

GoogleDataLink

User Manual and Reference Guide

Introduction

Features

The GoogleDataLink package provides a set of *Mathematica* functions that allow you to:

For Google Docs:

- Read from and write data and formulas to Google Docs spreadsheet cells and cell ranges
- Query/update/add/delete tabular data in a Google Docs spreadsheet like in a SQL query.
- Retrieve contributor names, emails, last updates, etc. of shared Google Docs spreadsheets.
- Add/rename/remove worksheets
- Manipulate spreadsheet/worksheet structure

For Google Calendar:

- Search your own calendar as well as all calendars you have subscribed to for text strings, and return all matches
- Obtain a date-sorted list of all your calendar events ("Agenda View")
- Create new calendars
- Delete calendars
- Subscribe to existing calendars
- Unsubscribe from existing calendars
- Add new events to your calendar

Additional Features:

- An interactive expandable/collapsible tree view of any *Mathematica* expression.
- A symbol browser that allows to search and filter from all standard *Mathematica* (3,600+), package, and all user-defined symbols (functions, options, attributes) along with their evaluated values – anything known to the kernel.

About

GoogleDataLink

Version 1.1

Copyright © 2012 Andreas Lauschke Consulting

<http://www.lauschkeconsulting.com>

Table of Contents

Introduction.....	2
Features	2
Installation.....	5
Requirements.....	5
Installation and Configuration.....	5
Look-and-Feel Configuration.....	7
Working in Mathematica	8
Getting Started	8
Loading the Package	8
Google Docs: Data Retrieval and Manipulation Functions.....	10
Assigning and Retrieving Data: Cell- and Range-based Operations.....	10
Assigning and Retrieving Data: List-based Operations.....	16
GoogleDataLink Functions: Returning Mathematica Expressions.....	23
GoogleDataLink Functions: Returning Java Objects.....	26
Google Calendar.....	27
Other Utility Functions.....	30
Interactive Tree Representation/Inspection of Mathematica Expressions.....	30
Interactive Symbol Browser of Mathematica Expressions.....	34

Installation

Requirements

- GoogleData libraries, 3.0 or later. Available from <http://code.google.com/p/gdata-java-client/downloads/list>
Download the gdata-src.java-xxx.zip file.
 - Current version is 1.46.0
- *Mathematica* 6 or later. Available from www.wolfram.com.
 - Current version is *Mathematica* 8.0.4.
- Java 7 or later. Available from www.java.com.
 - Current version is Java 7 update 2.
- GoogleDataLink 1.1 or later. Available from www.lauschkeconsulting.com.
 - Current version is GoogleDataLink 1.1.

Mathematica 7 and 8 have Java 6 bundled with it, *Mathematica* 6 has Java 5 bundled with it.

Installation and Configuration

- Create a new directory called GoogleDataLink in your user directory. Place all files from the GoogleDataLink distribution (3 files) in the GoogleDataLink directory.
- If you haven't done so, download the Google Docs libraries, for example from <http://code.google.com/p/gdata-java-client/downloads/list>
Download the gdata.java-xxx.zip file, unzip the files to a directory on your local filesystem. Make a note of that location for later.
- Edit the file GoogleDataLinkconfig.m in the GoogleDataLink directory with a text editor to set the variable GoogleDataLinklocation to the location of your local GoogleDataLink installation. You can use *Mathematica's* ToFileName[...] command or assign a string with the directory location. In the latter case ensure you are using proper syntax for your operating system, e. g. “../..” on Unix/Linux and “..\.” on Windows.

- In `GoogleDataLinkconfig.m` set the variables `googledocslibraries1` and `googledocslibraries2` to the lib directory of the Google Docs API library from the second step above.
 - `GoogleDataLinklibraries1` points to `../gdata/java/lib`.
 - `GoogleDataLinklibraries2` points to `../gdata/java/deps`.
- In `GoogleDataLinkconfig.m` set the variable `jlinklocation` to the location of the file `JLink.jar` of your local *Mathematica* installation. With a default install, this is

`../SystemFiles/Links/JLink/JLink.jar`
- In `GoogleDataLinkconfig.m` set the variables `GoogleDocsUserId` and `GoogleDocsPassword` to your Google UserID and your Google Password. If you change your Google password, do not forget to update the file `GoogleDataLinkconfig.m`.
- If you want to modify the Java runtime arguments used with GoogleDocs Link, set the variable `commandline` in the file `GoogleDataLinkconfig.m` to the string representation of the Java runtime arguments that you may want to modify from the default values, e. g. to specify a particular Java runtime you want to use (not the one that is included in the *Mathematica* distribution) or to set additional runtime options (memory settings, JIT-compilation, etc.). Assigning to `commandline` will automatically reinstall the Java runtime. The default is to not use a special command line and not to reinstall the Java runtime (i. e. the variable assignment is missing in the file).

Look-and-Feel Configuration

GoogleDataLink supports most third-party add-on/plug-in look-and-feels. To install a third-party look-and-feel, download the .jar file from the third-party supplier, place it in the GoogleDataLink directory, identify the name of the entry point class, and place a line

```
laf="<full name space class name to entry point class>"
```

in a text file lookandfeel.m in the GoogleDataLink directory. Example:

```
laf="com.jgoodies.looks.plastic.Plastic3DLookAndFeel"
```

This will enable the look-and-feel immediately for the *Mathematica* package.

If no file lookandfeel.m exists in the GoogleDataLink directory or laf is assigned the string "default", GoogleDataLink will use the system's default look-and-feel.

For a good overview of various free and commercial look-and-feels, visit <http://www.javootoo.com>

The following Look-and-Feels have been tested to work with GoogleDataLink:

- JGoodies Plastic3D
- JGoodies PlasticXP
- JGoodies Plastic
- JGoodies Windows
- Office2003 (Windows only)
- OfficeXP (Windows only)
- VisualStudio2005 (Windows only)
- Nimrod
- Fh
- Tiny
- Tonic
- Tonic Slim
- Infonode
- Napkin
- SquareNess
- EaSynth

which can be downloaded from javootoo.

The Alloy look-and-feel has also been tested to work with GoogleDataLink, which can be obtained from <http://lookandfeel.incors.com/>.

Working in *Mathematica*

Getting Started

Loading the Package

To start using the link from *Mathematica*, you must first load the GoogleDataLink package.

With *Mathematica* version 6 and above:

```
Get@ToFileName[{$HomeDirectory, "GoogleDataLink"}, "GoogleDataLink.m"]
```

With *Mathematica* version 7 and above:

```
Get@FileNameJoin[{$HomeDirectory, "GoogleDataLink", "GoogleDataLink.m"}]
```

This will start the Mathematica package and set up a link to your Google Documents and attempt authentication with the Google UserID and Google Password you have set up in the configuration file. If you the authentication fails, you will get an error message:



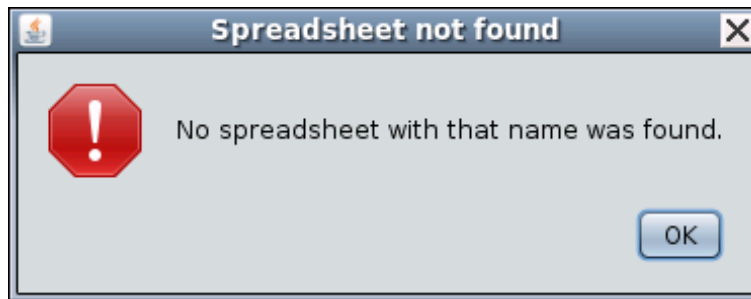
Next you have to define which of your GoogleDocs spreadsheets is your “active” spreadsheet. GoogleDataLink is based on the concept of an active spreadsheet that you are working on. Whenever you want to start working on another spreadsheet from your GoogleDocs spreadsheets, you have to set the active spreadsheet to the new spreadsheet.

```
GDLSetActiveSpreadsheet["MyFirstSpreadsheet"]
```

This sets the active GoogleDataLink spreadsheet to the spreadsheet called “MyFirstSpreadsheet”.

If there is no spreadsheet with that name, an error is thrown:

`GDLSetActiveSpreadsheet["ThisSpreadsheetdoesnotexist"]`



Google Docs: Data Retrieval and Manipulation Functions

Assigning and Retrieving Data: Cell- and Range-based Operations

The function `GDL[<cell>]` or alternatively `GDL[<row>,<column>]` can be used like a variable in *Mathematica*.

An assignment:

```
GDL["A12"]=56.8
```

```
« JavaObject[com.google.gdata.data.spreadsheet.CellEntry]»
```

or

```
GDL[12,1]=56.8
```

```
« JavaObject[com.google.gdata.data.spreadsheet.CellEntry]»
```

10		
11		
12	56.8	
13		
14		
...		

A retrieval:

```
GDL["A12"]
```

```
56.8
```

or

```
GDL[12,1]
```

```
56.8
```

This can be done for data of types integer, real, and string:

```
GDL["A13"]="Good Morning"
```

```
« JavaObject[com.google.gdata.data.spreadsheet.CellEntry]»
```

or

```
GDL[13,1]="Good Morning"
```

```
« JavaObject[com.google.gdata.data.spreadsheet.CellEntry]»
```

10		
11		
12		56.8
13	Good Morning	
14		
15		
16		

```
GDL["A13"]
```

```
"Good Morning"
```

or

```
GDL[13,1]
```

```
"Good Morning"
```

GoogleDataLink automatically converts numeric data from a symbolic *Mathematica* expression to display it in Google Docs spreadsheets:

```
GDL[15,1]=Pi;
```

```
GDL[16,1]=Sqrt@5;
```

```
GDL[17,1]=64/10;
```

10		
11		
12		56.8
13	Good Morning	
14		
15		3.14159
16		2.23607
17		6.4
18		
19		

You can also set formulas with GoogleDataLink, not just pass data:

GDL[19,1]="=A15+A16"

« JavaObject[com.google.gdata.data.spreadsheet.CellEntry]»

10		
11		
12		56.8
13	Good Morning	
14		
15		3.14159
16		2.23607
17		6.4
18		
19		5.37766
20		
21		

These tasks can also be performed with the explicit functions GDLSetCell[] and GDLGetCell[]. With these functions it is also possible to specify the worksheet to be used for the operation, for example

GDLGetCell[3,6,"mythirdworksheet"]

or

GDLGetCell["F3","mythirdworksheet"]

or

```
GDL[3,6,"mythirdworksheet"]
```

or

```
GDL["F3","mythirdworksheet"]
```

returns the data in cell 3,6 (or "F3") in the worksheet "mythirdworksheet" and

```
GDLSetCell[6,8,"new data","myfourthworksheet"]
```

or

```
GDLSetCell["H6","new data","myfourthworksheet"]
```

or

```
GDL[6,8,"myfourthworksheet"]="new data"
```

or

```
GDL["H6","myfourthworksheet"]="new data"
```

writes the string "new data" into cell 6,8 (or "H6") in the worksheet "myfourthworksheet".

The worksheets can also be accessed by index with the functions

```
GDLGetCell[<row>.<column>,<index>]
```

```
GDLSetCell[<row>.<column>,<value>,<index>]
```

```
GDLGetCell[<cell>,<index>]
```

```
GDLSetCell[<cell>,<value>,<index>]
```

or the equivalent GDL[] "assignment versions".

The function

```
GDLGetRange[<firstrow>,<firstcolumn>,<lastrow>,<lastcolumn>]
```

or equivalently

```
GDLGetRange[<cellrange>]
```

retrieves all values in the specified range.

	A	B	C	D
1	46			0.7
2	5656			0.7
3				
4		55		
5	33	557		
6	333	6		
7				
8		61		
9				
10				
11				
12	56.8			
13	Good Morning		8	
14		555		
15	3.14159265358979			
16	2.23606797749979			
17	6.4			
18				
19	5.37766063108958			
20				

GDLGetRange[1,1,20,4]

{46,0.7,0.7,3,55,33,557,333,6,61,56.8,Good Morning,8,555,3.14159,2.23607,6.4,5.37766}

or equivalently

GDLGetRange["A1:D20"]

{46,0.7,0.7,3,55,33,557,333,6,61,56.8,Good Morning,8,555,3.14159,2.23607,6.4,5.37766}

Analogously, the function GDLGetRange[] can also access a particular worksheet, by index or by name:

GDLGetRange[<firstrow>,<firstcolumn>,<lastrow>,<lastcolumn>,<name>]

GDLGetRange[<firstrow>,<firstcolumn>,<lastrow>,<lastcolumn>,<index>]

GDLGetRange[<cellrange>,<name>]

GDLGetRange[<cellrange>,<index>]

However, for the retrieval and manipulation (queries, updates, deletions, insertions, etc.) of larger tabular data the GoogleDataLink functions that make use of the list interface of the Google Data API are suited much better, see the next section.

The function `GDLIsCellEmpty[]` returns `True` if the specified cell is empty (blank), and `False` if the specified cell is not empty (blank).

```
GDLIsCellEmpty["B5","thirdsheet"]
```

`False`

```
GDLIsCellEmpty["B15","thirdsheet"]
```

`True`

`GDLIsCellEmpty[]` can also be used with row and column indices, worksheet indices, and default index (0 for first sheet):

```
GDLIsCellEmpty[<row>,<column>]
```

```
GDLIsCellEmpty[<row>,<column>,<sheet index>]
```

```
GDLIsCellEmpty[<row>,<column>,<sheet name>]
```

```
GDLIsCellEmpty[<cell>]
```

```
GDLIsCellEmpty[<cell>,<sheet index>]
```

```
GDLIsCellEmpty[<cell>,<sheet name>]
```

The Function `GDLGetLastEdited[]` returns the time stamp(s) the specified worksheet(s) was/were last edited.

`GDLGetLastEdited[<index>]` returns the time stamp the worksheet with the specified index was last edited. If `<name>` is a worksheet name, then `GDLGetLastEdited[<name>]` returns the time stamp the worksheet with that name was last edited. If `<name>` is the name of the active spreadsheet or is missing (as in `GDLGetLastEdited[]`) then this function will return a list of pairs of worksheet names and the respective times they were last edited.

`GDLGetLastEdited[]` used the *Mathematica* date/time format.

Assigning and Retrieving Data: List-based Operations

The Google Data API provides a list interface that can be used to retrieve, update, delete, and insert a whole set of cell entries at once. This is particularly efficient because a whole array of data can be manipulated with just one operation. In GoogleDataLink these functions have the word "List" in them.

To use the list interface from the Google Data API the data in the worksheet must meet the following criteria:

The list feed treats the first row of the worksheet as a header row. The columns are now referred to under the header name/label (similar to the use of a database table in a relational database). The first row should NOT contain any actual data, but only the column labels. If the first row contains data, then these data elements become the column labels, and the data will be missing from the actual data table returned.

The list feed contains all rows after the first row up to the first blank row. The first blank row terminates the data set. If expected data is not appearing in a feed, check the worksheet manually to see whether there is an unexpected blank row in the middle of the data. In particular, if the second row of the spreadsheet is blank, then the list feed will contain no data.

A row in a list feed has as many columns as the worksheet itself.

There can be only one table for the list feed for any given worksheet.

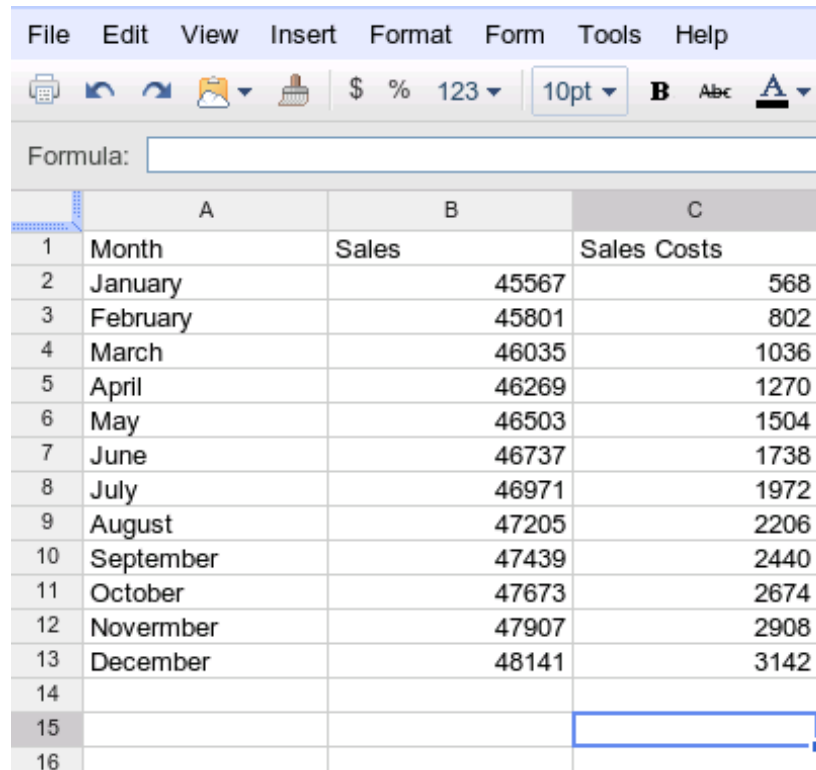
The column labels, when retrieved, are returned by the Google Data API as all lowercase and spaces omitted. This is NOT a bug of GoogleDataLink.

GDLGetList[] returns a list of the column labels and the data matrix.

GDLGetList[] uses the first worksheet of the active spreadsheet.

GDLGetList[<index>] uses the worksheet with the specified index of the active spreadsheet.

GDLGetList[<name>] uses the worksheet with the specified name of the active worksheet.



	A	B	C
1	Month	Sales	Sales Costs
2	January	45567	568
3	February	45801	802
4	March	46035	1036
5	April	46269	1270
6	May	46503	1504
7	June	46737	1738
8	July	46971	1972
9	August	47205	2206
10	September	47439	2440
11	October	47673	2674
12	November	47907	2908
13	December	48141	3142
14			
15			
16			

```
In[3]:= {header, data} = GDLGetList["thirdsheet"],  
header  
data // TableForm
```

```
Out[4]= {month, sales, salescosts}
```

```
Out[5]//TableForm=  
January 45 567 568  
February 45 801 802  
March 46 035 1036  
April 46 269 1270  
May 46 503 1504  
June 46 737 1738  
July 46 971 1972  
August 47 205 2206  
September 47 439 2440  
October 47 673 2674  
November 47 907 2908  
December 48 141 3142
```

GDLListQuery[] returns a list of the column labels and the data matrix, filtered by the query.

GDLListQuery[] uses the first worksheet of the active spreadsheet.

GDLListQuery[<index>] uses the worksheet with the specified index of the active spreadsheet.

GDLListQuery[<name>] uses the worksheet with the specified name of the active worksheet.

	A	B	C
1	Month	Sales	Sales Costs
2	January	45567	568
3	February	45801	802
4	March	46035	1036
5	April	46269	1270
6	May	46503	1504
7	June	46737	1738
8	July	46971	1972
9	August	47205	2206
10	September	47439	2440
11	October	47673	2674
12	November	47907	2908
13	December	48141	3142
14			
15			
16			

```
In[6]:= {header, data} = GDLListQuery["sales > 47000 and salescosts < 2900", "thirdsheet"],
header
data // TableForm
```

```
Out[7]= {month, sales, salescosts}
```

```
Out[8]//TableForm=
August    47205 2206
September 47439 2440
October   47673 2674
```

GDLListUpdateByQuery[] updates the data that is filtered by the query in the worksheet (this can affect more than one row). This function should be used with extreme care as the existing data will be overwritten without asking for confirmation.

GDLListUpdateByQuery[] uses the first worksheet of the active spreadsheet.

GDLListUpdateByQuery[<index>] uses the worksheet with the specified index of the active spreadsheet.

GDLListUpdateByQuery[<name>] uses the worksheet with the specified name of the active worksheet.

```
In[10]:= GDLListUpdateByQuery["sales > 47000 and salescosts < 2900", "sales",  
12345.67, "thirdsheet"]
```

```
Out[10]= 12345.670000000000000000000000
```

	A	B	C
1	Month	Sales	Sales Costs
2	January	45567	568
3	February	45801	802
4	March	46035	1036
5	April	46269	1270
6	May	46503	1504
7	June	46737	1738
8	July	46971	1972
9	August	12345.67	2206
10	September	12345.67	2440
11	October	12345.67	2674
12	November	47907	2908
13	December	48141	3142
14			
15			
16			

GDLListUpdateByRowIndex[] does the same as GDLListUpdateByQuery[], but uses a row index instead of a query to identify the row in which to update data. Unlike GDLListUpdateByQuery[] this can affect only one row. This function should also be used with extreme care as the existing data will be overwritten without asking for confirmation.

GDLListUpdateByRowIndex[] uses the first worksheet of the active spreadsheet.
GDLListUpdateByRowIndex[<index>] uses the worksheet with the specified index of the active spreadsheet.
GDLListUpdateByRowIndex[<name>] uses the worksheet with the specified name of the active worksheet.

GDLListDeleteByRowIndex[] deletes the row with the specified row index.

GDLListDeleteByRowIndex[] uses the first worksheet of the active spreadsheet.
GDLListDeleteByRowIndex[<index>] uses the worksheet with the specified index of the active spreadsheet.
GDLListDeleteByRowIndex[<name>] uses the worksheet with the specified name of the active worksheet.

```
In[13]:= GDLListDeleteByRowIndex[9, "thirdsheet"]  
Out[13]= 9
```

	A	B	C
1	Month	Sales	Sales Costs
2	January	45567	568
3	February	45801	802
4	March	46035	1036
5	April	46269	1270
6	May	46503	1504
7	June	46737	1738
8	July	46971	1972
9	August	47205	2206
10	September	47439	2440
11	November	47907	2908
12	December	48141	3142
13			
14			
15			

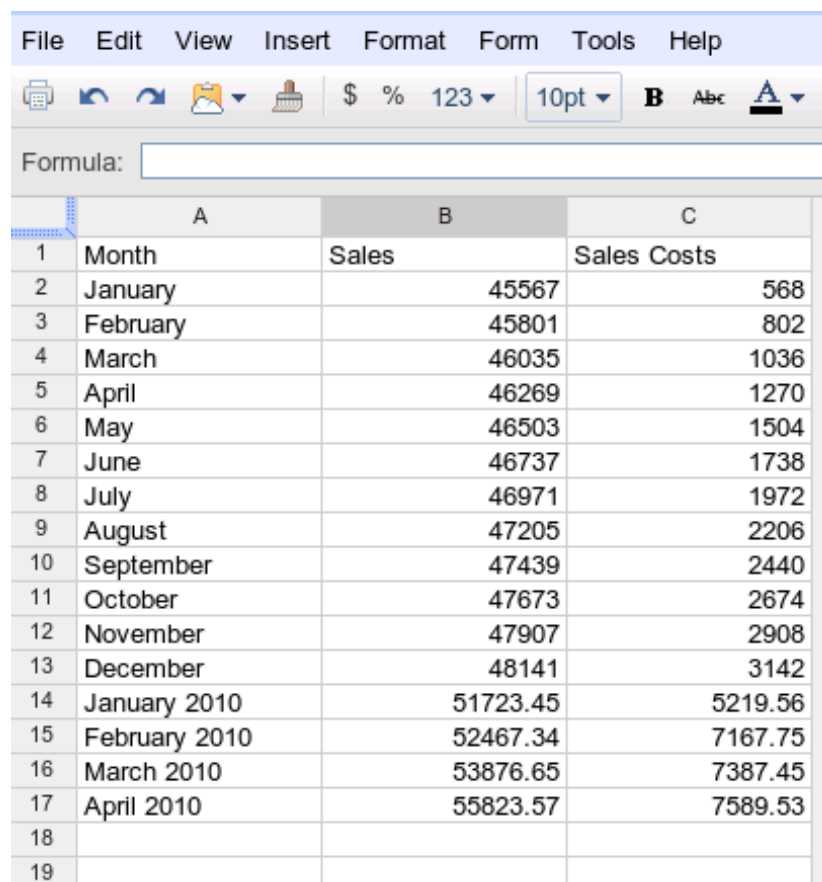
GDLListAddRows[] adds the lines at the bottom of the list.

GDLListAddRows[] uses the first worksheet of the active spreadsheet.

GDLListAddRows[<index>] uses the worksheet with the specified index of the active spreadsheet.

GDLListAddRows[<name>] uses the worksheet with the specified name of the active worksheet.

```
In[19]:= GDLListAddRows[{"February 2010", 52467.34, 7167.75}, {"March 2010", 53876.65, 7387.45}, {"April 2010", 55823.57, 7589.53}], "thirdsheet"]
```



	A	B	C
1	Month	Sales	Sales Costs
2	January	45567	568
3	February	45801	802
4	March	46035	1036
5	April	46269	1270
6	May	46503	1504
7	June	46737	1738
8	July	46971	1972
9	August	47205	2206
10	September	47439	2440
11	October	47673	2674
12	November	47907	2908
13	December	48141	3142
14	January 2010	51723.45	5219.56
15	February 2010	52467.34	7167.75
16	March 2010	53876.65	7387.45
17	April 2010	55823.57	7589.53
18			
19			

GoogleDataLink Functions: Returning Mathematica Expressions

GDLGetSpreadsheetTitles[] returns a list of all Google Docs spreadsheet titles:

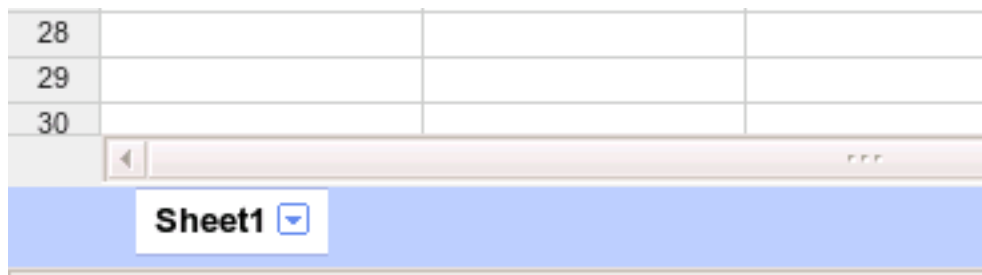
```
GDLGetSpreadsheetTitles[]
```

```
{secondspreadsheet,MyFirstSpreadsheet}
```

GDLGetAuthorNames[] returns a list of all author names for the active spreadsheet.

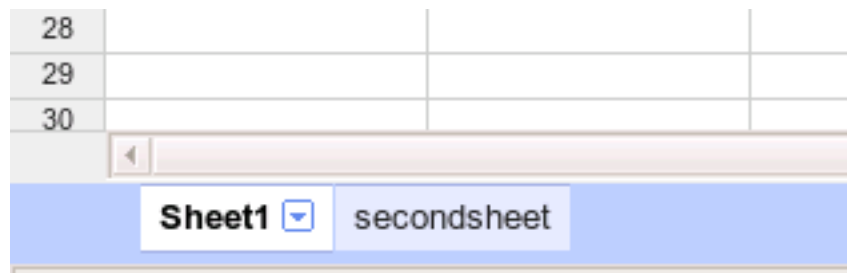
GDLGetAuthorEmails[] returns a list of all author emails for the active spreadsheet.

The function GDLAddWorksheet[<name>,<rows>,<columns>] adds a new worksheet to the active spreadsheet with the specified number of rows and columns.



```
GDLAddWorksheet["secondsheet",200,3]
```

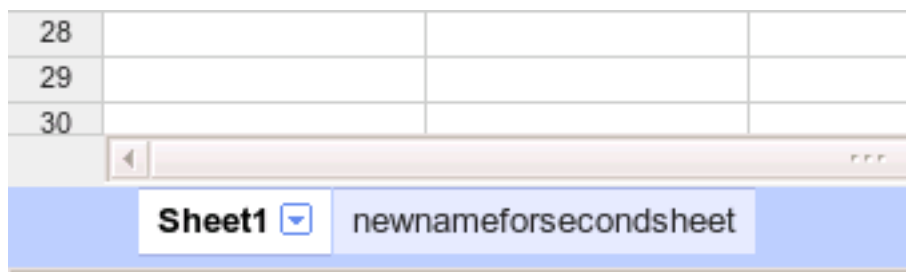
```
« JavaObject[com.google.gdata.data.spreadsheet.WorksheetEntry]»
```



The function GDLSetWorksheetName[] sets the name of the specified worksheet.

GDLSetWorksheetName["secondsheet","newnameforsecondsheet"]

newnameforsecondsheet



The function GDLGetWorksheetNames[] returns a list with the names of the active spreadsheet.

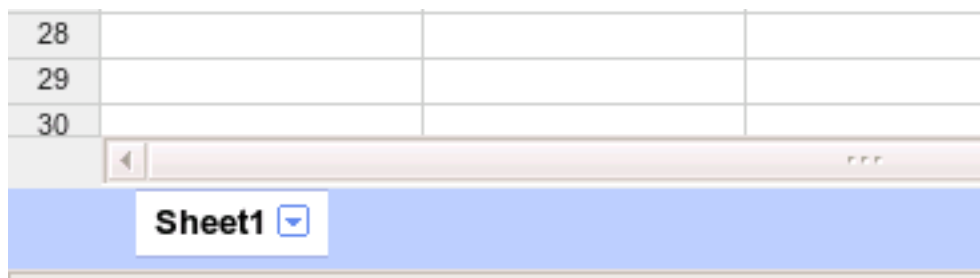
GDLGetWorksheetNames[]

{Sheet1,newnameforsecondsheet}

The function GDLRemoveWorksheet[<worksheetname>] removes an existing worksheet.

GDLRemoveWorksheet["newnameforsecondsheet"]

« JavaObject[com.google.gdata.data.spreadsheet.SpreadsheetEntry]»



The function GDLGetColumnCount[] returns the number of columns in the selected worksheet of the active spreadsheet. The worksheet can be specified by index or by name, and the first worksheet is used as a default of omitted.

The function GDLGetRowCount[] returns the number of rows in the selected worksheet of the active spreadsheet. The worksheet can be specified by index or by name, and the first worksheet is used as a default of omitted.

The function GDLSetColumnCount[] sets the number of columns in the selected worksheet of the active spreadsheet. The worksheet can be specified by index or by name, and the first worksheet is used as a default of omitted. If the new column count is less than the old one, all data in the removed columns is lost.

The function `GDLSetRowCount[]` sets the number of columns in the selected worksheet of the active spreadsheet. The worksheet can be specified by index or by name, and the first worksheet is used as a default if omitted. If the new row count is less than the old one, all data in the removed rows is lost.

GoogleDataLink Functions: Returning Java Objects

Some GoogleDataLink functions return Java objects that can be used further in *Mathematica* with J/Link to allow the user to extend the features provided by GoogleDataLink:

GoogleDataLink is the Java object representing the GoogleDataLink. You can inspect all Fields, Methods, Events of the GoogleDataLink object with the JLink functions Fields, Methods, and Events.

GDLGetSpreadsheets[] returns a Java LinkedList with all the Google Docs spreadsheets:

GDLGetSpreadsheets[]

« JavaObject[java.util.LinkedList]»

These spreadsheets can be obtained individually as Java objects:

GDLGetSpreadsheets[]@get[0]

« JavaObject[com.google.gdata.data.spreadsheet.SpreadsheetEntry]»

GDLGetWorksheets[] returns a Java LinkedList with all worksheets of the active spreadsheet:

GDLGetWorksheets[]

« JavaObject[java.util.LinkedList]»

The functions GDLShowSymbols[], GDLShowUserSymbols[], and GDLTree[] (see next section) also return their respective JFrame objects.

All objects can always be inspected with the JLink functions Fields, Methods, and Events.

Google Calendar

Get a list of all calendars you own or have subscribed to:

GDLGetAllCalendarNames[]

Get a list of all calendars you own:

GDLGetOwnCalendarNames[]

Search all calendars you own and that you have subscribed to for a text string:

GDLGetEventByTextQuery[<search text>]

This will return all events in all calendars that have a match in the entry title, entry description, or entry location fields of the event.

The following shows a list of all Queen's Birthday Holidays in Australia (after having subscribed to the Australian Holidays Calendar):

```
Grid[GDLGetEventByTextQuery["Queen"], Alignment → Left, Frame → All]
```

Queen's Birthday [All Except WA]	{2010, 6, 14}	{2010, 6, 15}		
Queen's Birthday [All Except WA]	{2011, 6, 13}	{2011, 6, 14}		
Queen's Birthday	{2011, 10, 28}	{2011, 10, 29}		
Queen's Birthday [All Except WA]	{2012, 6, 11}	{2012, 6, 12}		
Queen's Birthday [WA]	{2012, 10, 1}	{2012, 10, 2}		

The following shows the local weather forecast – Chicago, Independence Day – (after having subscribed to the local weather forecast calendar)

```
Grid[GDLGetEventByTextQuery["Forecast"], Alignment → Left, Frame → All]
```

Forecast for Chicago, IL (27° 17°)	{2011, 7, 3}	{2011, 7, 4}		
Forecast for Chicago, IL (28° 18°)	{2011, 7, 4}	{2011, 7, 5}		
Forecast for Chicago, IL (30° 22°)	{2011, 7, 5}	{2011, 7, 6}		
Forecast for Chicago, IL (26° 17°)	{2011, 7, 6}	{2011, 7, 7}		

The following shows the Labor Day and Thanksgiving Holidays for the US (after having subscribed to the US Holidays Calencar)

```
Grid[Join[GDLGetEventByTextQuery["Labor"], GDLGetEventByTextQuery["Thanksgiving"]], Alignment → Left, Frame → All]
```

Labor Day	{2010, 9, 6}	{2010, 9, 7}		
Labor Day	{2011, 9, 5}	{2011, 9, 6}		
Labor Day	{2012, 9, 3}	{2012, 9, 4}		
Thanksgiving	{2010, 11, 25}	{2010, 11, 26}		
Thanksgiving	{2011, 11, 24}	{2011, 11, 25}		
Thanksgiving	{2012, 11, 22}	{2012, 11, 23}		

Add a new event to your primary calendar, using Google's natural language parser:

```
GDLQuickAddEvent["Tennis with John Aug 13 3:15pm-3:30pm"]
```



Create a new calendar, specifying an html color for the calendar to use:

```
GDLCreateNewCalendar["Little League Schedule", "Some Calendar Description", "America/Los_Angeles", False, "#2952A3", "Oakland"]
```

Delete a calendar:

```
GDLDeleteCalendar["Little League Schedule"]
```

Note that you must be the owner of a calendar to be able to delete it.

Subscribe to the Australian Holidays Calendar:

```
GDLSubscribeToCalendar["https://www.google.com/calendar/feeds/en.australian%23holiday%40group.v.calendar.google.com/public/basic"]
```

Unsubscribe from the Australian Holidays Calendar:

```
GDLUnsubscribeFromCalendar["https://www.google.com/calendar/feeds/en.australian%23holiday%40group.v.calendar.google.com/public/basic"]
```

Get a list of all events from all your calendars (the ones you own and the ones you have subscribed to) between July 1st, 2011, and July 16, 2012 (aka: "Agenda View"):

```
GDLGetEventsByDateRange[{2011,7,1,0,0,0},{2012,7,16,0,0,0}]
```

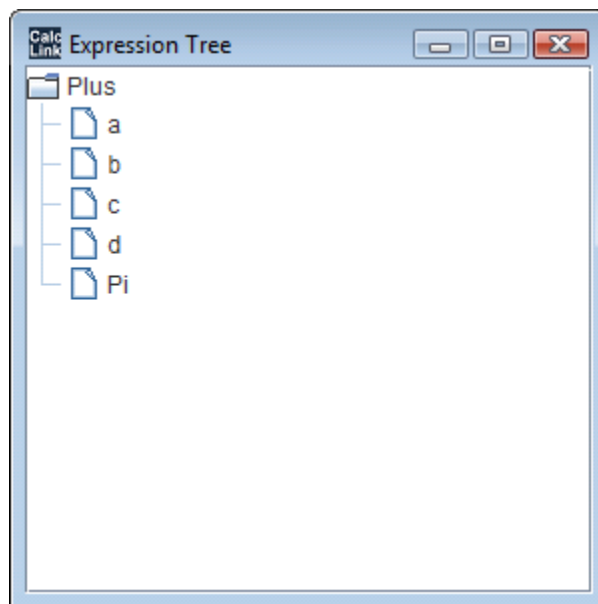
Other Utility Functions

Interactive Tree Representation/Inspection of *Mathematica* Expressions

The function `GDLTree[]` creates a new window containing a tree of the expression using expandable/collapsible tree nodes. At every node the name of the head of the expression at that level is shown. The node can be expanded to display all its nodes or leaves. Only leaves (symbols that are atomic, i. e. `AtomQ[]` is True) can not be expanded anymore.

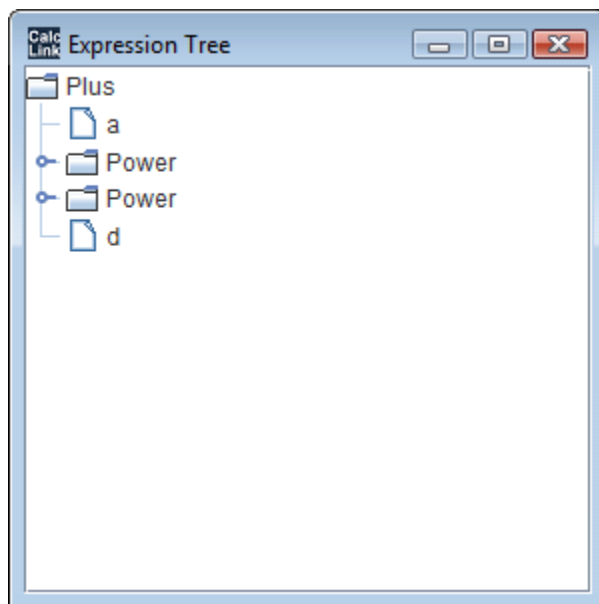
A very simple example involving only leaves under the root node:

```
GDLTree[a + b + c + d + Pi]
```



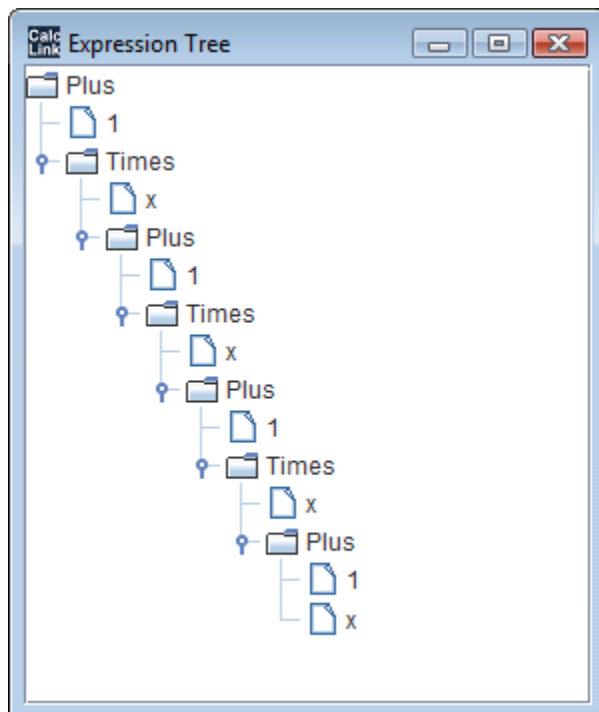
a and d are atomic, b^2 and c^3 are not:

`GDLTree[a + b^2 + c^2 + d]`



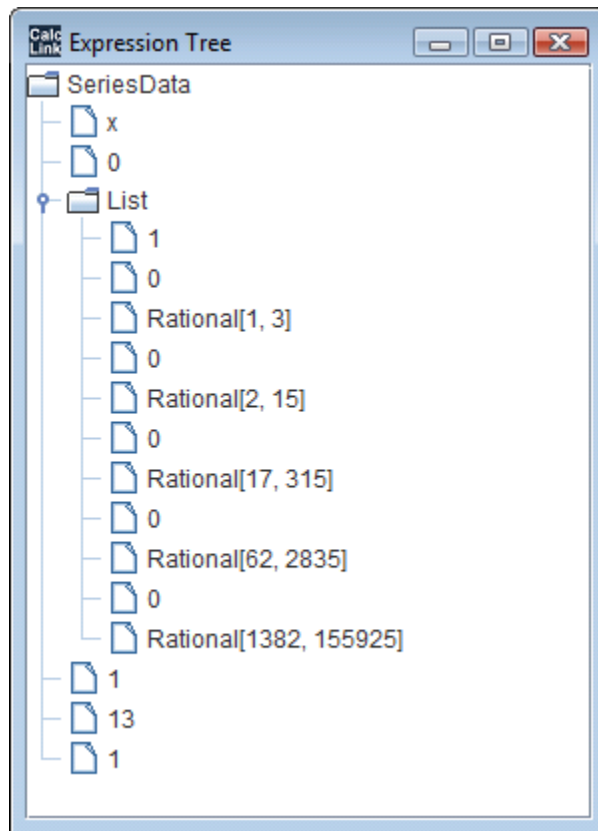
`HornerForm[]` creates an expression that has two nesting levels per order of the polynomial (minus 1).

`GDLTree[HornerForm[1 + x + x^2 + x^3, x]]`



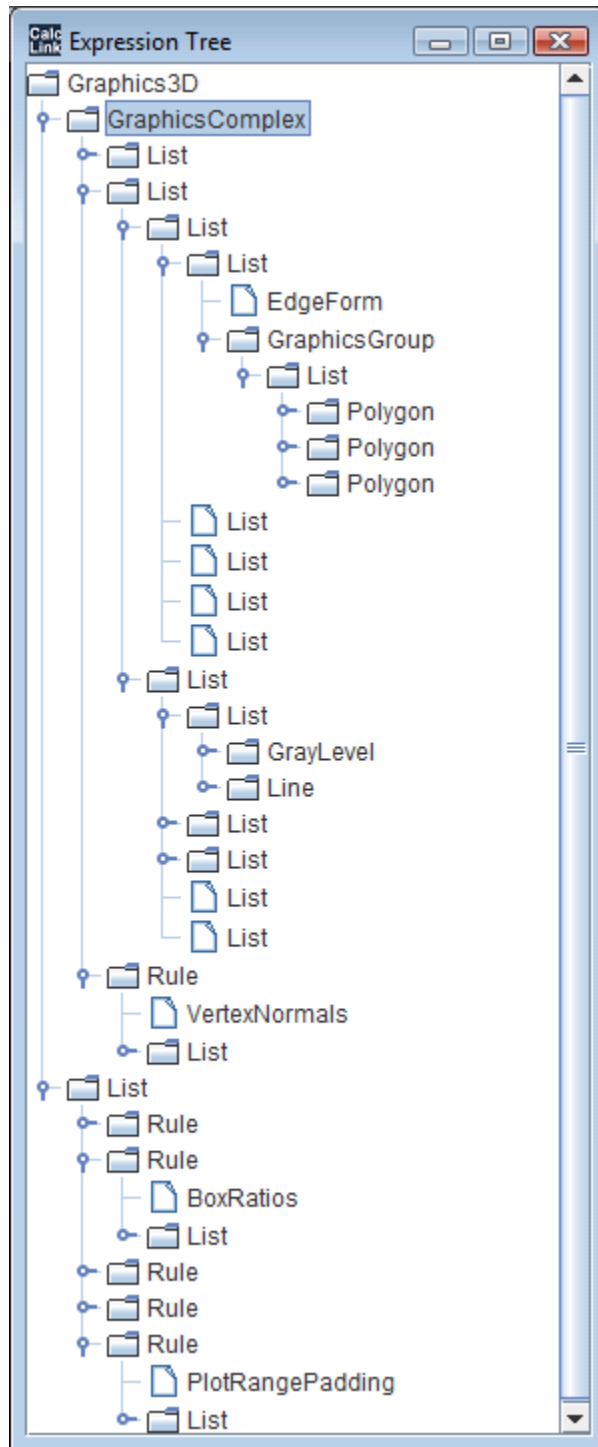
Inspect the coefficients of the Taylor series visually:

```
GDLTree[Series[Tan@x, {x, 0, 12}]]
```



The real usefulness of `GDLTree[]` becomes evident when used on complex, deeply nested expression structures. Sometimes *Mathematica* expressions can be so complex that it is very difficult to understand the nested symbol structure, so `GDLTree[]` makes it possible to "zoom in" on a branch of interest, while leaving others collapsed.

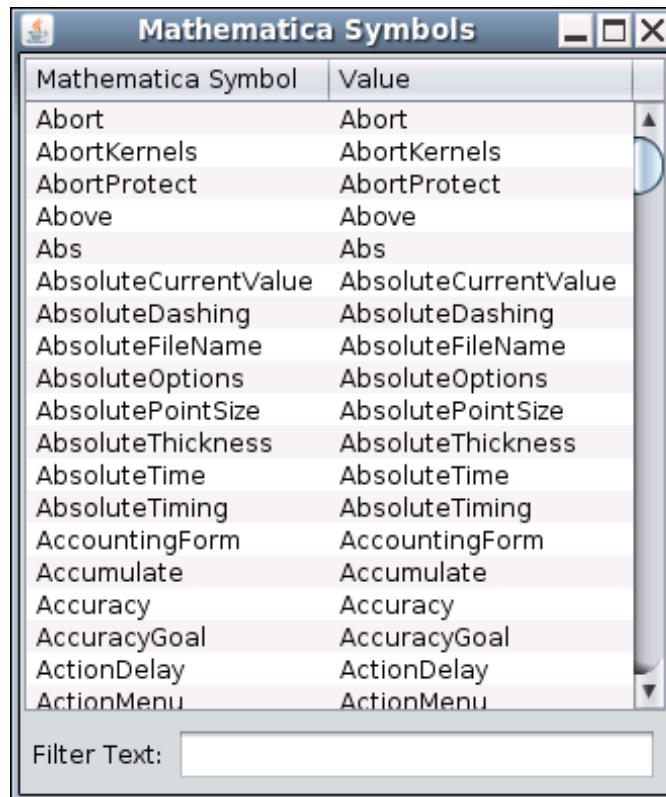
`GDLTree[Plot3D[Sin@x Sin@y, {x, -Pi, Pi}, {y, -Pi, Pi}]]`



While the *Mathematica* function `TreeForm[]` shows the expression in a "top-down" fashion in a possibly more "intuitive" and visually appealing form, the `GoogleDataLink` function `GDLTree[]` allows for interactive and selective inspection of the branches/leaves of a complex nested expression.

Interactive Symbol Browser of *Mathematica* Expressions

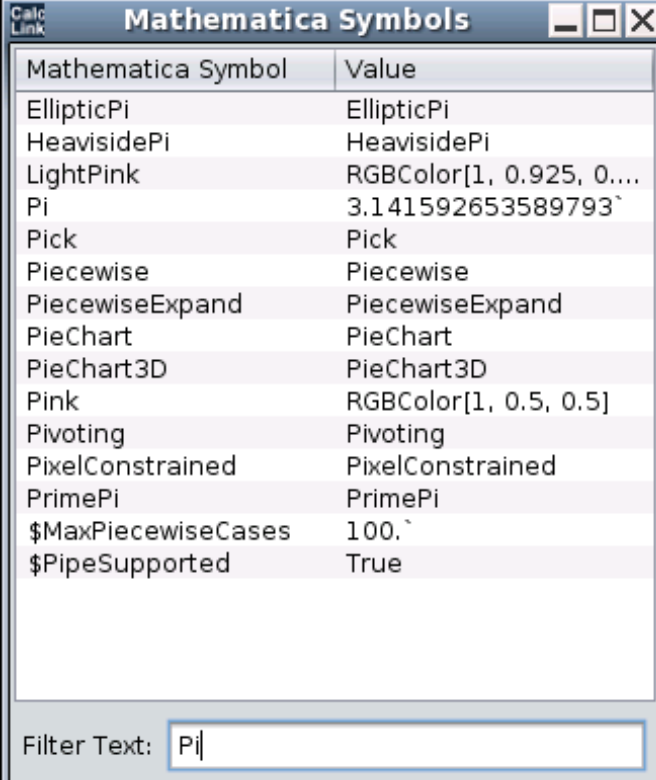
The interactive symbol browser is a very convenient tool to quickly find symbols with a certain name, or that contain a certain name.



The filter text field makes it possible to quickly find the *Mathematica* symbols and their respective values one is looking for. As the user is typing, the list of symbols is narrowed down to those that match the strings typed so far. This happens instantly and "live".

The filter text field uses Perl 5 regular expression matching, making it extremely efficient to find arbitrary patterns of *Mathematica* symbol names instantly, even very complex patterns.

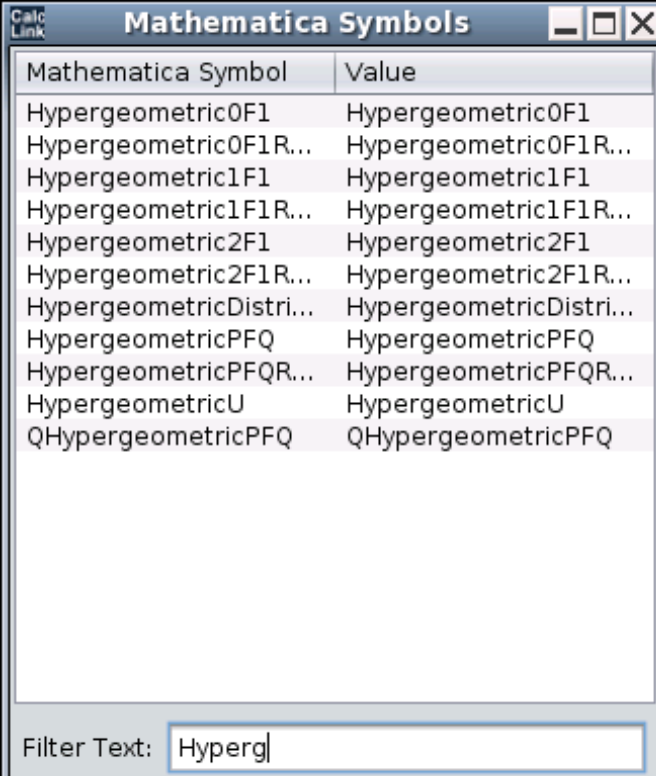
This shows all *Mathematica* symbols that contain "Pi":



The screenshot shows a window titled "Mathematica Symbols" with a search filter of "Pi". The table lists the following symbols and their values:

Mathematica Symbol	Value
EllipticPi	EllipticPi
HeavisidePi	HeavisidePi
LightPink	RGBColor[1, 0.925, 0....
Pi	3.141592653589793`
Pick	Pick
Piecewise	Piecewise
PiecewiseExpand	PiecewiseExpand
PieChart	PieChart
PieChart3D	PieChart3D
Pink	RGBColor[1, 0.5, 0.5]
Pivoting	Pivoting
PixelConstrained	PixelConstrained
PrimePi	PrimePi
\$MaxPiecewiseCases	100.`
\$PipeSupported	True

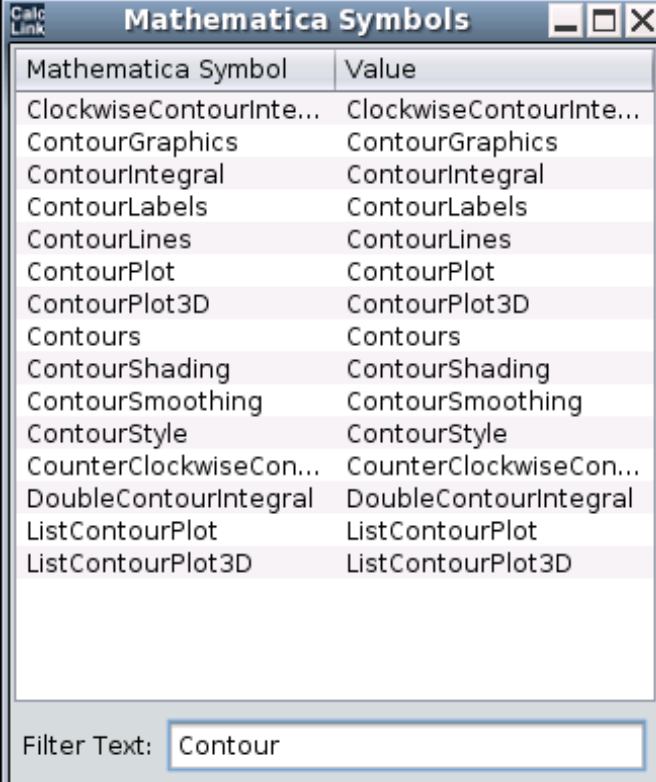
This shows all hypergeometric functions in *Mathematica*.



The screenshot shows a window titled "Mathematica Symbols" with a search filter of "Hyperg". The table lists the following symbols and their values:

Mathematica Symbol	Value
Hypergeometric0F1	Hypergeometric0F1
Hypergeometric0F1R...	Hypergeometric0F1R...
Hypergeometric1F1	Hypergeometric1F1
Hypergeometric1F1R...	Hypergeometric1F1R...
Hypergeometric2F1	Hypergeometric2F1
Hypergeometric2F1R...	Hypergeometric2F1R...
HypergeometricDistri...	HypergeometricDistri...
HypergeometricPFQ	HypergeometricPFQ
HypergeometricPFQR...	HypergeometricPFQR...
HypergeometricU	HypergeometricU
QHypergeometricPFQ	QHypergeometricPFQ


These are all *Mathematica* symbols that contain "Contour".



The screenshot shows a window titled "Mathematica Symbols" with a search filter of "Contour". The table below lists the symbols and their corresponding values.

Mathematica Symbol	Value
ClockwiseContourInte...	ClockwiseContourInte...
ContourGraphics	ContourGraphics
ContourIntegral	ContourIntegral
ContourLabels	ContourLabels
ContourLines	ContourLines
ContourPlot	ContourPlot
ContourPlot3D	ContourPlot3D
Contours	Contours
ContourShading	ContourShading
ContourSmoothing	ContourSmoothing
ContourStyle	ContourStyle
CounterClockwiseCon...	CounterClockwiseCon...
DoubleContourIntegral	DoubleContourIntegral
ListContourPlot	ListContourPlot
ListContourPlot3D	ListContourPlot3D

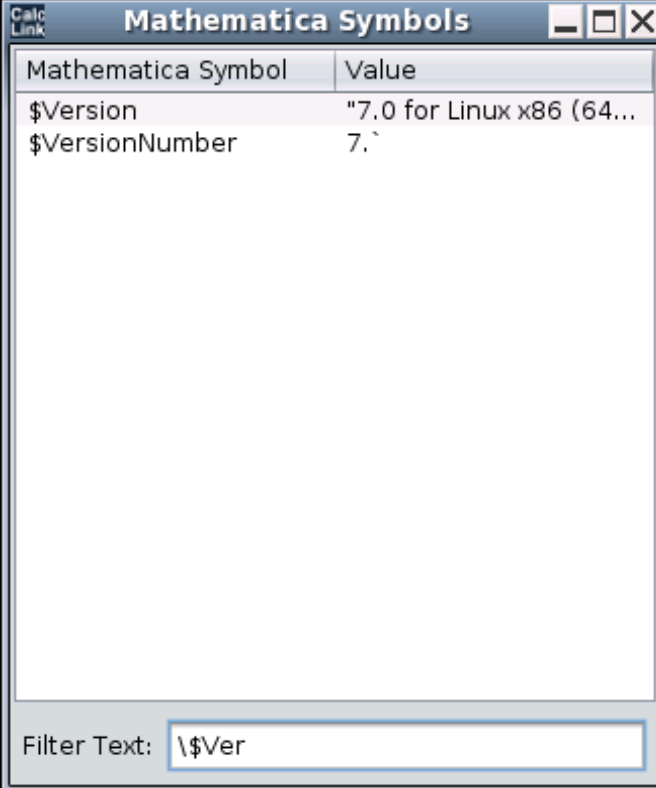
These are all *Mathematica* symbols that end with a digit character:



The screenshot shows a window titled "Mathematica Symbols" with a search filter of "\d\$". The table below lists the symbols and their corresponding values.

Mathematica Symbol	Value
AppellF1	AppellF1
Factorial2	Factorial2
googledocslinklibrari...	"/home/mooniac/javaf...
googledocslinklibrari...	"/home/mooniac/javaf...
HankelH1	HankelH1
HankelH2	HankelH2
Hypergeometric0F1	Hypergeometric0F1
Hypergeometric1F1	Hypergeometric1F1
Hypergeometric2F1	Hypergeometric2F1
Log10	Log10
Log2	Log2
SphericalHankelH1	SphericalHankelH1
SphericalHankelH2	SphericalHankelH2
SpheroidalS1	SpheroidalS1
SpheroidalS2	SpheroidalS2
StirlingS1	StirlingS1
StirlingS2	StirlingS2

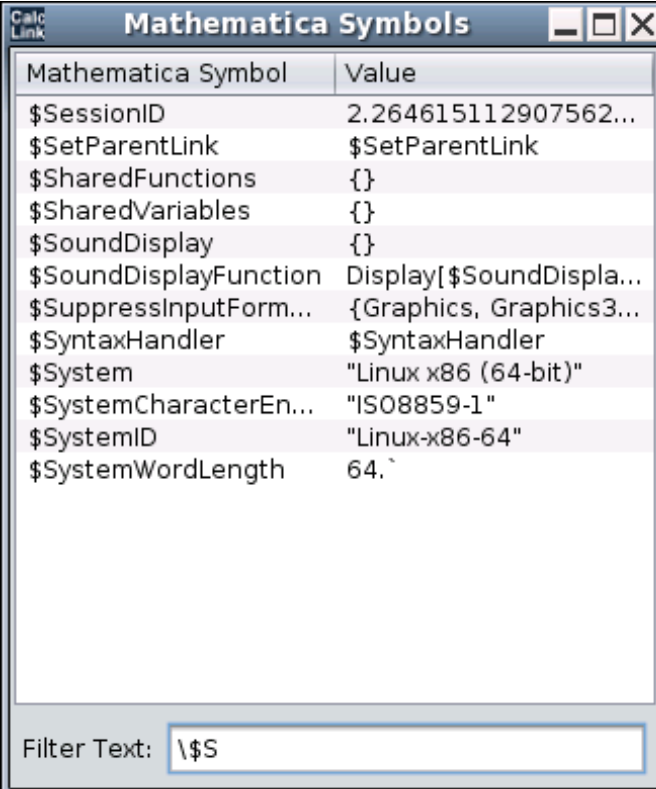
Perl 5 uses “\” as an escape character:



A screenshot of a Mathematica Symbols window. The window title is "Mathematica Symbols" and it has standard window controls. The main area contains a table with two columns: "Mathematica Symbol" and "Value". The table lists two symbols: "\$Version" with value "7.0 for Linux x86 (64..." and "\$VersionNumber" with value "7.". At the bottom, there is a "Filter Text:" label followed by a text input field containing the text "\\$Ver".

Mathematica Symbol	Value
\$Version	"7.0 for Linux x86 (64..."
\$VersionNumber	7.`

Filter Text: \\$Ver

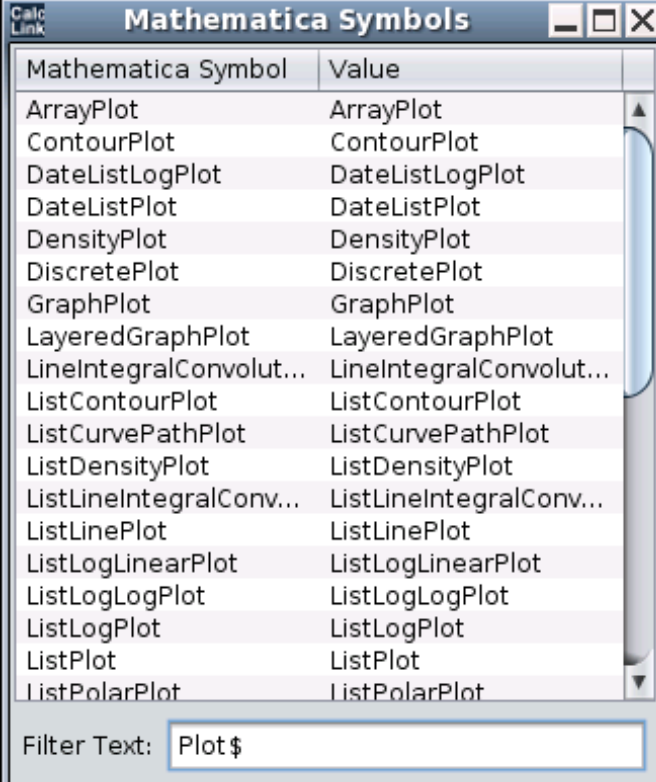


A screenshot of a Mathematica Symbols window. The window title is "Mathematica Symbols" and it has standard window controls. The main area contains a table with two columns: "Mathematica Symbol" and "Value". The table lists various system symbols such as "\$SessionID", "\$SetParentLink", "\$SharedFunctions", "\$SharedVariables", "\$SoundDisplay", "\$SoundDisplayFunction", "\$SuppressInputForm...", "\$SyntaxHandler", "\$System", "\$SystemCharacterEn...", "\$SystemID", and "\$SystemWordLength". At the bottom, there is a "Filter Text:" label followed by a text input field containing the text "\\$S".

Mathematica Symbol	Value
\$SessionID	2.264615112907562...
\$SetParentLink	\$SetParentLink
\$SharedFunctions	{}
\$SharedVariables	{}
\$SoundDisplay	{}
\$SoundDisplayFunction	Display[\$SoundDispla...
\$SuppressInputForm...	{Graphics, Graphics3...
\$SyntaxHandler	\$SyntaxHandler
\$System	"Linux x86 (64-bit)"
\$SystemCharacterEn...	"ISO8859-1"
\$SystemID	"Linux-x86-64"
\$SystemWordLength	64.`

Filter Text: \\$S

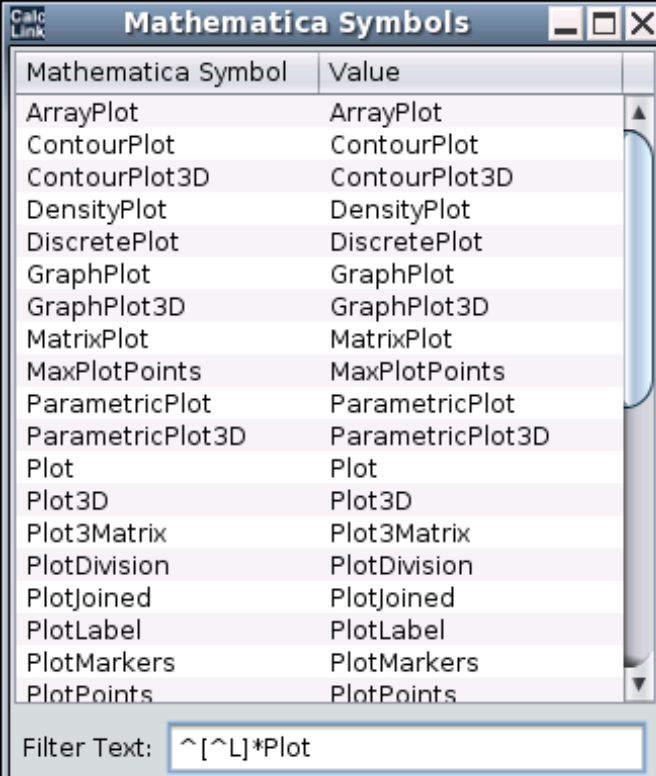
This shows all *Mathematica* symbols that end with "Plot".



The screenshot shows a dialog box titled "Mathematica Symbols" with a "Filter Text" field containing "Plot\$". The table below lists the symbols that match this filter.

Mathematica Symbol	Value
ArrayPlot	ArrayPlot
ContourPlot	ContourPlot
DateListLogPlot	DateListLogPlot
DateListPlot	DateListPlot
DensityPlot	DensityPlot
DiscretePlot	DiscretePlot
GraphPlot	GraphPlot
LayeredGraphPlot	LayeredGraphPlot
LineIntegralConvolut...	LineIntegralConvolut...
ListContourPlot	ListContourPlot
ListCurvePathPlot	ListCurvePathPlot
ListDensityPlot	ListDensityPlot
ListLineIntegralConv...	ListLineIntegralConv...
ListLinePlot	ListLinePlot
ListLogLinearPlot	ListLogLinearPlot
ListLogLogPlot	ListLogLogPlot
ListLogPlot	ListLogPlot
ListPlot	ListPlot
ListPolarPlot	ListPolarPlot

This shows all *Mathematica* symbols that contain "Plot" but don't begin with "L".



The screenshot shows a dialog box titled "Mathematica Symbols" with a "Filter Text" field containing "^([L])*Plot". The table below lists the symbols that match this filter.

Mathematica Symbol	Value
ArrayPlot	ArrayPlot
ContourPlot	ContourPlot
ContourPlot3D	ContourPlot3D
DensityPlot	DensityPlot
DiscretePlot	DiscretePlot
GraphPlot	GraphPlot
GraphPlot3D	GraphPlot3D
MatrixPlot	MatrixPlot
MaxPlotPoints	MaxPlotPoints
ParametricPlot	ParametricPlot
ParametricPlot3D	ParametricPlot3D
Plot	Plot
Plot3D	Plot3D
Plot3Matrix	Plot3Matrix
PlotDivision	PlotDivision
PlotJoined	PlotJoined
PlotLabel	PlotLabel
PlotMarkers	PlotMarkers
PlotPoints	PlotPoints

The GoogleDataLink function `GDLShowUserSymbols[]` does the same as `GDLShowSymbols[]`, but it only shows the user and package symbols, `GDLShowSymbols[]` shows all symbols.