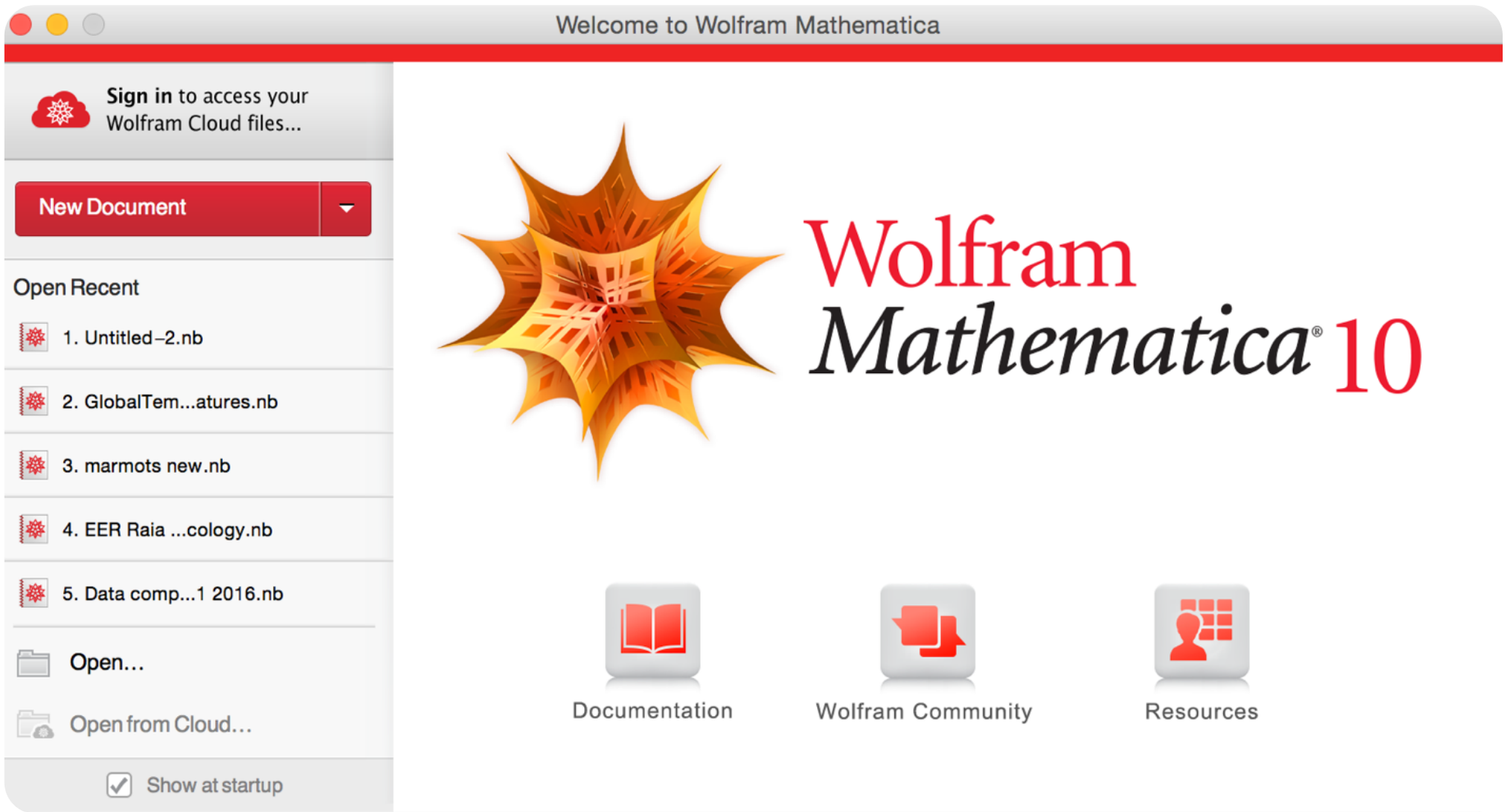
Mathematica

An Introduction



(c) Wolfram Research

Uses for *Mathematica*

Calculations, simple or complicated

```
In[1]:= 12 + 20
Out[1]= 32
```

```
In[2]:= ((10 * 5) + (20 * 30)) / 10^0.5
Out[2]= 205.548
```

Mathematical functions

```
In[7]:= Log[25] // N
Out[7]= 3.21888
```

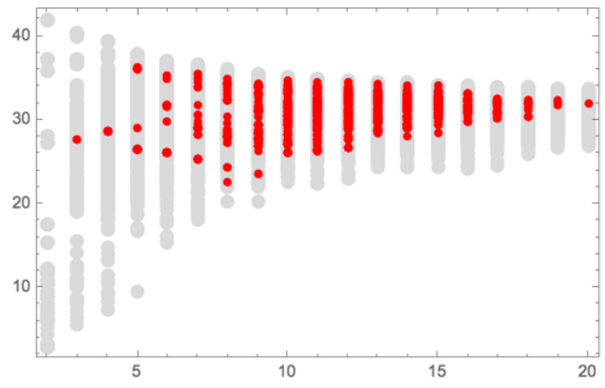
Statistical analysis

```
Out[15]= {ANOVA → |      DF   SumOfSq     MeanSq      FRatio     PValue
                  | Model  1    0.0682131   0.0682131   0.834059   0.363343
                  | Error  98   8.01489     0.0817845
                  | Total  99   8.0831

                  |      All        0.506327
          CellMeans → | Model[1]    0.477453 |,
                  | Model[2]    0.529952 |
```

Programming, simple or complicated

```
In[3]:= Do[Print["Species " <> ToString[x]], {x, 8}]
Species 1
Species 2
Species 3
Species 4
Species 5
Species 6
Species 7
Species 8
```

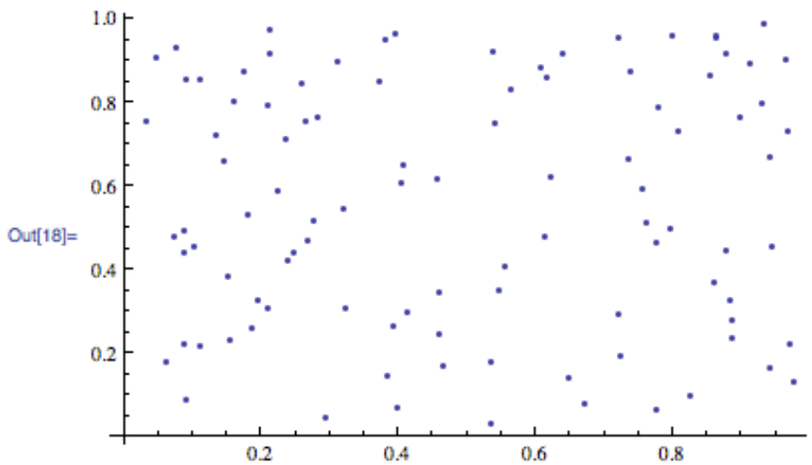
```
PhyloTmp =
  Table[
   Flatten[{faunas[[x, 1, {1, 3, 4}]],
     MeanPairwiseDivergence[ToString[#] & /@ faunas[[x, 1 ;;, 2]], VarY]}],
   {x, Length[faunas]}];
PhyloData = Select[PhyloTmp, #[[4]] > 2 && #[[5]] > 12 && #[[5]] < 39 &];
Do[PhyloData[[x, {3, 2}]] =
  GeoGridPosition[GeoPosition[{PhyloData[[x, 3]], PhyloData[[x, 2]]}],
    {"LambertAzimuthal", {"Centering" → {70, -100}}}][[1]],
 {x, Length[PhyloData]}]

Graphics[
 {{LightGray, PointSize[0.025], Point[Flatten[randommeandivergence, 1]]},
  {Red, PointSize[0.015], Point[PhyloData[[1 ;;, {4, 5}]]]}}, Frame → True,
 AspectRatio → 1 / GoldenRatio]
```



Graphics in *Mathematica*

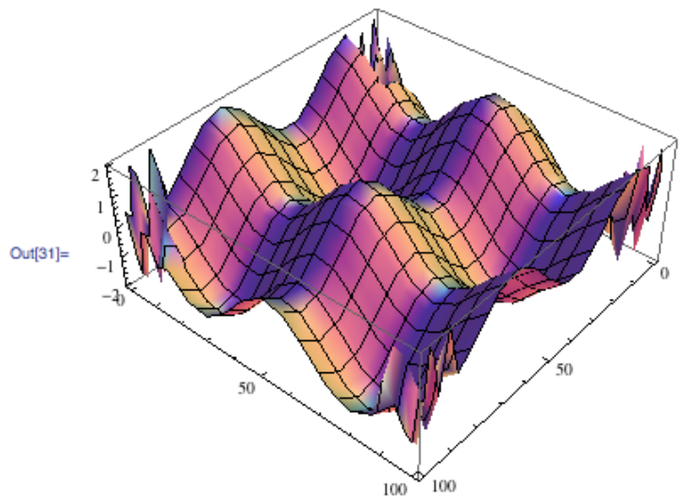
Plots of data

```
In[18]:= ListPlot[data]
```



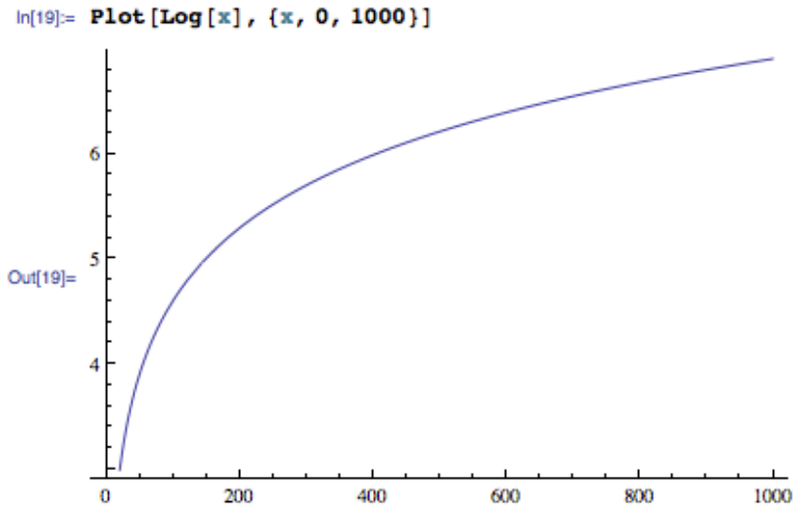
Three-dimensional plots

```
In[31]:= Plot3D[{Sin[x] + Cos[y]}, {x, 0, 100}, {y, 0, 100}]
```



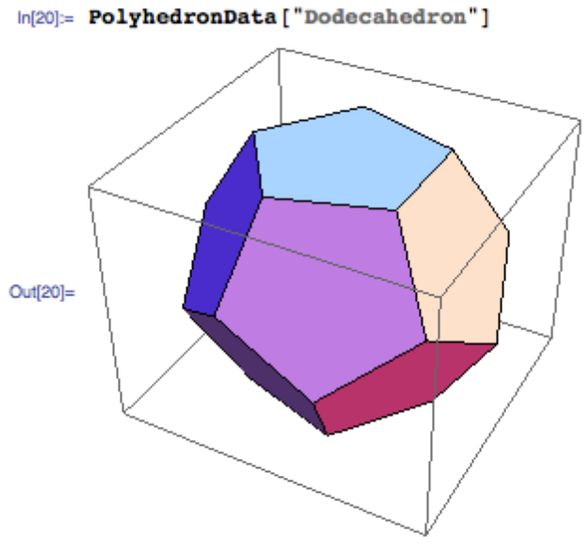
Plots of functions

```
In[19]:= Plot[Log[x], {x, 0, 1000}]
```



Specialized objects

```
In[20]:= PolyhedronData["Dodecahedron"]
```



Getting help

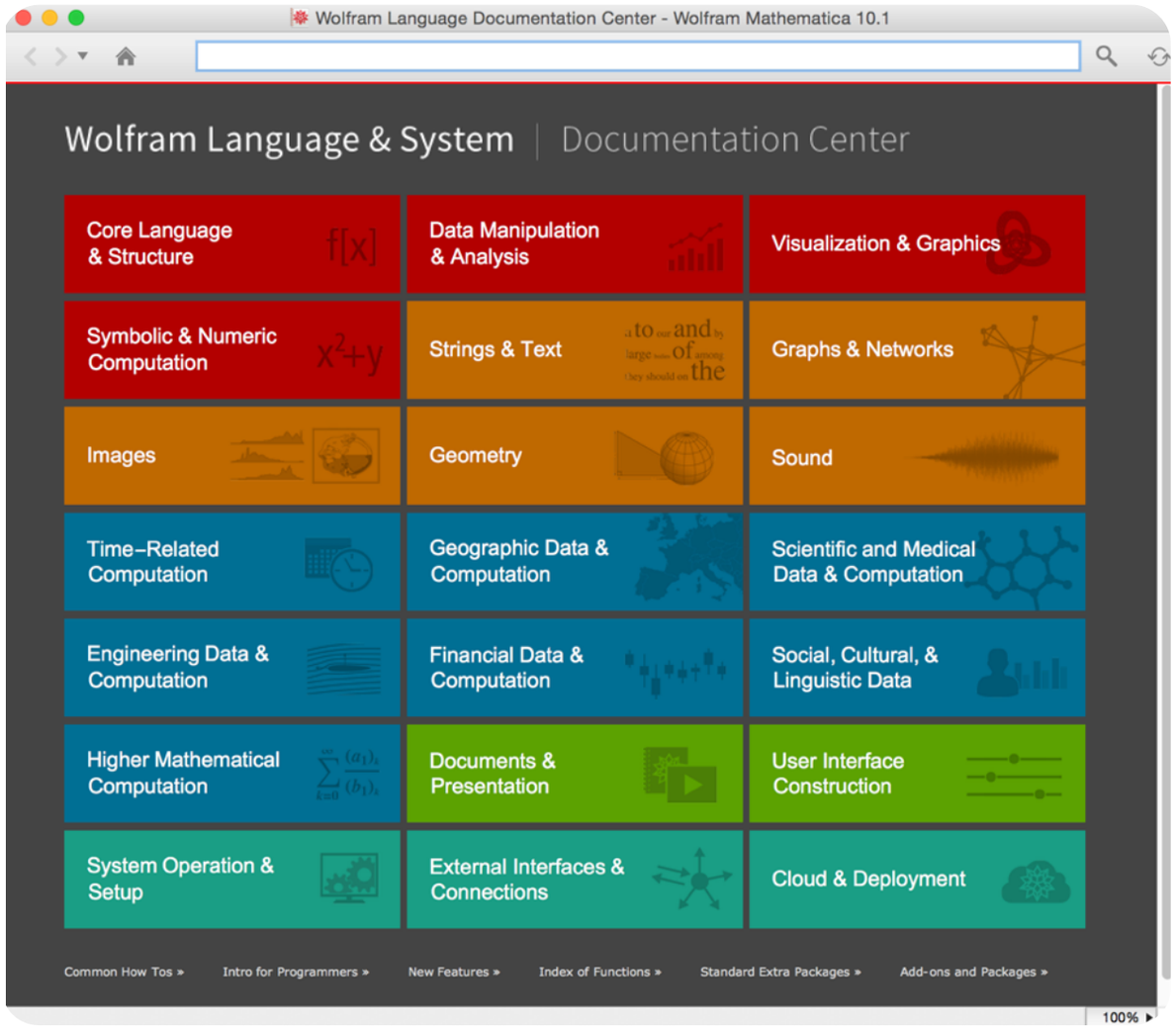
Mathematica help files can be browsed or searched from the Documentation Center of the Help menu

Function names are always made of up complete words, no spaces, with the first word capitalized

Search for functions you hope exist: "*Histogram*", "*LinearRegression*", "*PrincipalComponents*", "*GenomeData*"

Note Function Browser and Mathematica Book help buttons at top left of the Documentation Center

Lynda.com has three useful training courses for *Mathematica 10* from Curt Frye ("Up and Running", "Essential Training", "Advanced Analysis"). Access Lynda through one.iu.edu.



Kernels and Notebooks

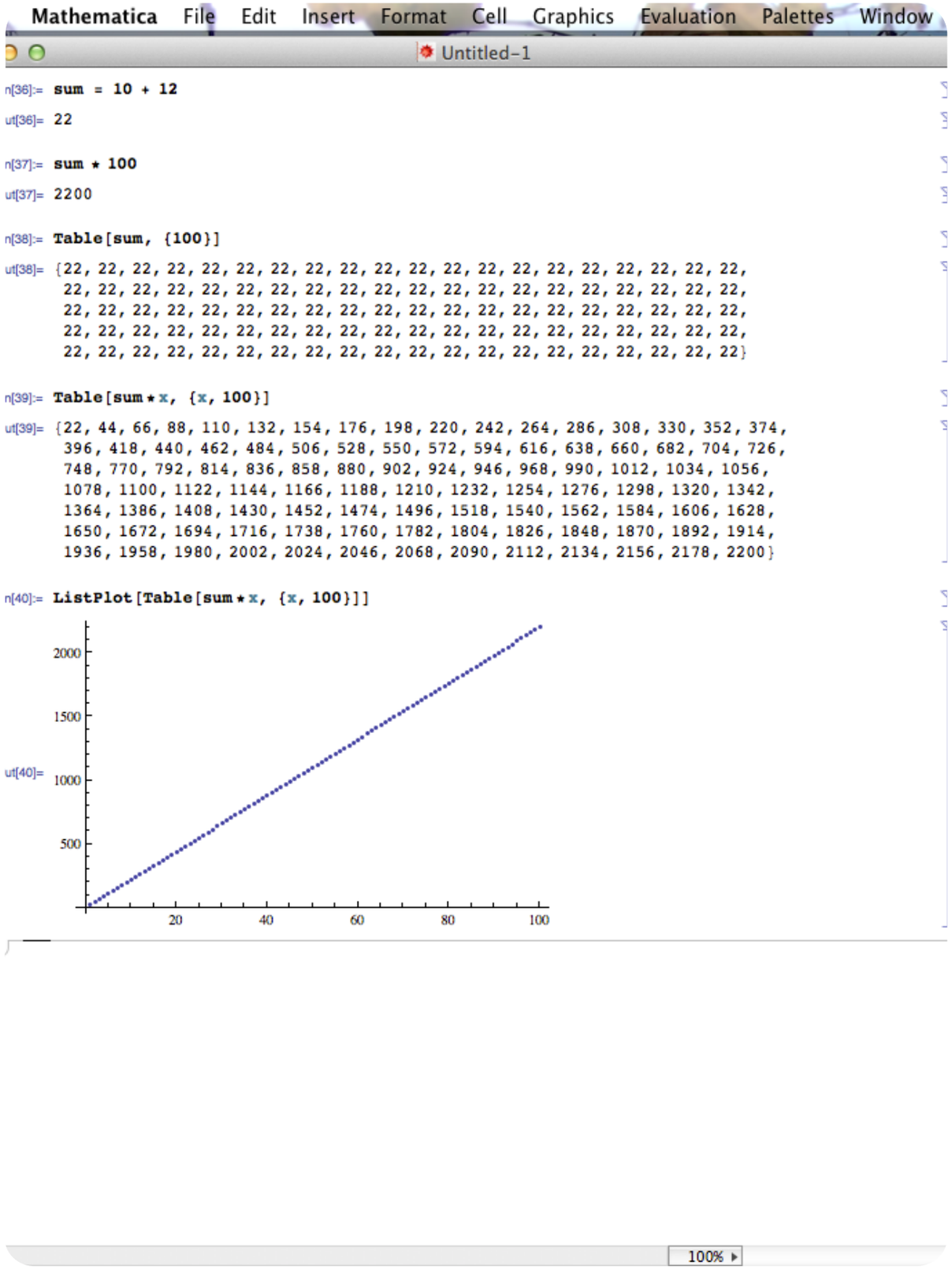
Mathematica has two components, the *kernel* and the *notebook*

The *kernel* is the invisible part of the program that does all the calculations

The *notebook* is the main user interface, its purpose is to allow you to perform analyses and to save them for re-use or for later reference

You can work with many notebooks at once. They share information between them because they interface with the same kernel

For advanced work you can work with two kernels, which allows you to run two sets of calculations in different notebooks at the same time



A typical Mathematica notebook

Notebooks and cells

Notebooks are organized into *cells*

Default cells are for calculations, with input entered by you followed by output created by the kernel

Cells must be executed to obtain output: Shift + Enter to execute

Cells may be executed more than once, and the input can be changed between executions

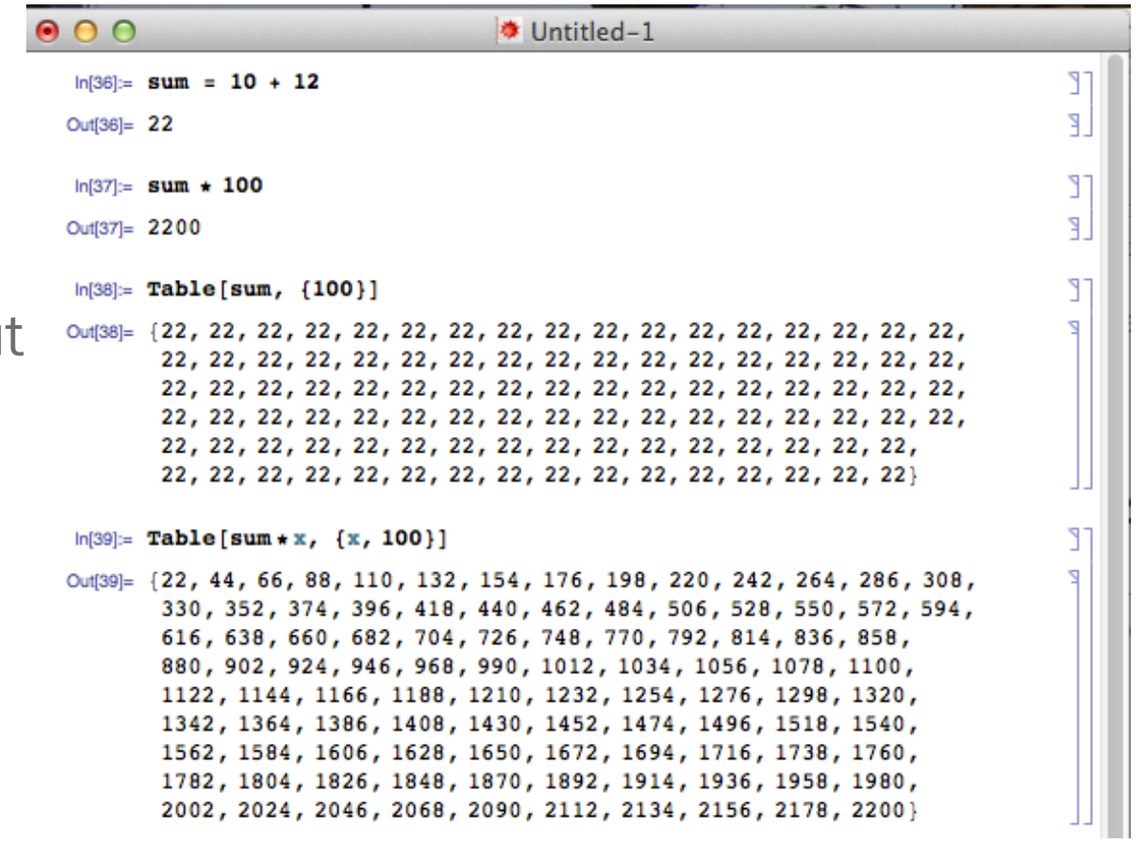
Brackets in the right margin show cell boundaries and distinguish between input and output

Uses for brackets:

1. monitor calculations (bracket is highlighted while the kernel is executing)

2. select entire cell for deletion

3. hide output by double clicking



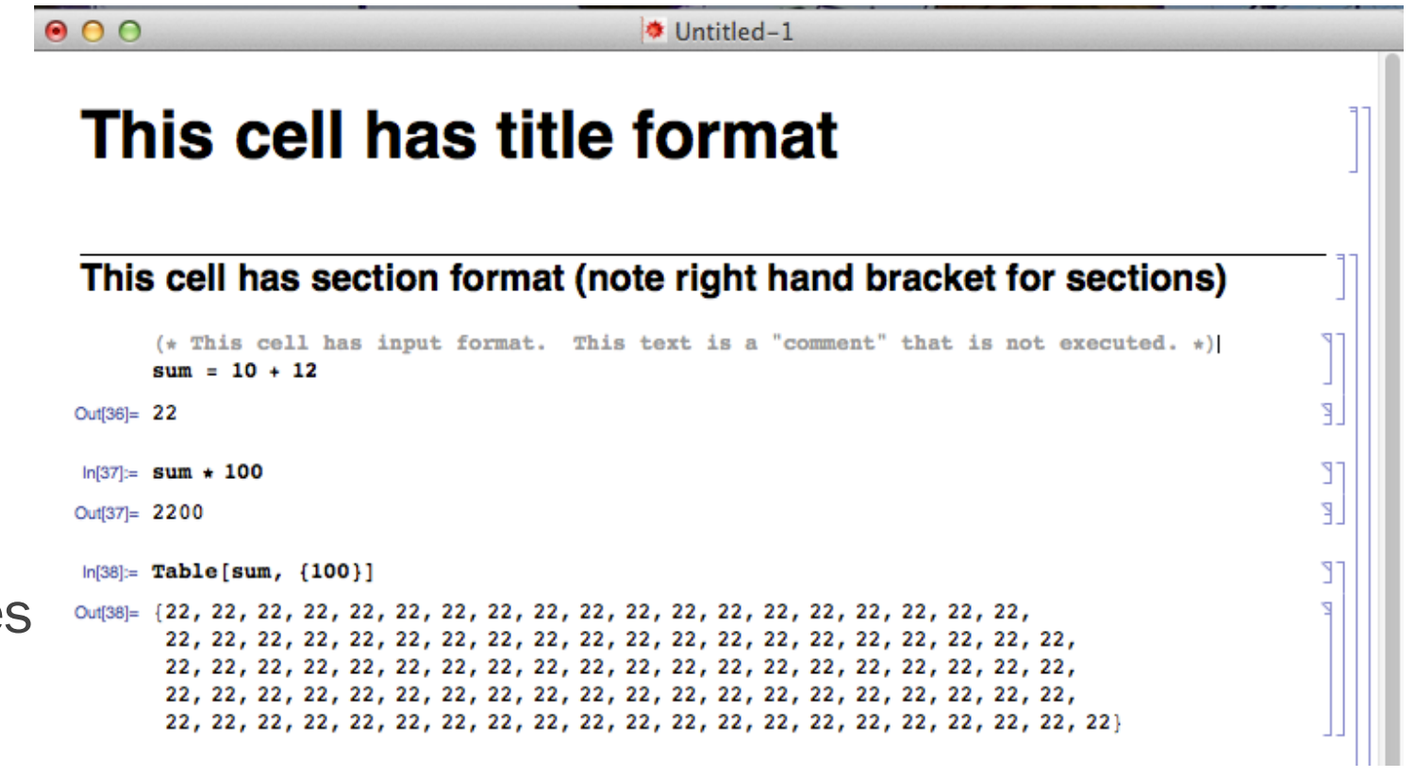
Formatting notebooks

Notebooks can be formatted like a word processor document

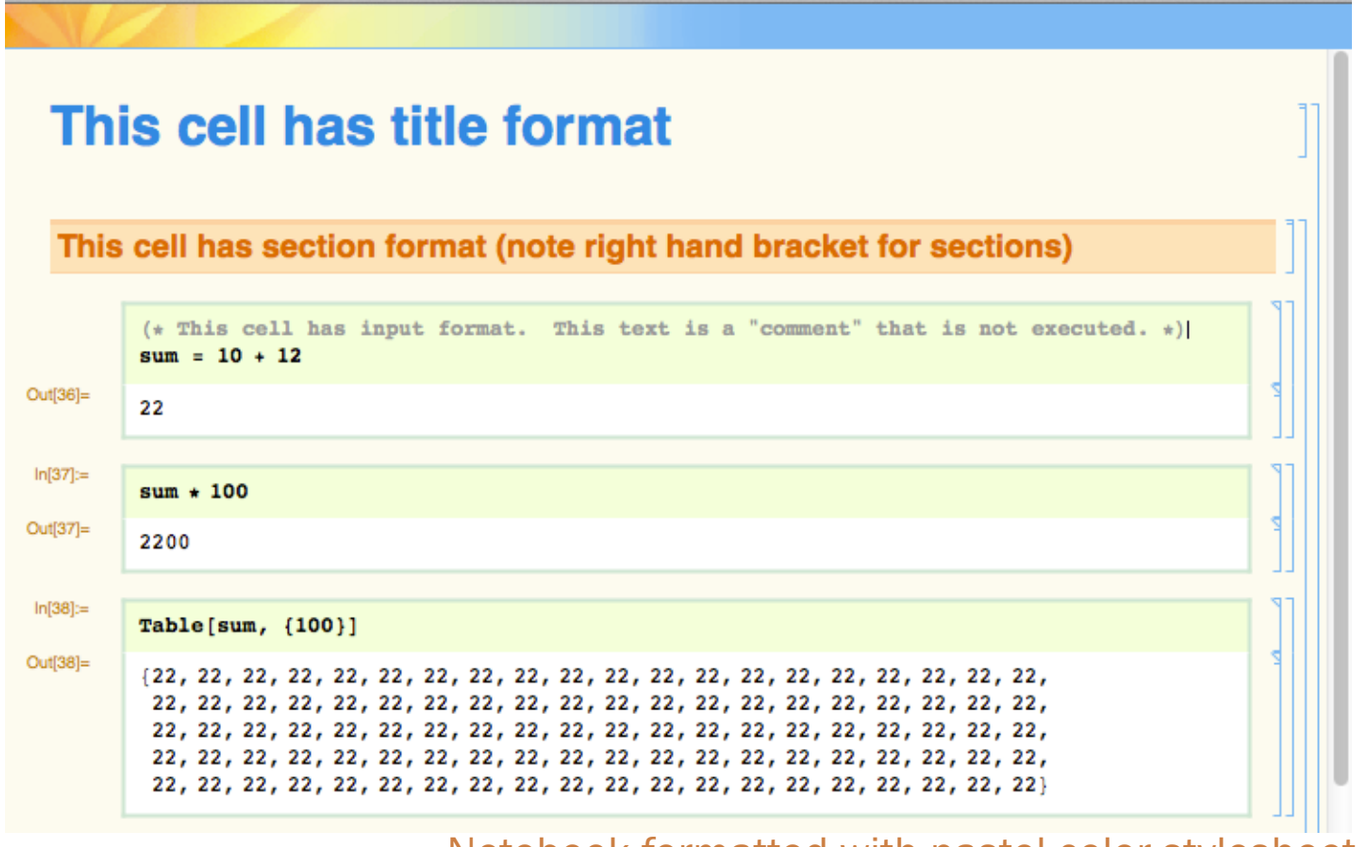
Individual cells can be formatted as titles text, section headings, or input (input is the default)

Use Format | Style menu to format individual cells

Use Format | Stylesheet menu to format the whole notebook



Notebook formatted with default stylesheet



Notebook formatted with pastel color stylesheet

Functions

Functions are key to *Mathematica:* functions receive information or data, process it, and return a result

Functions are called by their name, usually composed of complete English words describing what the function does, with no spaces and first letters capitalized

Function names are followed by square brackets, in which one or more arguments is entered:

FunctionName[argument]

For example, the *ListPlot[]* function takes a matrix of x,y values as its argument:

ListPlot[{{1,2},{3,4}}]

Mathematica's help files give descriptions and examples of every function

Options for functions

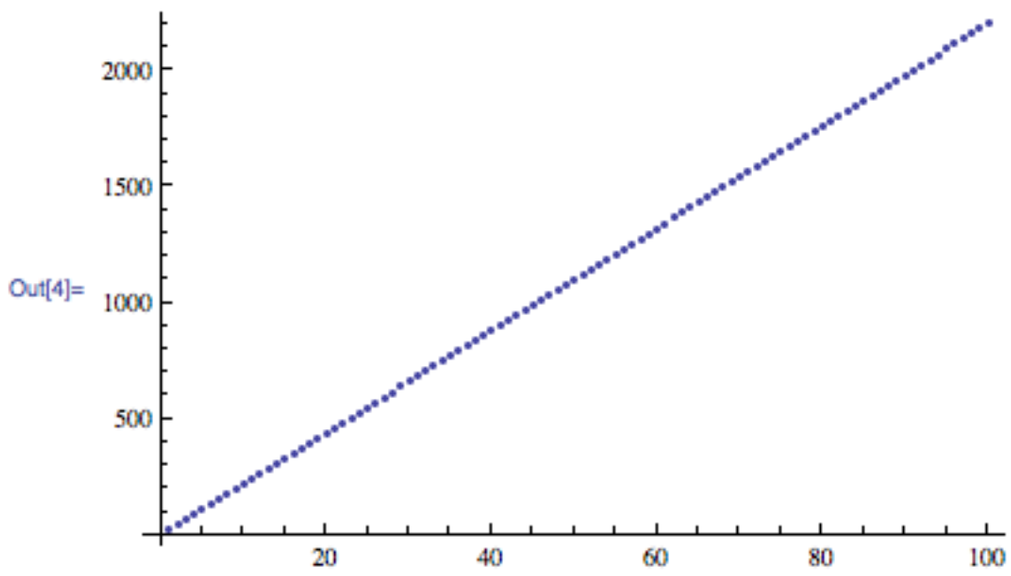
Many functions have options that are entered as arguments

Options usually have the format *OptionName -> Value*

Find options with *Options[FunctionName]* or in Documentation Center

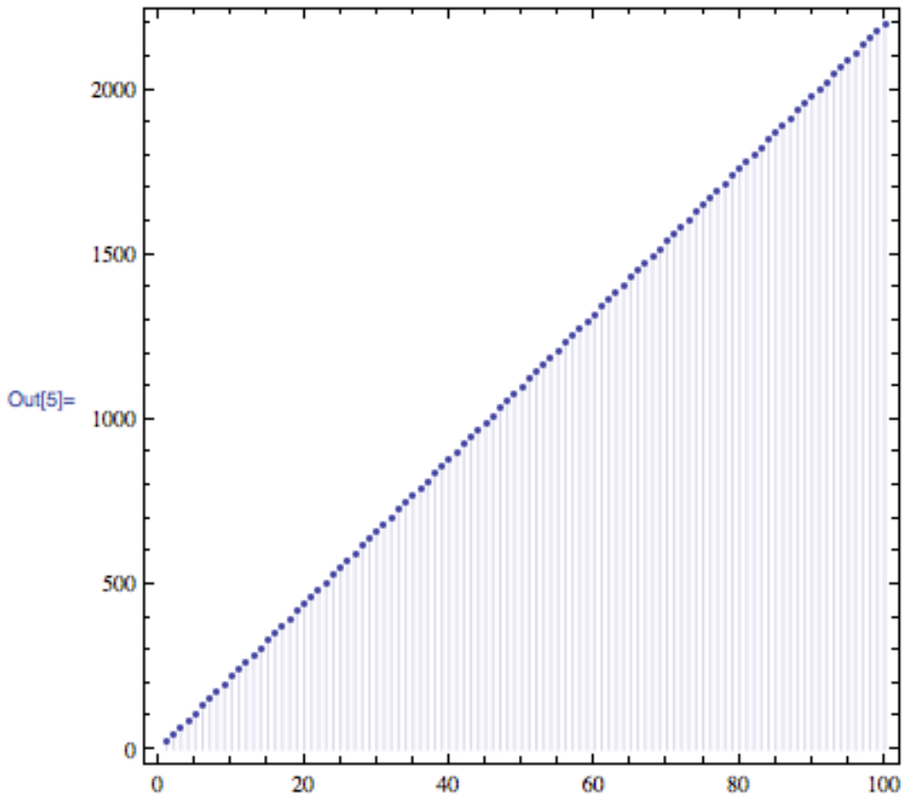
Listplot with no options

```
In[4]:= ListPlot[data]
```



Listplot with three options

```
In[5]:= ListPlot[data, Frame → True, AspectRatio → 1, Filling → Axis]
```



Variables

Variables are also key to *Mathematica*, allowing you to store information

Variables do not have brackets or options

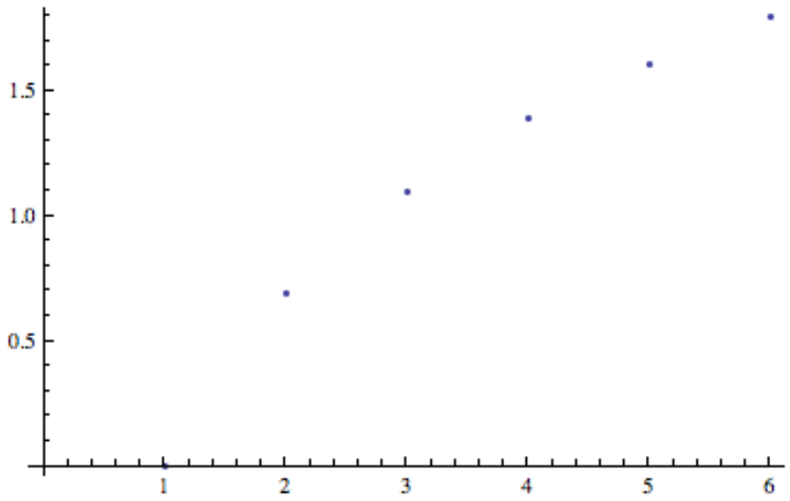
You create variables, giving them a name and putting something into them

Here a variable called *data* is used to store a number, a sequence of numbers, the natural log of a sequence of numbers, and data imported from an Excel file. A variable called *mygraph* is used to store a graphic

You can retrieve what is inside a variable by executing it (the graph is displayed again by executing *mygraph*)

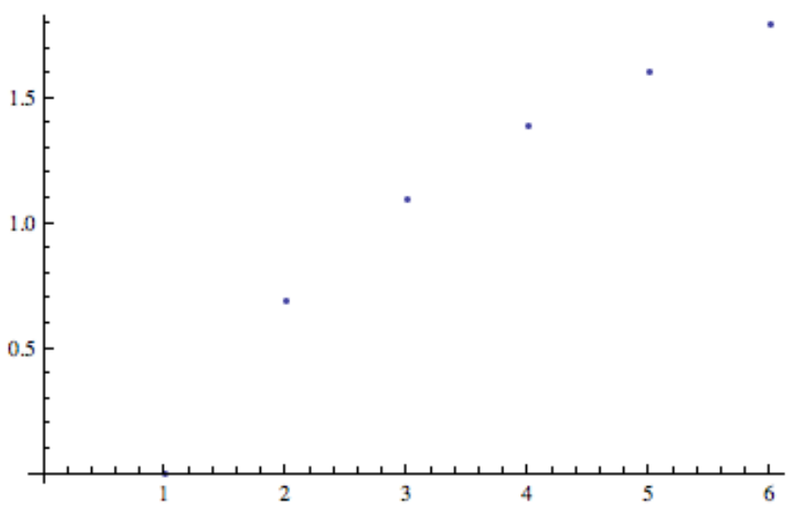
```
In[12]:= data = 1
Out[12]= 1

In[13]:= data = {1, 2, 3, 4, 5, 6}
Out[13]= {1, 2, 3, 4, 5, 6}

In[14]:= data = Log[{1, 2, 3, 4, 5, 6}]
Out[14]= {0, Log[2], Log[3], Log[4], Log[5], Log[6]}

In[15]:= mygraph = ListPlot[data]
```

Out[15]=




```
data = Import["/Users/pdavidpolly/Documents/Stat Data.xls"];

In[16]:= mygraph
```

Out[16]=



Parts of variables

When a variable has more than one item stored, you can get specific parts using double square brackets after the variable name

data returns all the items in *data*

data[[1]] returns only the first item in *data*

data[[1;;3]] returns items 1 to 3

For more examples look at the Documentation Center under the function *Part[]* and under the tutorial *GettingPiecesOfLists*

```
In[22]:= data = {10, 20, 30, 40, 50, 60}
Out[22]= {10, 20, 30, 40, 50, 60}

In[23]:= data
Out[23]= {10, 20, 30, 40, 50, 60}

In[24]:= data[[1]]
Out[24]= 10

In[25]:= data[[3 ;; 5]]
Out[25]= {30, 40, 50}
```

Lists, Matrices, and other Multidimensional data

You will often work with “lists”, which is Mathematica’s term for any group of several items

Some lists have only one element (scalar), some have a long row of elements (vector), some have columns and rows of data (matrix or array)

```
In[12]:= data = 1
Out[12]= 1

In[26]:= data = {10, 20, 30, 40, 50, 60}
Out[26]= {10, 20, 30, 40, 50, 60}

In[27]:= data = {{1, 2}, {1, 3}, {3, 1}}
Out[27]= {{1, 2}, {1, 3}, {3, 1}}
```

You can get columns, rows, or elements from the list using the double square bracket system

```
In[28]:= data = {{{1, 2}, {1, 3}, {3, 1}}, {{1, 2}, {1, 3}, {3, 1}}}
Out[28]= {{{1, 2}, {1, 3}, {3, 1}}, {{1, 2}, {1, 3}, {3, 1}}}

In[30]:= data[[1, 2 ;; 3]]
Out[30]= {{1, 3}, {3, 1}}
```

See Documentation Center under:

1. ListsOverview

2. HandlingArraysOfData

Special formatting tags

You can control the display of output in many ways by putting special tags at the end of a line of input

semicolon (;) prevents output from being displayed

//N forces numbers to be displayed in decimal form

//MatrixForm displays tables of data in rows and columns

```
In[34]:= data = {10, 20, 30, 40, 50, 60}
Out[34]= {10, 20, 30, 40, 50, 60}

In[36]:= data
Out[36]= {10, 20, 30, 40, 50, 60}

In[37]:= data;

In[38]:= data = Log[{1, 2, 3, 4, 5, 6}]
Out[38]= {0, Log[2], Log[3], Log[4], Log[5], Log[6]}

In[39]:= data
Out[39]= {0, Log[2], Log[3], Log[4], Log[5], Log[6]}

In[40]:= data // N
Out[40]= {0., 0.693147, 1.09861, 1.38629, 1.60944, 1.79176}

In[41]:= data = {{1, 2}, {1, 3}, {3, 1}}
Out[41]= {{1, 2}, {1, 3}, {3, 1}}

In[42]:= data
Out[42]= {{1, 2}, {1, 3}, {3, 1}}

In[43]:= data // MatrixForm
Out[43]//MatrixForm=
( 1  2 )
( 1  3 )
( 3  1 )
```

Importing and exporting data

Mathematica has an extensive range of file types that can be imported and exported: text files, Excel files, Word files, PDFs, Illustrator, JPEG, etc.

Import[FilePath]

Export[FilePath, "type"]

Note the helpful file path chooser found on the Insert menu

```
In[49]:= data =
           Import[
            "/Users/pdavidpolly/Documents/Lectures/G562 Geometric Morphometrics/Sample
              Data/CaumulAndPolly2005.xls"];

In[50]:= Export[
          "/Users/pdavidpolly/Documents/Lectures/G562 Geometric Morphometrics/mygraph.pdf",
          "PDF"]

Out[50]= /Users/pdavidpolly/Documents/Lectures/G562 Geometric Morphometrics/mygraph.pdf

In[51]:= mypic =
           Import[
            "/Users/pdavidpolly/Documents/Lectures/G562 Geometric Morphometrics/Example
              Images/JPG/DSCN4141.JPG"];

In[52]:= mypic

Out[52]=
```

Simple graphics

ListPlot[]

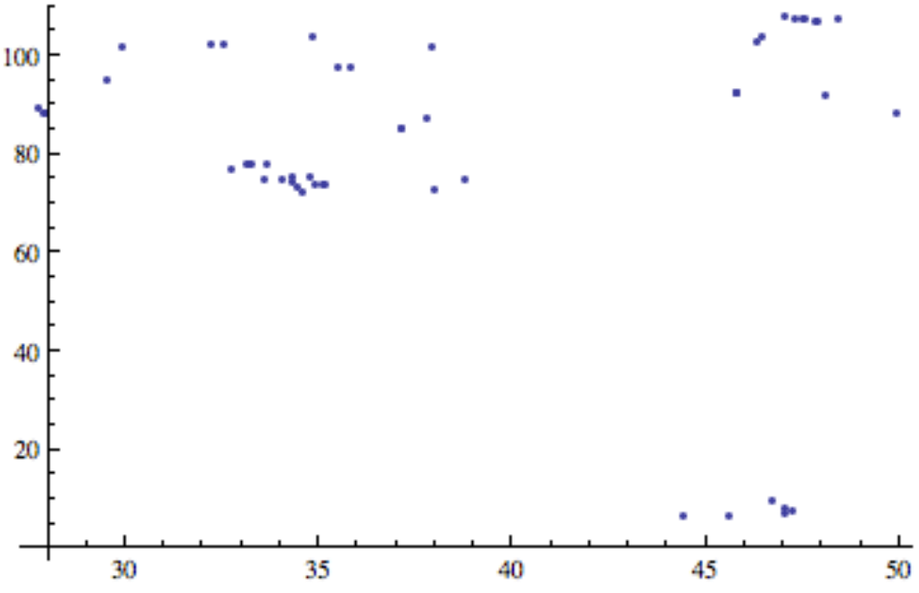
Plot[]

Histogram[]

BarChart[]

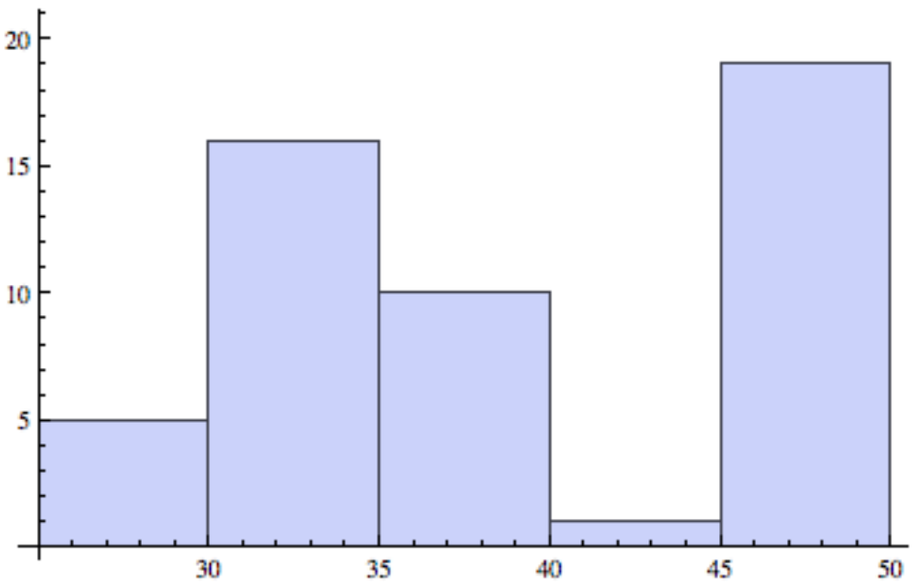
```
In[57]:= ListPlot[data[[1, 2 ;;, 6 ;; 7]]]
```

Out[57]=




```
In[58]:= Histogram[data[[1, 2 ;;, 6]]]
```

Out[58]=



Loops: programming structure for repeating things

Use Table[], Map[], or Do[] to carry out repeated tasks

Table[*lines to be repeated* , *{iterator}]*

where the lines to be repeated consist of other Mathematica functions or lists of functions separated by semicolons

iterator is a special construction that creates a temporary counting variable and specifies number of times to repeat

Simple: {10} (repeats 10 times)
With variable: {x,10} (repeats while incrementing x from 1 to 10 in steps of 1)
Full: {x,1,10,1} (repeats while incrementing x from 1 to 10 in steps of 1)
Full: {x,10,2,-2} (repeats while incrementing x backward from 10 to 2 in steps of 2)

```
In[2]:= Table["hello", {10}]
Out[2]= {hello, hello, hello, hello, hello, hello, hello, hello, hello, hello}

In[3]:= Table[x, {x, 10}]
Out[3]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

In[5]:= Table[x, {x, 1, 10, 1}]
Out[5]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

In[6]:= Table[x, {x, 10, 1, -1}]
Out[6]= {10, 9, 8, 7, 6, 5, 4, 3, 2, 1}

In[7]:= Table[x, {x, 10, 1, -2}]
Out[7]= {10, 8, 6, 4, 2}

In[10]:= min = 7;
         max = 14;
         bin = 2;
         Table[x, {x, min, max, bin}]
Out[13]= {7, 9, 11, 13}
```

Conditional statements

Is equal?	==
Is unequal?	!=
Greater than?	>
Less than?	<
And	&&
Or	||

If[statement is true, then this, or else this]

```
myage = 65.5;
If[ myage > 50, Print[“my age is older”], Print[“my age is not older”]

If[ myage > 55 && myage < 65, Print[“my age is in the bin”], Print[“my age is
outside the bin”]
```

Working with Strings

Strings are entities of characters, as opposed to numbers. You can manipulate strings in Mathematica as well as numbers. For example:

mytext = “Species”;

You can combine strings by joining them with the StringJoin[] function or <> (which do the same thing):

```
In[16]:= StringJoin[mytext, " Name"]
Out[16]= Species Name
```

```
In[17]:= mytext <> " Name"
Out[17]= Species Name
```

You can create a list of labels using Table[] and ToString[], the latter of which converts numbers to strings so they can be joined to other strings:

```
In[18]:= Table[mytext <> " " <> ToString[x], {x, 5}]
Out[18]= {Species 1, Species 2, Species 3, Species 4, Species 5}
```

Random numbers

Mathematica has many functions for generating random numbers.

```
(* Random real number from 0 to 1 *)

In[4]:= RandomReal[]
Out[4]= 0.9513

(* random real number from 100 to 1000 *)

In[5]:= RandomReal[{100, 1000}]
Out[5]= 505.785

(* 10 random real numbers from 100 to 1000 *)

In[10]:= RandomReal[{100, 1000}, 10]
Out[10]= {178.469, 576.318, 234.461, 549.177, 178.544, 581.808, 823.167, 515.409, 828.69, 951.191}

(* Random number drawn from a normal distribution with a mean of 10 and standard
 deviation of 100 *)

In[11]:= Random[NormalDistribution[10, 100]]
Out[11]= 154.744

(* 10 pairs of random numbers between 0 and 1 *)

In[12]:= Table[RandomReal[{0, 1}, 2], {10}]
Out[12]= {{0.0245927, 0.630284}, {0.260035, 0.591502}, {0.38211, 0.146923},
  {0.891077, 0.0315945}, {0.75184, 0.567132}, {0.553506, 0.443656},
  {0.614652, 0.300159}, {0.791076, 0.0654448}, {0.19977, 0.272843}, {0.291167, 0.958036}}
```

```
In[14]:= ListPlot[Table[RandomReal[{0, 1}, 2], {1000}]]
```

Out[14]=

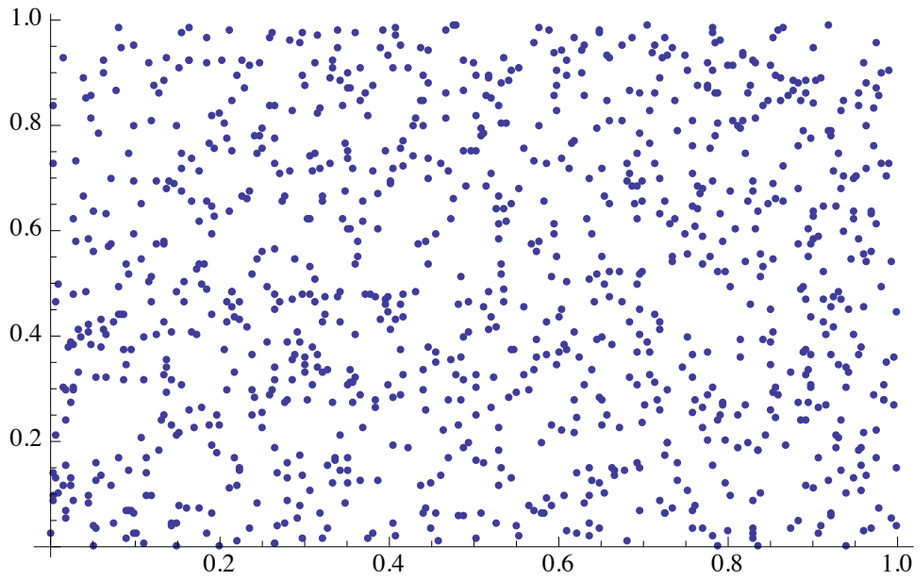


Photo credit

Defining your own function

You can create your own customized functions to perform operations that you use a lot.

The syntax uses ":=" to define the operation of the function.

The input parameters are defined as variables with an underscore after them.

The *Module* function shields the variables used in the custom function from the rest of the notebook (it keeps them from clashing).

Custom functions usually end with *Return*, which is a function that returns something to the user in response to the input parameters.

Module function (closes after the Return function)

Function name

Input parameters

Internal variables

```
In[17]:= MyFunction[x_, y_] := Module[{i, j},
           i = 10;
           j = i * (x + y);
           Return[j];
          ]

In[18]:= MyFunction[1, 2]

Out[18]= 30
```

This example takes two numbers as input, adds them together and multiplies them by 10, and stores the result in the temporary internal variable *j*. The value is returned to the user at the end of the function.