Mathematica Quick Guide

**Brackets**

*Single Square Brackets*  
[ ]  
Used with functions. Place input parameters and options inside the brackets. Individual parameters and options are separated by commas.

*Curly Brackets*  
{ }  
Used to create a list. Items in a list are separated by commas. A table is a list of lists, where the first list is the first row, the second list is the second row, and so on.

*Double Square Brackets*  
[[ ]]  
Used to select items from a variable that contains a list, table, or matrix. Use numbers, double semicolons (;;), and commas to select rows, columns, or individual cells. Rows first, columns second, separated by commas. \( Data[[\text{row, col}]] \).

*Parentheses*  
( )  
Parentheses in Mathematica are like in Algebra, they determine the order in which a calculation is carried out. \( (1 + 2) / 3 \)

**Functions to repeat things or to create programming loops**

*Table*  
\[ \text{Table[stuff to repeat, \{iterator\}]} \]  
Table repeats everything that comes before the comma as many times as required by the format of the iterator. Iterator often has a incremental variable that is used on the left side. For example,

\[ \text{Table} [x, \{x, 1, 10, 2\}] \]

has an iterator that starts and 1, ends at 10, and jumps by increments of two. For each iteration, \( x \) is set to the next value in the sequence. Here: \( x=1, x=3, x=5, x=7, x=9 \). This example would print these values because \( x \) is listed on the left side of the comma. The results are presented in a list, like this:

\[ \{1, 3, 5, 7, 9\} \]
\textit{Do[ stuff to repeat, \{ iterator \}]}  \hspace{1cm} \text{Do is exactly like table, except that it doesn't save the results or report them unless told to do so by the functions before the comma.}

\textit{Map[ function[#] \&, list ]}  \hspace{1cm} \text{Map repeats a function for every item in a list by sequentially substituting the next item for the \# symbol. The \& indicates that everything that comes before it should be repeated. For example:}

\[
\text{Map[ \# * 150 \&, \{10, 20, 30, 40\}]
\]

\text{Will multiply each of the numbers in the list at the right by 150. The results will be returned as a list of numbers:}

\text{\{1500, 3000, 4500, 6000\}}

\textit{Function[#]\& /@ list}  \hspace{1cm} \text{This is an alternative form of the \textit{Map[]} function. It behaves the same except you don't need to literally type “Map” or its square brackets and you substitute /@ for the comma separating the function and the list.}

\textbf{Functions for transforming lists and tables}

\textit{Flatten[list]}  \hspace{1cm} \text{Flatten collapses lists of lists (for example, a table) into a single list of items.}

\textit{Partition[list, n]}  \hspace{1cm} \text{This divides a simple list into parts of length } n. \text{ For example:}

\text{Partition[list, 2]} \text{ groups items in the list into pairs.}

\textit{Transpose[list]}  \hspace{1cm} \text{This function takes a table-shaped list of rows and columns and switches it to columns and rows.}

\textbf{Graphics Function}

\textit{ListPlot[list]}  \hspace{1cm} \text{This function produces a bare-bones bivariate scatter plot. It has options to adjust the size of the plot, to add labels to the axes, to change colors, etc.}

\textit{Graphics[ lists of “primitives” ]}  \hspace{1cm} \text{This function allows you to assemble custom graphics composed of lists of “primitives”, which are functions for Point[], Text[], Disk[], Line[], Polygon[], etc. You can build almost any kind of complicated graphic by assembling different kinds of primitives. They must all be enclosed in curly brackets if you use more than one. Lists within lists are ok.}
Especially useful functions

Select – This function selects items from a list or table by a criterion that you specify (e.g., \( #==10 \), \( #>2 \), OddQ, EvenQ, etc.)

Cases – Similar to select, but uses special Pattern Matching criteria (search Mathematica Help Guide for Pattern Matching for more info)

GatherBy – Similar to the Group function in MySQL. This function groups data by one of the columns.

Sort – Sorts data. See options for advanced ways to sort.

Map – Repeats the same operation on every item in a list. (see above)

Table – Builds a list by repeating operations and saving the result. Can be used as a looping structure in a program.

Do – Repeats operations. Similar to Table, but doesn’t automatically save results.

Graphics – This is the main command for creating graphics. It has many, many options and is very flexible.