### Sorting Algorithms Visualizer

### Selection Sort, Bubble Sort and Insertion Sort

Photo by [Sophie Elvis](https://unsplash.com/@thetechnomaid?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com?utm_source=medium&utm_medium=referral" \t "_blank)

We are all familiar with what sorting a group of numbers mean, i.e whether to arrange a list of numbers in ascending order or in descending order. Normally, we can do it manually. But what if we have a long list of numbers that need to be sorted?

This is when Sorting algorithms come into the picture. Algorithms like Bubble Sort, Insertion Sort, Selection Sort, Quick Sort, and Heap Sort are some of the sorting techniques which make it easier to sort a given input list of numbers.

Here’s a brief description of the above-said sorting algorithms which will make it easier for you to implement and understand in the Algorithm visualizer.

**1)Selection Sort:** The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.

a] The subarray which is already sorted.

b] Remaining subarray which is unsorted. In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.

**Time Complexity:**

**Best Case: O(n2)**

**Worst Case: O(n2)**

**Space Complexity: O(1)**

**2)Bubble Sort:** This is one of the simplest sorting algorithms to exist so far which carries out its execution by repeatedly swapping the elements adjacent to each other if they are not in the correct order. This algorithm is mostly used for small data sets ie the list of numbers which has fewer elements in it.The reason is, the average and worst-case time complexity of this algorithm are quite high.

**Time Complexity:**

**Best case: The array is already sorted => O(n)**

**Worst case: O(n2)**

**Space Complexity: O(1)**

**3)Insertion Sort:** This is also a simple sorting technique in which the given array is virtually split into two parts: a sorted part and an unsorted part. Values/ numbers from the unsorted part are taken and then placed at the correct position in the sorted part of the array. This process keeps on repeating until all the numbers in the array are at the correct positions.

**Time Complexity:**

**Best case: O(n)**

**Worst case: O(n2)**

**Space Complexity: O(1)**

**4)Quick Sort:** This is a Divide and Conquer algorithm. It picks a random element as a pivot and partitions the given list of numbers around the picked pivot.

Pivot can be considered as any one of the following:

1. Taking the first element as the pivot.
2. Taking the last element as the pivot.
3. Pick a random element as a pivot.
4. Pick the median of the list as the pivot.

**Time Complexity:**

**Best case: O (n\*logn)**

**Worst case: O(n2)**

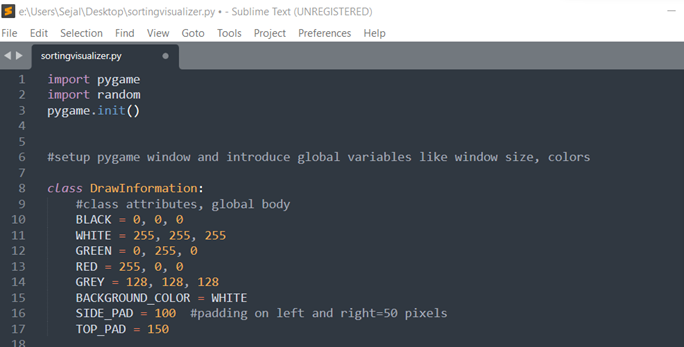
**Space Complexity: O(log n)**

I’ve used python language to implement the Sorting Algorithm Visualizer. This visualizer contains options like Reset — to reset the sorting operation, Start Sorting — to begin the sort, Ascending order — to sort the list in ascending order, Descending order — to sort the list in descending order, an option to carry out Insertion sort, an option for Bubble sort and one for Selection sort.

**Firstly, need to import two libraries namely:** [**pygame**](https://en.wikipedia.org/wiki/Pygame) **and random.**

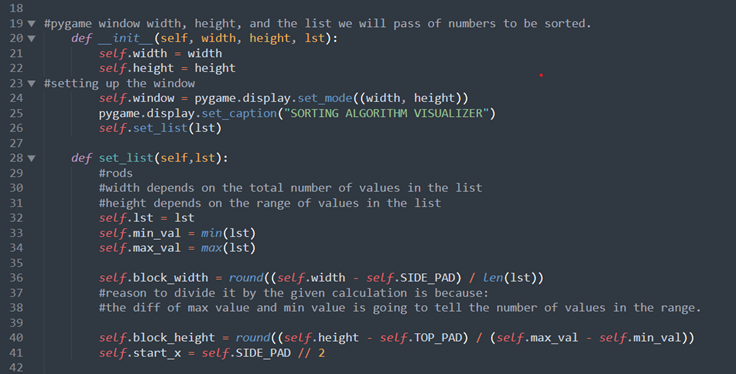
* [**pygame**](https://en.wikipedia.org/wiki/Pygame) **library** in python is an open-source module for the Python programming language specifically intended to help programmers make games and other multimedia applications. It can run across many platforms and operating systems.
* [**random**](https://python.fandom.com/wiki/Random) **library** in python is a built-in module that can be used to generate random numbers.

To begin with, the setup of [pygame](https://en.wikipedia.org/wiki/Pygame) window needs to be done. For that, many global variables like window size, and colors to be used in the visualization are declared under a class.

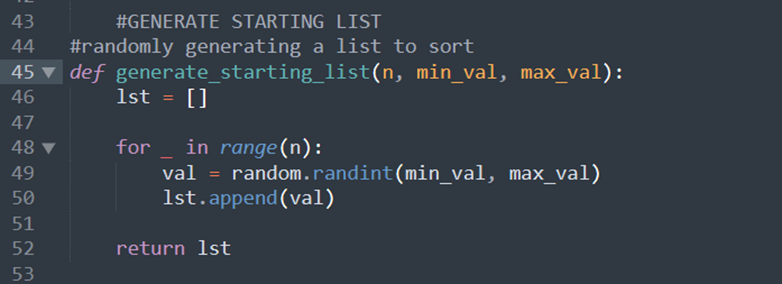
Setting up global variables under DrawInformation class

Next, define the width and height of the bars, where each bar represents a number present in the list. Here, padding also needs to be defined to get a better view of the visualization window.

**NOTE:** The top-left corner of the screen is coordinate (0,0). As you go down, y coordinate increases, and as you go horizontally the x coordinate increases.

Building up of the pygame window

Next, define a list that consists of random numbers. This is the list on which we’ll apply the sorting algorithms. Here, the user can also give his/her own list of numbers, for that other piece of code is to be implemented.

Generating starting list

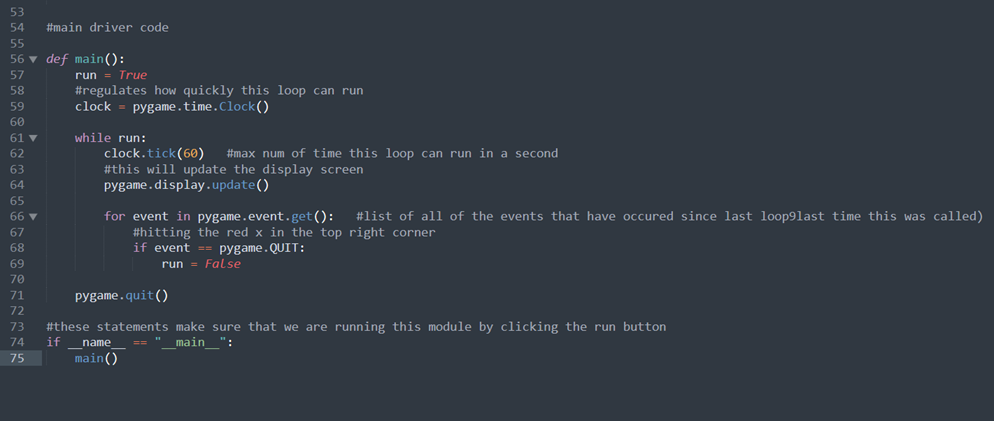
Here, min\_val is the minimum number present in the list, and max\_val represents the maximum number present in the list. The variable ’n’ represents the total number of elements the list should contain.

This function ‘generate\_starting\_list’ will return a list of random numbers which are already shuffled.

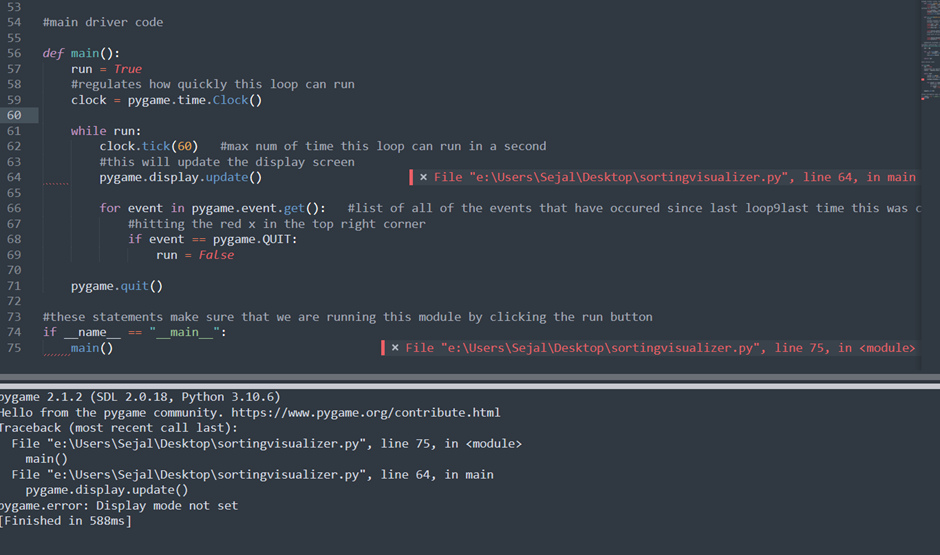
We’ll now implement the main driver code. Here, we are going to render the screen, set up the main event loop which is going to allow us to click buttons, and then draw the list onto the screen.

Whenever you implement something in pygame, you need a loop that runs constantly in the background to keep the program continuously under execution.

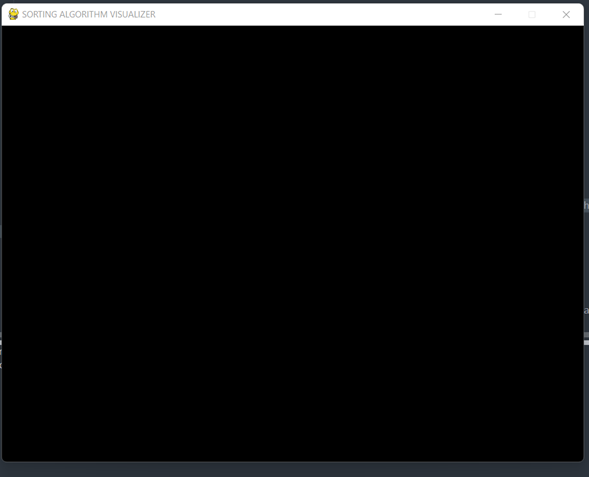
Next, define the main function which handles the execution of the entire code.

main function

**Problem 1**: When you run the above-written code, you’ll receive an error message stating “Display mode not set”. This error appeared because we’ve not yet instantiated the window where we will be drawing stuff. So, before updating the window, first, we need to create it. And to create the window, we need to first instantiate the DrawInformation() function.

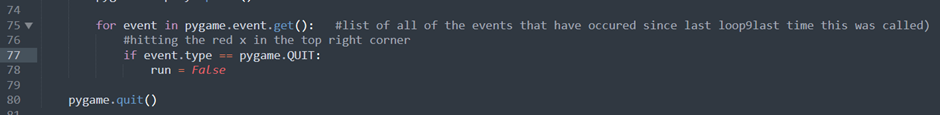


After generating the starting list and instantiating the DrawInformation() method, the pygame window named “ Sorting Algorithm Visualizer” appears on the screen.

pygame window

**Problem 2**: But the problem with this is that the close window button doesn’t work in this case. You’ll have to manually close it.

To solve this type of issue, add event type property rather than only defining the event only.



Now, the pygame window closes when the close button is pressed.

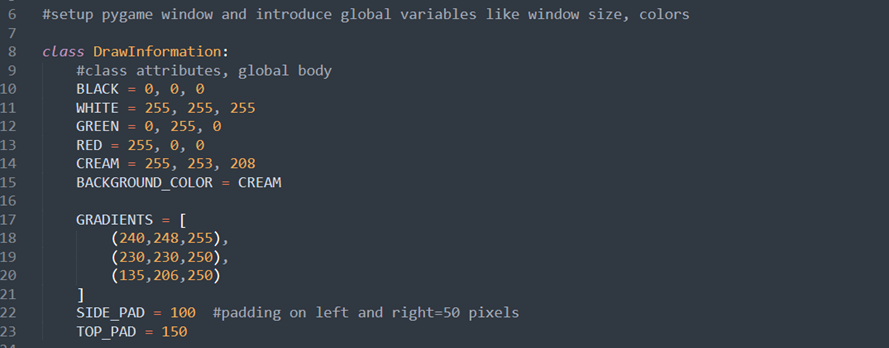
**Drawing the list:**

Two functions need to be defined: One function for generally drawing the window, and the other for drawing the list specifically.

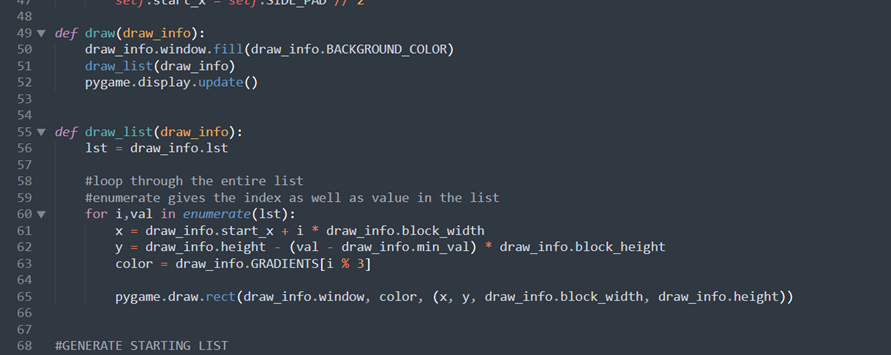
In pygame, the way drawing works is to draw a bunch of stuff on the screen, and then as soon as you want to apply and actually see it, you update the display.

To draw the list, we have to look at every single element of the list. We have to determine the height and x-coordinate of the element and draw a rectangle(block) representing it and we also need to make sure that we are drawing all the elements' rectangles in a slightly different color. For that, we’ll define 3 colors — GRADIENTS.

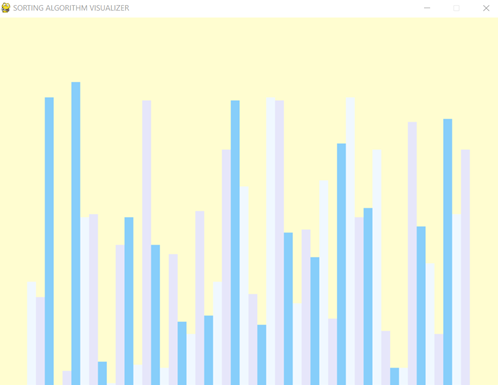
Here, I’ve updated the background color to cream and defined the gradient colors for the blocks which represent the numbers in the list.

defining gradient colors — represents numbers in the list

Defining the draw and draw\_list functions.

defining draw and draw\_list functions

Output so far is:

updated pygame window

The next task is to set the Sorting Controls. That means some keys on the keyboard will be given some functionality to carry out:

**R — Reset the list**

**A — Arrange the list in Ascending Order**

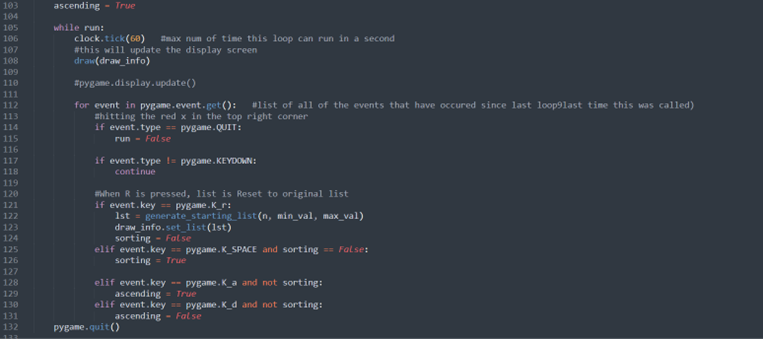
**D — Arrange the list in Descending Order**

**B — Bubble Sort**

**I — Insertion Sort**

**S — Selection Sort**

**SPACE — Start Sorting**



**event.type != pygame.KEYDOWN** meansFor not pressing any key, we are going to continue i.e we are going to the next event in for loop.

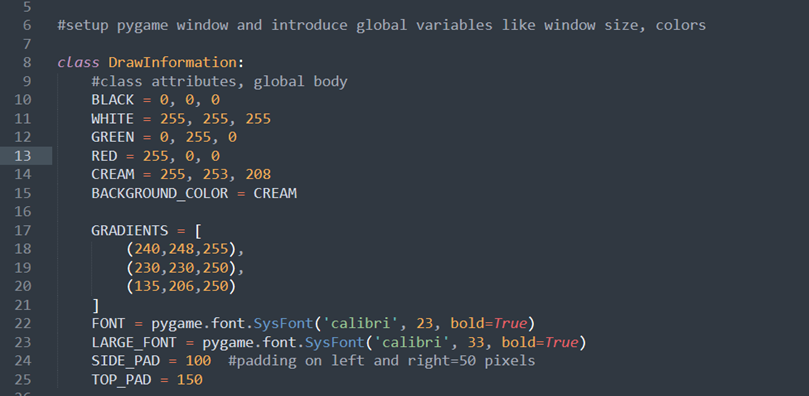
If you do have a keydown i.e if any key is pressed, another if statement will be executed.

For example, if we want to sort the list in ascending order, we’ll press A. We check this by writing it as **event.key == pygame.K\_a**

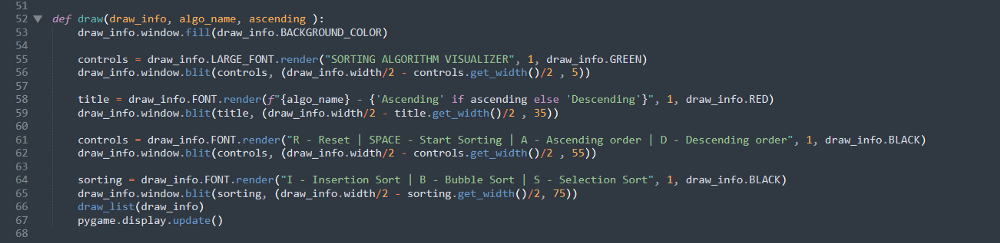
Similarly, for SPACE, we’ll write the condition as:

**event.key == pygame.K\_SPACE**

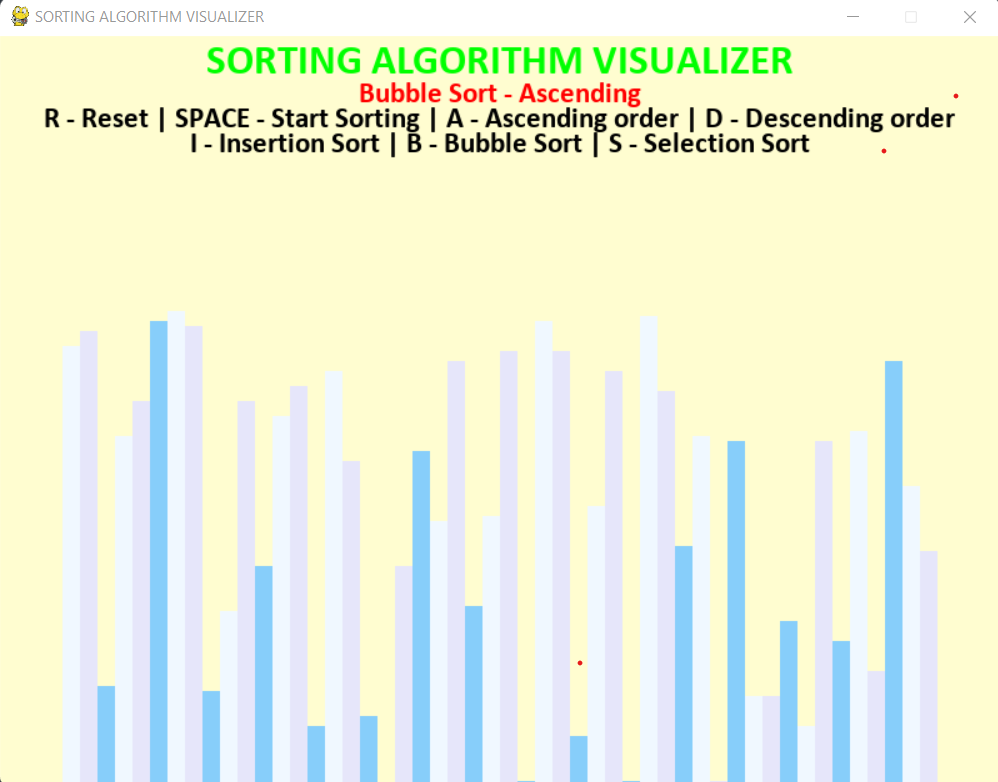
**To display the contents on the top of the pygame screen, we define FONT.**

defining font sizes

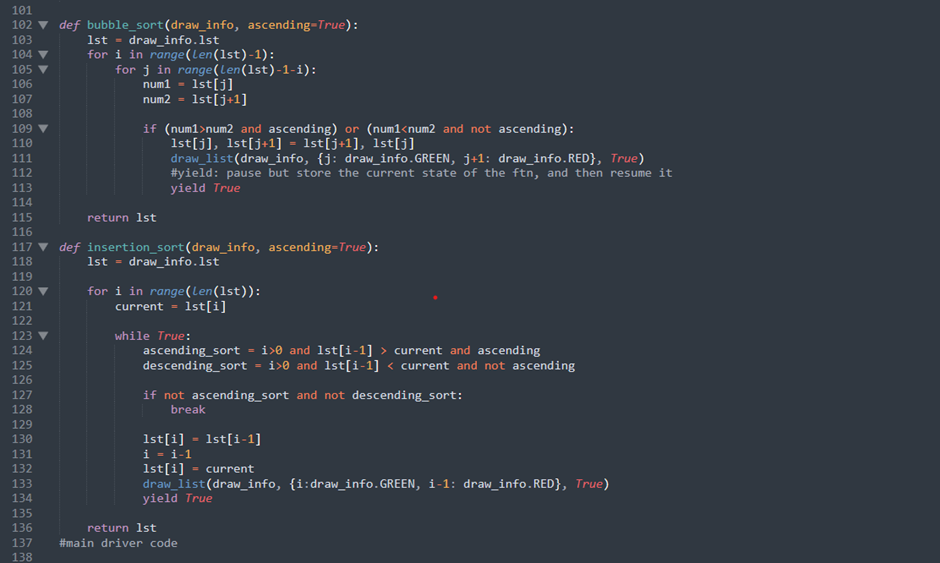
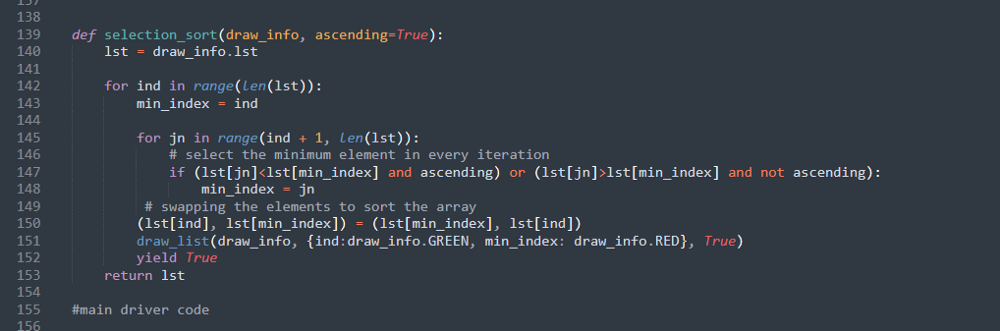
And then, write down the stuff you want to display on the window, in the draw\_list function.



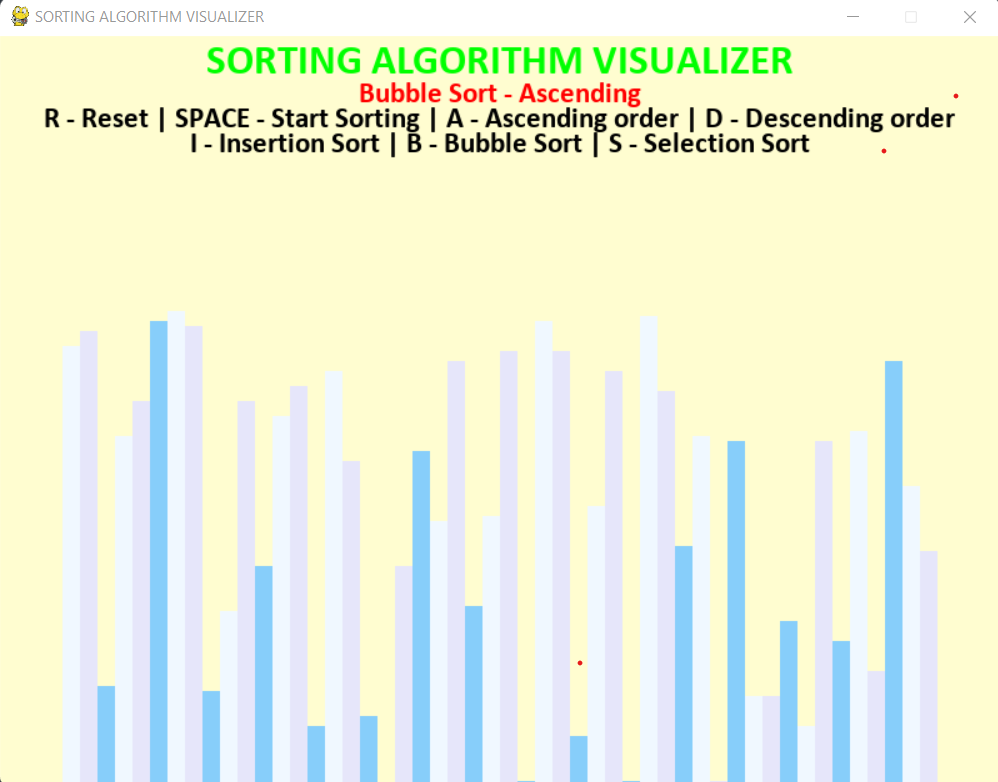
**Output till here:**



Next up, make two functions namely bubble\_sort, insertion\_sort and selection\_sort for covering up the sorting algorithm.

Bubble sort and Insertion sort functionsSelection sort

**The final Output for the code is :**



Final Output on the screen

Visualization videos links are posted below:

1] Bubble Sort: Ascending Order — <https://youtu.be/rnJ74ZZgvpU>

2] Bubble Sort: Descending Order — <https://youtu.be/qmwpXWF_ijg>

3] Insertion Sort: Ascending Order — <https://youtu.be/jGzB37X7pfs>

4] Insertion Sort: Descending Order — <https://youtu.be/J8UNIiMUcX0>

5] Selection Sort: Ascending Order — <https://youtu.be/X0zSlPk6Nek>

6] Selection Sort: Descending Order — <https://youtu.be/D_qVjSWY5Qc>

Now, you have a visual overview of how bubble sort and Insertion sort works. Of course, you can implement other sorting algorithms and see how they work and understand them better by visualizing them.

I hope you enjoyed reading it and try it yourself. Let me know your thoughts and your ideas about it!

The entire code is available on my github profile: [**https://github.com/SejalLoya/Sorting-Algorithm-Visualizer**](https://github.com/SejalLoya/Sorting-Algorithm-Visualizer)

I’ve referred “Tech with Tim” Youtube channel for creating this project.

**The reference link for the project is:** [**https://www.youtube.com/c/TechWithTim**](https://www.youtube.com/c/TechWithTim)