Insertion Sort visualization using Matplotlib in Python:

```
import matplotlib.pyplot as plt
import numpy as np
import time
def insertion_sort_visualization(data):
    fig, ax = plt.subplots()
    bars = ax.bar(range(len(data)), data)
    ax.set_xlim(0, len(data))
    ax.set_ylim(0, int(1.1 * max(data)))
    ax.set_title("Insertion Sort Visualization")
    ax.set_xlabel("Index")
    ax.set_ylabel("Value")
    for i in range(1, len(data)):
        key = data[i]
        j = i - 1
        while j >= 0 and data[j] > key:
            data[j + 1] = data[j]
            j -= 1
            bars[j + 1].set_height(data[j + 1])
            plt.pause(0.1)
        data[j + 1] = key
        bars[j + 1].set_height(data[j + 1])
        plt.pause(0.1)
    plt.show()
np.random.seed(42)
data = np.random.randint(low=1, high=100, size=10)
insertion_sort_visualization(data)
```

In the above code, we first import the required libraries: `matplotlib.pyplot` as `plt` for visualization and `numpy` as `np` for generating random data.

The `insertion_sort_visualization()` function takes an array of data as input and performs the insertion sort algorithm. It creates a bar plot using Matplotlib to visualize the sorting process. The plot is updated at each step to show the swapping and the sorted portion of the array.

The main steps in the `insertion_sort_visualization()` function are as follows:

1. Create the initial bar plot using `ax.bar()`.

2. Iterate over the array, starting from the second element.

3. Compare the current element with the elements before it and shift them if necessary.

4. Update the plot to visualize the swapping or the sorted portion of the array using `bars[j].set_height()`.

5. Pause for a short time (`plt.pause(0.1)`) to visualize each step of the sorting process.

Finally, we generate random data using `np.random.randint()` and call the `insertion_sort_visualization()` function with the data array.

Approaches for Insertion Sort Visualization:

1. Matplotlib Bar Plot: The code example above uses the Matplotlib bar plot to visualize the insertion sort algorithm. It represents each element as a bar and updates the plot to show the sorting process.

2. Animation Libraries: Instead of updating the plot at each step, you can utilize animation libraries like `matplotlib.animation` or external libraries like `manim` to create a smoother animation of the sorting process.

3. GUI Frameworks: If you prefer a graphical user interface (GUI), you can use GUI frameworks like Tkinter or PyQt to create a window and visualize the sorting process interactively. You can update the bar plot in response to button clicks or timer events.

Remember to adapt and modify the code based on your requirements or preferred visualization techniques.