks on the x-axis at  $\frac{\pi}{6}, \frac{2\pi}{6}, \frac{3\pi}{6}, \frac{4\pi}{6}, \frac{5\pi}{6}$  and on the y-axis at 1, 2, 3.

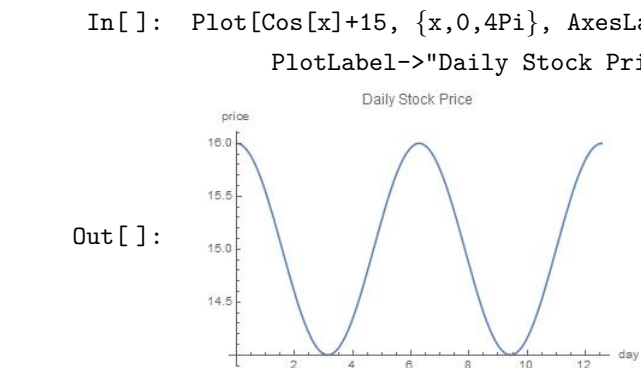


You can specify the labels to be shown at given tick marks.

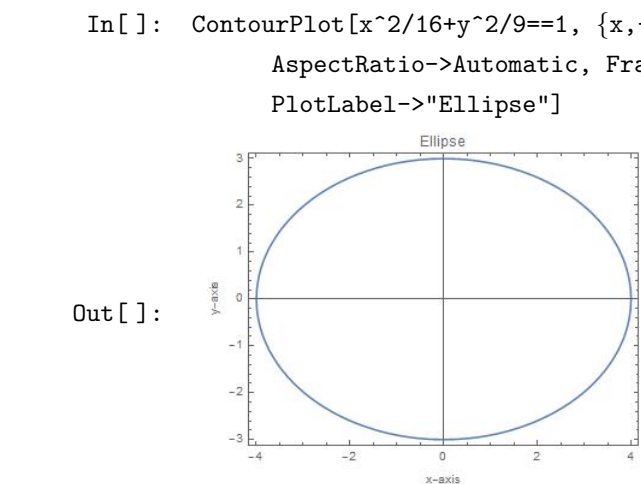




**Labels** The AxesLabel and PlotLabel options allow you to specify labels for one or both of the axes, or for the output graphic as a whole, respectively.

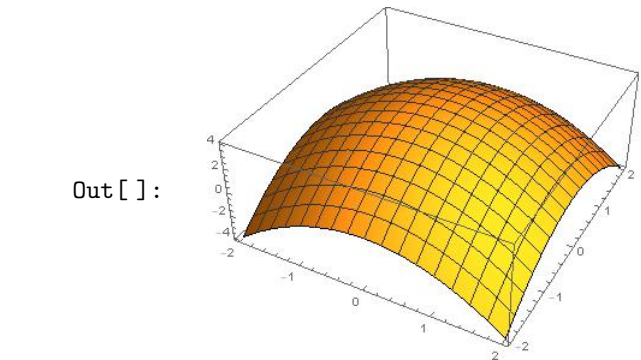
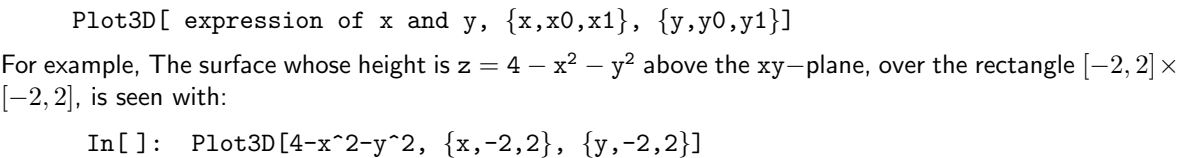


If your picture is framed, then you must use the FrameLabel option in place of AxesLabel.

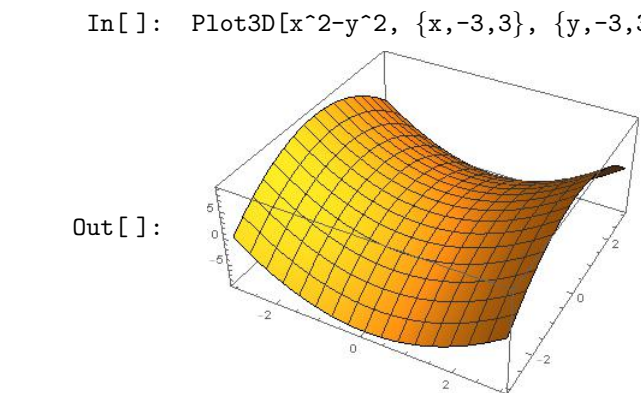


9.3 Making Graphs in Space

The easiest way to sketch a surface in three dimensions (3D) is to use the Plot3D command. You input an expression that gives the height of a surface above the  $xy$ -plane, in terms of the independent variables  $x$  and  $y$ . You must also specify intervals  $x \in [x_0, x_1]$  and  $y \in [y_0, y_1]$ . The Plot3D command then has the form:



The graph of the function  $f(x,y) = x^2 - y^2$  looks like a saddle if we take, for example,  $x,y \in [-3, 3]$  as shown below.



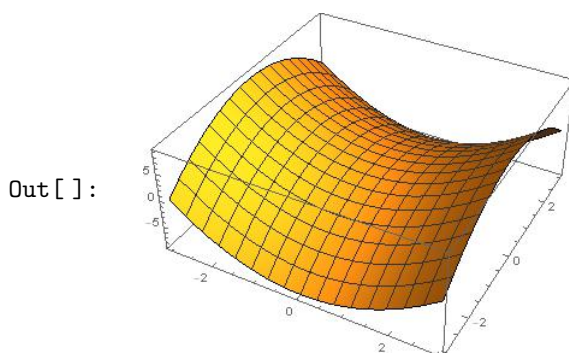
There are a number of options that you can use to add more character to the graph of a function  $z = f(x, y)$  with Plot3D. For example,

1. `AxesLabel` : Provides names to label each of the three axes.
2. `BoxRatios` : Scales the graphic so that it appears within a certain lengths of a box.
3. `PlotRange` : Specify the range of one or more of the variables  $x$ ,  $y$ , and  $z$ .
4. `Mesh→False` : Omit the mesh on the graph.
5. `PlotStyle→Green` : Color the graph green.

For example see how the sombrero looks like using some of Plot3D options. Note that the sombrero is

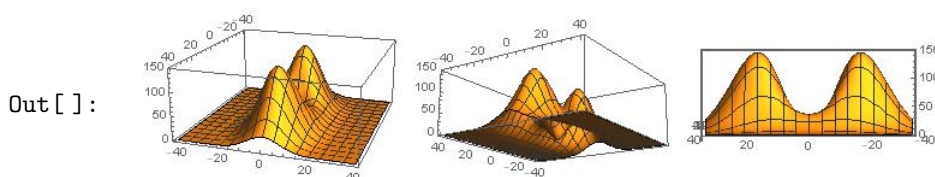
given by the function  $f(x, y) = \frac{\sin(\sqrt{x^2 + y^2})}{\sqrt{x^2 + y^2}}$ .

```
In[]: Plot3D[Sin[Sqrt[x^2+y^2]]/Sqrt[x^2+y^2], {x,-7,7},
           {y,-7,7}, PlotPoints->50, PlotRange->All, AxesLabel->{"x","y","z"},
           Mesh->False, BoxRatios->{1,1,1}]
```



Mathematica includes very useful Notebook feature for working with 3D graphics. As you move over a 3D graphic with the mouse, you will see that the cursor changes. If you now click and drag with the mouse, you will be able to rotate the graphic in real time. This lets you view the surface from many different angles. Every three-dimensional picture drawn in Mathematica will be shown from a specified `ViewPoint`. Specifying a `ViewPoint` is the same as describing how your eye is located with respect to the graphic being drawn. Let us show you how it is done. You input the `ViewPoint` option in the form `ViewPoint->{a,b,c}`, where the values  $a$ ,  $b$ , and  $c$  describe the position of your eye relative to the object that you are viewing. Positive values of  $a$ ,  $b$ , and  $c$  place you "in front of", "to the right of", and "above" the object, with respect to the direction of the positive  $x$ -,  $y$ - and  $z$ -coordinate axes, respectively. Negative values place you "behind", "to the left of", and "below" the object. A value of zero for any of these three places you "at the center". Let us think of the graph of  $f(x, y) = (x^2 + y^2) e^{-\frac{x^2}{400} - \frac{y^2}{100}}$  as being twin mountains. Let us take a "helicopter ride" around these mountains using three `ViewPoints`:  $\{3, 1, 1\}$ ,  $\{3, 2, -1\}$ , and  $\{0, 100, 0\}$ , respectively.

```
In[]: f[x_,y_] := (x^2+y^2) Exp[-x^2/400 - y^2/100];
In[]: vp1=Plot3D[f[x, y],{x,-40,40},{y,-40,40},ViewPoint->{3,1,1}];
In[]: vp2=Plot3D[f[x, y],{x,-40,40},{y,-40,40},ViewPoint->{3,2,-1}];
In[]: vp3=Plot3D[f[x, y],{x,-40,40},{y,-40,40},ViewPoint->{0,100,0}];
In[]: GraphicsRow[{vp1, vp2, vp3}, ImageSize -> Full]
```



## 9.4 Surfaces in Cylindrical and Spherical Coordinates

Surfaces are sometimes more easily described in terms of the cylindrical or spherical coordinate systems. You can draw such a surface easily with the Mathematica commands `RevolutionPlot3D` and `SphericalPlot3D` commands, respectively.

Points in the cylindrical coordinate system are described by quantities  $r$ ,  $\theta$ , and  $z$ , where

$r$  is the horizontal radial distance of the point from the  $z$ -axis;

$\theta$  is the horizontal angle measured from the  $x$ -axis; and

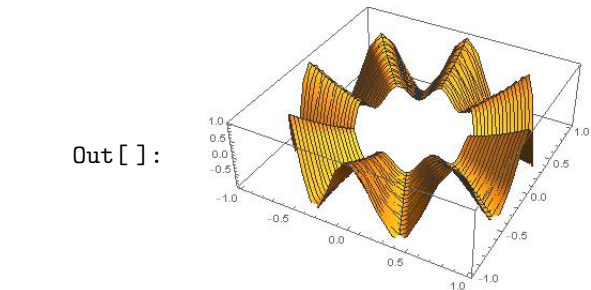
$z$  is the  $z$ –coordinate in standard rectangular coordinates.

To draw the surface  $z = f(r, \theta)$  for  $r \in [r_0, r_1]$  and  $\theta \in [\theta_0, \theta_1]$ , you enter:

```
RevolutionPlot3D[f[r,θ],{r,r1,r2},{θ,θ1,θ2}]
```

For example, to see the surface  $z = r^2 \cos(8\theta)$ , for  $r \in [0.5, 1]$  and  $\theta \in [0, 2\pi]$ :

```
In[]: RevolutionPlot3D[r^2 Cos[8 theta], {r,0.5,1}, {theta,0,2Pi}]
```



In Mathematica, points in the spherical coordinate system are described by quantities  $\rho$ ,  $\phi$ , and  $\theta$ , where

- $\rho$  is the radial distance in space of the point from the origin;
- $\theta$  is the vertical angle measured from the positive  $z$ –axis; and
- $\phi$  is the horizontal angle measured from the  $x$ –axis.

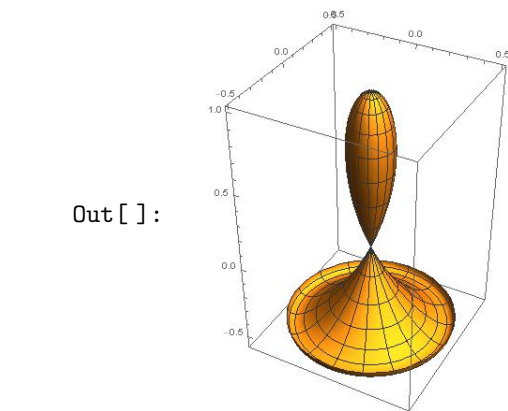
Note that Mathematica uses  $\phi$  and  $\theta$  for the horizontal and vertical angles, respectively, in spherical coordinates. This is exactly the opposite of the notation used in almost every Calculus book.

To draw the surface  $\rho = f(\theta, \phi)$ , where  $\theta \in [\theta_0, \theta_1]$  and  $\phi \in [\phi_0, \phi_1]$ , you will enter:

```
SphericalPlot3D[f[θ,φ],{θ,θ1,θ2},{φ,φ1,φ2}]
```

For example, to see the surface defined by  $\rho = \cos(\theta) \cos(4\theta)$ , for  $\theta \in [0, \frac{\pi}{4}]$ , and  $\phi \in [0, 2\pi]$ , use:

```
In[]: SphericalPlot3D[Cos[theta]Cos[4theta], {theta,0,Pi/4},{phi,0,2Pi}]
```



When you use the `SphericalPlot3D` command, you must enter the interval for the vertical angle `theta` first, then the interval for the horizontal angle `phi`. If you do not use this order, the picture will be incorrect.

**Plotting Multiple Surfaces** The `Plot3D` command allows you to draw more than one surface above a rectangle  $x \in [a, b]$  and  $y \in [c, d]$  in a single graphic. You use it in the form

```
Plot3D[{surface1, surface2, ...},{x,a,b},{y,c,d}]
```

and you still can attach options just as before. For example, the intersection of the paraboloid  $z = x^2 + y^2$  with the plane  $z = 2x + 5y$  is a circle. You can see this quite easily with

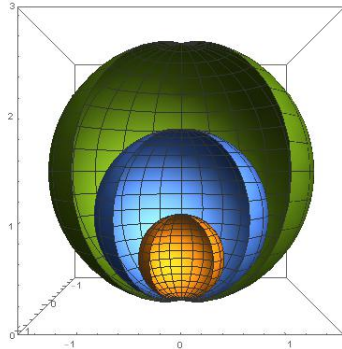
```
In[]: Plot3D[{x^2+y^2, 2x+5y}, {x,-6,6}, {y,-6,6}]
```

Out[]:

SphericalPlot3D similarly allows you to plot more than one surface, using the same syntax extension as the Plot3D command uses. For example, we can see a very nice "cut-away" image of the 3 spheres  $r = \cos(\theta)$ ,  $r = 2\cos(\theta)$ , and  $r = 3\cos(\theta)$  by restricting the range of  $\phi$ , to stay between  $\frac{\pi}{4}$  and  $\frac{7\pi}{4}$  and looking at them from in front of the x-axis:

```
In[]: SphericalPlot3D[{Cos[theta], 2Cos[theta], 3Cos[theta]},
  {theta,0,Pi/2}, {phi,Pi/4,7Pi/4}, ViewPoint->{2, 0, 0}]
```

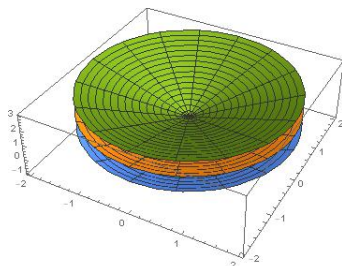
Out[]:



Finally, although RevolutionPlot3D also allows multiple equations in the same graphic, the syntax is different. You must use one extra layer of curly braces  $\{$  and  $\}$ . For example, in cylindrical coordinates, the equation  $z = r$ , for  $1 \leq r \leq 2$  and  $0 \leq \theta \leq 2\pi$ , describes a portion of a cone, also the equations  $z = r \pm 1$  are just vertical translations of the cone. Put the three together and you have got three bowls stacked up.

```
In[]: RevolutionPlot3D[{ {r}, {r-1}, {r+1}}, {r,0,2}, {theta,0,2Pi}]
```

Out[]:



## 9.5 Changing Coordinate Systems

Changing coordinate systems can involve two very different operations. One is recomputing coordinate values that correspond to the same point. The other is re-expressing a field in terms of new variables. The Wolfram Language provides functions to perform both these operations.

**Converting Points** Two coordinate systems are related by a mapping that takes coordinate values in the old system and returns coordinate values in the new system. The function `CoordinateTransformData` returns information about mappings between the coordinate systems. For example, the following converts the point  $(r, \theta)$  in polar coordinates to the corresponding  $(x, y)$  Cartesian coordinates.

```
In[]: CoordinateTransformData["Polar"->"Cartesian", "Mapping", {r, theta}]
```

Out[]:  $\{r \cos(\theta), r \sin(\theta)\}$

Conversely, we write

```
In[]: CoordinateTransformData["Cartesian"->"Polar", "Mapping", {x, y}]
```

Out[]:  $\left\{ \sqrt{x^2 + y^2}, \tan^{-1} \left( \frac{y}{x} \right) \right\}$

To convert a point  $(\rho, \theta, \phi)$  in spherical coordinates to the corresponding  $(x, y, z)$  Cartesian coordinates, and vice versa, we use

```
In[]: CoordinateTransformData["Spherical"->"Cartesian", "Mapping", {rho, theta, phi}]
```

Out[]:  $\{\rho \sin(\theta) \cos(\phi), \rho \sin(\theta) \sin(\phi), \rho \cos(\theta)\}$

```
In[]: CoordinateTransformData["Cartesian"->"Spherical", "Mapping", {x, y, z}]
```

Out[]:  $\left\{ \sqrt{x^2 + y^2 + z^2}, \tan^{-1} \left( \frac{\sqrt{x^2 + y^2}}{z} \right), \tan^{-1} \left( \frac{y}{x} \right) \right\}$

Also, we convert a point  $(r, \theta, z)$  in cylindrical coordinates to the corresponding  $(x, y, z)$  Cartesian coordinates, and vice versa by writing

```

In[]: CoordinateTransformData["Cylindrical"->"Cartesian","Mapping", {r,θ,z}]
Out[]: {r cos(θ), r sin(θ), z}
In[]: CoordinateTransformData["Cartesian"->"Cylindrical","Mapping", {x,y,z}]
Out[]: {√(x²+y²), tan⁻¹(y/x), z}

```

**Example 9.1.** Convert the point  $(1, 1, 0)$  in Cartesian coordinates to the corresponding values in spherical coordinates.

```

In[]: CoordinateTransformData["Cartesian"->"Spherical","Mapping",{1,1,0}]
Out[]: {√2, π/2, π/4}

```

**Example 9.2.** Convert the point  $(1, \frac{\pi}{3}, \frac{\pi}{4})$  in spherical coordinates to the corresponding cylindrical coordinates.

```

In[]: CoordinateTransform["Spherical"->"Cylindrical",{1,Pi/3,Pi/4}]
Out[]: {√3/2, π/4, 1/2}

```

**Example 9.3.** Convert the points  $(1, 1)$ ,  $(1, -1)$ , and  $(0, -2)$  from Cartesian coordinates to the corresponding polar coordinates.

```

In[]: CoordinateTransform["Cartesian"->"Polar",{1,1},{1,-1},{0,-2}]
Out[]: {{√2, π/4}, {√2, -π/4}, {2, -π/2}}

```

**Transforming Fields** When transforming fields between two coordinate systems, a field given in terms of variables in the old system is re-expressed in terms of variables in the new system. In addition to the mapping between the systems, several additional steps are needed: solving for the old variables in terms of the new, the substituting in these expressions. All of these steps are performed by the command `TransformedField`. For example, this converts the scalar field  $x^2 + y^2 + z^2$  from Cartesian  $(x, y, z)$  to cylindrical  $(r, \theta, \xi)$  coordinates.

```

In[]: TransformedField["Cartesian"->"Cylindrical", x^2+y^2+z^2, {x,y,z}->
      {r,θ,ξ}] // Simplify
Out[]: ξ² + r²

```

The following converts the hyperbola  $x^2 - y^2$  from Cartesian  $(x, y)$  to polar  $(r, \theta)$  coordinates.

```

In[]: TransformedField["Cartesian"->"Polar",x^2-y^2, {x,y}->{r,θ}]/Simplify
Out[]: r² cos(2θ)

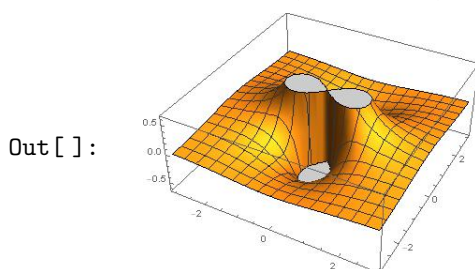
```

**Example 9.4.** In some cases, a surface given in rectangular coordinates will look better if you draw it using cylindrical or spherical coordinates. For example, the surface defined by the equation  $z = \frac{x^2 - y^2}{(x^2 + y^2)^2}$  can be plotted with:

```

In[]: Plot3D[(x^2-y^2)/(x^2+y^2)^2,{x,-3,3},{y,-3,3}]

```



The picture is choppy especially near the origin. However, if we use cylindrical coordinates, the surface becomes

```

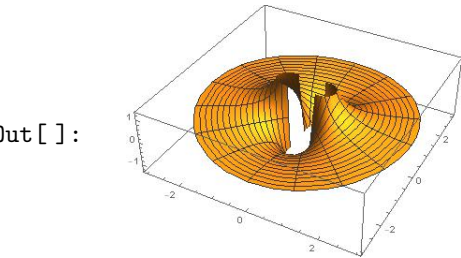
In[]: f=TransformedField["Cartesian"->"Cylindrical",
      (x^2-y^2)/(x^2+y^2)^2, {x,y,z}->{r,θ,ξ}] // Simplify
Out[]: cos(2θ)/r²

```

Now, we will do the plot.



```
In[ ]: RevolutionPlot3D[f,{r,0.5,3},{θ,0,2Pi}]
```



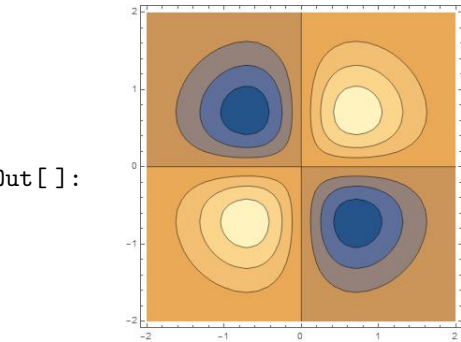
9.6    Level Curves and Level Surfaces

In Mathematica, the level curves (contours) of a function  $f(x,y)$  are plotted with the `ContourPlot` command. To see the level curves inside the rectangle  $x \in [a,b]$  and  $y \in [c,d]$ , you use the command with this syntax:

```
ContourPlot[f[x,y],{x,a,b},{y,c,d}]
```

For example, here are some level curves of  $f(x,y) = xye^{-x^2-y^2}$  near the origin:

```
In[ ]: ContourPlot[x y Exp[-x^2-y^2], {x,-2,2}, {y,-2,2}]
```



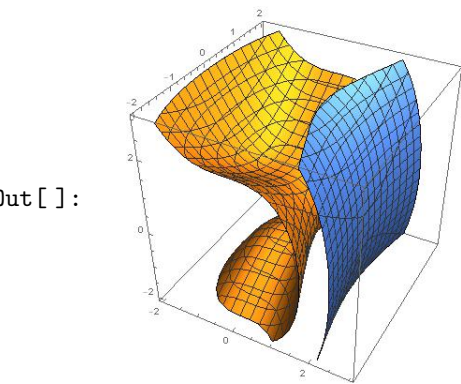
By default, Mathematica automatically sketches an appropriate number of level curves and colors the areas between these curves. Lighter shades represent higher levels, while darker shades represent lower levels. By moving the mouse over the graphic above, Mathematica will show you (using a tooltip at the cursor) exactly what contour levels have been drawn.

**Level Surfaces in Space** If  $f(x,y,z)$  is a function of three variables defined over a region  $x \in [x_0,x_1]$ ,  $y \in [y_0,y_1]$ , and  $z \in [z_0,z_1]$  then the level surface of  $f$  at level  $c$  can be seen with the `ContourPlot3D` command.

```
ContourPlot3D[function, {x,x0,x1}, {y,y0,y1}, {z,z0,z1}, Contours->{c}]
```

You can specify more than one level surface to be shown in the same graphic by writing `Contours->{the levels}`, where the levels are separated by commas. For example, We can see the level surfaces of  $f(x,y,z) = x^3 - y^2 + z^2$  at the levels of 1 and 10 with the following.

```
In[ ]: ContourPlot3D[x^3-y^2+z^2,{x,-2,3},{y,-2,2},{z,-2,3}, Contours->{1,10}]
```



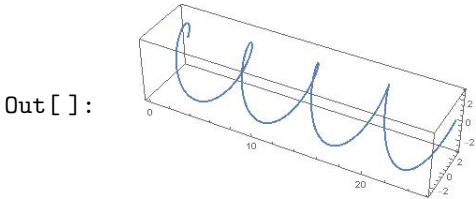
9.7    Parametric Curves and Surfaces in Space

You use the `ParametricPlot3D` command to draw a space curve. To see the curve given parametrically as  $(x(t),y(t),z(t))$ , for  $t \in [a,b]$ , type:

```
ParametricPlot3D[{x(t),y(t),z(t)},{t,a,b}]
```

This format is similar to the ParametricPlot command used for plane curves. Here, however, the curve is defined with three parametric functions rather than two. For example, the helix, given parametrically by  $(t, 3 \cos t, 3 \sin t)$ , for  $t \in [0, 8\pi]$ , is drawn with:

```
In[ ]: ParametricPlot3D[{t, 3Cos[t], 3Sin[t]}, {t, 0, 8Pi}]
```

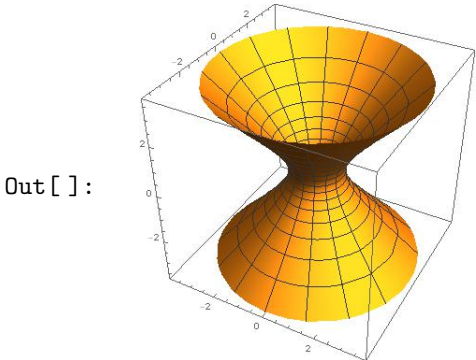


ParametricPlot3D can also be used to draw a surface in space. If a surface is defined parametrically by  $(x(u, v), y(u, v), z(u, v))$  for  $u \in [a, b]$  and  $v \in [c, d]$ , you enter:

```
ParametricPlot3D[{x(u, v), y(u, v), z(u, v)}, {u, a, b}, {v, c, d}]
```

For example, to see a portion of the one-sheeted hyperboloid given parametrically by  $(\cos u \cosh v, \sin u \cosh v, \sinh v)$ , for  $u \in [0, 2\pi]$  and  $v \in [-2, 2]$ , write:

```
In[ ]: ParametricPlot3D[{Cos[u] Cosh[v], Sin[u] Cosh[v], Sinh[v]}, {u, 0, 2Pi}, {v, -2, 2}]
```



9.8 Visualizing Data

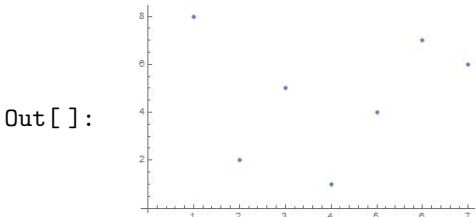
In Mathematica, curly braces are used to represent lists, regardless of the type of elements in the list. Lists can be created several different ways as we have seen before in chapter 4.

It is common to store lists in variables; this allows the lists to be easily referenced in subsequent calculations. For example, by assigning a list to the symbol data, this variable can be used in other calculations or commands where the list is needed.

```
In[ ]: data = {8, 2, 5, 1, 4, 7, 6}
Out[ ]: {8, 2, 5, 1, 4, 7, 6}
```

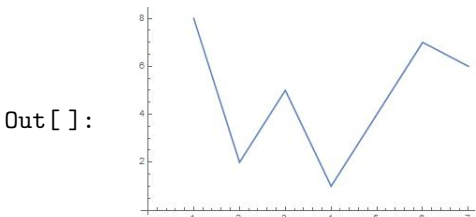
Just as the Wolfram Language has many commands available to visualize all types of mathematical functions and surfaces, so too does it have many commands available to visualize lists and datasets. One of the most common commands to visualize data is ListPlot, which displays the data as individual points.

```
In[ ]: ListPlot[data]
```

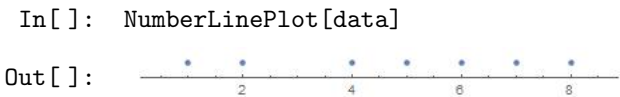


When values jump around, it is usually harder to understand if you do not join them up. ListLinePlot plots a list, joining up values.

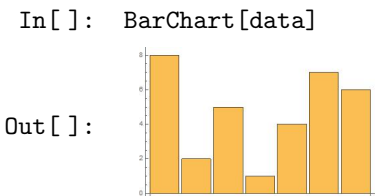
```
In[ ]: ListLinePlot[data]
```



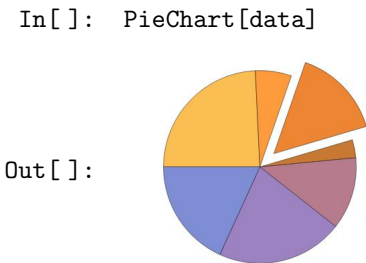
If you just want to know which numbers appear, you can plot them on a number line.



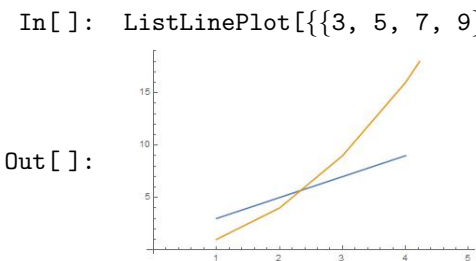
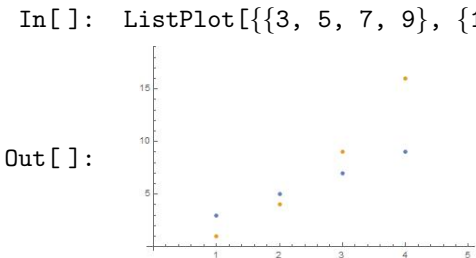
Making a bar chart can be useful too:



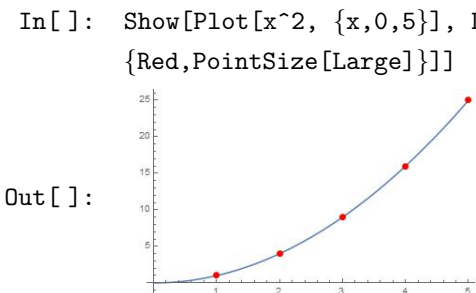
So long as the list is not too long, a pie chart can be useful:



**Multiple Datasets** While the examples thus far have concentrated on visualizing a single dataset, the visualization commands can also be used to visualize multiple datasets. Multiple datasets can be constructed by placing several different lists into a single "parent" list that encompasses them all. This larger list can be passed to a command like `ListPlot` to visualize each of the sublists as a separate dataset. The following example shows the use of `ListPlot` to plot two datasets, which are automatically given different colors to easily tell them apart.



The following example depicts a use of the `Show` command to combine a data visualization from `ListPlot` and a function visualization from `Plot`.

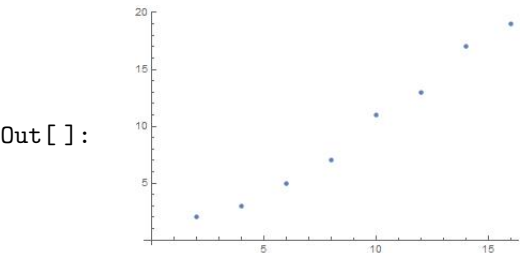


Note that we add an option to `ListPlot` called `PlotStyle`, which allows us to color the data points in red and change their size to large, in order to make them stand out.

Many measurements are commonly represented as two-dimensional datasets. The data visualization commands in the Wolfram Language are designed to work with both one- and multidimensional data. When a command like `ListPlot` is given a one-dimensional list, it is assumed that the list contains y values that correspond to x values 1, 2, and so on. `ListPlot` also accepts a list of (x,y) pairs instead of single height values for y coordinates.

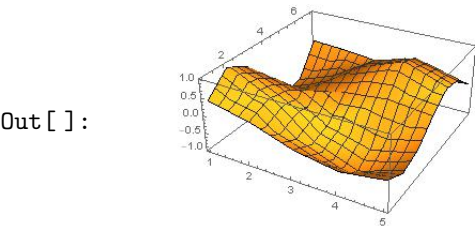


```
In[]: ListPlot[Table[{2i,Prime[i]}, {i,8}]]
```



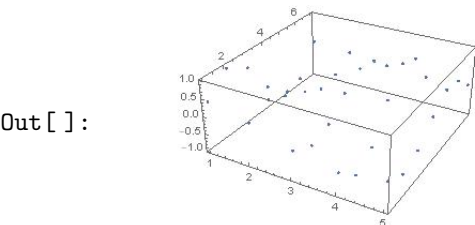
Visualizing lists or datasets in three dimensions is just as easy as visualizing them in one or two dimensions. Many of the plotting commands shown thus far have 3D equivalents, like `ListPlot` and `ListPlot3D`. The `ListPlot3D` command visualizes a 3D surface based on the values from that dataset, and this 3D surface has the same interactivity (rotation, panning, zooming) as other objects.

```
In[]: ListPlot3D@Table[Cos[i] Cos[j], {i,2,8},{j,-3,1}]
```



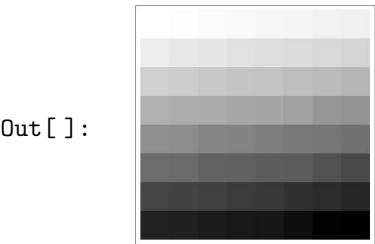
To visualize only the discrete data points and not a connecting mesh between the points, `ListPointPlot3D` can be used.

```
In[]: ListPointPlot3D@Table[Cos[i] Cos[j], {i,2,8},{j,-3,1}]
```



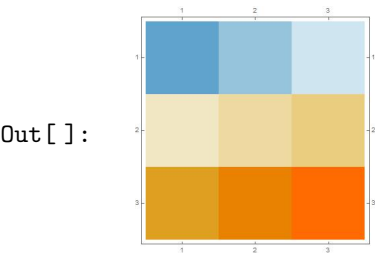
**Visualize a Matrix** Vectors and matrices have special visualization commands. `ArrayPlot` draws a representation of an array, coloring squares that represent larger values with darker colors. The following is the `ArrayPlot` of the  $8 \times 8$  matrix of the first 64 consecutive prime numbers.

```
In[]: ArrayPlot[Partition[Prime[Range[64]],8]]
```



`MatrixPlot` follows a similar logic, where the position of a value in a matrix determines the coloration of that position in the plot, with negative values shown in cool tones, like blue, and positive values shown in warm tones, like orange. The higher the magnitude of a value, the more intense its corresponding color is for that position.

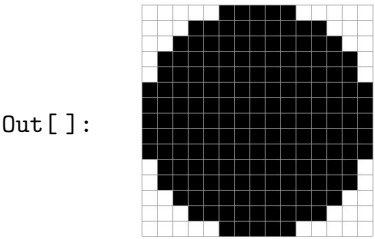
```
In[]: MatrixPlot[{{-10,-5,-1}, {2,4,6}, {20,30,40}}]
```



**Shape Matrices**   Mathematica provides certain shapes of matrices as follows.

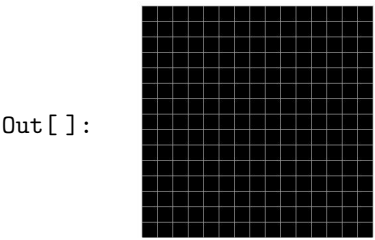
1. `DiskMatrix[r]` gives a matrix whose elements are 1 in a disk-shaped region of radius  $r$ , and are otherwise 0.

```
In[ ]: ArrayPlot[DiskMatrix[7], Mesh -> All]
```



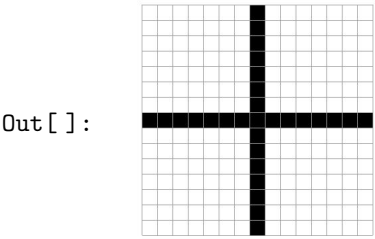
2. `BoxMatrix[r]` gives a  $(2r + 1) \times (2r + 1)$  matrix of 1's.

```
In[ ]: ArrayPlot[BoxMatrix[7], Mesh -> All]
```



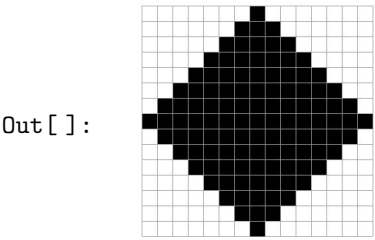
3. `CrossMatrix[r]` gives a matrix whose elements are 1 in a centered cross-shaped region that extends  $r$  positions along each index direction, and are 0 otherwise.

```
In[ ]: ArrayPlot[CrossMatrix[7], Mesh -> All]
```



4. `DiamondMatrix[r]` gives a matrix whose elements are 1 in a diamond-shaped region that extends  $r$  index positions to each side, and are 0 otherwise.

```
In[ ]: ArrayPlot[DiamondMatrix[7], Mesh -> All]
```



**9.9   Advanced Graphics**

In addition to using commands such as `Plot` and `ParametricPlot` to create 2D pictures, we can ask Mathematica to directly draw simple objects such as points, lines, circles, and rectangles. These objects are called graphics primitives, and they are the lowest-level commands used to produce every 2D Mathematica picture. To work with graphics primitives, you have to first define them with the `Graphics` command:

```
Graphics[ graphics primitive ]
```

or, to group several graphic primitives together, use

```
Graphics[ {graphics primitive1, graphics primitive2, ...} ]
```

3D graphics primitives are also available in Mathematica. You use the `Graphics3D` command both to define and show them. Some of the graphics primitives we have used often are the following.

1. `Point[p]` is a graphics and geometry primitive that represents a point at  $p$ .

```
In[ ]: Graphics[Point[{-2,1}]]
```

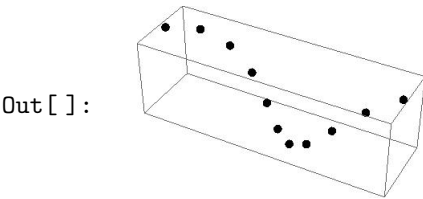


```
In[ ]: Graphics[Point[Table[{t,Sin[t]}, {t,0,2Pi, 2Pi/10}]]]
```



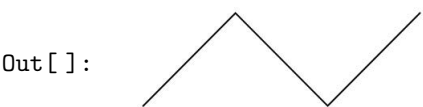
```
In[ ]: pts=Table[{t,Sin[t],Cos[t]}, {t,0,2Pi,2Pi/10}];
```

```
In[ ]: Graphics3D[Point[pts]]
```



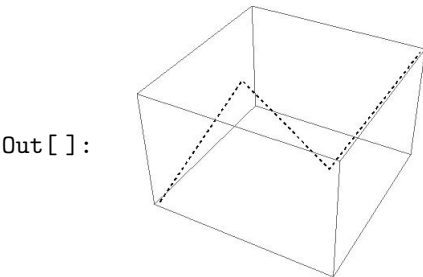
2. `Line[{p1,p2,...}]` represents the line segments joining a sequence for points  $p_i$ .

```
In[ ]: Graphics[{Thick, Line[{1,0}, {2,1}, {3,0}, {4,1}]]]
```



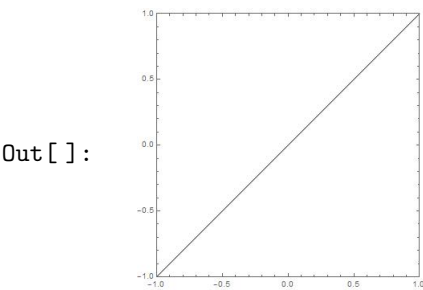
```
In[ ]: line=Line[{1,1,-1},{2,2,1},{3,3,-1},{4,4,1}];
```

```
In[ ]: Graphics3D[{Thick,Dashed,line}]
```

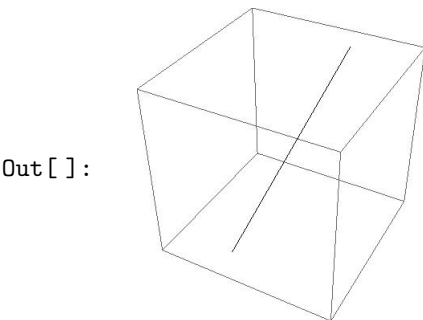


3. `InfiniteLine[{p1,p2}]` represents the infinite straight line passing through the points  $p_1$  and  $p_2$ .

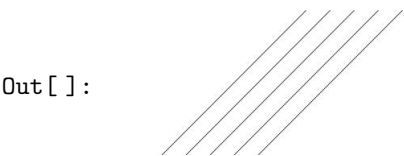
```
In[ ]: Graphics[InfiniteLine[{0,0},{1,1}], Frame->True]
```



```
In[ ]: Graphics3D[InfiniteLine[{0,0,0},{1,2,3}]]
```



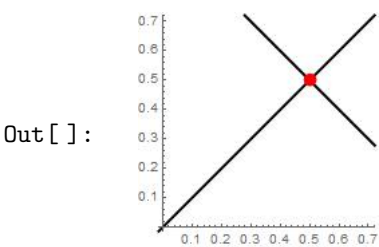
```
In[ ]: lines=Table[InfiniteLine[{0,y},{1,1}],{y,5}];
In[ ]: Graphics[lines,PlotRange->{{-6,6},{0,6}}]
```



**Example 9.5.** Find the point of intersection for the two lines  $\ell_1$  that passes through the point  $(0,0)$  in the direction of the vector  $(1,1)$ , and the line  $\ell_2$  that passes through the points  $(0,1)$ ,  $(1,0)$ .

```
In[ ]: line1 = InfiniteLine[{0,0},{1,1}];
In[ ]: line2 = InfiniteLine[0, 1, 1, 0];
In[ ]: sol=Solve[Element[{x,y},line1]&& Element[{x,y},line2],{x,y}]
```

```
Out[ ]: {{x -> 1/2, y -> 1/2}}
In[ ]: Graphics[{{Thick,line1,line2},{PointSize[0.06],Red,
Point[{x,y}/.sol]}},Axes->True,ImageSize->Small]
```

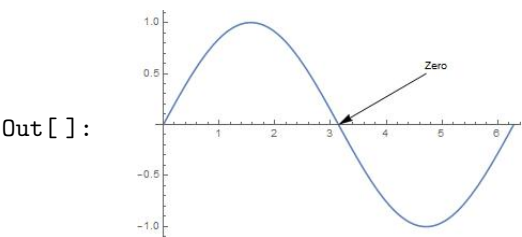


4. `Arrow[{pt1,pt2}]` is a graphics primitive that represents an arrow from  $pt_1$  to  $pt_2$ .

```
In[ ]: Graphics[Arrow[{{1,0},{2,1}},{3,0},{4,1}}]]
```

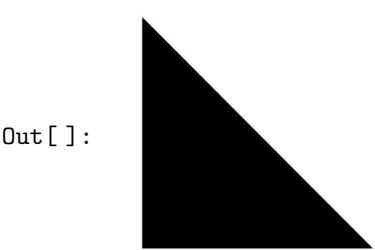


```
In[ ]: Plot[Sin[x],{x,0,2Pi},Epilog->{Arrow[{{3Pi/2,1/2},{Pi,0}}],
Text["Zero",{3Pi/2,1/2},{-1,-1}]}]
```

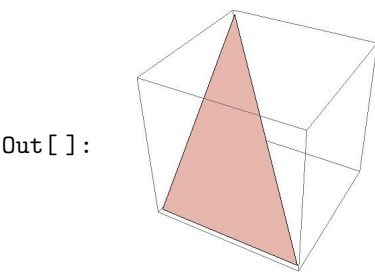


5. `Triangle[{p1,p2,p3}]` represents a filled triangle with corner points  $p_1$ ,  $p_2$ , and  $p_3$ .

```
In[ ]: Graphics[Triangle[]]
```



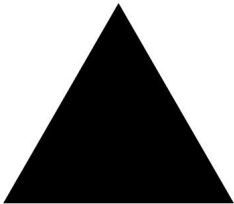
```
In[ ]: Graphics3D[Triangle[{{0,0,0},{1,0,0},{0,1,1}}]]
```



**Example 9.6.** To define an equilateral triangle by side length  $s$ , we use the command `SSSTriangle[a,b,c]` which returns a filled triangle with sides of lengths  $a$ ,  $b$ , and  $c$ .

```
In[]: EquilateralTriangle[s_] := SSSTriangle[s,s,s]
In[]: EquilateralTriangle[2]
Out[]: Triangle[{{0,0},{2,0},{1,√3}}]
In[]: Graphics@EquilateralTriangle[2]
```

Out[]:



**Example 9.7.** To define an isosceles triangle by side length  $s$  that adjacent to both equals angles  $\alpha$ , we may use either the command `IsoscelesTriangle[a,s,b]` which returns a filled triangle with angles  $a$  and  $b$  and side length  $c$ , and  $c$  is adjacent to both angles, or we use the command `SASTriangle[s1,a,s2]` which returns a filled triangle with sides of length  $s1$  and  $s1$  and angle  $a$  between them.

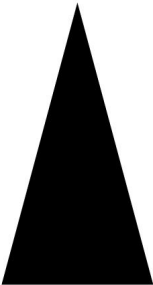
```
In[]: IsoscelesTriangle[a_,s_,b_] := ASATriangle[a,s,b]
In[]: IsoscelesTriangle[Pi/6,3,Pi/6]
Out[]: Triangle[{{0,0},{3,0},{3/2,√3/2}}]
In[]: Graphics@IsoscelesTriangle[Pi/6,3,Pi/6]
```

Out[]:



```
In[]: IsoscelesTriangle[s1_,a_,s2_] := SASTriangle[s1,a,s2]
In[]: Graphics@IsoscelesTriangle[2, Pi/6, 2]
```

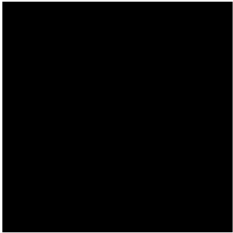
Out[]:



6. `Rectangle[{xmin,ymin},{xmax,ymax}]` represents an axis-aligned filled rectangle from  $\{xmin,ymin\}$  to  $\{xmax,ymax\}$ .

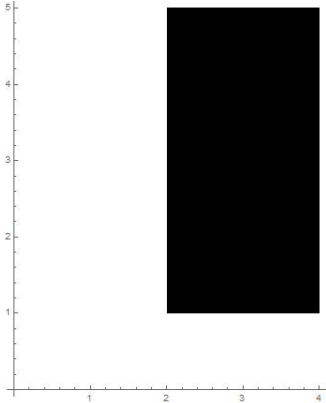
```
In[]: Graphics[Rectangle[]] (* this command defines a unit square *)
```

Out[]:



```
In[]: Graphics[Rectangle[{2,1},{4,5}],Axes->True,AxesOrigin->{0,0}]
```

Out[]:



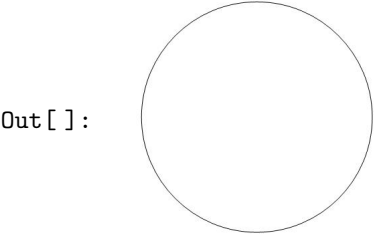
7. `RegularPolygon[n]` gives the regular polygon with  $n$  vertices equally spaced around the unit circle. `RegularPolygon[n,r]` gives the regular polygon of radius  $r$ .

```
In[ ]: Table[Graphics[RegularPolygon[k]],{k,3,8}]
```



8. `Circle[{x,y},r]` represents a circle of radius  $r$  centered at  $\{x,y\}$ . The command `Circle[{x,y},{rx,ry}]` gives an axis-aligned ellipse with semi-axes lengths  $rx$  and  $ry$ . While `Circle[{x,y},...,{ $\theta_1$ , $\theta_2$ }]` gives a circular or ellipse arc from angle  $\theta_1$  to  $\theta_2$ .

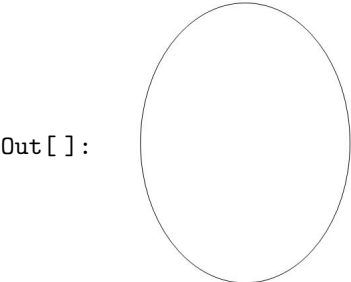
```
In[ ]: Graphics[Circle[]] (* Unit Circle *)
```



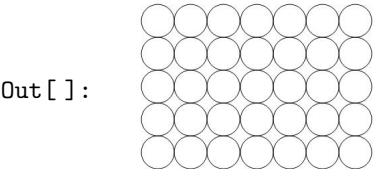
```
In[ ]: Graphics[Circle[{0,0},1,{Pi/6,3Pi/4}]] (* Circular Arc *)
```



```
In[ ]: Graphics[Circle[{0,0},{3,4}]] (* An Ellipse *)
```

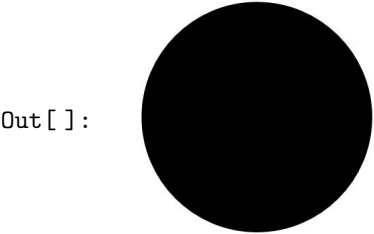


```
In[ ]: Graphics[Table[Circle[{i,j},1/2],{i,7},{j,5}]]
```



9. `Disk[{x,y},r]` represents a disk of radius  $r$  centered at  $\{x,y\}$ .

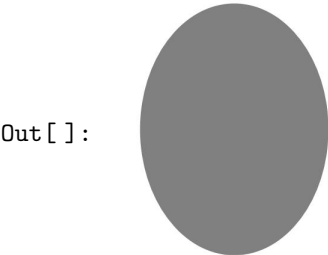
```
In[ ]: Graphics[Disk[]] (* Unit Disk *)
```



```
In[ ]: Graphics[{Orange,Disk[{0,0},1,{Pi/4,3Pi/4}]] (* Disk Sector *)
```



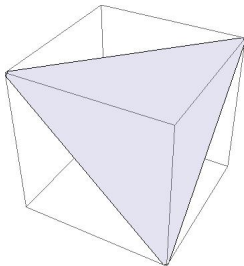
```
In[ ]: Graphics[{Gray,Disk[{0,0},{3,4}]] (* An Elliptical Disk *)
```



10. `InfinitePlane[{p1,p2,p3}]` represents the plane passing through the points `p1`, `p2`, and `p3`.  
`InfinitePlane[p,{v1,v2}]` represents the plane passing through the point `p` in the directions `v1` and `v2`.

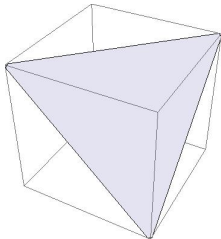
To represent a plane through the points  $(1,0,0)$ ,  $(1,1,1)$ , and  $(0,0,1)$   
`In[ ]:`   `Graphics3D[InfinitePlane[{1,0,0},{1,1,1},{0,0,1}]]`

`Out[ ]:`



To define the plane passing through the point  $(1,0,0)$  in the directions  $(0,1,1)$  and  $(1,0,1)$   
`In[ ]:`   `Graphics3D[InfinitePlane[{1,0,0},{0,1,1},{-1,0,1}]]`

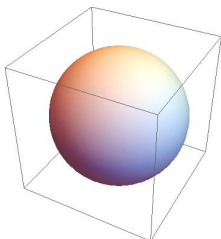
`Out[ ]:`



11. `Ball[p,r]` represents a ball of radius `r` centered at the point `p`.

A unit ball at the origin:  
`In[ ]:`   `Graphics3D[Ball[{0,0,0}]]`

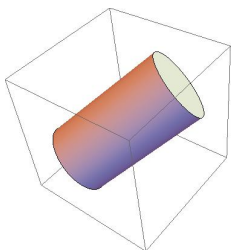
`Out[ ]:`



12. `Cylinder[{x1,y1,z1},{x2,y2,z2}],r]` represents a cylinder of radius `r` around the line from  $(x1,y1,z1)$  to  $(x2,y2,z2)$ .

Cylinder from the origin to  $\{1,1,1\}$  with radius  $\frac{1}{2}$ .  
`In[ ]:`   `Graphics3D[Cylinder[{0,0,0},{1,1,1}],1/2]]`

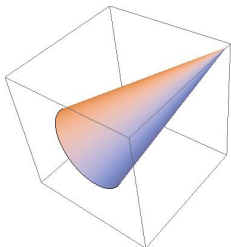
`Out[ ]:`



13. `Cone[{x1,y1,z1},{x2,y2,z2}],r]` represents a cone with a base of radius `r` centered at  $(x1,y1,z1)$  and a tip at  $(x2,y2,z2)$ .

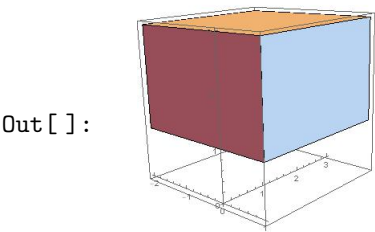
Cone from the origin to  $\{1,1,1\}$  with radius  $\frac{1}{2}$  at its base  
`In[ ]:`   `Graphics3D[Cone[{0,0,0},{1,1,1}],1/2]]`

`Out[ ]:`



14. `Cuboid[pmin,pmax]` represents an axis-aligned filled cuboid with lower corner `pmin` and upper corner `pmax`.

```
In[ ]: Graphics3D[Cuboid[{-2,0,1},{1,3,3}],Axes->True,AxesOrigin->{0,0,0}]
```



9.10   Region Integrals Measures

The Wolfram Language supports a broad range of standard properties and measures for geometric regions, including integral measures such as length, area, volume, and centroid.

**Arc Length**   The command `ArcLength[reg]` gives the length of the one dimensional region `reg`. For example, the arc length of the function  $f(x) = \frac{x^3}{6} - \frac{1}{2x}$  on the interval  $[1, 2]$  is

```
In[ ]: ArcLength[x^3/6+1/(2x), {x,1,2}]
Out[ ]: 17/12
```

The length can be computed using the polar representation of  $r(\theta) = 2 - 2 \cos \theta$  on  $[0, 2\pi]$  as follows.

```
In[ ]: ArcLength[{2-2Cos[theta],theta},{theta,0,2Pi},"Polar"]
Out[ ]: 16
```

Length of one revolution of the helix given parametrically is

```
In[ ]: ArcLength[{Cos[t],Sin[t],t},{t,0,2Pi}]
Out[ ]: 2√2π
```

The length of the line connecting the points  $(0, 0)$ ,  $(1, 1)$ , and  $(3, -1)$  is

```
In[ ]: ArcLength[Line[{0,0},{1,1},{3,-1}]]
Out[ ]: 3√2
```

The length of a circle with radius  $r$  is

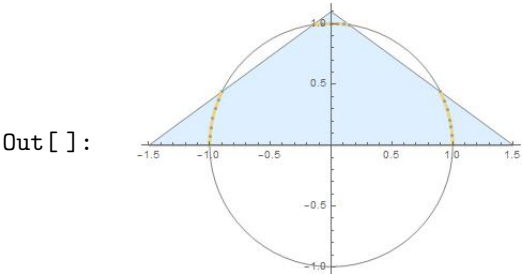
```
In[ ]: ArcLength[Circle[{x,y},r]]
Out[ ]: 2πr
```

The length of an ellipse is

```
In[ ]: N@ArcLength[Circle[{0,0},{2,3}]]
Out[ ]: 15.8654
```

The arc length of a circle intersected with a triangle:

```
In[ ]: R1 = Circle[];
In[ ]: R2 = Triangle[{{-3/2,0},{3/2,0},{0,11/10}}];
In[ ]: R3 = RegionIntersection[R1,R2];
```



```
In[ ]: ArcLength[R3]
Out[ ]: 2 (sin^-1 (3/692 (121 - 5√295)) + cos^-1 (3/692 (5√295 + 121)))
```

The perimeter of the triangle whose two of its angles are  $\frac{\pi}{6}$  and  $\frac{\pi}{3}$ , and the length of the side adjacent to both angles is of length 2, is

```
In[ ]: ArcLength[RegionBoundary[ASATriangle[Pi/6,2,Pi/3]]]
Out[ ]: √3 + 3
```

where the command `RegionBoundary` represents the boundary of the region `reg`.



**Area** The command `Area[reg]` gives the area of the two dimensional region `reg`. For example, the area of the disk with radius `r` is

```
In[]: Area[Disk[{x,y}, r]]
Out[]:  $\pi r^2$ 
```

The surface area of a sphere (ball) with radius `r` is

```
In[]: Area[RegionBoundary[Ball[{x,y,z},r]]]
In[]: Area[Sphere[{x,y,z},r]]
Out[]:  $4\pi r^2$ 
```

The area of an annulus with inner radius 1 and outer radius 2 equals

```
In[]: Area[{r Sin[theta], r Cos[theta]}, {r,1,2}, {theta,0,2Pi}]
Out[]:  $3\pi$ 
```

The area of the triangle whose vertices at the points  $(-\frac{1}{2}, -1)$ ,  $(1, 2)$ , and  $(3, 0)$  is

```
In[]: Area[Triangle[{{-1/2,-1},{1,2},{3,0}}]]
Out[]:  $\frac{9}{2}$ 
```

The area of a triangle with sides of length `a`, `b`, and `c` is

```
In[]: Area[SSSTriangle[a,b,c]]
Out[]:  $\frac{1}{4}\sqrt{(a+b-c)(a-b+c)(-a+b+c)(a+b+c)}$ 
```

The area of an ellipse centered at  $(c_x, c_y)$  is

```
In[]: Area[Disk[{cx,cy},{a,b}]]
Out[]:  $ab\pi$ 
```

**Volume** The command `Volume[reg]` gives the volume of the three dimensional region `reg`. For example, the volume of a ball in 3D of radius `r` is

```
In[]: Volume[Ball[{a,b,c},r]]
Out[]:  $\frac{4\pi r^3}{3}$ 
```

The volume and the surface area of a cone of height `h` and radius `r` of its circular base are

```
In[]: PowerExpand@Volume[Cone[{0,0,0},{0,0,h}],r]]
Out[]:  $\frac{1}{3}\pi hr^2$ 
In[]: Area[RegionBoundary[Cone[{0,0,0},{0,0,h}],r]]
Out[]:  $\pi r \left( \sqrt{h^2 + r^2} + r \right)$ 
```

**Centroid** The centroid or geometric center of a plane figure is the arithmetic mean ("average") position of all the points in the shape. `RegionCentroid[reg]` gives the centroid of the region `reg`. For example, the centroid of the disk with center  $(a,b)$  and radius `r` is


```
In[]: RegionCentroid[Disk[{a,b},r]]
Out[]: {a,b}
```

The centroid of the rectangle is


```
In[]: RegionCentroid[Rectangle[{a,b},{c,d}]]
Out[]:  $\left\{ \frac{a+c}{2}, \frac{b+d}{2} \right\}$ 
```

9.11   Graphs and Networks

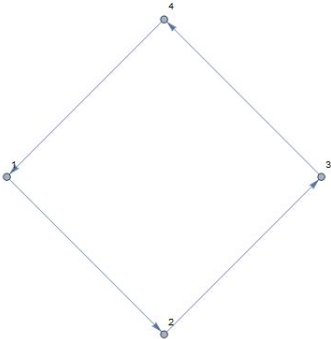
A graph is a way of showing connections between things - say, how web pages are linked, or how people form a social network. Let us start with a very simple graph, in which 1 connects to 2, 2 to 3, and 3 to 4. Each of the connections is represented by  $\rightarrow$  (typed as `->`).

```
In[]: Graph[{1->2,2->3,3->4}]
Out[]: 
```

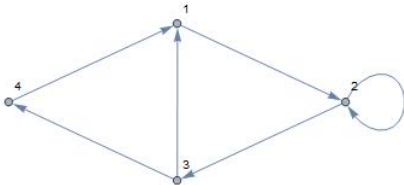
Automatically label all the "vertices":

```
In[]: Graph[{1->2,2->3,3->4},VertexLabels->All]
Out[]: 
```

Let us add one more connection: to connect 4 to 1. Now we have a loop.

```
In[]: Graph[{1->2,2->3,3->4,4->1},VertexLabels->All]
Out[]: 
```

Add two more connections, including one connecting 2 right back to 2:

```
In[]: Graph[{1->2,2->3,3->4,4->1,3->1,2->2},VertexLabels->All]
Out[]: 
```

As we add connections, the Wolfram Language chooses to place the vertices or nodes of the graph differently. All that really matters for the meaning, however, is how the vertices are connected. And if you do not specify otherwise, the Wolfram Language will try to lay the graph out so it is as untangled and easy to understand as possible.

You can do computations on the graph, say finding the shortest path that gets from 4 to 2, always following the arrows.

```
In[]: FindShortestPath[Graph[{1->2,2->3,3->4,4->1,3->1,2->2}],4,2]
Out[]: {4,1,2}
```

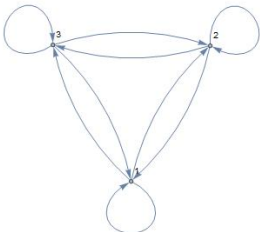
Now let us make another graph. This time let us have 3 nodes, and let us have a connection between every one of them. Start by making an array of all possible connections between 3 objects:

```
In[]: Table[i->j,{i,3},{j,3}]
Out[]: {{1->1,1->2,1->3},{2->1,2->2,2->3},{3->1,3->2,3->3}}
```

The result here is a list of lists. But what Graph needs is just a single list of connections. We can get that by using Flatten to "flatten" out the sublists.

```
In[]: Flatten[Table[i->j,{i,3},{j,3}]]
Out[]: {1->1,1->2,1->3,2->1,2->2,2->3,3->1,3->2,3->3}
```

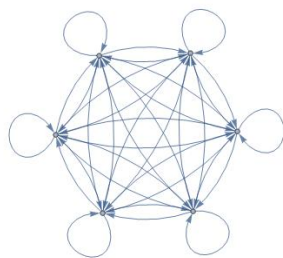
Now, show the graph of these connections:

```
In[]: Graph[Flatten[Table[i->j,{i,3},{j,3}]],VertexLabels->All]
Out[]: 
```

The following generates the completely connected graph with 6 nodes:

```
In[]: Graph[Flatten[Table[i->j,{i,6},{j,6}]]]
```

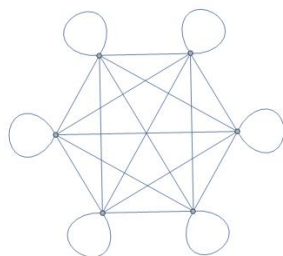
```
Out[]:
```



Sometimes the "direction" of a connection does not matter, so we can drop the arrows.

```
In[]: UndirectedGraph[Flatten[Table[i->j,{i,6},{j,6}]]]
```

```
Out[]:
```



## 9.12 Exercises

1. The standard normal curve used in probability and statistics is defined by the function  $f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ . Sketch the graph of  $f$  for  $x \in [-3, 3]$ .
2. The graphs of a function and its inverse are symmetric with respect to the line  $y = x$ . Plot the functions  $f(x) = x^2$ ,  $x \in [0, 2]$ , and its inverse  $f^{-1}(x) = \sqrt{x}$ ,  $x \in [0, 4]$ , and the line  $y = x$  and observe the symmetry.
3. Sketch the graphs of  $y = x^2$ ,  $y = -x^2$ , and  $y = x^2 \sin(10x)$ ,  $x \in [-2\pi, 2\pi]$ , on a single set of axes enclosed by a frame.
4. Sketch the parabola  $y = x^2 - 9$  and a circle of radius 3 centered at the origin.
5. The curve traced by a point on a circle as the circle rolls along a straight line is called a cycloid and has parametric equations  $x = r(\theta - \sin \theta)$  and  $y = r(1 - \cos \theta)$  where  $r$  represents the radius of the circle. Plot the cycloid formed as a circle of radius 1 makes four complete revolutions.
6. The polar graph  $r = \theta$  is called the Spiral of Archimedes. Sketch the graph for  $\theta \in [0, 10\pi]$ .
7. Sketch the graph defined by the equation  $y^2 = x^3(2 - x)$  where  $x \in [0, 2]$  and  $y \in [-2, 2]$ .
8. Plot the first 50 prime numbers.
9. Make a number line plot of the first 20 elements of the sequence given by the rule  $\frac{1}{n}$ .
10. Plot the functions  $y = x^2$  and  $y = 8 - x^2$  and color the region enclosed between them using the plot option `Filling`.
11. Plot the graph of the function  $e^{-x^2-y^2}$  above the rectangle  $[-2, 2] \times [-2, 2]$ .
12. Graph the paraboloid  $z = x^2 + y^2$  with the plane  $z + y = 12$  above the rectangle  $[-5, 5] \times [-5, 5]$ . Do not draw axes or a surrounding box.
13. Sketch the space curve defined by the parametric equations

$$\begin{aligned} x(t) &= (4 + \sin 20t) \cos t \\ y(t) &= (4 + \sin 20t) \sin t \\ z(t) &= \cos 20t \end{aligned}$$

where  $t \in [0, 2\pi]$ . This curve is called a toroidal spiral since it lies on the surface of a torus.

14. Draw the "ice cream cone" formed by the cone  $z = 3\sqrt{x^2 + y^2}$  and the upper half of the sphere  $x^2 + y^2 + (z - 9)^2 = 9$ . Use cylindrical coordinates.
15. Sketch the graph of the surface  $\rho = 1 + \sin 4\theta \sin \phi$  given in spherical coordinates where  $\theta \in [0, 2\pi]$  and  $\phi \in [0, \pi]$ .

16. Draw a contour plot of  $f(x, y) = \sin x + \sin y$  on the square  $x, y \in [-4\pi, 4\pi]$ .
17. Let  $f(x, y, z) = 5x^2 + 2y^2 + z^2$ . Draw the level surfaces  $f(x, y, z) = k$  for  $k=1, 4, 9, 16$ , and 25. Sketch the surfaces only for  $y \geq 0$  so that all the surfaces will be visible.
18. A triangle has angles  $\frac{\pi}{6}$  and  $\frac{\pi}{4}$ , and a side of length 1 is adjacent to only one of the angles. Find the perimeter of this triangle, then find its centroid and display the position of the centroid inside the triangle.
19. Make a graph with 4 nodes in which every node is connected.
20. For the graph  $\{1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4, 4 \rightarrow 1, 3 \rightarrow 1, 2 \rightarrow 2\}$ , make a grid giving the shortest paths between every pair of nodes, with the start node as row and end node as column.

## References

1. **Getting Started with Mathematica**,  
*C-K. Cheung, Gerard E. Keough, Robert H. Gross, Charles Landraitis*,  
Wiley; 3rd edition (March 16, 2009).
2. **Mathematica: A Problem-Centered Approach**,  
*Roozbeh Hazrat*,  
Springer; 2nd ed. 2015 edition (January 12, 2016).
3. **Programming with Mathematica: An Introduction**,  
*Paul Wellin*,  
Cambridge University Press; 1st edition (February 25, 2013).
4. **Hands-On Start to Wolfram Mathematica**,  
*Cliff Hastings, Kelvin Mischo, Michael Morrison*,  
Wolfram Media, Inc. (September 15, 2015).
5. **An Elementary Introduction to the Wolfram Language**,  
*Stephen Wolfram*,  
Wolfram Media, Inc. (December 11, 2015).
6. **Mathematica by Example**,  
*Martha L. Abell, James P. Braselton*,  
Academic Press; 4th edition (September 23, 2008).
7. **Schaum's Outline of Mathematica**,  
*Eugene Don*,  
McGraw-Hill Education; 2nd edition (May 19, 2009)
8. **The Student's Introduction to MATHEMATICA**,  
*Bruce F. Torrence, Eve A. Torrence*,  
Cambridge University Press; 2nd edition (February 2, 2009)

## Index

Add Column, 76  
Add Row, 76  
Algebraics, 30  
Alignment, 96  
AllTrue, 43  
And, 29  
Apart, 47  
Apply, 149  
ArcLength, 126  
Area, 128  
ArrayPlot, 116  
Array, 24, 74  
Arrow, 120  
AspectRatio, 95  
Assumptions, 47  
Automatic, 95  
AxesLabel, 102, 103  
Ball, 125  
BarChart, 113  
Basic Commands, 76  
Basic Math Assistant, 76  
BetaDistribution, 145  
BinomialDistribution, 145  
Binomial, 41  
Block, 158  
BooleanTable, 29  
Booleans, 30  
BoxMatrix, 117  
BoxRatios, 103  
CDF, 145  
Ceiling, 38  
CharacteristicPolynomial, 83  
ChiDistribution, 145  
ChiSquareDistribution, 145  
Circle, 123  
Clear[Global' \*], 18  
Clear, 18  
CoefficientArrays, 84  
CoefficientList, 50  
Coefficient, 50  
Complement, 31  
Complexes, 30  
CompositeQ, 34  
Conditioned, 146  
Cone, 125  
ConstantArray, 78  
ContourPlot3D, 111  
ContourPlot, 100  
CoordinateTransformData, 107  
CoprimeQ, 37  
CountRoots, 51  
Count, 27  
CreatePalette, 9  
CrossMatrix, 117  
Cross, 75  
CubeRoot, 12  
Cuboid, 126  
Curl, 140  
Cylinder, 125  
CylindricalDecomposition, 138  
DSolveValue, 90  
DSolve, 89, 141  
Dashing, 97  
Default, 157  
DeleteDuplicates, 31  
Det, 81  
DiagonalMatrix, 79  
DiamondMatrix, 117  
DigitCount, 36  
Direction, 59  
DiscreteLimit, 68  
DisjointQ, 32  
DiskMatrix, 116  
Disk, 123  
Distributed, 146  
Dividers, 96  
Divisible, 37  
Divisors, 35  
Div, 140  
Dot, 75, 80  
Dt, 62  
D, 60, 135  
Eigenvalues, 83  
Eigenvectors, 83  
Element, 30  
Equivalent, 33  
Evaluate, 147  
EvenQ, 39  
Exists, 32  
Expand, 45  
Expectation, 146  
ExponentialDistribution, 145  
Extension, 45  
Factor, 45  
False, 29  
Fibonacci, 38  
Filling, 132  
FindInstance, 57  
FindRoot, 54  
FindSequenceFunction, 68  
FindShortestPath, 130  
First, 27  
Fit, 146  
Flatten, 131  
Floor, 38  
ForAll, 32  
FrameLabel, 102  
FromDigits, 39  
FullForm, 149  
FullSimplify, 32, 49  
FunctionDomain, 58  
FunctionRange, 58  
GCD, 37  
GeometricMean, 144  
GoldenRatio, 98  
Grad, 135  
GraphicsGrid, 96  
GraphicsRow, 96  
Graphics, 117  
Graph, 129  
HarmonicMean, 144  
IdentityMatrix, 79  
If, 77, 156  
Implies, 33  
InfiniteLine, 119  
InfinitePlane, 124  
IntegerDigits, 39  
IntegerExponent, 35  
IntegerLength, 36  
IntegerPartitions, 42  
IntegerReverse, 39  
Integers, 30  
Integrate, 63, 137  
InterpolatingFunction, 92

InterpolatingPolynomial, 147  
 Intersection, 31  
 InverseCDF, 145  
 InverseLaplaceTransform, 92  
 Inverse, 81  
 IsoscelesTriangle, 121  
 Join, 85  
 LCM, 37  
 LaplaceTransform, 92  
 Last, 27  
 Length, 26  
 Limit, 58  
 LinearSolve, 83  
 Line, 118  
 ListLinePlot, 113  
 ListPlot3D, 115  
 ListPlot, 112  
 ListPointPlot3D, 115  
 Log10[x], 13  
 Log2[x], 13  
 Mapping, 108  
 Map, 26, 150  
 MatchQ, 152  
 MatrixForm, 76  
 MatrixPlot, 116  
 MatrixPower, 81  
 MatrixRank, 82  
 Maximize, 63, 136  
 Max, 27  
 MeanDeviation, 144  
 Mean, 144  
 Median, 144  
 MersennePrimeExponentQ, 44  
 MersennePrimeExponent, 44  
 Mesh, 103  
 MinMax, 27  
 Minimize, 63, 136  
 Min, 27  
 Mod, 36  
 NDSolve, 91  
 NIntegrate, 65, 139  
 NSolve, 54  
 NSum, 69  
 N[expr,n], 13  
 Negative, 155  
 NestList, 151  
 NestWhileList, 151  
 Nest, 151  
 NextPrime, 35  
 NonNegative, 155  
 NormalDistribution, 145  
 Normalize, 87  
 Normal, 71, 84  
 Norm, 75  
 NullSpace, 82  
 Null, 156  
 NumberLinePlot, 113  
 N, 13  
 Or, 29  
 PDF, 145  
 PalindromeQ, 43  
 ParametricPlot3D, 111  
 ParametricPlot, 97  
 Partition, 77  
 Part, 24  
 PerfectNumberQ, 42  
 PerfectNumber, 42  
 PieChart, 113  
 Plot3D, 102  
 PlotLabel, 102  
 PlotRange, 94  
 PlotStyle, 96, 114  
 Plot, 94  
 Plus, 149  
 Point, 118  
 PoissonDistribution, 145  
 PolarPlot, 99  
 Positive, 155  
 PowersRepresentations, 57  
 PrimePi, 35  
 PrimeQ, 34  
 Primes, 30  
 Prime, 34  
 Probability, 145  
 Product, 71  
 Projection, 88  
 QuotientRemainder, 36  
 Quotient, 36  
 RSolveValue, 68  
 Range, 23  
 Rationals, 30  
 Reals, 30  
 Rectangle, 122  
 RecurrenceTable, 68  
 RecursionLimit, 158  
 Reduce, 55  
 RegionBoundary, 127  
 RegionCentroid, 129  
 RegularPolygon, 122  
 Reverse, 26  
 RevolutionPlot3D, 105  
 Root, 54  
 Round, 38  
 RowReduce, 85  
 SASTriangle, 121  
 SSSTriangle, 121  
 Select, 39  
 Series, 71  
 Show, 95  
 Simplify, 46  
 Solve, 51  
 Sort, 27  
 SphericalPlot3D, 105  
 StandardDeviation, 144  
 StudentTDistribution, 145  
 SubsetQ, 31  
 Subsets, 31  
 SumConvergence, 70  
 Sum, 69  
 Surd[x,n], 13  
 Table, 23  
 TautologyQ, 30  
 Thickness, 97  
 Ticks, 101  
 Times, 149  
 Together, 47  
 Total, 27  
 TransformedField, 109  
 Transpose, 81  
 Triangle, 120  
 TrigExpand, 48  
 TrigFactor, 48  
 TrigReduce, 48  
 True, 29  
 Tr, 82  
 UniformDistribution, 145  
 Union, 31  
 Variance, 144  
 VectorAngle, 76  
 VectorQ, 74

VertexLabels, 129  
 ViewPoint, 104  
 Volume, 129

abort evaluation, 7  
 absolute value, 11  
 addition, 10  
 addition of fractions, 45  
 alternating series, 70  
 analytic solutions of ODEs, 89  
 angle of vectors, 75  
 annulus, 128  
 anonymous functions, 21  
 antiderivative, 63  
 arbitrary constant, 63, 89  
 arc length, 126  
 area, 126  
 arithmetic mean, 129  
 arrow, 120  
 arrow symbol, 20  
 aspect ratio, 94  
 augmented matrix, 84

ball, 125  
 bar chart, 113  
 base-10 logarithm, 13  
 base-2 logarithm, 13  
 basic arithmetic, 10  
 basic commands, 9  
 basic math assistant, 8  
 binomial coefficient, 41  
 boolean expressions, 29  
 boundary conditions, 142  
 boundary of a region, 127  
 boundary value problem, 90  
 built-in functions, 11

calculator, 10  
 calculus, 58  
 Cartesian coordinates, 107  
 Cauchy-Euler ODE, 90  
 Cayley-Hamilton theorem, 88  
 cell menu, 7  
 centroid, 126  
 characteristic polynomial, 82  
 circles, 117  
 Collatz function, 156  
 collection of terms, 45  
 column vector, 74  
 combined graphic, 96  
 comment, 13  
 complement, 31  
 composite number, 34  
 conditional function, 77  
 conditional probability, 146  
 cone, 107, 125  
 consistent, 84  
 constant matrix, 78  
 contradiction, 33  
 converge, 68  
 coordinate systems, 107  
 coprime, 37  
 critical numbers, 66  
 cross product, 75  
 cross-shaped region, 117  
 cube root, 12  
 cumulative distribution function, 145  
 curl, 140  
 curly braces, 7  
 cyclic number, 41

cycloid, 132  
 cylinder, 125  
 cylindrical coordinates, 105

decimal point, 10  
 default values, 157  
 definite integral, 64  
 delete all output, 7  
 derivative, 60  
 descriptive statistics, 144  
 determinant, 81  
 diagonal matrix, 78  
 diagonalizable matrix, 88  
 diamond-shaped region, 117  
 different values, 29  
 direction, 131  
 directional derivative, 136  
 disjoint sets, 32  
 disk sector, 124  
 disk-shaped region, 116  
 diverge, 68  
 divergence of vector field, 140  
 divergent series, 69  
 dividend, 36  
 division, 10  
 divisor, 36  
 document center, 14  
 domains, 30  
 dot product, 75  
 double integrals, 137  
 Dudeney number, 42  
 duplication, 31

eigenvalue problem, 82  
 eigenvalues, 82  
 eigenvectors, 82  
 elementary row operations, 84  
 ellipse, 123  
 elliptical disk, 124  
 equation solving, 51  
 equilateral triangle, 121  
 Euclidean division, 36  
 evaluation menu, 7  
 expected value, 146  
 exponential, 11

factorial, 29  
 Fibonacci Sequence, 38  
 finding roots, 51  
 flatten, 131  
 framed picture, 102  
 front end, 5  
 function domain, 58  
 function range, 58  
 functions of several variables, 19  
 functions producing lists, 23  
 Fundamental Theorem of Calculus, 64

G. H. Hardy, 57  
 Gaussian elimination, 84  
 general term, 68  
 geometric center, 129  
 geometric series, 69  
 Goldbach partition, 43  
 Goldbach's conjecture, 43  
 gradient, 135  
 graph, 129  
 graph of a function, 94  
 graphics primitives, 117  
 greater than, 29  
 greatest common divisor, 37



happy number, 152, 158  
 harmonic function, 143  
 Harshad number, 40  
 head of expression, 149  
 heat equation, 142  
 helix, 126  
 help menu, 14  
 Heron's formula, 19  
 higher-level operations, 10  
 higher-order derivatives, 60, 135  
 higher-order implicit derivative, 62  
 highest power, 35  
 Hilbert matrix, 87  
 homogeneous ODE, 90  
 horizontal angle measured, 105  
 horizontal radial distance, 105  
 horizontal range, 94  
 hyperbolic functions, 11  
 hyperboloid, 112  
  
 ice cream cone, 133  
 identity matrix, 78  
 immediate assignment, 16  
 implicit differentiation, 62  
 implicit equation plot, 100  
 inconsistent, 84  
 inequalities, 56  
 infinite sum, 69  
 infinity, 11  
 infix, 14  
 initial conditions, 142  
 initial value problem, 90  
 input, 6  
 integer factorization, 35  
 integer partition, 42  
 integration, 63  
 integration by parts, 63  
 interpolation, 147  
 intersection, 31  
 interval of convergence, 70  
 inverse hyperbolic, 11  
 inverse trigonometric, 11  
 invertible, 81  
 isosceles triangle, 121  
 iterated integral, 137  
 iterated triple integral, 139  
  
 kernel, 5  
  
 labels, 102  
 Laplace transform, 92  
 Laplace's equation, 143  
 leading coefficients, 84  
 least common multiple, 37  
 least-square fit, 147  
 left-hand limit, 59  
 length, 126  
 less than, 29  
 level curves, 110  
 level surfaces, 110  
 limits, 58, 134  
 line integrals, 140  
 linear systems, 53, 83  
 lines, 117  
 list, 23  
 listable functions, 26  
 logarithm to base  $a$ , 11  
 logic, 29  
 lower-triangular matrix, 78  
  
 mathematical constants, 11  
 MathLink interface, 6  
 matrices, 76  
 matrix power, 81  
 maximum, 63  
 mean, 144  
 Mean-Value Theorem, 61  
 measures of central tendency, 144  
 measures of variation, 144  
 median, 144  
 Mersenne prime, 43  
 minimum, 63  
 modulo operation, 36  
 modulus, 36  
 multiple datasets, 114  
 multiplication, 10  
 mutually prime, 37  
  
 natural logarithm, 11  
 negation, 29  
 nested lists, 76  
 nested loops, 151  
 networks, 129  
 Newton's Method, 68  
 nonlinear exact ODE, 90  
 nonlinear ODE, 89  
 nonsingular, 81  
 norm, 75  
 normal line, 66  
 notebook interface, 5  
 notebooks, 5  
 $n$ th root, 13  
 null space, 82  
 number of real roots, 51  
 numerical notations, 13  
  
 optimization, 136  
 ordinary differential equation, 89  
 outer layer, 51  
 output, 6  
 overdetermined, 83  
  
 $p$  series, 70  
 palettes, 8  
 palettes menu, 8  
 palindromic number, 43  
 parabola, 132  
 paraboloid, 132  
 parametric curves, 111  
 parametric plot, 97  
 parametric surfaces, 111  
 parentheses, 7  
 partial derivatives, 135  
 partial differential equations, 141  
 partial fraction, 47  
 particular solution, 91  
 pattern matching, 152  
 perfect number, 42  
 perimeter, 127  
 pie chart, 113  
 piecewise functions, 20  
 plane, 124  
 plotting curves, 97  
 point of intersection, 119  
 points, 117  
 polar plot, 99  
 polynomials expansion, 45  
 polynomials factoring, 45  
 postfix, 14  
 power, 10

power set, 31  
 precedence, 10  
 prefix, 14  
 prime factorization, 35  
 prime number, 34  
 prime palindromic, 41  
 probability density function, 145  
 probability distributions, 145  
 product, 71  
 proportion, 94  
 pure functions, 21  
  
 quadratic function, 21  
 quantifiers, 32  
 quotient, 36  
  
 radial distance in space, 105  
 radian, 11  
 rank of a matrix, 82  
 rational function, 47  
 reciprocal, 24  
 rectangles, 117  
 recursion formulas, 68  
 recursive function, 157  
 Recursive Sequence, 68  
 reduced row echelon form, 84  
 regression, 146  
 regular polygon, 122  
 relatively prime, 37  
 remainder, 36  
 replace, 149  
 repunits, 42  
 right-hand limit, 59  
 row reduction, 84  
 row vector, 74  
  
 S. Ramanujan, 57  
 saddle, 103  
 scientific notations, 13  
 Sequences, 68  
 sets, 31  
 shortest path, 130  
 singular matrix, 82  
 smooth curves, 134  
 social number, 158  
 sombrero, 103  
 spherical coordinates, 105  
 Spiral of Archimedes, 132  
 square brackets, 7  
 square root, 11  
 standard normal curve, 132  
 substitution, 63  
 substitution rule, 20  
 subtraction, 10  
 sum, 69  
 surface integrals, 140  
 symbol &, 21  
 symmetric, 132  
 symmetric matrix, 87  
  
 tangent line, 61  
 tautology, 30  
 taxicab number, 57  
 Taylor polynomials, 71  
 Taylor series, 70  
 text-based interface, 5  
 tick marks, 101  
 toroidal spiral, 132  
 total derivative, 62  
 trace, 82  
 transcendental functions, 47  
  
 transforming fields, 109  
 transpose, 81  
 triangular matrix, 78  
 tridiagonal matrix, 87  
 trigonometric functions, 11  
 triple integrals, 137  
 truth tables, 29  
 twin primes, 43  
  
 underdetermined, 83  
 underscore character, 18  
 union, 31  
 unit ball, 125  
 unit disk, 123  
 unit matrix, 79  
 unit square, 122  
 unit vector, 87  
 universal set, 31  
 upper-triangular matrix, 78  
  
 vector field, 140  
 vector projection, 88  
 vector-valued function, 140  
 vectors, 74  
 vertical angle measured, 105  
 vertical range, 94  
 visualize a matrix, 116  
 volume, 126  
  
 wave equation, 142  
 Wolfram Language, 5