Sorting Visuals

DSA Project-1 for Impact Training

Introduction

- **Sorting Visuals** is a visualizer project that helps us to understand the concepts of different **sorting algorithms** in a very **fun** & **easy** way.
- It uses **Python** & **Pygame** development.
- By controlling the **types** & **order** of sorting lets the user to understand multiple concepts at the same time

Problem domain

- **Sorting algorithms** are one of the most interesting topics of **DSA** to learn but are not as simple as they look.
- Basically, **sorting process** the main definition of it is same for all the **Sorting algorithms** but the method and way to perform them is different and so is to **understand them**.

Solution domain

- **Sorting Visuals** helps us to understand **sorting algorithms** in an easier & fun way by combining both **Logic** with **Visuals.**
- It shows how the base logic of sorting works using these all different ways and methods.

Requirements

- **Software requirements:**
    - PyCharm
    - Install Pygame
- **Hardware requirements:**

- **Processor:** Any modern-day processor
- **Memory:** 2 GB is enough
- **Storage:** 200 MB is enough

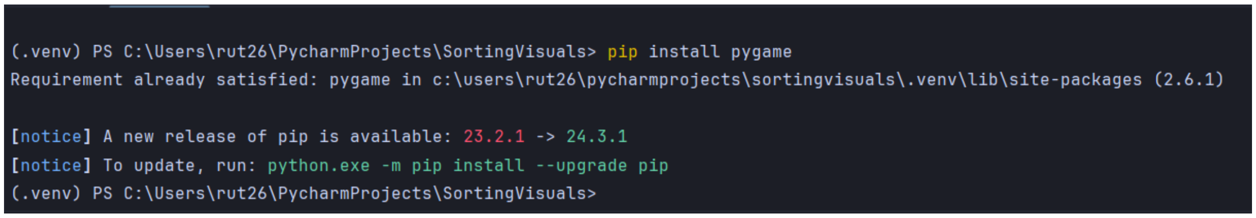
DSA Concept

- The main **DSA** concept used for this project is **sorting.**
- **Sorting** simply means to set the randomly given list of numbers in order.
- The order can be either ascending or descending based on the situation.
- This project particularly has 3 main **sorting algorithms:**
  1. **Insertion sort**
  2. **Bubble sort**
  3. **Merge sort**

Methodology (includes Screenshot)

- Breakdown of the whole process into different steps:

STEP – 1:



```
(.venv) PS C:\Users\rut26\PycharmProjects\SortingVisuals> pip install pygame
Requirement already satisfied: pygame in c:\users\rut26\pycharmprojects\sortingvisuals\.venv\lib\site-packages (2.6.1)

[notice] A new release of pip is available: 23.2.1 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(.venv) PS C:\Users\rut26\PycharmProjects\SortingVisuals>
```

First, we must set-up **PyCharm** as the IDE and code editor and install **Python.**

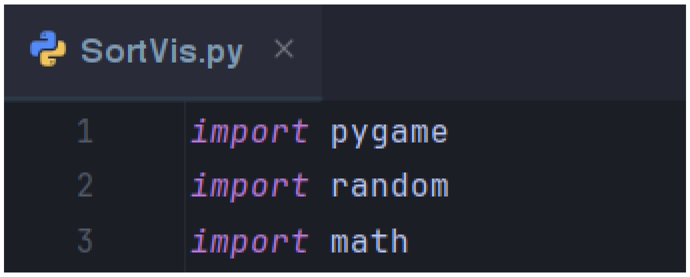
Next do set up and create a new project with a python file, name it **SortingVis.**

Now comes the most important part to install **Pygame** as the whole UI and controls are created and displayed by it.

To install **Pygame** this command is used:

pip install pygame

STEP – 2:



SortVis.py

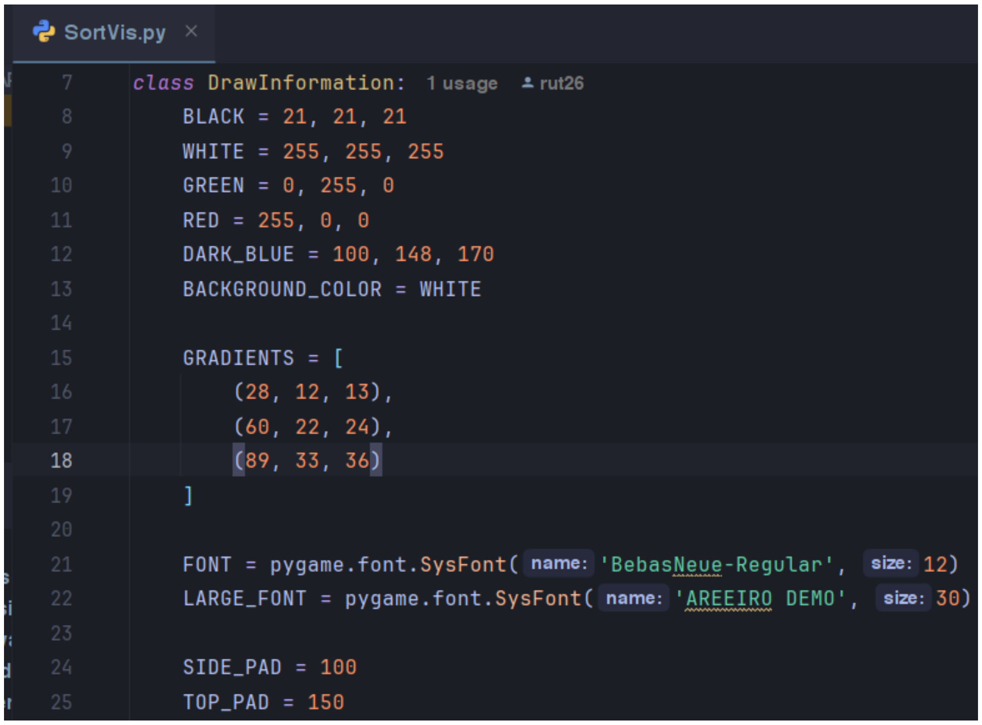
```python
import pygame
import random
import math
```

Now import this necessary **Python libraries,** here 3 main libraries are need:

1. **Pygame**
2. **Random**
3. **Math**

These libraries save time by providing built-in functions instead of writing everything from scratch and make the development process faster.

STEP – 3:



SortVis.py

```python
class DrawInformation:  1 usage  rut26
    BLACK = 21, 21, 21
    WHITE = 255, 255, 255
    GREEN = 0, 255, 0
    RED = 255, 0, 0
    DARK_BLUE = 100, 148, 170
    BACKGROUND_COLOR = WHITE

    GRADIENTS = [
        (28, 12, 13),
        (60, 22, 24),
        (89, 33, 36)
    ]

    FONT = pygame.font.SysFont( name: 'BebasNeue-Regular',  size: 12)
    LARGE_FONT = pygame.font.SysFont( name: 'AREEIRO DEMO',  size: 30)

    SIDE_PAD = 100
    TOP_PAD = 150
```

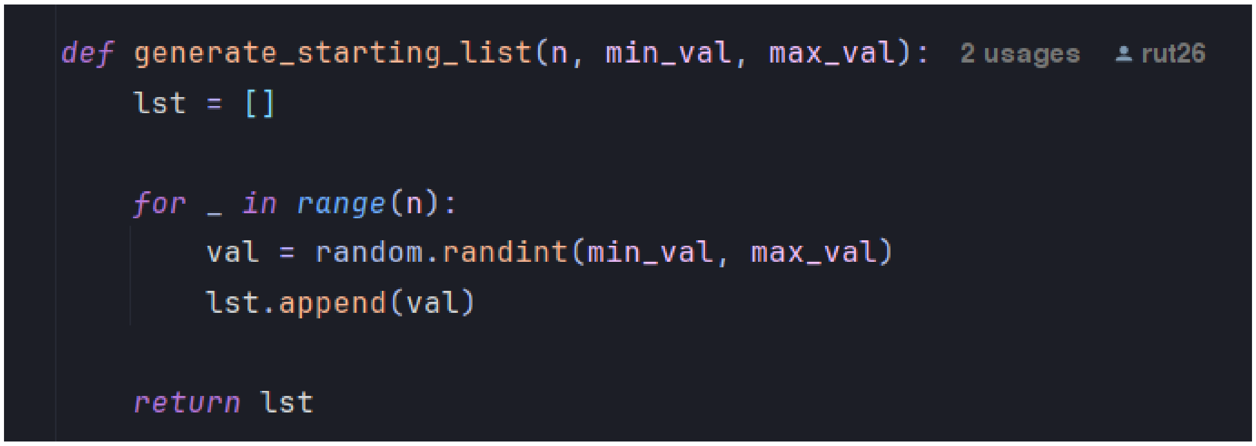
Create the **DrawInformation** class to manage colors, fonts, padding, and the data visualization window.

Here we define 3 different shades of dark Red for the rectangle bars.

Before that we declare 6 variables of 6 different colors that will help us in the UI part.

And at last, we take 2 variables for padding.

STEP – 4:

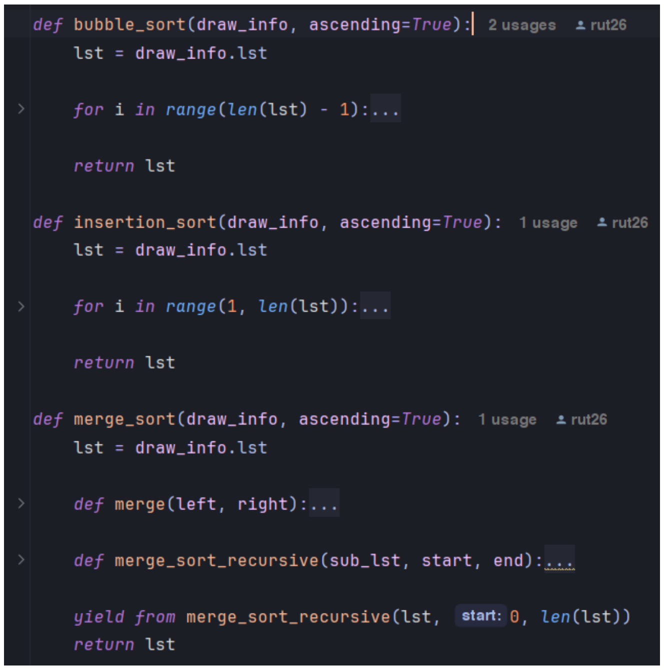


```
def generate_starting_list(n, min_val, max_val):  2 usages  rut26
    lst = []

    for _ in range(n):
        val = random.randint(min_val, max_val)
        lst.append(val)

    return lst
```

Now comes the main logic part of the code.

Here we take a **generate_starting_list** function that will randomly generate list on which the sorting process will be performed.

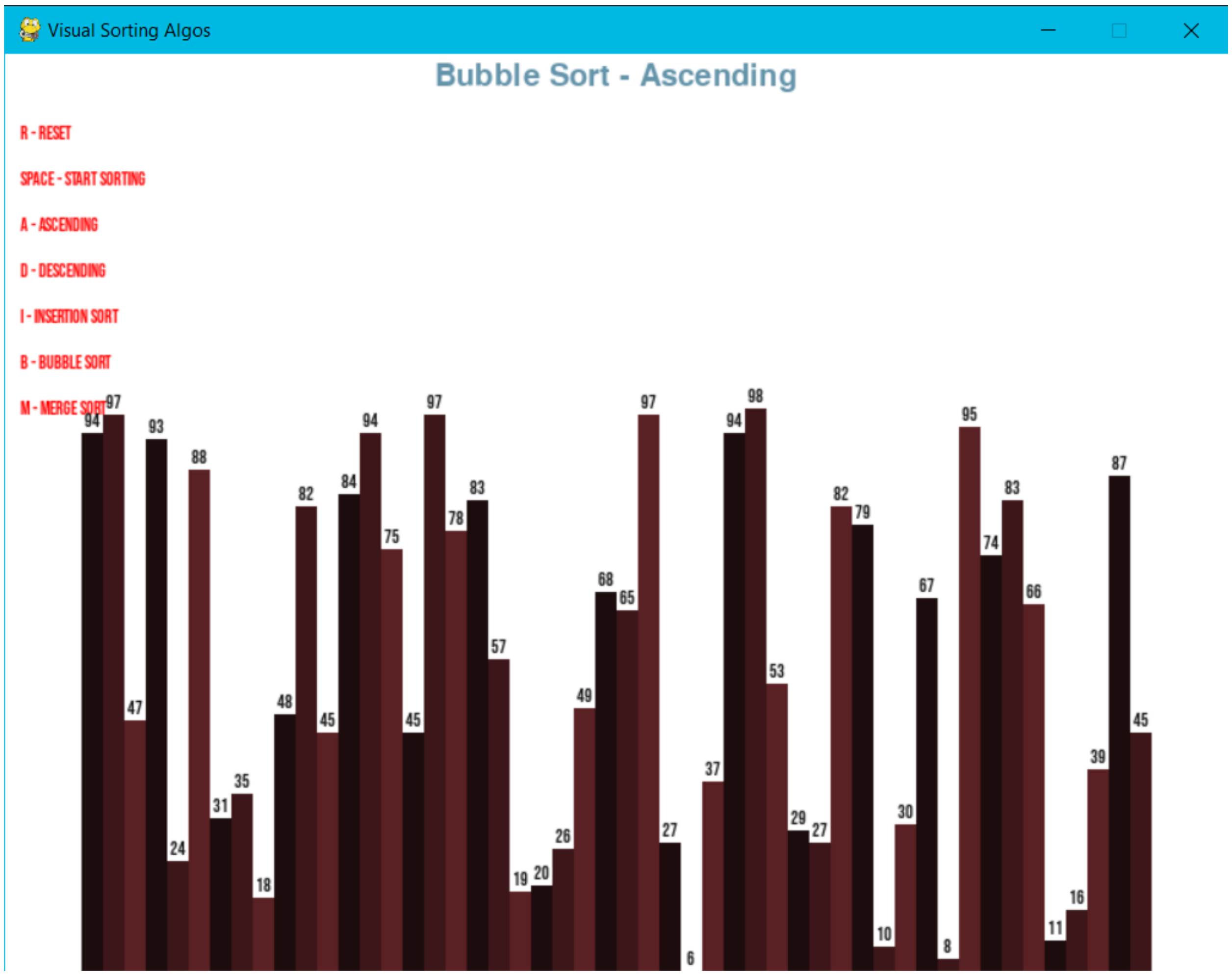
STEP – 5:



```
def bubble_sort(draw_info, ascending=True):  2 usages  rut26
    lst = draw_info.lst

>   for i in range(len(lst) - 1):...

    return lst

def insertion_sort(draw_info, ascending=True):  1 usage  rut26
    lst = draw_info.lst

>   for i in range(1, len(lst)):...

    return lst

def merge_sort(draw_info, ascending=True):  1 usage  rut26
    lst = draw_info.lst

>   def merge(left, right):...

>   def merge_sort_recursive(sub_lst, start, end):...

    yield from merge_sort_recursive(lst, start: 0, len(lst))
    return lst
```

Here is the main and most crucial part of the code, this part will handle the logic for all three sorting algorithms.

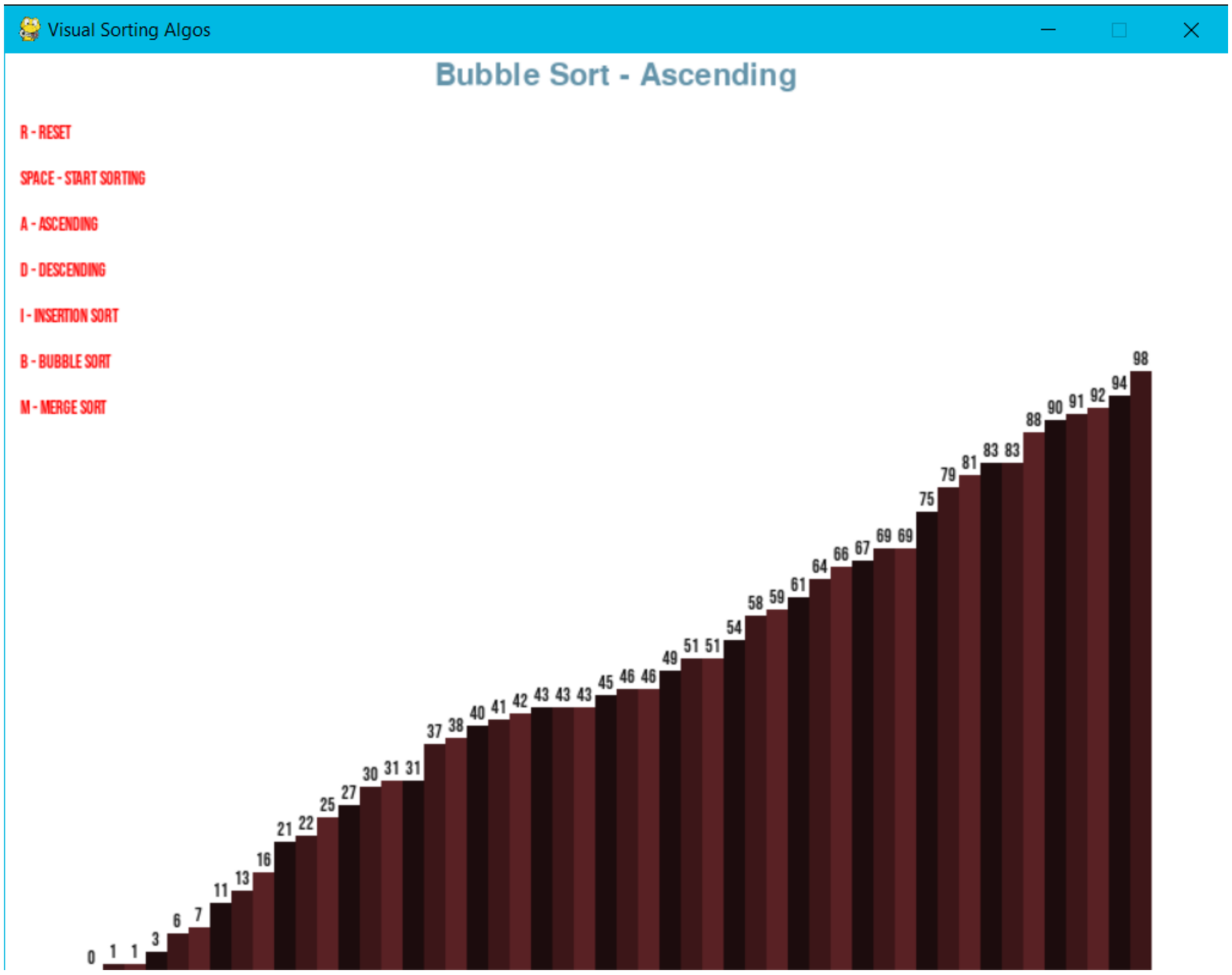
STEP – 6:



After some final changes and error solving, here is the final window in which the project will run.

All the control instructions are stated in the window with the main title that is being solved.

Here are the elements after the sorting process:



Summary

- **Sorting** is one of the most **important** concepts of **DSA.**
- This **DSA** project helps us to understand the **Sorting algorithms** more easily by using **visualization** and lets us control the title and ascending to descending order.
- This project has **3** main sorting algorithms like **Insertion sort, Bubble sort, Merge sort**.
- All 3 sorts have different **logic**, **method**, **Time** and **Space Complexities**.
- It has different **controls** to use different methods and ways to perform **sorting.**
- **Pygame** is used to create windows and create UI for this project.
- It takes random numbers as a list, even in random order and that further is used to perform sorting.

Conclusion

- At the end **visuals projects** like this help us to understand concepts easily and makes it more fun to learn.
- This project particularly solves the problem of understanding complex concept of **Sorting algorithms.**
- I learned multiple concepts throughout the development of this project and even my understanding for sorting algorithms.