11-Python-Intro to Sort

Introduction to Sort

3 December 2025

Sorting Things Out

This article outlines a suggestion that categorization may be hardwired into our brains. In other words, thousands of years of evoluation may shape the urge for us to sort things out based on different characteristics.

Outlining the steps we use to sort through collections can be quite challenging. We're going to look at one particular algorithm that is probably the most straightforward.

Selection Sort

Selection sort can be used to order data in ascending order (or descending).

For this example, consider the following data, organized in a list. Note that these are numbers:

```
[1, 43, 55, -11, 100, 34]
```

Selection sort (in ascending order) can be described in this way:

1. Starting at the beginning of the list
2. Set this entry as the lowest
3. Checking the rest of the list:
    1. If the current entry is lower than the lowest number
        1. Mark this location as the lowest
        2. Set the lowest number to the current number
    2. Move to the next item
    3. Repeat
4. When we've reached the end, swap this entry with the lowest location
5. Move to the next entry and repeat steps 2 to 4
6. If we've reached the end, we're done!

Let's see how that works with the list above.

```
[1, 43, 55, -11, 100, 34]
```

```
Setup
-----
 0   1   2    3    4   5
[1, 43, 55, -11, 100, 34]

lowest_num   = 1
lowest_index = 0

First Comparison (* indicates the entry we're sorting, %% indicates the
current entry we're comparing)
----------------
 *  %%   2    3    4   5
[1, 43, 55, -11, 100, 34]

lowest_num   = 1
current_num  = 43
lowest_index = 0

"current_num isn't lower, move on to the next one"


2nd Comparison
--------------
 *   1  %%    3    4   5
[1, 43, 55, -11, 100, 34]

lowest_num   = 1
current_num  = 55
lowest_index = 0

"current_num still isn't lower, move on to the next one"

3rd Comparison
--------------
 *   1   2   %%    4   5
[1, 43, 55, -11, 100, 34]

lowest_num   = 1
current_num  = -11
lowest_index = 0

"current_num is now lowest! set lowest_num to -11, and lowest_index to 3"
```

```
New values:
lowest_num = -11
lowest_index = 3

4th and 5th Comparison (we'll combine these two because we can see that
they're not the smallest)
--------------
 *   1   2    3   %%  %%
[1, 43, 55, -11, 100, 34]

lowest_num   = -11
current_num  = 100  ... current_num = 34
lowest_index = -3

"These two aren't lowest, we've reached the end!"

SWAP!   (Smallest number is indicated with an S)
-----
 *   1   2    S    4   5
[1, 43, 55, -11, 100, 34]

"Swap * with S."

New Values:   (-- indicates that this part of the list is already sorted)
  --   1   2  3    4   5
[-11, 43, 55, 1, 100, 34]
```