Sure! Let's break down insertion sort in a simple and easy-to-understand way.

### Insertion Sort Explained

Imagine you have a deck of playing cards, and you want to sort them in your hand. You pick up the cards one by one and place them in the correct position relative to the already sorted cards in your hand. This is essentially how insertion sort works.

Here's a step-by-step explanation:

1. \*\*Start with the First Card:\*\*

- You start with the first card in your hand. Since it's the only card, it's already sorted.

2. \*\*Pick the Next Card:\*\*

- You pick the next card from the deck.

3. \*\*Find the Correct Position:\*\*

- Compare the new card with the cards already in your hand (sorted portion).

- Shift the cards in your hand to the right until you find the correct position for the new card.

4. \*\*Insert the Card:\*\*

- Place the new card in its correct position.

5. \*\*Repeat:\*\*

- Repeat the process for each card in the deck until all cards are sorted.

### Visual Example

Let's say we have the following list of numbers: `[5, 2, 4, 6, 1, 3]`

1. Start with the first element (5). It's already sorted: `[5]`

2. Pick the next element (2):

- Compare 2 with 5 and find that 2 should be before 5.

- Move 5 to the right and insert 2: `[2, 5]`

3. Pick the next element (4):

- Compare 4 with 5 and find that 4 should be before 5.

- Move 5 to the right and insert 4: `[2, 4, 5]`

4. Pick the next element (6):

- 6 is already in the correct position: `[2, 4, 5, 6]`

5. Pick the next element (1):

- Compare 1 with 6, 5, 4, and 2.

- Move 6, 5, 4, and 2 to the right and insert 1: `[1, 2, 4, 5, 6]`

6. Pick the next element (3):

- Compare 3 with 6, 5, and 4.

- Move 6, 5, and 4 to the right and insert 3: `[1, 2, 3, 4, 5, 6]`

Now the list is sorted!

### Python Code

Here's the insertion sort algorithm implemented in Python:

```python

def insertion\_sort(arr):

n = len(arr)

for i in range(1, n):

value = arr[i]

hole = i

# Shift elements to the right to make room for the new element

while hole > 0 and arr[hole - 1] > value:

arr[hole] = arr[hole - 1]

hole -= 1

# Insert the new element into the correct position

arr[hole] = value

return arr

# Example usage

numbers = [5, 2, 4, 6, 1, 3]

sorted\_numbers = insertion\_sort(numbers)

print("Sorted array:", sorted\_numbers)

```

### Explanation of Code

1. \*\*Initialize Variables:\*\*

- `n = len(arr)`: Calculate the length of the array.

2. \*\*Main Loop:\*\*

- `for i in range(1, n)`: Start from the second element and iterate through the array.

3. \*\*Inner Loop:\*\*

- `value = arr[i]`: Store the current element to be inserted.

- `hole = i`: Initialize the position to insert the current element.

- `while hole > 0 and arr[hole - 1] > value`: Compare and shift elements to the right to make space for the current element.

- `arr[hole] = value`: Insert the current element into the correct position.

The process continues until the entire array is sorted.