LISTS AND SEQUENCES

North American GeoGebra Journal Editorial Staff

1 LISTS

A **list** in GeoGebra is an ordered set that can contain repetitions. For example, $\{1, 2, 3\}$ and $\{2, 3, 1\}$ are both lists, but they are not equal since the elements of the list (namely 1, 2, and 3) are in a different order. Note, too, that lists needn't be lists of numbers. Instead, they can contain other objects such as numbers, coordinate pairs, polygons, and conics. We explore several examples below.

1.1 Defining Lists in GeoGebra

To define a list in GeoGebra, enter items in curly brackets.

Example 1. *A list containing the six elements* $\{1, 2, 3, 3, 3, 2\}$ *in GeoGebra would be*

`L1 = {1,2,3,3,3,2}`.

Example 2. *A list of points* $\{(0, 1), (2, 3), (4, 5)\}$ *in GeoGebra would be*

`L2 = {(0,1),(2,3),(4,5)}`.

Example 3. *It is also possible to create a list of lists. For instance, using* `L1` *and* `L2` *as defined above*

`L3 = {L1,L2}`.

In GeoGebra, this results in the following definition: `L3={{1,2,3,3,3,2},{(0,1),(2,3),(4,5)}}`

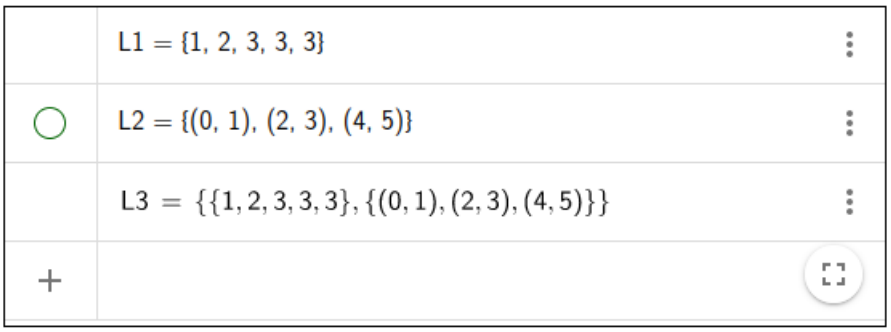

Figure 1. Example lists in GeoGebra (available at https://tinyurl.com/list-examples).

In some cases it does not make sense to allow repetitions as it would geometrically not produce an unique graphic. For example, the list of points in Example 2 would not produce a different image than `L4={(0,1),(0,1),(2,3),(4,5)}`.

To obtain the nth element of a list `L`:

```
Element[L,n].
```

2 List Operations

Several operations can be performed lists including comparisons, artihmetic, sorting, and searching (see Section 4 for more comprehensive list of operations). Let's first consider some comparisons.

2.1 Comparison Operators

To determine if two lists are equal, use the double equal sign, i.e., ==. The result is a boolean expression (`True` or `False`), and thus can be used in a logical expression. List equality is illustrated in Example 4.

Example 4. *Let* $\{1, 2, 3\}$ *and* $\{2, 1, 3\}$ *be two lists of integers. Then, to check if the two lists are equal we would have,*

$$\{1,2,3\}==\{2,1,3\}.$$

The operator, `!=`, is used to determine when two lists are *not* equal. We illustrate the use of both of these operators in a short video tutorial at `https://tinyurl.com/eq-not-eq`.

2.2 Sort and Unique

Often it is desirable to sort a list or select only the unique elements. The `sort` and `unique` commands are used to that end. For instance, we can sort list `L1` or display its unique elements using the commands illustrated below.

Example 5. *Recall that* `L1 = {1,2,3,3,3,2}`. *The following commands sort and display unique elements from* `L1`*, respectively.*

- `L4=sort(L1)`
- `L5=unique(L1)`

The results of these operations are illustrated in Figure 2.

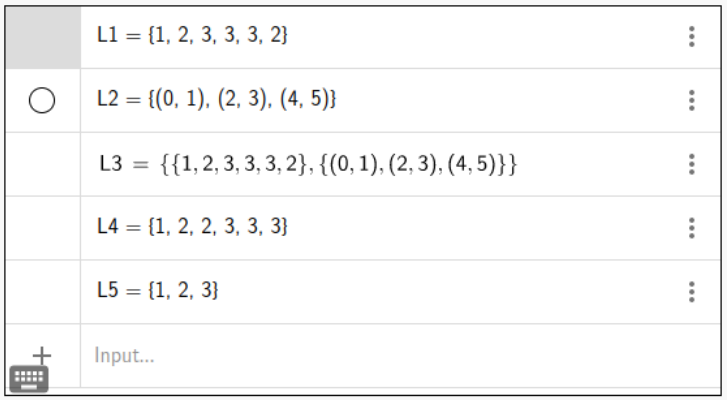

Figure 2. Results of `Sort` and `Unique` commands (available at https://tinyurl.com/sort-unique).

2.3 Vectorized Commands

Lists can be used as input to standard mathematical functions to apply the function to the entire list at once. In particular, given a list `L` and function $f(x)$, GeoGebra supports computation of $f(L)$. Some examples include,

Example 6. *Let* $L = \{1, 2, 3\}$. *The following will return lists:* `L^2`, `sin(L)`, `2^L`, *and* `sqrt[L]`.

GeoGebra has built-in statistical functions that can be applied to lists. These include: mean (`mean`), median (`Median`), mode (`Mode`), lower quartile (`Q1`), upper quartile (`Q3`), maximum (`Max`), minimum (`Min`), standard deviation `stdev`, and variance (`Variance`).

Example 7. *Let* $Grades = \{80, 75, 80, 95, 65, 75, 80\}$. *To calculate the mean, median, and mode of the grades, enter at the command prompt:* `mean(Grades)`, `Median(Grades)`, *and* `Mode(Grades)`.

2.4 Matrices

Matrices are represented in GeoGebra as a "list of lists". In particular, a matrix is a *list* of rows where each row is a list of elements in the row. An example of a 3×3 matrix is given below.

Example 8. *Define the matrix below using lists in GeoGebra.*

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

`A = {{1,2,3},{4,5,6},{7,8,9}}`.

3 SEQUENCES

The `Sequence` command creates a list. The syntax is given by:

`Sequence[f(k),k,start,end]`.

where `f(k)` is an expression of the index `k` that starts and ends at the values `start` and `end`.
The follow examples illustrate the versatility sequence command.

Example 9. *A sequence of points five points on the vertical line* $x = 2$ *can be generated by the command,*

`Sequence[(2,k),i,1,5]`.

An optional increment parameter can be added to change the step size as shown with the additional parameter 0.25 in the following.

Example 10. *A sequence of*

`Sequence[(2,k),i,1,5,0.25]`.

Example 11. *Simpler syntax can be used to generate a list of consecutive integers. In particular, the list* $\{1, 2, 3, 4, 5\}$ *is generated by,*

```
Sequence[5].
```

Example 12. *The first five powers of two (i.e.,* $2, 4, 8, 16, 32$*) are therefore,*

```
2^Sequence[5].
```

Example 13. *The sequence command isn't limited to generating numerical sequences. The following example from the GeoGebra Institute of Hong Kong (ober) illustrates the use of the sequence command to generate animations through repeated rotation.*

```
Sequence[Rotate[poly1, k*t, A], k, 1, 10].
```

The result of this command is illustrated in Figure 3.

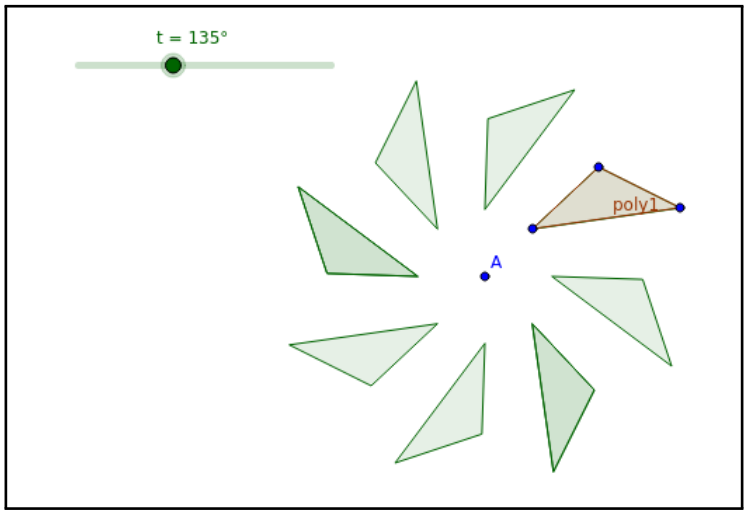

Figure 3. The result of rotating `poly1` 10 times with the sequence command (available at https://tinyurl.com/rotation-sequence).

4 List and Sequence Commands

In the following section, we provide a number of popular list and sequence commands for your reference (N.A., 2012). Interested readers are encouraged to visit the GeoGebra Reference Wiki at `https://wiki.geogebra.org/en/Sequence_Command` for more details.

- `Append[List, Object]` – Appends the object to the list.

  Example: `Append[{1,2,3},4]` → `{1,2,3,4}`

- `CountIf[Condition, List]` – Counts elements in list if condition is satisfied.

  Example: `CountIf[x<3,{1,2,3,4,5}]` → 2, (i.e., the number of elements less than 3).

- `Element[List, n]` – Returns the nth element in the list.

  Example: `Element[{1,4,3,2},2]` → (i.e., 4).

- `First[List, n]` – Returns list with the first n elements.

  Example: `First[{1, 2, 3, 4, 5}, 3]` → `{1,2,3}`.

- `Insert[Object, List, Position]` – Inserts the object into the list at given position.

  Example: `Insert[5,{1,4,3,2},2]` → `{1,5,4,3,2}`.

- `Intersection[List1,List2]` – Returns a list with common elements between the two lists.

  Example: `Intersection[{1,4,3,2}, {1,2,7,9}]` → `{1,2}`.

- `IterationList[Function, Number x, Number n]` – Returns list length $n + 1$: $x, f(x), f(f(x)), \ldots, f^{(n)}(x)$ where the notation $f^{(n)}$ is function composition n-times.

  Example: Suppose $f(x) = x^2$. Then `IterationList[f, 2, 3]` → `{2,4,16,256}`.

- `Join[List of lists]` – Joins two lists into one.

  Example: `Join[{1,2,3}, {4,5,6}]` → `{1,2,3,4,5,6}`

- `KeepIf[Condition, List]` – Creates list containing only elements meeting given condition.

  Example: `KeepIf[x>2,{1,2,3,4,5}]` → `{3,4,5}`

- `Last[List]` – Returns the last element in the list.

  Example: `Last[{1,2,3}]` → `{3}`.

- `Length[List]` – Returns the length of the list.

  Example: `Length[{1,2,3}]` → `3`

- `Min[List]` – Returns the minimum of the list.

  Example: `Min[{1,2,3}]` → `1`.

- `Max[List]` – Returns the maximum of the list.

  Example: `Max[{1,2,3}]` → `3`.

- `Product[List]` – Returns the product of all numbers in the list.

  Example: `Product[{1,2,3}]` → `6`.

- `Reverse[List]` – Returns the list in revere order.

  Example: `Reverse[{1,2,3}]` → `{3,2,1}`.

- `Sequence[Expression, Index, Lower, Upper, <increment>]` – Returns a list containing the sequence using the given expression with index over the range lower to upper. The increment value is optional.

  Example: `Sequence[2*i+1, i, 0, 5]` produces the list of the first six odd numbers starting at 1, i.e., `{1,3,5,7,9,11}`.

- `Sort[List]` – Sorts the list. If the list consists of ordered pairs, the elements are sorted by the x-coordinate. If strings are given, then sorted alphabetically.

  Example: `Sort[{3,1,7}]` $\rightarrow$ `{1,3,7}`

  `Sort[{(3,2),(1,7),(4,1)}]` $\rightarrow$ `{(1,7),(3,2),(4,1)}`

  `Sort[{"red", "green", "blue"}]` $\rightarrow$ `{"blue", "green", "red"}`

- `Sum[List]` – Calculates the sum of all numbers in the list.

  Example: `Sum[{3,1,7}]` $\rightarrow$ `11`.

- `Sum[List]` – Calculates the sum of all numbers in the list.

  Example: `Sum[{3,1,7}]` $\rightarrow$ `11`.

- `Union[List 1, List 2]` – Returns a list with all elements of the two lists, duplicates removed.

  Example: `Union[{1,4,3,2}, {1,2,7,9}]` returns `{1,2,3,4,7,9}`.

REFERENCES

GeoGebra Institute of Hong Kong (2021, October). Using sequence command. `https://www.geogebra.org/m/Jkx7gakV`. Accessed: 2021-10-10.

N.A. (2012). Getting to lists (presentation handout). In *Proceedings of the 2012 GeoGebra Midwest Conference*.