🟦 Insertionsort

1️⃣ Grundidee

Insertionsort baut die sortierte Liste **schrittweise von links nach rechts** auf. Jedes neue Element wird an der **richtigen Position im bereits sortierten Teil** eingefügt.

- Ähnlich wie Sortieren von Spielkarten in der Hand
- Sehr effizient für **kleine oder fast sortierte** Datenmengen

---

2️⃣ Voraussetzungen

- ❌ keine
- funktioniert auf **beliebigen vergleichbaren Elementen**

3️⃣ Laufzeiten & Eigenschaften

Eigenschaft	Wert
Best Case	O(n)
Average Case	$O(n^2)$
Worst Case	$O(n^2)$
Speicherbedarf	O(1)
In-place	ja
Stabil	ja

4️⃣ Schritt-für-Schritt-Beispiel

Ausgangsliste:

```
[5, 3, 4, 1]
```

Schritt 1

- 3 wird vor 5 eingefügt → [3, 5, 4, 1]

Schritt 2

- 4 wird zwischen 3 und 5 eingefügt → [3, 4, 5, 1]

Schritt 3

- 1 wird ganz nach vorne verschoben → [1, 3, 4, 5]

Ergebnis:

```
[1, 3, 4, 5]
```

Algorithmus in Worten

Der Algorithmus beginnt beim zweiten Element der Liste.
Dieses Element wird als Einfügeelement betrachtet.
Es wird mit den Elementen links davon verglichen.
Solange diese größer sind, werden sie eine Position nach rechts verschoben.
Das Einfügeelement wird anschließend an der entstandenen freien Position eingefügt.
Dieser Vorgang wird für jedes weitere Element wiederholt, bis die gesamte Liste sortiert ist.\

5️⃣ Besonderheiten / Prüfungsrelevante Hinweise

- Sehr gut bei **fast sortierten Listen**
- Oft verwendet als Teilschritt in **Hybridsortierverfahren**
- Kann mit **binärer Suche** kombiniert werden (Binary Insertion Sort)

6️⃣ Vor- und Nachteile

Vorteile

- stabil
- in-place
- sehr gut für kleine n

Nachteile

- ineffizient bei großen, ungeordneten Datenmengen

🧠 Merksatz für die Prüfung

Insertionsort fügt Elemente schrittweise in eine sortierte Teilliste ein und ist stabil.

7️⃣ Python-Implementierung

In [1]:
```python
def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i - 1

        while j >= 0 and arr[j] > key:
            arr[j + 1] = arr[j]
            j -= 1

        arr[j + 1] = key

    return arr
```