This script visually demonstrates the **Bubble Sort** algorithm using Matplotlib for plotting the bar heights in real-time. It's suitable for environments like Google Colab, where GUI libraries like Tkinter cannot be used.

**Purpose:**

This script visually demonstrates the **Bubble Sort** algorithm using Matplotlib for plotting the bar heights in real-time. It's suitable for environments like Google Colab, where GUI libraries like Tkinter cannot be used.

**Code Breakdown:**

**1. Importing Libraries**

import matplotlib.pyplot as plt

import random

import time

* **matplotlib.pyplot**: Used for plotting the bar chart to represent the array being sorted.
* **random**: Generates random integers for the initial dataset.
* **time**: Introduces a delay to create an animation effect.

**2. Bubble Sort Visualization Function**

def bubble\_sort\_visualization(data):

n = len(data)

for i in range(n - 1): # Outer loop for each pass

for j in range(n - i - 1): # Inner loop for comparison

if data[j] > data[j + 1]: # Compare adjacent elements

data[j], data[j + 1] = data[j + 1], data[j] # Swap if out of order

visualize(data) # Visualize the current state of the array

time.sleep(0.1) # Pause for 0.1 seconds to slow down visualization

* **Outer loop**: Iterates n-1 times, where n is the number of elements. This is because the last element gets sorted in each pass.
* **Inner loop**: Compares adjacent elements (data[j] and data[j + 1]) and swaps them if they’re in the wrong order.
* **visualize(data)**: Called after every swap to redraw the bar chart showing the updated array.
* **time.sleep(0.1)**: Adds a delay to make the changes visible as an animation.

**3. Visualization Function**

def visualize(data):

plt.bar(range(len(data)), data, color="yellow") # Draws the array as a bar chart

plt.pause(0.05) # Briefly pauses to update the plot

plt.clf() # Clears the previous plot to redraw

* **plt.bar**: Creates a bar chart where the indices are on the x-axis, and values (heights) are on the y-axis.
* **plt.pause(0.05)**: Allows the current plot to be shown for a short duration before being cleared.
* **plt.clf()**: Clears the figure so the next frame can be drawn without overlap.

**4. Generate Random Data**

data = [random.randint(10, 100) for \_ in range(20)]

* Creates a list of 20 random integers between 10 and 100.

**5. Initialize the Plot**

plt.ion() # Turn on interactive mode for live updates

visualize(data) # Show the initial array

* **plt.ion()**: Enables Matplotlib's interactive mode for dynamic updating of plots.
* **visualize(data)**: Displays the initial state of the array.

**6. Call Bubble Sort Visualization**

bubble\_sort\_visualization(data)

plt.ioff() # Turn off interactive mode after sorting

plt.show() # Display the final sorted array

* **bubble\_sort\_visualization(data)**: Sorts the array and visualizes the process.
* **plt.ioff()**: Disables interactive mode.
* **plt.show()**: Displays the final sorted bar chart.

**Output Explanation**

1. Initially, a bar chart displays the random, unsorted array.
2. As the sorting progresses, the bars are updated to show the swapping of elements in real-time.
3. After all passes of the Bubble Sort algorithm, the bars are arranged in ascending order.

