# L2\_2.1 Sortierung: Bubble Sort

## **Sortieralgorithmus**

Die Sortierung nimmt eine große Bedeutung in der Programmierung ein.

* Bis zu ¼ der Rechenzeit von Großrechnern entfallen auf die Sortierung (Studie der Stanford University)
* Mehr als 40 verschiedene Sortierverfahren sind bekannt und werden genutzt
* Sortierte Daten lassen sich viel schneller durchsuchen (z. B. Stichwortverzeichnis)
* Sortieren ist ein Bestandteil zahlreicher Algorithmen (Greedy-Algorithmus: Beispiel Herausgabe von Wechselgeld)
* Sortieren dient der Visualisierung von Daten (z. B. Medaillenspiegel bei Olympia)

Um Sortieralgorithmen zu bewerten und voneinander abzugrenzen werden oft der Worst und der Best Case verglichen. Es wird folglich die Frage gestellt, wie lange man im besten/schnellsten Fall für die Sortierung eines Arrays braucht und wie lange es im schlechtesten/langsamsten Fall dauert.

## **Bubble Sort**

Der Bubble Sort ist ein vergleichsbasierter Sortieralgorithmus, der auf dem folgenden Prinzip basiert:

1. Starte mit dem ersten Array-Element. Ist es größer als das rechte Nachbarelement, so werden beide getauscht.
2. Fahre fort mit dem zweiten Array-Element. Ist es größer als das rechte Nachbarelement, so werden beide getauscht.
3. Wiederhole den Vorgang bis zum Ende des Arrays.
4. Wiederhole den Vorgang so oft, bis das Array sortiert ist. Das letzte Element des vorherigen Durchlaufs muss nun nicht mehr betrachtet werden, da das größte Element ans Ende des Arrays „aufgestiegen“ ist.

**Umsetzung**



**Erklärung zum Ablauf des Algorithmus**

**Bubble-Phase 1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a[0] | a[1] | a[2] | a[3] | a[4] |
| 5 | 4 | 9 | 1 | 3 |
| 5 > 4 ? Ja 🡪 Tausch | |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 5 | 9 | 1 | 3 |
|  | 5 > 9 ? Nein 🡪 kein Tausch | |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 5 | 9 | 1 | 3 |
|  |  | 9 > 1 ? Ja 🡪 Tausch | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 5 | 1 | 9 | 3 |
|  |  |  | 9 > 3 ? Ja 🡪 Tausch | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 5 | 1 | 3 | 9 |

🡪 Ende erste Bubble-Phase. Ab a[4] ist das Array sortiert.

**Bubble-Phase 2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a[0] | a[1] | a[2] | a[3] | a[4] |
| 4 | 5 | 1 | 3 | 9 |
| 4 > 5 ? Nein 🡪 kein Tausch | |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 5 | 1 | 3 | 9 |
|  | 5 > 1 ? Ja 🡪 Tausch | |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 1 | 5 | 3 | 9 |
|  |  | 5 > 3 ? Ja 🡪 Tausch | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 1 | 3 | 5 | 9 |

🡪 Ende zweite Bubble-Phase. Ab a[3] ist das Array sortiert

**Bubble-Phase 3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a[0] | a[1] | a[2] | a[3] | a[4] |
| 4 | 1 | 3 | 5 | 9 |
| 4 > 1 ? Ja 🡪 Tausch | |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 4 | 3 | 5 | 9 |
|  | 4 > 3 ? Ja 🡪 Tausch | |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 5 | 9 |

🡪 Ende der dritten Bubble-Phase. Ab a[2] ist das Array sortiert.

**Bubble-Phase 4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a[0] | a[1] | a[2] | a[3] | a[4] |
| 1 | 3 | 4 | 5 | 9 |
| 1 > 4 ? Nein 🡪 Kein Tausch | |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 5 | 9 |

🡪 Ende der vierten Bubble-Phase. Das Array ist sortiert.

## **Ausgabe eines Arrays**

Für die Ausgabe aller Elemente eines Arrays kann die for-Schleife eingesetzt werden. Sie basiert auf einer Variablen für den Index und wird mit der Increment-Anweisung automatisch um die Schrittweite 1 erhöht, bis der letzte Index des Arrays erreicht ist.

laenge = len(meinArray)

for i in range(laenge):

print(meinArray[i])