New Features of *Mathematica* 7

Rolf Mertig
rolfm@gluonvision.com

Outline

- General News
- Mathematics and Algorithm News
- Core news
- Parallel Computing with Mathematica
- Accessing Integrated Data Sources
- Final Thoughts

General News

Kernel & FrontEnd changes

New Help

New Palettes

New Functions and Options

```
Length[Names["System`*"]]
```

3432

In *Mathematica* 6.0.3: 2959

Mathematics and Algorithm News

Product

```
? Product
```

Product[f, {i, i_{max}}] evaluates the product $\prod_{i=1}^{i_{max}} f$.

Product[f, {i, i_{min}, i_{max}}] starts with $i = i_{min}$.
Product[f, {i, i_{min}, i_{max}, di}] uses steps di.
Product[f, {i, {i_1, i_2, ...}}] uses successive values i_1, i_2, ...

Product[f, {i, i_{min}, i_{max}}, {j, j_{min}, j_{max}}, ...] evaluates the multiple product $\prod_{i=i_{min}}^{i_{max}} \prod_{j=j_{min}}^{j_{max}} \ldots f$.

Product[f, i] gives the indefinite product $\prod_i f$. ≫

```
Product[g[i], i]
```

$$\prod_{i} g[i]$$

DiscreteRatio is the inverse for indefinite products:

```
DiscreteRatio[Product[f[i], i], i]
```

```
f[i]
```

```
Product[DiscreteRatio[f[i], i], i]
```

```
f[i]
```

Sum

```
? Sum
```

Sum[f, {i, i_{max}}] evaluates the sum $\sum_{i=1}^{i_{max}} f$.

Sum[f, {i, i_{min}, i_{max}}] starts with $i = i_{min}$.
Sum[f, {i, i_{min}, i_{max}, di}] uses steps di.
Sum[f, {i, {i_1, i_2, ...}}] uses successive values i_1, i_2, … .

Sum[f, {i, i_{min}, i_{max}}, {j, j_{min}, j_{max}}, ...] evaluates the multiple sum $\sum_{i=i_{min}}^{i_{max}} \sum_{j=j_{min}}^{j_{max}} \ldots f$.

Sum[f, i] gives the indefinite sum $\sum_i f$. ≫

```
Sum[i^3, i]
```

$$\frac{1}{4}(-1+i)^2 i^2$$

$$\sum_{i=1}^{n} \frac{1}{i^r}$$

```
HarmonicNumber[n, r]
```

```
Sum[n^2 HarmonicNumber[n], {n, m}]
```

$$\frac{1}{36}\left(-5\,m-9\,m^2-4\,m^3+6\,m\,\text{HarmonicNumber}[1+m]+18\,m^2\,\text{HarmonicNumber}[1+m]+12\,m^3\,\text{HarmonicNumber}[1+m]\right)$$

```
f[m_] = Sum[n^-2 HarmonicNumber[n], {n, m}]
```

```
DifferenceRoot[Function[{ẏ, ṅ},
  {-ṅ^2 (1 + ṅ) ẏ[ṅ] + (1 + ṅ) (3 + 5 ṅ + 3 ṅ^2) ẏ[1 + ṅ] + (-11 - 20 ṅ - 13 ṅ^2 - 3 ṅ^3) ẏ[2 + ṅ] +
     (2 + ṅ)^3 ẏ[3 + ṅ] == 0, ẏ[1] == 0, ẏ[2] == 1, ẏ[3] == 11/8}]][1 + m]
```

```
f[3]
```

$$\frac{341}{216}$$

DifferenceRoot

```
? DifferenceRoot
```

DifferenceRoot[*lde*] represents a function that
solves the linear difference equation specified by *lde*[*a*, *n*]. ≫

Define Pell numbers:

```
PellNumber =
  DifferenceRoot[Function[{y, n}, {-y[n] - 2 y[1 + n] + y[2 + n] == 0, y[0] == 0, y[1] == 1}]];

Table[PellNumber[n], {n, 0, 10}]
```

```
{0, 1, 2, 5, 12, 29, 70, 169, 408, 985, 2378}
```

DifferenceDelta, DiscreteShift,DiscreteRatio

```
? DifferenceDelta
```

DifferenceDelta[*f*, *i*] gives the discrete difference $\triangle_i f = f(i+1) - f(i)$.
DifferenceDelta[*f*, {*i*, *n*}] gives the multiple difference $\triangle_i^n(f)$.
DifferenceDelta[*f*, {*i*, *n*, *h*}] gives the multiple difference with step *h*.
DifferenceDelta[*f*, *i*, *j*, …]
computes the partial difference with respect to *i*, *j*, … . ≫

```
? DiscreteShift
```

DiscreteShift[*f*, *i*] gives the discrete shift $E_i(f(i)) = f(i+1)$.
DiscreteShift[*f*, {*i*, *n*}] gives the multiple shift $E_i^n f$.
DiscreteShift[*f*, {*i*, *n*, *h*}] gives the multiple shift of step *h*.
DiscreteShift[*f*, *i*, *j*, ...] computes partial shifts with respect to *i*, *j*, ... ≫

```
? DiscreteRatio
```

DiscreteRatio[*f*, *i*] gives the discrete ratio $\frac{f(i+1)}{f(i)}$.
DiscreteRatio[*f*, {*i*, *n*}] gives the multiple discrete ratio.
DiscreteRatio[*f*, {*i*, *n*, *h*}] gives the multiple discrete ratio with step *h*.
DiscreteRatio[*f*, *i*, *j*, ...] computes
the partial difference ratio with respect to *i*, *j*, ... ≫

GeneratingFunction

```
? GeneratingFunction
```

GeneratingFunction[*expr*, *n*, *x*] gives the generating function in *x* for the
sequence whose n^{th} series coefficient is given by the expression *expr*.
GeneratingFunction[*expr*, {n_1, n_2, … }, {x_1, x_2, … }] gives the multidimensional
generating function in x_1, x_2, … whose n_1, n_2, … coefficient is given by *expr*. ≫

The generating function for the sequence whose n^{th} term is 1:

```
GeneratingFunction[1, n, x]
```

$$\frac{1}{1-x}$$

All coefficients in the series are 1:

```
Series[%, {x, 0, 10}]
```

$$1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{10} + O[x]^{11}$$

Univariate generating function:

```
GeneratingFunction[1/n!^2, n, x]
```

```
BesselI[0, 2 Sqrt[x]]
```

Multivariate:

```
GeneratingFunction[1/((n+1)! m!), {n, m}, {x, y}]
```

$$\frac{e^y\,(-1+e^x)}{x}$$

SumConvergence

```
? SumConvergence
```

SumConvergence[f, n] gives conditions for the sum $\sum_{n}^{\infty} f$ to be convergent.

SumConvergence[f, {n_1, n_2, ...}] gives

conditions for the multiple sum $\sum_{n_1}^{\infty}\sum_{n_2}^{\infty}\ldots f$ to be convergent. ≫

```
Options@SumConvergence
```

```
{Method → Automatic, Assumptions :→ $Assumptions, Direction → 1}
```

Basic Examples (2)

Test for convergence of the sum $\sum_n^\infty \frac{1}{n}$:

```
SumConvergence[1 / n, n]
```

```
False
```

Test the convergence of $\sum_n^\infty \frac{3^n n^2}{n!}$:

```
SumConvergence[3^n n^2 / n!, n]
```

```
True
```

Find the condition for convergence of $\sum_n^\infty \frac{1}{n^\alpha}$:

```
SumConvergence[1 / n^α, n]
```

```
Re[α] > 1
```

Casoratian

```
? Casoratian
```

Casoratian[{y_1, y_2, ...}, n] gives the
Casoratian determinant for the sequences y_1, y_2, ... depending on n.
Casoratian[eqn, y, n] gives the Casoratian determinant for the basis of the
solutions of the linear difference equation eqn involving y[n + m].
Casoratian[eqns, {y_1, y_2, ...}, n] gives the Casoratian determinant
for the system of linear difference equations eqns. ≫

- The Casoratian determinant is defined as: Det[Table[DiscreteShift[y_i, {n, j}], {i, m}, {j, 0, m - 1}]].
- Linear independence of the functions y_1, y_2, ... is equivalent to the vanishing of the Casoratian.

The following sequences are linearly dependent:

```
Casoratian[{2^n, 2^(n + 3)}, n]
```

```
0
```

ZTransform

```
? ZTransform
```

ZTransform[*expr*, *n*, *z*] gives the Z transform of *expr*.
ZTransform[*expr*, {n_1, n_2, ...}, {z_1, z_2, ...}]
gives the multidimensional Z transform of *expr*. ≫

FindSequenceFunction

```
? FindSequenceFunction
```

FindSequenceFunction[{a_1, a_2, a_3, ...}] attempts to find a simple function
that yields the sequence a_n when given successive integer arguments.
FindSequenceFunction[{{n_1, a_1}, {n_2, a_2}, ...}] attempts to find a
simple function that yields a_i when given argument n_i.
FindSequenceFunction[*list*, *n*] gives the function applied to *n*. ≫

```
FindSequenceFunction[
 {1, 2, 3, 5, 17, 305, 34 865, 24 918 065, 125 436 246 065, 5 056 710 181 206 065}, n]
```

$$1 + \sum_{K[1]=1}^{-1+n} \text{BarnesG}[1 + K[1]]$$

New Special Functions »

AngerJ ▪ WeberE ▪ DawsonF ▪ BarnesG ▪ LogBarnesG

HurwitzZeta ▪ HurwitzLerchPhi ▪ Haversine ▪ InverseHaversine

Core News

Gather

```
? Gather
```

Gather[*list*] gathers the elements of *list* into sublists of identical elements.
Gather[*list*, *test*] applies *test* to pairs of
elements to determine if they should be considered identical. ≫

```
Gather[{a, b, a, c, d, c}]
```

```
{{a, a}, {b}, {c, c}, {d}}
```

Gather elements that have equal integer parts:

```
Gather[Range[0, 3, 1 / 3], Floor[#1] == Floor[#2] &]
```

$$\left\{\left\{0, \frac{1}{3}, \frac{2}{3}\right\}, \left\{1, \frac{4}{3}, \frac{5}{3}\right\}, \left\{2, \frac{7}{3}, \frac{8}{3}\right\}, \{3\}\right\}$$

GatherBy

```
? GatherBy
```

GatherBy[*list*, *f*] gathers into sublists each set
of elements in *list* that gives the same value when *f* is applied.
GatherBy[*list*, {f_1, f_2, ...}] gathers *list* into nested sublists using f_i at level i. ≫

```
GatherBy[{1, 2, 3, 4, 5}, OddQ]
```

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

```
GatherBy[{a, b, c, d, e, d, a}]
```

```
{{a, a}, {b}, {c}, {d, d}, {e}}
```

```
data = Transpose[{RandomReal[5, 10], RandomChoice[{a, b, c}, 10], RandomReal[10, 10]}]
```

```
{{3.006, c, 4.20132}, {1.02151, b, 1.86177}, {4.60925, a, 1.24528}, {1.93872, a, 4.66082},
 {2.54989, b, 7.28555}, {4.15773, b, 8.55556}, {4.7963, c, 1.09682},
 {4.37411, b, 5.60706}, {0.468551, c, 7.62877}, {1.52394, a, 4.66188}}
```

Group elements based on the value of the second element:

```
gathered = GatherBy[data, #[[2]] &]
```

```
{{{3.006, c, 4.20132}, {4.7963, c, 1.09682}, {0.468551, c, 7.62877}}, {{1.02151, b, 1.86177},
  {2.54989, b, 7.28555}, {4.15773, b, 8.55556}, {4.37411, b, 5.60706}},
 {{4.60925, a, 1.24528}, {1.93872, a, 4.66082}, {1.52394, a, 4.66188}}}
```

SplitBy

```
Tuples[{1, 2}, 3]
```

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

```
SplitBy[%, First]
```

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

DeleteDuplicates

```
? DeleteDuplicates
```

DeleteDuplicates[*list*] deletes all duplicates from *list*.
DeleteDuplicates[*list*, *test*] applies *test* to pairs of
elements to determine whether they should be considered duplicates. ≫

```
DeleteDuplicates[{1, 7, 8, 4, 3, 4, 1, 9, 9, 2}]
```

```
{1, 7, 8, 4, 3, 9, 2}
```

ArrayPad

```
? ArrayPad
```

ArrayPad[*array*, *m*] gives an array with *m* 0s of padding on every side.
ArrayPad[*array*, *m*, *padding*] uses the specified padding.
ArrayPad[*array*, {*m*, *n*}, ...] pads
with *m* elements at the beginning and *n* elements at the end.
ArrayPad[*array*, {{m_1, n_1}, {m_2, n_2}, ...}, ...] pads with m_i, n_i elements at level *i* in *array*. ≫

```
ArrayPad[{a, b, c, d}, 1]
```

```
{0, a, b, c, d, 0}
```

```
ArrayPad[{{1, 2}, {3, 4}}, 2] // MatrixForm
```

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 3 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Ratios

```
? Ratios
```

Ratios[*list*] gives the successive ratios of elements in *list*.
Ratios[*list*, *n*] gives the n^{th} iterated ratios of *list*.
Ratios[*list*, {n_1, n_2, ...}] gives
the successive n_k^{th} ratios at level *k* in a nested list. ≫

```
Ratios[{a, b, c, d, e}]
```

$$\left\{\frac{b}{a}, \frac{c}{b}, \frac{d}{c}, \frac{e}{d}\right\}$$

Second ratios:

```
Ratios[{a, b, c, d, e}, 2]
```

$$\left\{\frac{a\,c}{b^2}, \frac{b\,d}{c^2}, \frac{c\,e}{d^2}\right\}$$

Quiet

```
Quiet[1 / 0]
```

```
ComplexInfinity
```

`Quiet[expr, "name"]`
quietens only the named group of messages.

$MessageGroups

```
$MessageGroups
```

```
{Compiler :> {Compile::cif, Compile::cpapot, Compile::cpbool, Compile::cpdsts,
   Compile::cpint, Compile::cpintlt, Compile::cpintlt2, Compile::cpiter,
   Compile::cplist, Compile::cpout, Compile::cppat, Compile::cprank, Compile::cpts,
   Compile::cptype, Compile::cret, Compile::nocomp, Compile::part, Compile::cxcoerce,
   Compile::realcoerce, CompiledFunction::cfex, CompiledFunction::cflist,
   CompiledFunction::cfn, CompiledFunction::cfnlts, CompiledFunction::orank,
   CompiledFunction::cfsa, CompiledFunction::cfse, CompiledFunction::cfta,
   CompiledFunction::cfte, CompiledFunction::cfsec, CompiledFunction::cftec},
 Graphics :> {General::plnr, General::pp3tr, General::pptr, General::pwfailed,
   General::nfunfail, Compiled::deprec, ContourPlot3D::njnum,
   ParametricPlot3D::njnum, Plot3D::njnum, RegionPlot3D::njnum,
   General::accbend, ContourPlot::accbend, ContourPlot3D::accbend,
   DensityPlot::accbend, LogLinearPlot::accbend, LogLogPlot::accbend,
   LogPlot::accbend, ParametricPlot::accbend, ParametricPlot3D::accbend,
   Plot::accbend, Plot3D::accbend, PolarPlot::accbend, RegionPlot3D::accbend},
 Obsolete :> {General::obs, General::obspkg, General::newpkg},
 Packing :> {General::unpack, General::unpack1, General::punpack, General::punpack1},
 Spelling :> {General::spell, General::spell1},
 Symbolics :> {Integrate::gener, Maximize::nopmet, Minimize::nopmet, Series::esss}}
```

```
$MessageGroups[[All, 1]]
```

```
{Compiler, Graphics, Obsolete, Packing, Spelling, Symbolics}
```

Applications :

Check for unpacking of arrays:

```
On["Packing"]
```

This produces rational numbers and has to unpack:

```
Range[5] / 2
```

```
Developer`FromPackedArray::unpack1 : Unpacking array with dimensions {5}. >>
```

$$\left\{\frac{1}{2}, 1, \frac{3}{2}, 2, \frac{5}{2}\right\}$$

Using inexact numbers leaves the array packed:

```
Range[5] / 2.
```

```
{0.5, 1., 1.5, 2., 2.5}
```

```
Off["Packing"]
```

Quiet the compiler warnings that allow the evaluation to proceed:

```
Quiet[Compile[{x}, Log[10., x]][1.*^1000], "Compiler"]
```

```
1000.
```

Parallel Computing with *Mathematica*

Mathematica 7 features built-in parallel computing capabilities.

Parallelize

Parallelize will automatically parallelize a computation, if possible.

```
Parallelize[Sum[i!, {i, 250}]]
```

```
3 245 839 829 646 932 305 706 781 334 085 387 423 045 886 707 102 864 239 947 155 598 639 623 741 978 145 ⁘
  433 246 772 997 970 167 407 723 040 950 499 976 939 221 384 174 708 291 790 681 731 284 266 065 051 411 ⁘
  141 491 719 683 469 842 975 613 108 179 345 674 527 330 409 620 311 188 916 439 014 240 407 034 202 858 ⁘
  900 618 509 602 850 024 516 214 062 716 639 508 513 531 377 860 650 846 620 316 252 734 419 111 128 139 ⁘
  999 992 301 626 150 519 817 601 422 193 407 122 960 285 162 792 537 296 857 574 089 249 364 533 084 506 ⁘
  365 165 064 392 031 273 304 659 518 229 406 508 273 079 478 684 738 621 107 994 804 323 593 105 039 052 ⁘
  556 442 336 528 920 420 940 313
```

Otherwise, it will perform a serial computation.

```
Parallelize[Integrate[(1 + x^4)/(x^9 + 3 x), dx]]
```

```
Parallelize::nopar1 :
  ∫(1 + x^4)/(x^9 + 3 x) dx cannot be parallelized; proceeding with sequential evaluation. >>
```

$$\frac{1}{24}\left(-2\sqrt{3}\,\text{ArcTan}\left[\text{Cot}\left[\frac{\pi}{8}\right]-\frac{x\,\text{Csc}\left[\frac{\pi}{8}\right]}{3^{1/8}}\right]-2\sqrt{3}\,\text{ArcTan}\left[\text{Cot}\left[\frac{\pi}{8}\right]+\frac{x\,\text{Csc}\left[\frac{\pi}{8}\right]}{3^{1/8}}\right]-2\sqrt{3}\,\text{ArcTan}\left[\frac{x\,\text{Sec}\left[\frac{\pi}{8}\right]}{3^{1/8}}-\text{Tan}\left[\frac{\pi}{8}\right]\right]+2\sqrt{3}\,\text{ArcTan}\left[\frac{x\,\text{Sec}\left[\frac{\pi}{8}\right]}{3^{1/8}}+\text{Tan}\left[\frac{\pi}{8}\right]\right]+8\,\text{Log}[x]-\text{Log}\left[3+x^8\right]\right)$$

Parallel Computing with *Mathematica*

Scalable parallelization

Parallel computing scales across your available local cores or remote cores. Let's take six really large numbers.

```
nums = {2 949 139 351 597 575 118 259 944 040 385 088 406 471 149 304 313 950 803 916 170,
    2 745 201 680 660 241 417 682 118 990 009 429 336 956 428 174 177 365 148 868 001,
    2 233 795 066 652 699 157 538 868 797 967 201 925 661 196 164 476 818 038 296 004,
    1 811 407 300 094 750 906 313 647 154 418 422 327 145 787 739 155 366 439 445 521,
    2 945 655 125 896 943 888 734 546 306 600 095 101 787 359 723 357 126 960 817 429,
    1 993 289 517 437 145 602 616 895 020 850 006 297 493 179 390 622 764 388 547 884};
```

And now we'll ask *Mathematica* to factor them using two different methods:

1. Serial processing.
2. Parallel processing.

```
Map[FactorInteger, nums] // AbsoluteTiming
```

```
{11.962311, {{{2, 1}, {5, 1}, {19, 1}, {31, 1}, {3461, 1}, {16 993, 1}, {57 641, 1}, {69 911, 1},
    {339 527, 1}, {1 521 293, 1}, {2 765 123, 1}, {6 660 396 343, 1}, {222 090 310 009, 1}},
  {{7, 1}, {10 889 738 124 169, 1}, {23 140 659 239 483 475 743, 1},
    {1 556 263 217 511 405 712 375 840 129, 1}}, {{2, 2}, {311, 1}, {14 923, 1}, {20 129, 1},
    {2 834 297 671, 1}, {2 109 109 599 418 258 809 392 701 205 440 382 828 563, 1}}, {{3, 1}, {467, 1},
    {3 274 151, 1}, {60 982 081 317 767, 1}, {6 475 555 514 768 899 485 570 247 647 000 461 513, 1}},
  {{3, 3}, {19, 1}, {15 727, 1}, {34 897, 1}, {1 863 413 972 883 765 097, 1},
    {5 614 630 825 010 234 162 074 522 774 931, 1}}, {{2, 2}, {31, 1}, {443, 1},
    {16 037 301 773, 1}, {2 262 630 671 992 589 783 347 826 982 337 592 338 627 672 619, 1}}}}
```

```
Parallelize[Map[FactorInteger, nums]] // AbsoluteTiming
```

```
{30.088362, {{{2, 1}, {5, 1}, {19, 1}, {31, 1}, {3461, 1}, {16 993, 1}, {57 641, 1}, {69 911, 1},
    {339 527, 1}, {1 521 293, 1}, {2 765 123, 1}, {6 660 396 343, 1}, {222 090 310 009, 1}},
  {{7, 1}, {10 889 738 124 169, 1}, {23 140 659 239 483 475 743, 1},
    {1 556 263 217 511 405 712 375 840 129, 1}}, {{2, 2}, {311, 1}, {14 923, 1}, {20 129, 1},
    {2 834 297 671, 1}, {2 109 109 599 418 258 809 392 701 205 440 382 828 563, 1}}, {{3, 1}, {467, 1},
    {3 274 151, 1}, {60 982 081 317 767, 1}, {6 475 555 514 768 899 485 570 247 647 000 461 513, 1}},
  {{3, 3}, {19, 1}, {15 727, 1}, {34 897, 1}, {1 863 413 972 883 765 097, 1},
    {5 614 630 825 010 234 162 074 522 774 931, 1}}, {{2, 2}, {31, 1}, {443, 1},
    {16 037 301 773, 1}, {2 262 630 671 992 589 783 347 826 982 337 592 338 627 672 619, 1}}}}
```

```
Grid[{
  {Style["# of cores", FontFamily → "Arial", Bold],
   Style["Time elapsed", FontFamily → "Arial", Bold]},
  {Style["1", FontFamily → "Arial"], Style["17.9487 sec", FontFamily → "Arial"]},
  {Style["2", FontFamily → "Arial"], Style["14.1600 sec", FontFamily → "Arial"]},
  {Style["4", FontFamily → "Arial"], Style["xx.xxxx sec", FontFamily → "Arial"]},
  {Style["8", FontFamily → "Arial"], Style["xx.xxxx sec", FontFamily → "Arial"]},
  {Style["16", FontFamily → "Arial"], Style["xx.xxxx sec", FontFamily → "Arial"]}

 },
 Dividers → {None, {2 → Black}}, Frame → {{11, 2} → True},
 Alignment → {{Left, Right}}]
```

# of cores	Time elapsed
1	17.9487 sec
2	14.1600 sec
4	xx.xxxx sec
8	xx.xxxx sec
16	xx.xxxx sec

Parallel Computing with *Mathematica*

ParallelTry

We can use ParallelTry to send multiple evaluations to kernels; this is useful when you want to pick a particular type of method,
for example, but don't know which is the most efficient.

```
AbsoluteTiming[ParallelTry[
  {#, FindMinimum[e^Sin[50 x] + Sin[60 e^y] + Sin[70 Sin[x]] +
      Sin[Sin[80 y]] - Sin[10 (x + y)] + 1/4 (x^2 + y^2), {x, y}, Method -> #]} &,
  {"Gradient", "ConjugateGradient", "InteriorPoint", "QuasiNewton", "Newton"}]]
```

```
{0.090967, {Gradient, {-0.618508, {x → 0.994175, y → 0.993418}}}}
```

Distributing function definitions

We can also define a function, distribute its definitions to the worker kernels, and then parallelize a computation.

```
factorsLength[n_] := Length[FactorInteger[n]];
DistributeDefinitions[factorsLength]
```

```
Table[With[{n = n}, ParallelSubmit[Length[FactorInteger[(10^n - 1) / 9]]]],
 {n, Range[45, 64]}]
```

```
WaitAll[%]
```

```
{9, 6, 2, 13, 4, 10, 8, 9, 4, 12, 8, 12, 6, 8, 2, 20, 7, 5, 13, 15}
```

Mathematica as a Programming Language

Wolfram *Workbench*

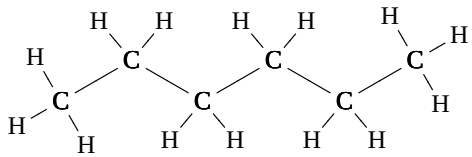
The Wolfram *Workbench* provides a powerful integrated development environment (IDE) that allows you to debug *Mathematica* code. Additionally, you can debug and work with mixed platform environments, such as Java, *JLink*, and grid*Mathematica*.

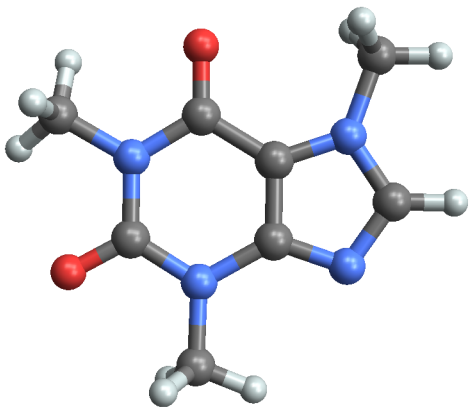
Working with Data

Physical and chemical data: elements, isotopes, chemicals, particles

```
Row[{
  Style[IsotopeData["Uranium235", "FullSymbol"],
   FontFamily → "Arial", FontSize → "Large"],
  Text["     "],
  ChemicalData["Hexane"],
  ChemicalData["Caffeine", "MoleculePlot"],
  Text["     "],
  Style[ParticleData["BStarZeroBar", "Symbol"], FontFamily → "Arial", FontSize → "Large"]
 }]
```

$${}^{235}_{92}U_{143}$$

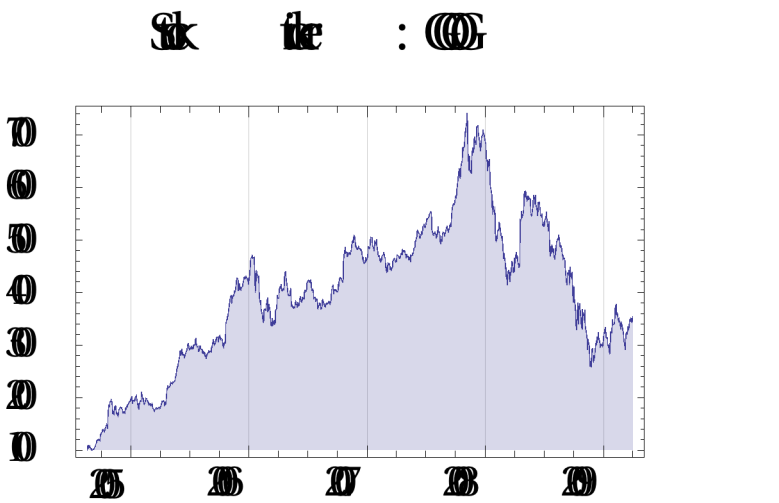




$$\overline{B}^{*0}$$

Financial, economic, and sociological data

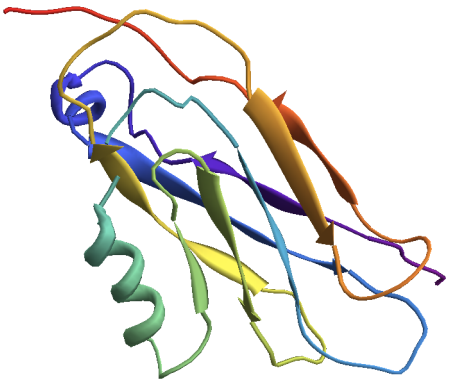
```
Row[{
  DateListPlot[FinancialData["GOOG", "2004"],
   Filling → Bottom, Joined → True, PlotLabel → "Stock ticker: GOOG"],
  Text["          "],
  DateListPlot[
   CountryData["France", {{"GDP"}, {1970, 2005}}], PlotLabel → "GDP of France"]
 }]
```

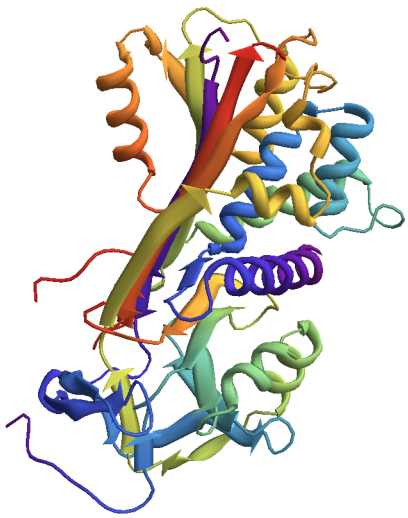


Life sciences:

genomes, proteins

```
Row[{
  ProteinData["A2M", "MoleculePlot"],
  ProteinData["SERPINA3", "MoleculePlot"]
 }]
```

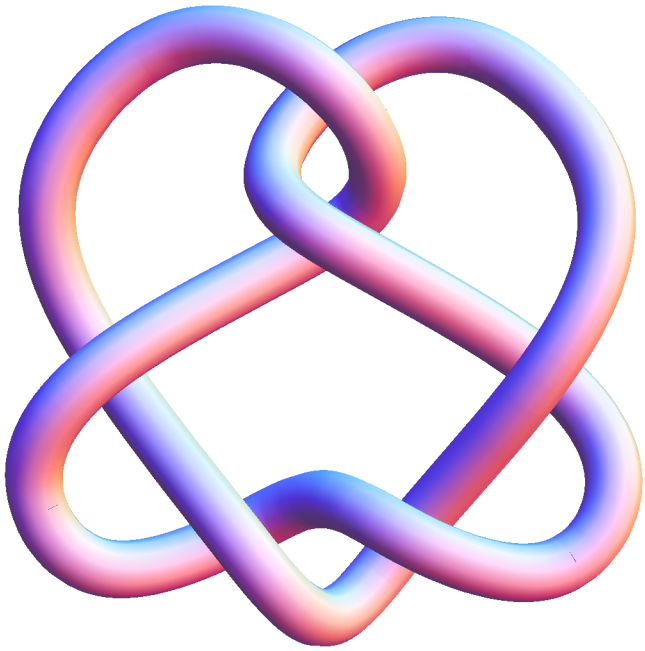


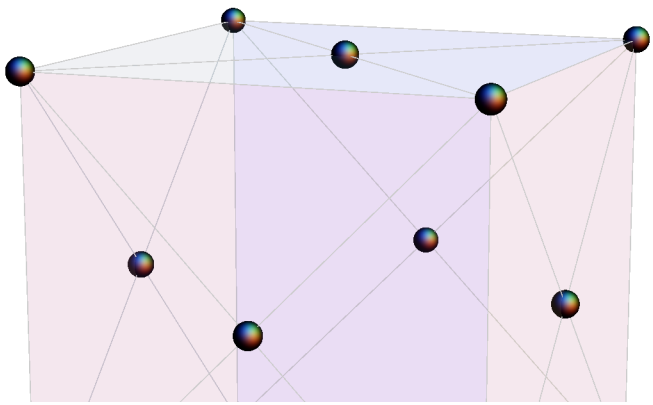


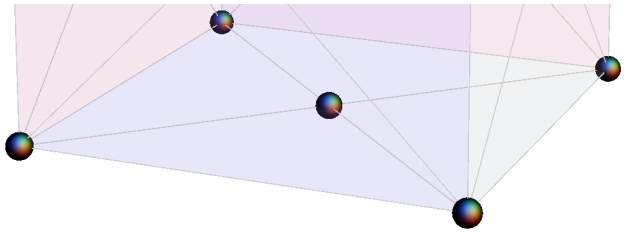
Mathematics:
finite groups, graphs, knots, lattices, polyhedra

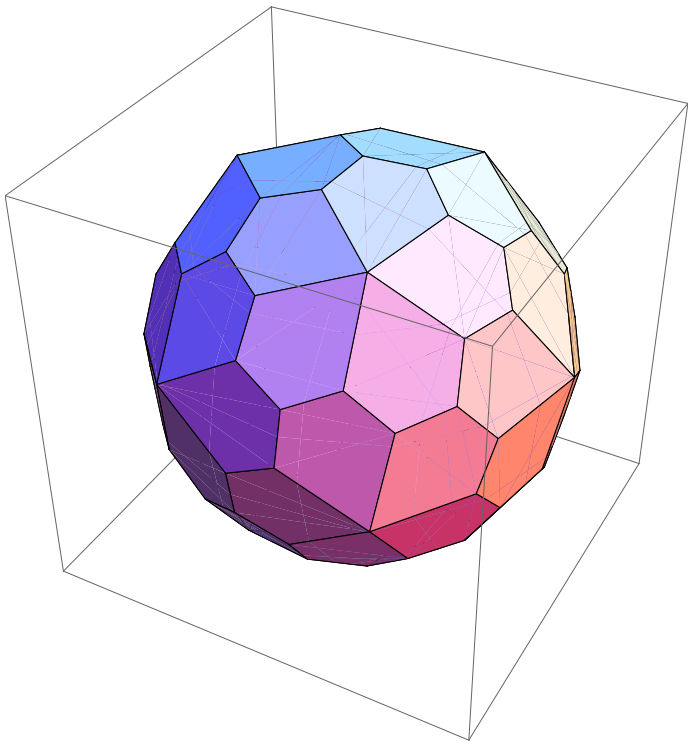
```
Row[{Style[Grid[FiniteGroupData["Quaternion", "MultiplicationTable"]],
   FontFamily → "Arial", FontSize → Medium],
  KnotData[{6, 1}],
  LatticeData["FaceCenteredCubic", "Image"],
  PolyhedronData["PentagonalHexecontahedron"]
 }]
```

1	2	3	4	5	6	7	8
2	5	4	7	6	1	8	3
3	8	5	2	7	4	1	6
4	3	6	5	8	7	2	1
5	6	7	8	1	2	3	4
6	1	8	3	2	5	4	7
7	4	1	6	3	8	5	2
8	7	2	1	4	3	6	5





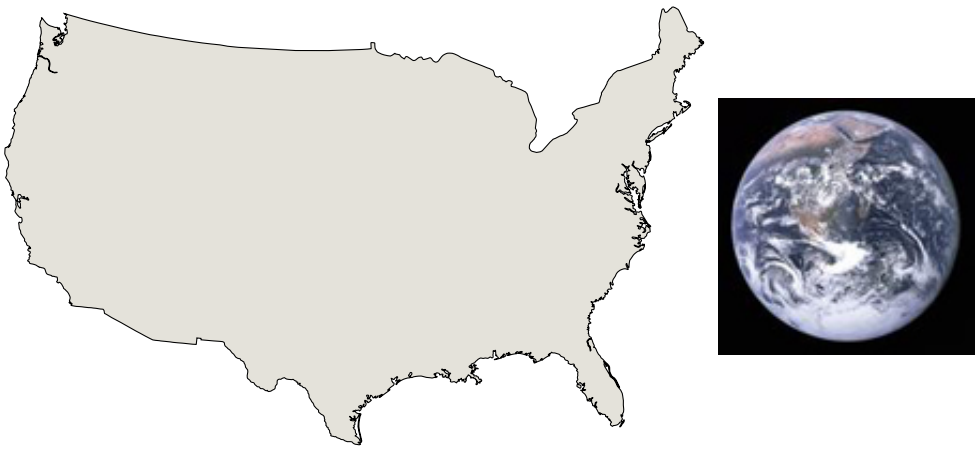




Geographical, meteorological, and astronomical data:

weather, geodesy, cities, countries, astronomical objects

```
Row[{
  CountryData["UnitedStates", "Shape"],
  AstronomicalData["Earth", "Image"]
 }]
```



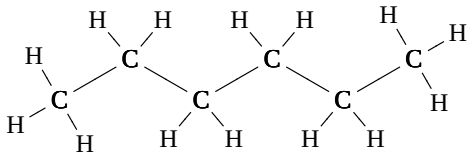
Working with Data

Chemical and Physical Data Sources

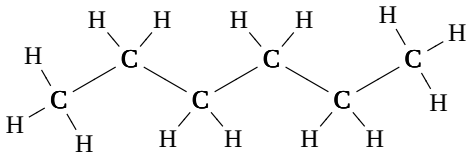
ChemicalData

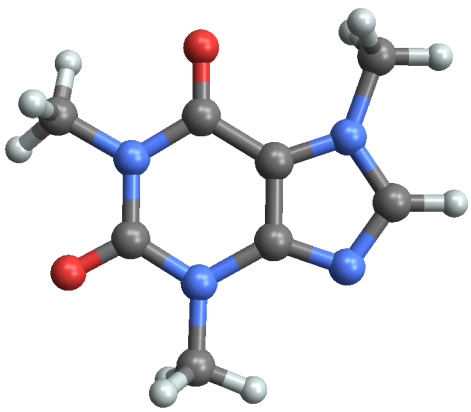
Get information about chemicals.

```
ChemicalData["Hexane"]
```



```
Row[{
  Style[IsotopeData["Uranium235", "FullSymbol"],
   FontFamily → "Arial", FontSize → "Large"],
  "      ",
  ChemicalData["Hexane"],
  ChemicalData["Caffeine", "MoleculePlot"],
  Style[ParticleData["BStarZeroBar", "Symbol"], FontFamily → "Arial", FontSize → "Large"]
 }]
```

$^{235}_{92}U_{143}$ 



$\overline{B}^{*0}$

Look up all information about a specific chemical.

```
ChemicalData["Caffeine", "Properties"]
```

```
{AcidityConstant, AcidityConstants, AdjacencyMatrix, AlternateNames, AtomPositions,
 AutoignitionPoint, BeilsteinNumber, BoilingPoint, BondTally, CASNumber,
 CHColorStructureDiagram, CHStructureDiagram, CIDNumber, Codons, ColorStructureDiagram,
 CombustionHeat, CompoundFormulaDisplay, CompoundFormulaString, CriticalPressure,
 CriticalTemperature, Density, DensityGramsPerCC, DielectricConstant,
 DOTHazardClass, DOTNumbers, EdgeRules, EdgeTypes, EGECNumber, ElementMassFraction,
 ElementTally, ElementTypes, EUNumber, FlashPoint, FlashPointFahrenheit,
 FormalCharges, FormattedName, GmelinNumber, HBondAcceptorCount, HBondDonorCount,
 HenryLawConstant, HildebrandSolubility, HildebrandSolubilitySI, InChI,
 IonEquivalents, Ions, IonTally, IsoelectricPoint, IsomericSMILES, IUPACName,
 LogAcidityConstant, LowerExplosiveLimit, MDLNumber, MeltingBehavior, MeltingPoint,
 Memberships, MolarVolume, MolecularFormulaDisplay, MolecularFormulaString,
 MolecularWeight, MoleculePlot, Name, NFPAFireRating, NFPAHazards, NFPAHealthRating,
 NFPALabel, NFPAReactivityRating, NonHydrogenCount, NonStandardIsotopeCount,
 NonStandardIsotopeNumbers, NonStandardIsotopeTally, NSCNumber, OdorThreshold,
 OdorType, PartitionCoefficient, pH, Phase, RefractiveIndex, Resistivity,
 RotatableBondCount, RTECSClasses, RTECSNumber, SideChainAcidityConstant, SMILES,
 Solubility, SolubilityType, SpaceFillingMoleculePlot, StandardName, StructureDiagram,
 SurfaceTension, TautomerCount, ThermalConductivity, TopologicalPolarSurfaceArea,
 UpperExplosiveLimit, VanDerWaalsConstants, VaporDensity, VaporizationHeat,
 VaporPressure, VaporPressureTorr, VertexCoordinates, VertexTypes, Viscosity}
```

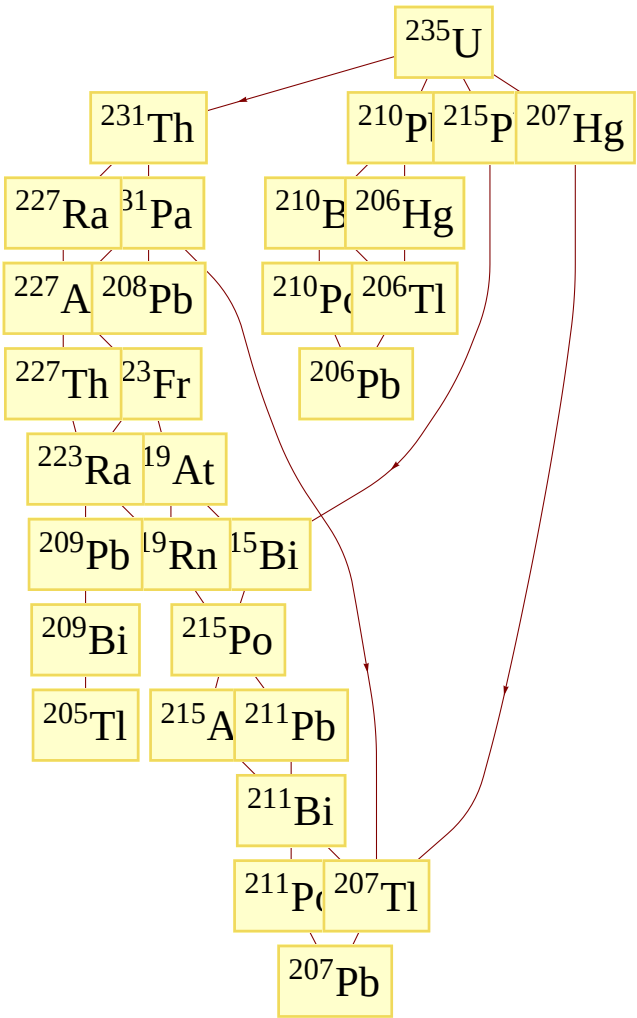
A more advanced example...a decay network for Uranium- 235.

```
DaughterNuclides[s_List] := DeleteCases[Union[
   Join @@ (IsotopeData[#1, "DaughterNuclides"] &) /@ DeleteCases[s, _Missing]], _Missing]

ReachableNuclides[s_List] := FixedPoint[Union[Join[#1, DaughterNuclides[#1]]] &, s]

DecayNetwork[iso_List] :=
 Join @@ (Thread[#1 → DaughterNuclides[{#1}]] &) /@ ReachableNuclides[iso]

U235Decay = Map[IsotopeData[#1, "Symbol"] &, DecayNetwork[{"Uranium235"}], {2}]
```

$$\{{}^{227}\text{Ac} \to {}^{223}\text{Fr},\ {}^{227}\text{Ac} \to {}^{227}\text{Th},\ {}^{215}\text{At} \to {}^{211}\text{Bi},\ {}^{219}\text{At} \to {}^{215}\text{Bi},\ {}^{219}\text{At} \to {}^{219}\text{Rn},\ {}^{209}\text{Bi} \to {}^{205}\text{Tl},\ {}^{210}\text{Bi} \to {}^{210}\text{Po},$$
$${}^{210}\text{Bi} \to {}^{206}\text{Tl},\ {}^{211}\text{Bi} \to {}^{211}\text{Po},\ {}^{211}\text{Bi} \to {}^{207}\text{Tl},\ {}^{215}\text{Bi} \to {}^{215}\text{Po},\ {}^{223}\text{Fr} \to {}^{219}\text{At},\ {}^{223}\text{Fr} \to {}^{223}\text{Ra},\ {}^{209}\text{Pb} \to {}^{209}\text{Bi},$$
$${}^{210}\text{Pb} \to {}^{210}\text{Bi},\ {}^{210}\text{Pb} \to {}^{206}\text{Hg},\ {}^{211}\text{Pb} \to {}^{211}\text{Bi},\ {}^{215}\text{Pb} \to {}^{215}\text{Bi},\ {}^{206}\text{Hg} \to {}^{206}\text{Tl},\ {}^{207}\text{Hg} \to {}^{207}\text{Tl},\ {}^{210}\text{Po} \to {}^{206}\text{Pb},$$
$${}^{211}\text{Po} \to {}^{207}\text{Pb},\ {}^{215}\text{Po} \to {}^{215}\text{At},\ {}^{215}\text{Po} \to {}^{211}\text{Pb},\ {}^{231}\text{Pa} \to {}^{227}\text{Ac},\ {}^{231}\text{Pa} \to {}^{208}\text{Pb},\ {}^{231}\text{Pa} \to {}^{207}\text{Tl},$$
$${}^{223}\text{Ra} \to {}^{209}\text{Pb},\ {}^{223}\text{Ra} \to {}^{219}\text{Rn},\ {}^{227}\text{Ra} \to {}^{227}\text{Ac},\ {}^{219}\text{Rn} \to {}^{215}\text{Po},\ {}^{206}\text{Tl} \to {}^{206}\text{Pb},\ {}^{207}\text{Tl} \to {}^{207}\text{Pb},$$
$${}^{227}\text{Th} \to {}^{223}\text{Ra},\ {}^{231}\text{Th} \to {}^{231}\text{Pa},\ {}^{231}\text{Th} \to {}^{227}\text{Ra},\ {}^{235}\text{U} \to {}^{210}\text{Pb},\ {}^{235}\text{U} \to {}^{215}\text{Pb},\ {}^{235}\text{U} \to {}^{207}\text{Hg},\ {}^{235}\text{U} \to {}^{231}\text{Th}\}$$

Draw the graph.

```
LayeredGraphPlot[U235Decay, VertexLabeling → True]
```



Working with Data

Financial, Economic, and Sociological Data

FinancialData

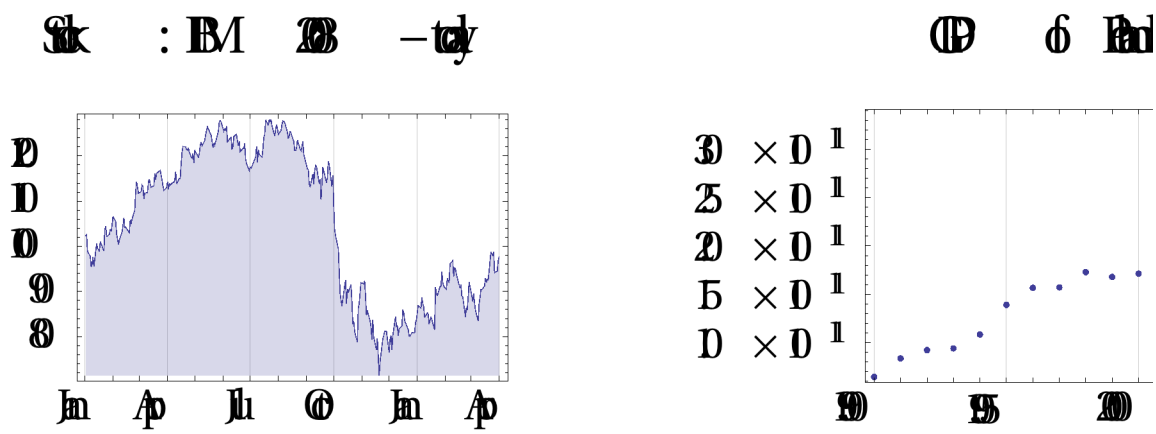
Get current information about a particular stock or financial instrument.

```
FinancialData["GOOG"]
```

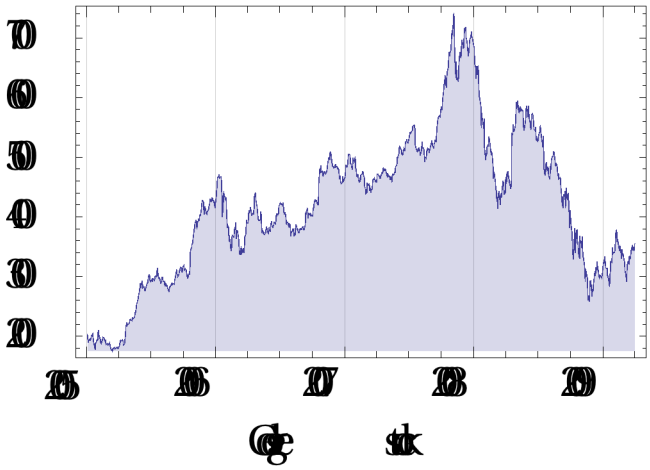
```
354.09
```

Data is delivered in a way that allows you to immediately feed it
into other parts of *Mathematica*.

```
GraphicsRow[{
  DateListPlot[FinancialData["IBM", "2008"],
   Filling → Bottom, Joined → True, PlotLabel → "Stock: IBM 2008-today"],
  DateListPlot[CountryData["Poland", {{"GDP"}, {1990, 2008}}],
   PlotLabel → "GDP of Poland"],
  DateListPlot[CountryData["Germany", {{"GDP"}, {1990, 2008}}],
   PlotLabel → "GDP of Germany"]
 }, ImageSize → 1000]
```



```
DateListPlot[FinancialData["GOOG", "2005"],
 Filling → Bottom, Joined → True, FrameLabel → "Google stock"]
```



CountryData

Get information about a particular country.

```
CountryData["Greece", "Shape"]
```

An incredible amount of properties are available.

```
CountryData["Greece", "Properties"]
```

```
{AdultPopulation, AgriculturalProducts, AgriculturalValueAdded, Airports,
 AlternateNames, AlternateStandardNames, AMRadioStations, AnnualBirths, AnnualDeaths,
 AnnualHIVAIDSDeaths, ArableLandArea, ArableLandFraction, Area, BirthRateFraction,
 BorderingCountries, BordersLengths, BoundaryLength, CallingCode, CapitalCity,
 CapitalLocation, CapitalLocationLink, CellularPhones, CenterCoordinates,
 CenterLocationLink, ChildPopulation, Classes, ClimateTypes, CoastlineLength,
 ConstructionValueAdded, Continent, Coordinates, Countries, CountryCode, CropsLandArea,
 CropsLandFraction, CurrencyCode, CurrencyName, CurrencyShortName, CurrencyUnit,
 CurrentAccountBalance, DeathRateFraction, Dependencies, DependencyParent,
 EconomicAid, ElderlyPopulation, ElectricalGridFrequency, ElectricalGridPlugImages,
 ElectricalGridPlugs, ElectricalGridSocketImages, ElectricalGridSockets,
 ElectricalGridVoltages, ElectricityConsumption, ElectricityExports, ElectricityImports,
 ElectricityProduction, EnvironmentalAgreements, EnvironmentalIssues, EthnicGroups,
 EthnicGroupsFractions, ExchangeRate, ExpenditureFractions, ExportCommodities,
 ExportPartners, ExportPartnersFractions, ExportValue, ExternalDebt,
 FemaleAdultPopulation, FemaleChildPopulation, FemaleElderlyPopulation,
 FemaleInfantMortalityFraction, FemaleLifeExpectancy, FemaleLiteracyFraction,
 FemaleMedianAge, FemalePopulation, FiscalYearDate, FixedInvestment, Flag,
 FlagDescription, FMRadioStations, ForeignExchangeReserves, ForeignOwnedShips,
 ForeignRegisteredShips, FullCoordinates, FullName, FullNativeName, FullPolygon,
 GDP, GDPAtParity, GDPPerCapita, GDPRealGrowth, GDPSectorFractions, GiniIndex,
 GovernmentConsumption, GovernmentDebt, GovernmentExpenditures, GovernmentReceipts,
 GovernmentSurplus, GrossInvestment, Groups, HighestElevation, HighestPoint,
 HIVAIDSDeathRateFraction, HIVAIDSFraction, HIVAIDSPopulation, HouseholdConsumption,
 ImportCommodities, ImportPartners, ImportPartnersFractions, ImportValue,
 IndependenceDate, IndependenceYear, IndustrialProductionGrowth, IndustrialValueAdded,
 InfantMortalityFraction, InfectiousDiseases, InflationRate, InternationalOrganizations,
 InternationalOrganizationsObserver, InternetCode, InternetHosts, InternetUsers,
 InventoryChange, IrrigatedLandArea, IrrigatedLandFraction, ISOName, LaborForce,
 LandArea, Languages, LanguagesDialects, LanguagesFractions, LargestCities,
 LifeExpectancy, LiteracyFraction, LowestElevation, LowestPoint, MajorIndustries,
 MajorPorts, MaleAdultPopulation, MaleChildPopulation, MaleElderlyPopulation,
 MaleInfantMortalityFraction, MaleLifeExpectancy, MaleLiteracyFraction, MaleMedianAge,
 MalePopulation, ManufacturingValueAdded, MaritimeClaims, MedianAge, Memberships,
 MerchantShips, MerchantShipsDeadWeight, MerchantShipsGross, MerchantShipTypes,
 MigrationRateFraction, MilitaryAgeFemales, MilitaryAgeMales, MilitaryAgePopulation,
 MilitaryAgeRate, MilitaryExpenditureFraction, MilitaryExpenditures, MilitaryFitFemales,
 MilitaryFitMales, MilitaryFitPopulation, MiscellaneousValueAdded, Name,
 NationalIncome, NationalityName, NativeName, NaturalGasConsumption, NaturalGasExports,
 NaturalGasImports, NaturalGasProduction, NaturalGasReserves, NaturalHazards,
 NaturalResources, OilConsumption, OilExports, OilImports, OilProduction,
 OilReserves, PavedAirportLengths, PavedAirports, PavedRoadLength, Phone, PhoneLines,
 Pipelines, Polygon, Population, PopulationGrowth, PovertyFraction, PriceIndex,
 RadioStations, RailwayGaugeLengths, RailwayGaugeRules, RailwayLength, RegionNames,
 Regions, Religions, ReligionsFractions, RoadLength, SchematicCoordinates,
 SchematicPolygon, SectorLaborFractions, Shape, ShortWaveRadioStations,
 SignedEnvironmentalAgreements, StandardName, SuffrageType, TelevisionStations,
 TerrainTypes, TimeZones, TotalConsumption, TotalFertilityRate, TradeValueAdded,
 TransportationValueAdded, UNCode, UnemploymentFraction, UNNumber, UnpavedAirportLengths,
 UnpavedAirports, UnpavedRoadLength, ValueAdded, WaterArea, WaterwayLength}
```

A more advanced example...plotting GDP vs. population.

```
Clear[lp, data]

data = (Tooltip[{CountryData[#1, "Population"], CountryData[#1, "GDP"]},
       CountryData[#1, "Name"]] &) /@ CountryData[];
data = DeleteCases[data, Tooltip[{___, _Missing, ___}, _]];
```

Fit the logarithms to a line and show on the graph.

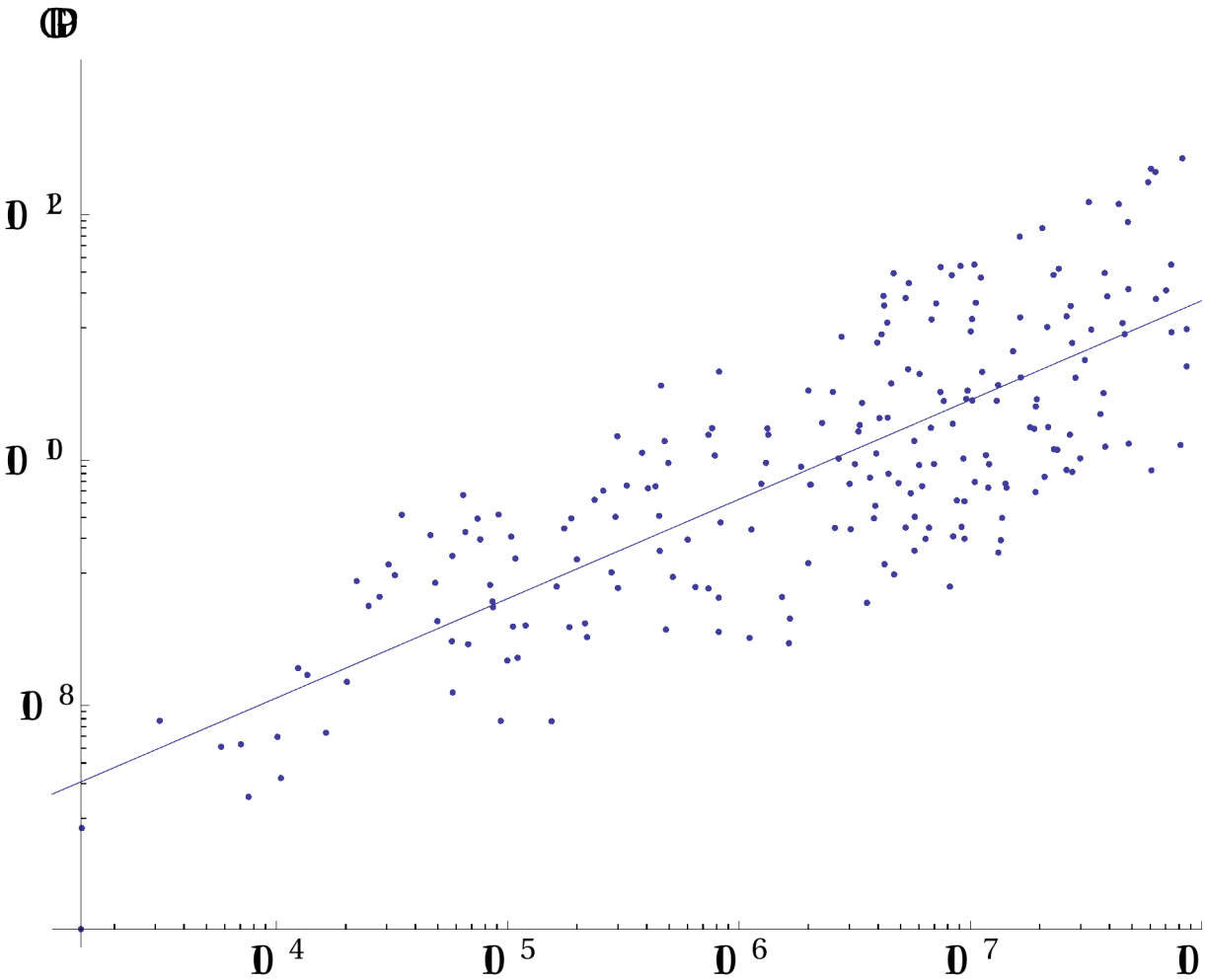
```
fit = Fit[Log[data[[All, 1]]], {1, lp}, lp]
```

```
11.1024 + 0.809705 lp
```

11.1024 + 0.809705 lp

Show the data and fit.

```
plot = ListLogLogPlot[data, AxesLabel → {"Population", "GDP"}];
Show[plot, Plot[fit, {lp, Log[10^3], Log[10^9]}], ImageSize → 800]
```



Working with Data

Life Sciences Data Sources

ProteinData

Get information about proteins.

```
ProteinData[]
```

```
A very large output was generated. Here is a sample of it:
{A1BG, A2M, NAT1, NAT2, SERPINA3, AADAC, AAMP, AANAT, AARS, ABAT, <<27 460>>,
 LOC100133313, LOC100133315, LOC100133319, LOC100133322, LOC100133327,
 LOC100133332, LOC100133333, LOC100133337Isoform1, LOC100133337Isoform2}
Show Less | Show More | Show Full Output | Set Size Limit...
```

```
(*ProteinData["SERPINA3","MoleculePlot"]*)
```

```
ProteinData["NAT1", "MolecularWeight"]
```

```
33 768
```

```
ProteinData["NAT1", "Name"]
```

```
N-acetyltransferase 1
```

```
ProteinData["NAT1", "BiologicalProcesses"]
```

```
{MetabolicProcess}
```

GenomeData

Get a list of genes *Mathematica* knows about.

```
GenomeData[]
```

A very large output was generated. Here is a sample of it:

```
{381, 3812, 3813, 3814, 3815, 5HT3c2, 7A5, A1BG, A1CF, A26A1, A26B1,
 A26B2, A26B3, A26C1A, A26C1B, <<39 890>>, ZSWIM7, ZUFSP, ZW10, ZWILCH,
 ZWINT, ZWINTAS, ZWS1, ZXDA, ZXDB, ZXDC, ZYG11A, ZYG11B, ZYX, ZZEF1, ZZZ3}
```

Show Less | Show More | Show Full Output | Set Size Limit...

We can choose a particular gene and look up properties about it.

```
GenomeData["ZXDB", "Chromosome"]
```

```
ChromosomeX
```

```
GenomeData["ZXDB", "Name"]
```

```
zinc finger, X-linked, duplicated B
```

Find the positions of a DNA sequence fragment:

```
(*GenomeLookup["CTCTCTAACTAAACT"]*)
```

```
{{{Chromosome1, 1}, {108 939 073, 108 939 087}},
 {{Chromosome1, -1}, {138 309 610, 138 309 624}},
 {{Chromosome5, -1}, {139 640 264, 139 640 278}},
 {{Chromosome8, 1}, {72 019 948, 72 019 962}}, {{Chromosome9, 1}, {110 092 060, 110 092 074}}}
```

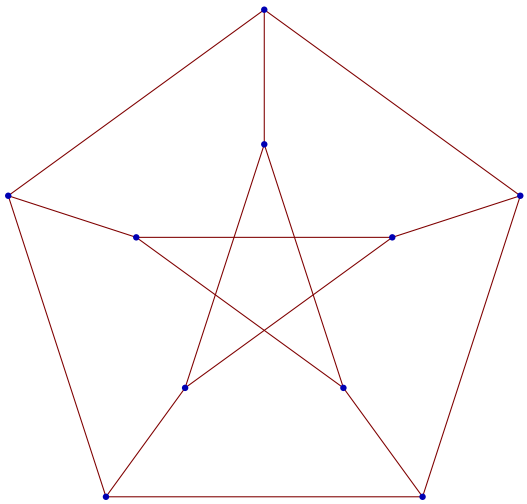
Working with Data

Mathematical Data Sources

GraphData

Get information about graphs.

```
GraphData["PetersenGraph"]
```



Now we can look at some properties...

```
GraphData["PetersenGraph", "Cubic"]
```

```
True
```

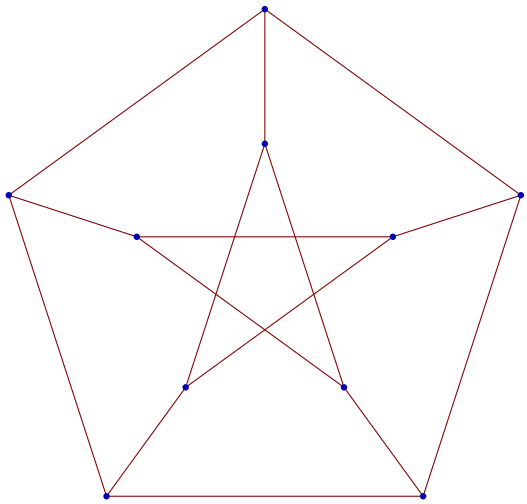
```
GraphData["PetersenGraph", "Spectrum"]
```

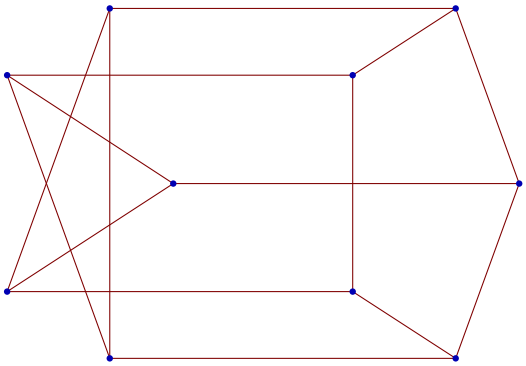
```
{-2, -2, -2, -2, 1, 1, 1, 1, 1, 3}
```

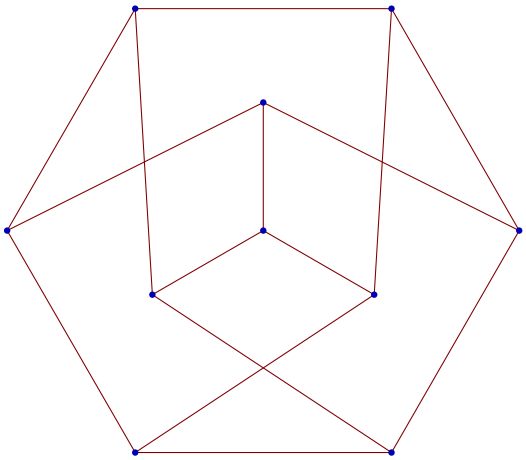
```
GraphData["PetersenGraph", "CycleCount"]
```

```
114
```

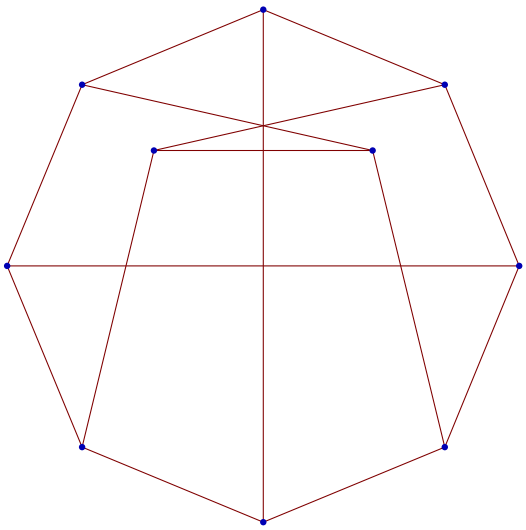
```
GraphData["PetersenGraph", "AllImages"]
```

{  ,

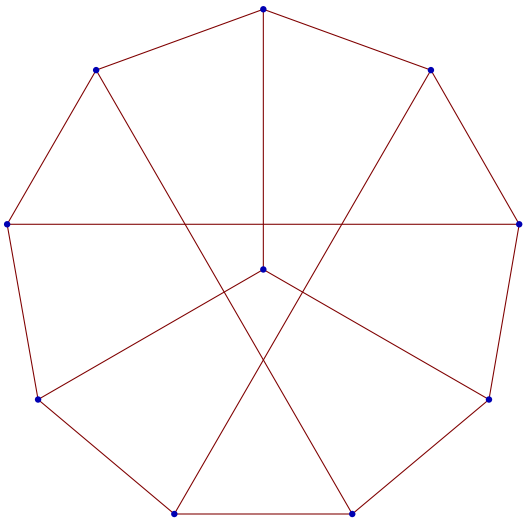
 ,



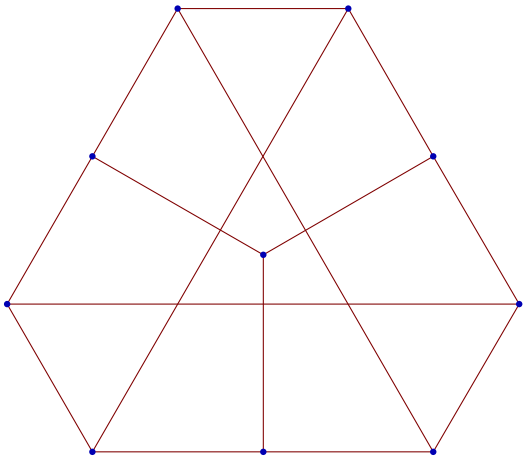
,



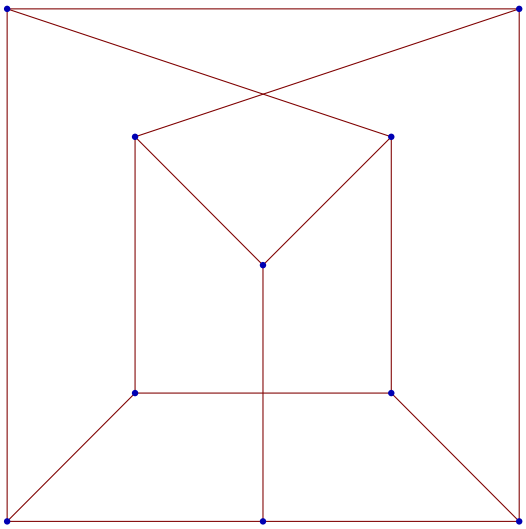
,



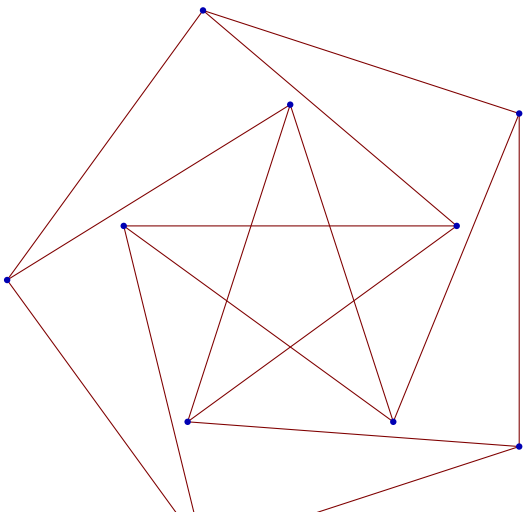
,



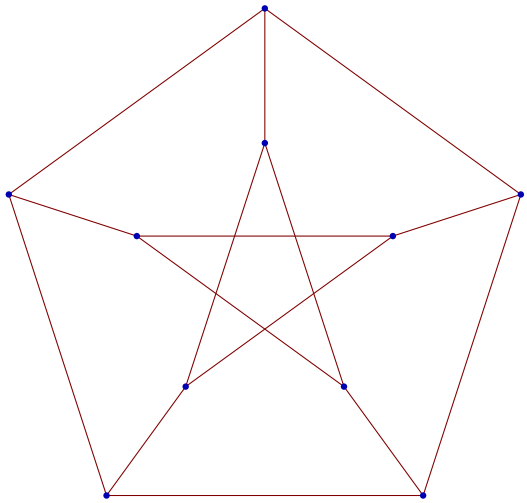
,

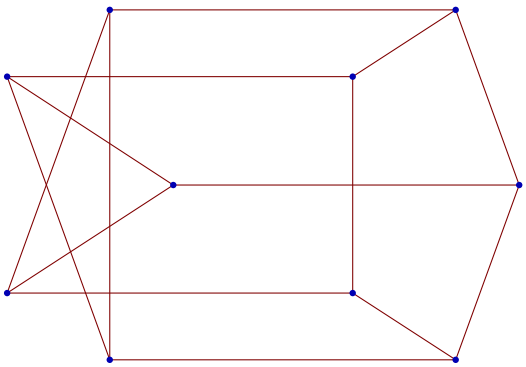


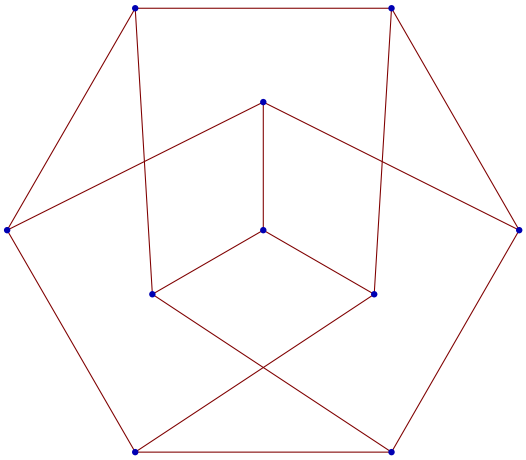
,



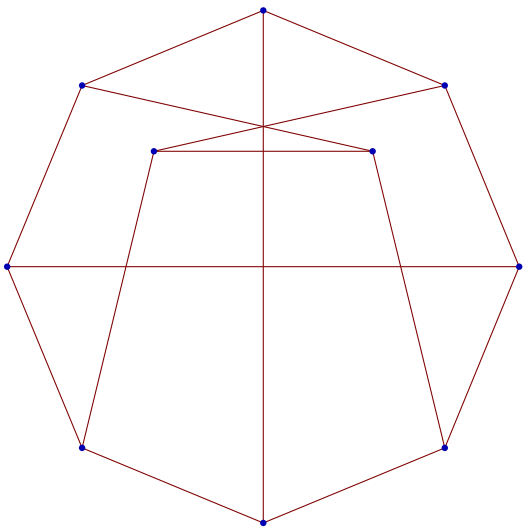
}

{  ,

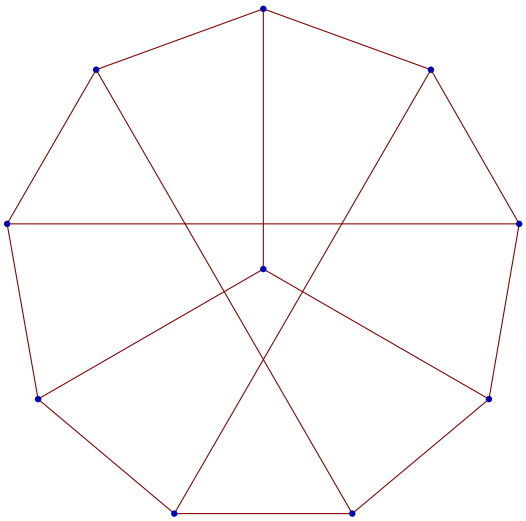
 ,



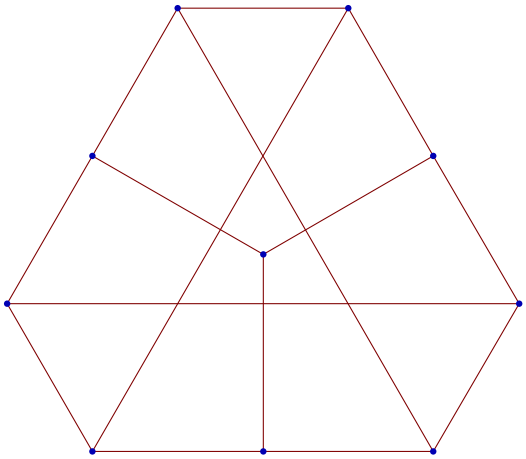
,



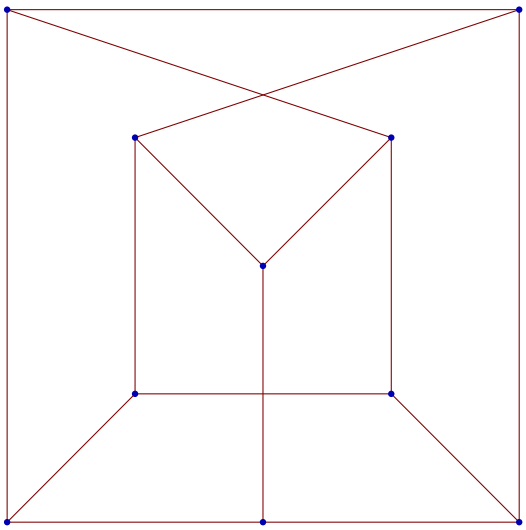
,



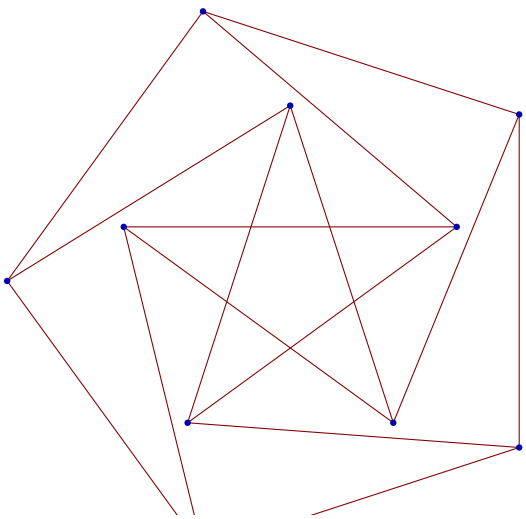
,



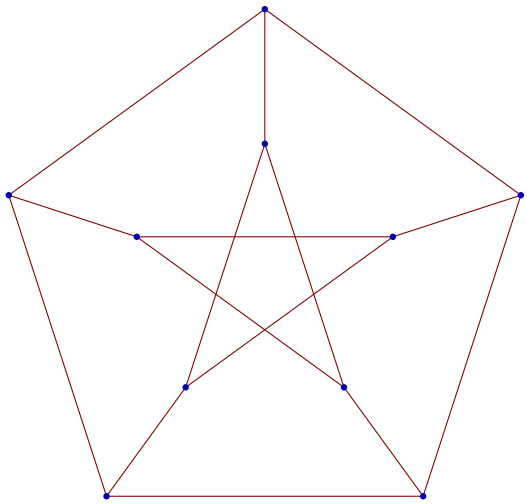
,

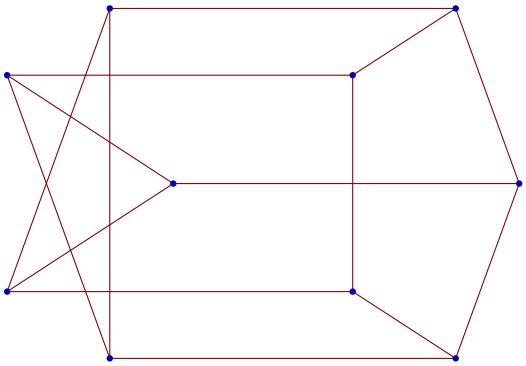


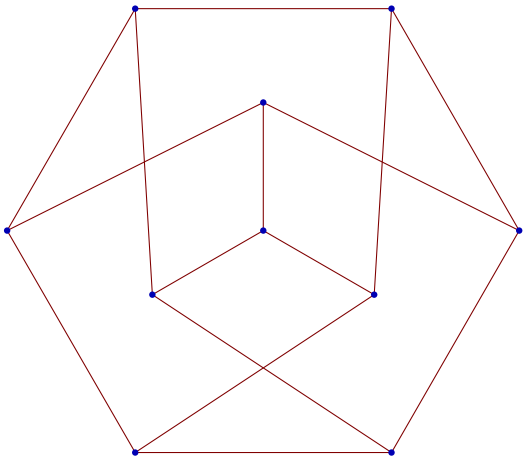
,



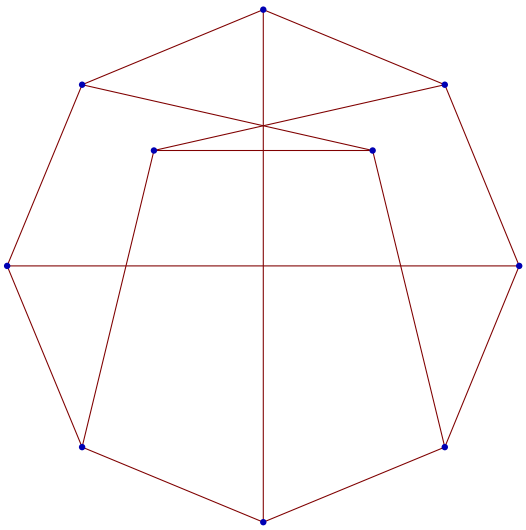
}

{  ,

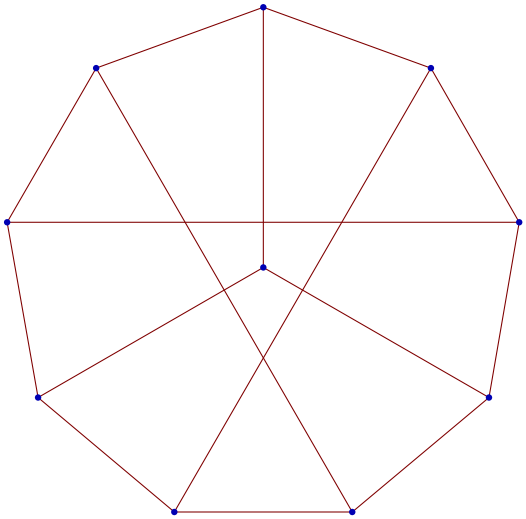
 ,



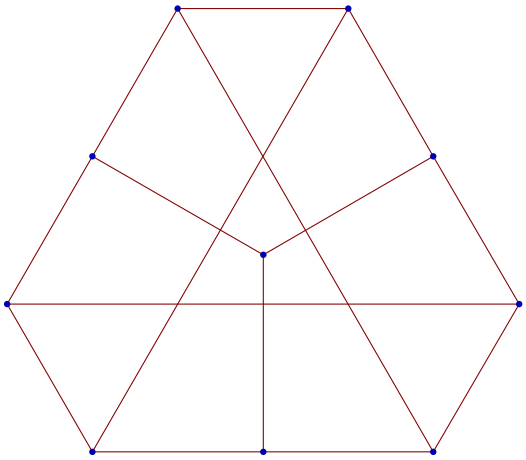
,



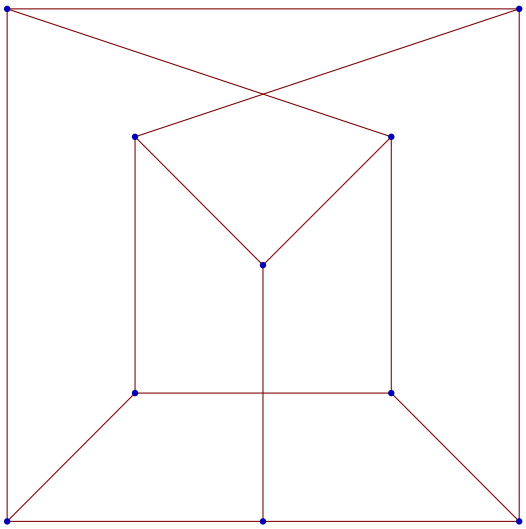
,



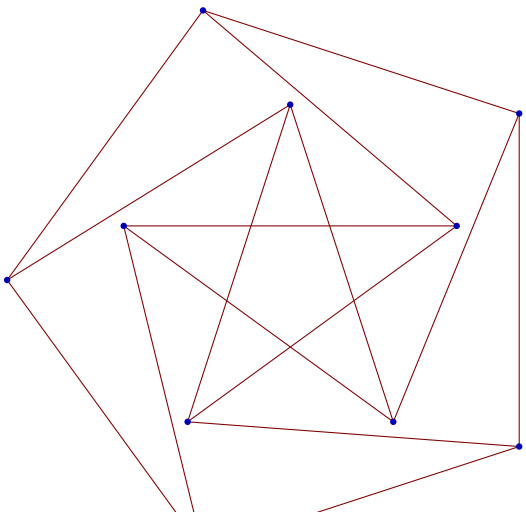
,



,



,



}

Working with Data

Geographical, Meteorological, and Astronomical Data Sources

WeatherData

Get weather information for a particular weather station.

```
(wd = WeatherData[]) // Length
```

```
17 168
```

```
(* WeatherData[{{"Buenos Aires","Argentina"},10}] *)
```

```
{SABE, SABA, AS890, SAEZ, SUCA, SADL, WMO87596, SUME, WMO87563, SAAG}
```

```
(* WeatherData["SABE","Properties"] *)
```

```
{AlternateStandardNames, CloudCoverFraction, CloudHeight, CloudTypes, Conditions,
 Coordinates, DewPoint, Elevation, Humidity, Latitude, Longitude, MaxTemperature,
 MaxWindSpeed, MeanDewPoint, MeanHumidity, MeanPressure, MeanStationPressure,
 MeanTemperature, MeanVisibility, MeanWindChill, MeanWindSpeed, Memberships,
 MinTemperature, NCDCID, PrecipitationAmount, PrecipitationRate, PrecipitationTypes,
 Pressure, PressureTendency, SnowAccumulation, SnowAccumulationRate,
 SnowDepth, StationName, StationPressure, Temperature, TotalPrecipitation,
 Visibility, WBANID, WindChill, WindDirection, WindGusts, WindSpeed, WMOID}
```

```
(* WeatherData["SABE","Coordinates"] *)
```

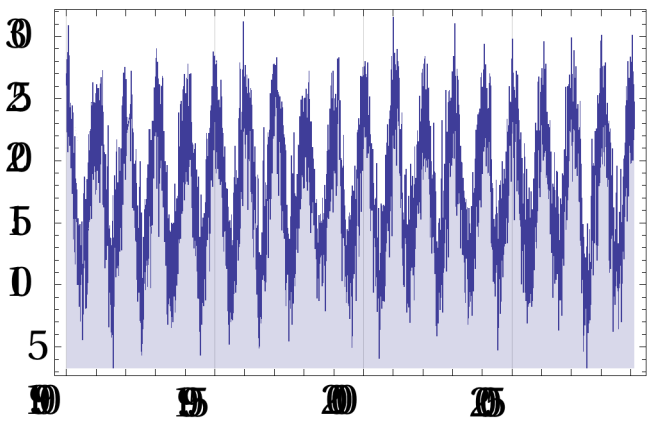
```
{-34.559, -58.416}
```

```
(* FindGeoLocation[] *)
```

```
{-34.48, -58.5}
```

Graph the mean temperature between 1990 and 2009

```
(*DateListPlot[WeatherData["SABE","MeanTemperature",{{1990,1,1},{2009,2,19},"Day"}],
 Joined→True,Filling→Bottom]*)
```



CityData

Search for information of a city.

```
CityData[{"Buenos Aires", "Argentina"}, "Properties"]
```

```
{AlternateNames, Coordinates, Country, Elevation, FullName, Latitude,
 LocationLink, Longitude, Name, Population, Region, RegionName, TimeZone}
```

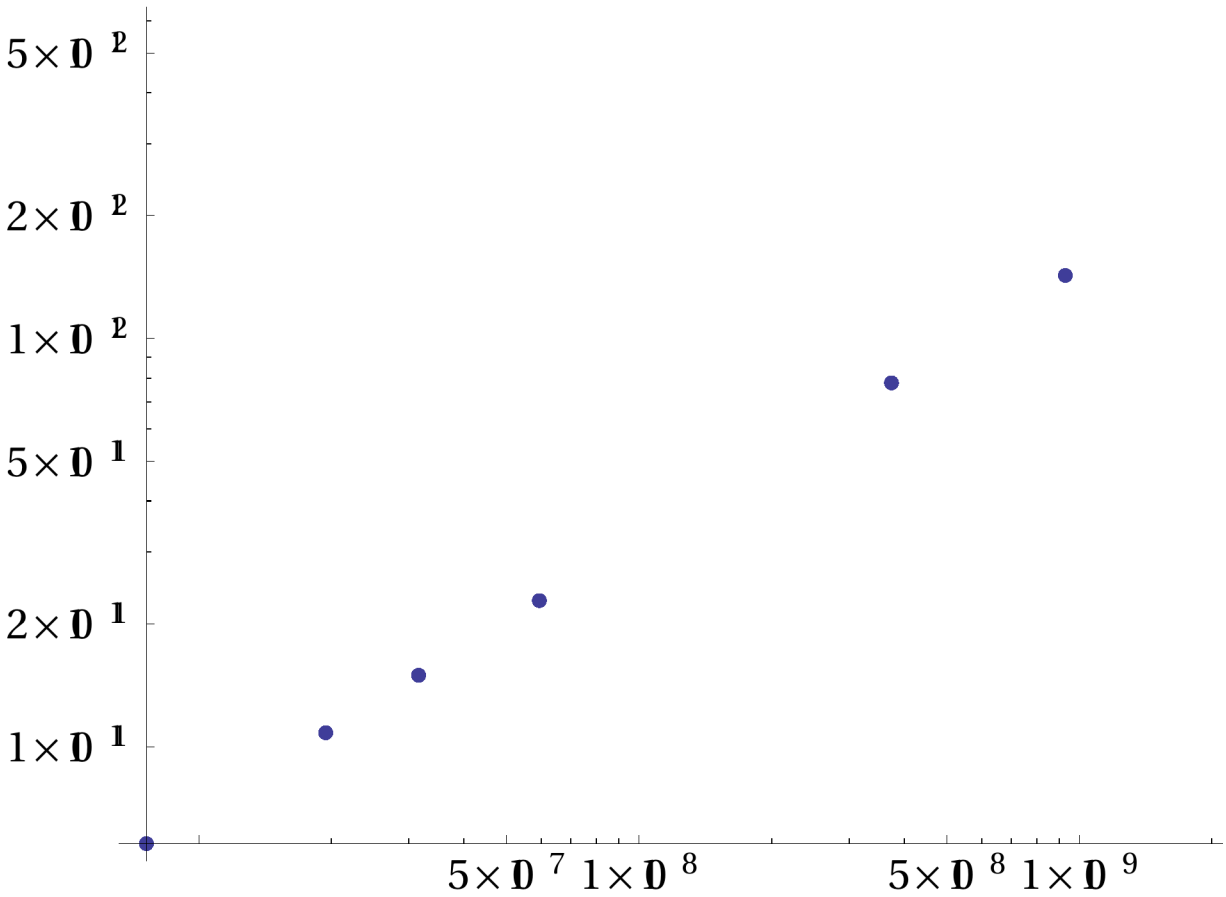
```
TableForm[
 ({#1, CityData[{"Buenos Aires", "Argentina"}, #1]} &) /@ CityData[All, "Properties"]]
```

AlternateNames	
Coordinates	−34.61 −58.37
Country	Argentina
Elevation	26
FullName	Buenos Aires, Buenos Aires, Argentina
Latitude	−34.61
LocationLink	http://maps.google.com/maps?q=−34.61−58.37&z=12&t=h
Longitude	−58.37
Name	Buenos Aires
Population	11 574 205
Region	BuenosAires
RegionName	Buenos Aires
StandardName	BuenosAires BuenosAires Argentina
TimeZone	−3

A more advanced example...illustrating Kepler's Third Law.

Log-log plot of orbital period vs. distance for the planets.

```
ListLogLogPlot[
 Tooltip[{AstronomicalData[#, "OrbitPeriod"], AstronomicalData[#, "SemimajorAxis"]},
    #] & /@ AstronomicalData["Planet"], PlotStyle → PointSize[Large], ImageSize → 700]
```



Working with Data

Importing, Exporting, and Connecting to Other Programs

Mathematica's expressions allow anything to be represented in some way. A broad variety of file formats can be imported from or exported to a file.

Importing & Exporting »

```
$ImportFormats
```

```
{3DS, ACO, AIFF, ApacheLog, AU, AVI, Base64, Binary, Bit, BMP, Byte, BYU, BZIP2, CDED,
 CDF, Character16, Character8, Complex128, Complex256, Complex64, CSV, CUR, DBF,
 DICOM, DIF, Directory, DXF, EDF, ExpressionML, FASTA, FITS, FLAC, GenBank, GeoTIFF,
 GIF, Graph6, GTOPO30, GZIP, HarwellBoeing, HDF, HDF5, HTML, ICO, Integer128,
 Integer16, Integer24, Integer32, Integer64, Integer8, JPEG, JPEG2000, JVX,
 LaTeX, List, LWO, MAT, MathML, MBOX, MDB, MGF, MMCIF, MOL, MOL2, MPS, MTP, MTX,
 MX, NB, NetCDF, NOFF, OBJ, ODS, OFF, Package, PBM, PCX, PDB, PDF, PGM, PLY, PNG,
 PNM, PPM, PXR, QuickTime, RawBitmap, Real128, Real32, Real64, RIB, RSS, RTF, SCT,
 SDF, SDTS, SDTSDEM, SHP, SMILES, SND, SP3, Sparse6, STL, String, SXC, Table, TAR,
 TerminatedString, Text, TGA, TIFF, TIGER, TSV, UnsignedInteger128, UnsignedInteger16,
 UnsignedInteger24, UnsignedInteger32, UnsignedInteger64, UnsignedInteger8, USGSDEM,
 UUE, VCF, WAV, Wave64, WDX, XBM, XHTML, XHTMLMathML, XLS, XML, XPORT, XYZ, ZIP}
```

```
Import[ToFileName[NotebookDirectory[], "chemtest.xls"]]
```

```
{{{0., 0.}, {1., 0.}, {2., 0.}, {3., 0.}, {4., 0.}, {5., 0.}, {6., 0.3}, {7., 0.7}, {8., 1.1},
  {9., 1.7}, {10., 2.8}, {11., 4.1}, {12., 5.9}, {13., 8.1}, {14., 11.3}, {15., 13.6},
  {16., 15.1}, {17., 16.}, {18., 16.6}, {19., 17.}, {20., 17.3}, {21., 17.5}, {22., 17.6},
  {23., 17.7}, {24., 17.7}, {25., 17.7}, {26., 17.7}, {27., 17.7}, {28., 17.7}, {29., 17.7}}}
```

```
$ExportFormats
```

```
{3DS, ACO, AIFF, AU, AVI, Base64, Binary, Bit, BMP, Byte, BYU, BZIP2, CDF, Character16,
 Character8, Complex128, Complex256, Complex64, CSV, DICOM, DIF, DXF, EMF, EPS,
 ExpressionML, FASTA, FITS, FLAC, FLV, GIF, Graph6, GZIP, HarwellBoeing, HDF, HDF5, HTML,
 Integer128, Integer16, Integer24, Integer32, Integer64, Integer8, JPEG, JPEG2000,
 JVX, List, LWO, MAT, MathML, Maya, MGF, MIDI, MOL, MOL2, MTX, MX, NB, NetCDF, NOFF, OBJ,
 OFF, Package, PBM, PCX, PDB, PDF, PGM, PLY, PNG, PNM, POV, PPM, PXR, RawBitmap, Real128,
 Real32, Real64, RIB, RTF, SCT, SDF, SND, Sparse6, STL, String, SVG, SWF, Table, TAR,
 TerminatedString, TeX, Text, TGA, TIFF, TSV, UnsignedInteger128, UnsignedInteger16,
 UnsignedInteger24, UnsignedInteger32, UnsignedInteger64, UnsignedInteger8, UUE,
 VRML, WAV, Wave64, WDX, WMF, X3D, XBM, XHTML, XHTMLMathML, XLS, XML, XYZ, ZIP, ZPR}
```

MathLink provides a general interface for connecting *Mathematica* to other programs. Links to Java (*J/Link*) and .NET (*.NET/Link*) have been built on top of *MathLink* to provide easy interactive connectivity to these runtimes.

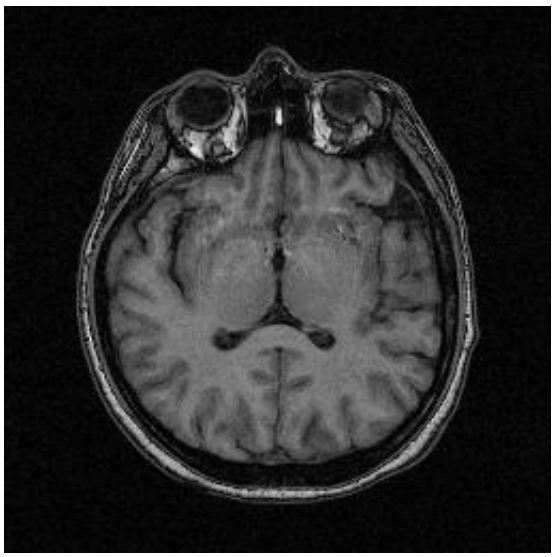
External Programs »

Working with Data

Mathematica for Image Processing

Images can be imported, copied and pasted, or even dragged and
dropped into *Mathematica*.

```
brain = Import[ToFileName[NotebookDirectory[], "brain.jpg"]]
```

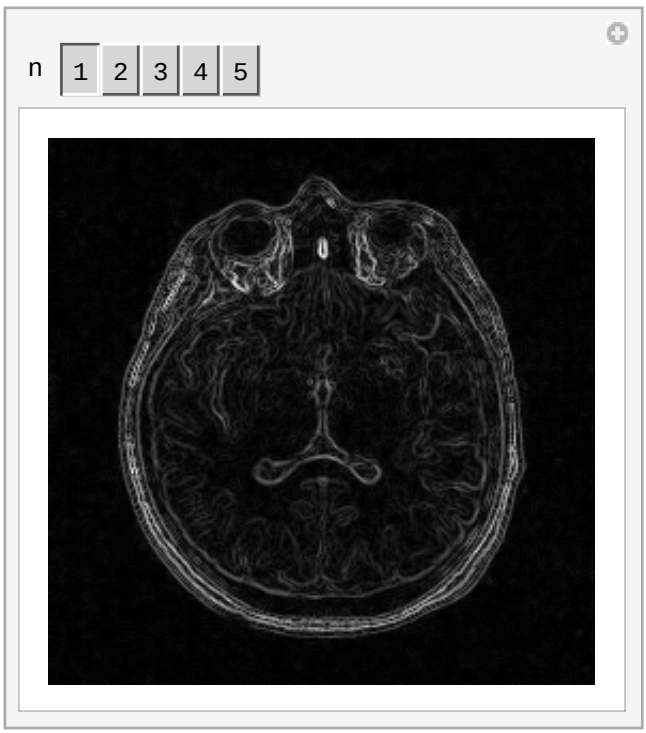


We can sharpen the image for inspection.

```
Manipulate[Sharpen[brain, n], {n, 1, 5}, SaveDefinitions → True]
```

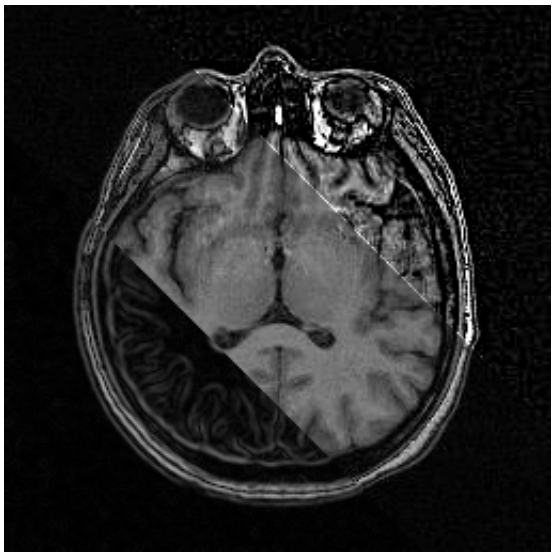
Or we can extract the edges to determine contours.

```
Manipulate[ImageAdjust[GradientFilter[brain, n]], {n, Range[5]}, SaveDefinitions → True]
```



We can even apply several effects to different parts of the image at the same time.

```
s = Sharpen[brain, 4];
g = GradientFilter[brain, 2] // ImageAdjust;
ds = UpperTriangularize[#, 59] & /@ ImageData[s, Interleaving → False];
dg = LowerTriangularize[#, -60] & /@ ImageData[g, Interleaving → False];
dh = Normal@SparseArray[{i_, j_} /; Abs[i - j] < 60 :> #[[i, j]],
      Reverse@ImageDimensions@brain] & /@ ImageData[brain, Interleaving → False];
Image[ds + dg + dh, Interleaving → False]
```



Many other effects are possible.

```
lena = ExampleData[{"TestImage", "Lena"}];
Grid[{
  {ColorNegate[lena],
   ImageTake[lena, 300],
   ImagePad[lena, 15, Red]},
  {Image[GraphicsGrid[ImagePartition[lena, 25]], ImageSize → 180],
   Image[ImageRotate[lena, 45 Degree]],
   ImageEffect[lena, "Charcoal"]},
  {Dilation[lena, 3],
   ImageApply[RotateRight, lena],
   ImageAdjust[lena, 6]}
 }]
```

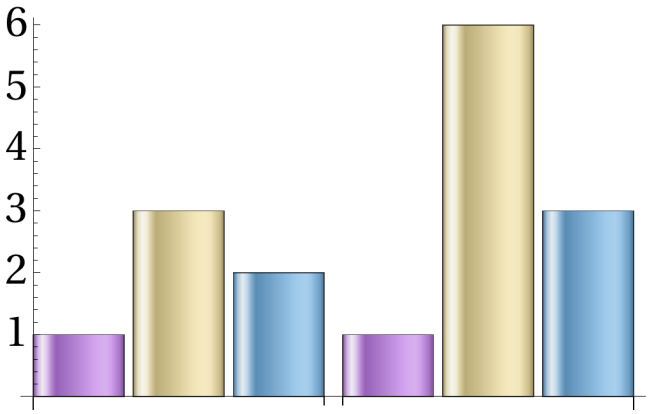
Working with Data

Statistics

Charting

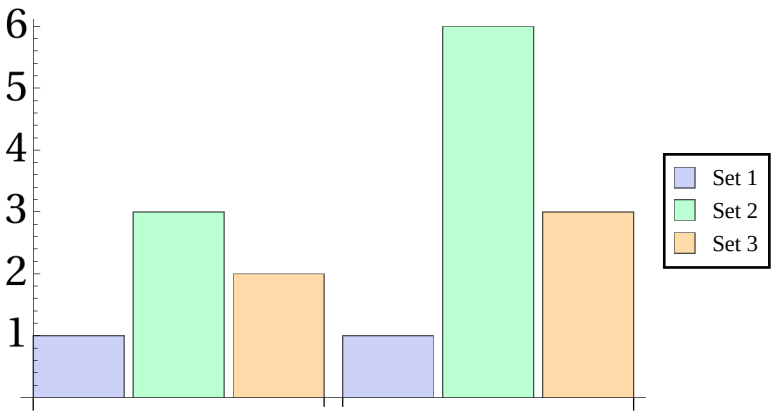
Create a bar chart.

```
BarChart[{{1, 3, 2}, {1, 6, 3}},
 ChartElementFunction → "GlassRectangle", ChartStyle → "Pastel"]
```



Add customizations, such as legends:

```
BarChart[{{1, 3, 2}, {1, 6, 3}}, ChartLegends → {"Set 1", "Set 2", "Set 3"}]
```



Or use your own graphics.

```
spikey =
```

```
;
BarChart[Range[10], ChartElements → spikey]
```

Pie charts are interactive.

```
PieChart[Range[5]]
```

Bubble charts allow you to visualize data sets with varying magnitudes.

```
BubbleChart[RandomInteger[{10, 80}, {10, 10, 3}]]
```

There are also 3D analogues of the above functions.

```
BarChart3D[{{1, 3, 2}, {1, 6, 3}}, ChartLegends → {"Set 1", "Set 2", "Set 3"}]
```

```
PieChart3D[Range[5], ChartStyle → "Rainbow"]
```

```
BubbleChart3D[RandomInteger[{10, 80}, {10, 10, 4}]]
```

There is also a new palette to help with charting.

Open up Chart Element Schemes palette to show an example of this.

```
BarChart[Range[10]]
```

Final Thoughts

Mathematica as a Teaching and Research Environment

Can collect all relevant research information in *Mathematica* notebooks:

- General notes
- Typeset formulas for publication
- Graphics
- Code implementation
- Direct conversion to HTML, TeX, PDF, etc.
- Field-specific application packages

In Conclusion

- A lot of new features in *Mathematica* 7
- All components of *Mathematica* are integrated.
- Instant interactivity makes exploring fast and fun.

Resources Available

- Learning Center
- Library
- MathWorld
- Blog
- Forums