Gruppe A
Insertion-Sort

Vorgehen:
Erarbeiten Sie zusammen das Verfahren „Insertion-Sort“. Lesen Sie dazu den kurzen Einführungstext auf dieser Seite und teilen Sie die übrigen Teile untereinander auf:

1. Textbeschreibung
2. Struktogrammdarstellung
3. Pseudo-Code

Erklären Sie sich das Verfahren gegenseitig. Überlegen Sie, wie Sie es den anderen Schüler:innen erklären können. Entwerfen Sie eine Präsentation und ein Handout zu dem Verfahren, das Sie den übrigen Schüler:innen zur Verfügung stellen können. Jedes Mitglied der Gruppe muss in der Lage sein, das Verfahren zu erklären.

Begründen Sie auch, warum das Verfahren eine Sortierung zur Folge hat. Zählen Sie die Vergleiche, die ausgeführt werden für:

a = [1, 2, 3, 4, 5, 6]
b = [6, 5, 4, 3, 2, 1]

Entscheiden Sie, ob dieses Sortierverfahren stabil ist. Begründen Sie Ihre Entscheidung.

Teilung des Arrays in einen sortierten und einen unsortierten Teil

Bei den Sortieralgorithmen, die wir besprechen, wird das Array in einen sortierten und einen unsortierten Teil geteilt. In Ihrer Präsentation sollte immer deutlich werden, welcher Teil sortiert und welcher Teil unsortiert ist.
Um es einheitlicher zu gestalten, sollten Sie den sortierten Teil mit grüner und den unsortierten Teil mit roter Farbe verdeutlichen.

Ausgangspunkt der Sortierung ist folgendes unsortiertes Array

array = [18, 10, 38, 11, 23, 24, 45, 11]

In der grafischen Darstellung:

18	10	38	11	23	24	45	11

Insertion-Sort: Textbeschreibung

Zunächst muss ein sortierter Bereich definiert werden. Der Einfachheit wegen wird das erste Element als sortierter Bereich festgelegt. Überlegen Sie, ob eine einelementige Liste wirklich sortiert ist.

18	10	38	11	23	24	45	11

Nun nimmt man das erste Element des unsortierten Teils und vergleicht es mit allen Elementen des sortierten Teils, bis man entweder am Beginn des Arrays ist oder das einzufügende Element größer ist. Da 18 > 10 ist, hat die sortierte Liste nach dem ersten Schritt diese Gestalt.

10	18	38	11	23	24	45	11

38 ist schnell abgearbeitet, da 38 > 18:

10	18	38	11	23	24	45	11

Nun zur 11:

- 11 < 38: wir vergleichen weiter
- 11 < 18: wir vergleichen weiter
- 11 > 10: wir ordnen die 11 nach der 10 ein.

10	11	18	38	23	24	45	11

Als nächstes wird 23 einsortiert:

- 23 < 38: wir vergleichen weiter
- 18 > 23: wir ordnen die 23 nach der 18 ein

10	11	18	23	38	24	45	11

Die 24 wird danach nach der 23 eingefügt und dann die 45 nach der 38.

10	11	18	23	24	38	45	11

Abschließend ist noch eine 11 einzusortieren:

- 11 < 45: wir vergleichen weiter
- 11 < 38: wir vergleichen weiter
- 11 < 24: wir vergleichen weiter
- 11 < 23: wir vergleichen weiter
- 11 < 18: wir vergleichen weiter
- 11 = 11: wir sortieren die 11 des unsortierten Teils hinter die 11 des sortierten Teils ein.

10	11	11	18	23	24	38	45

Die Liste ist sortiert.

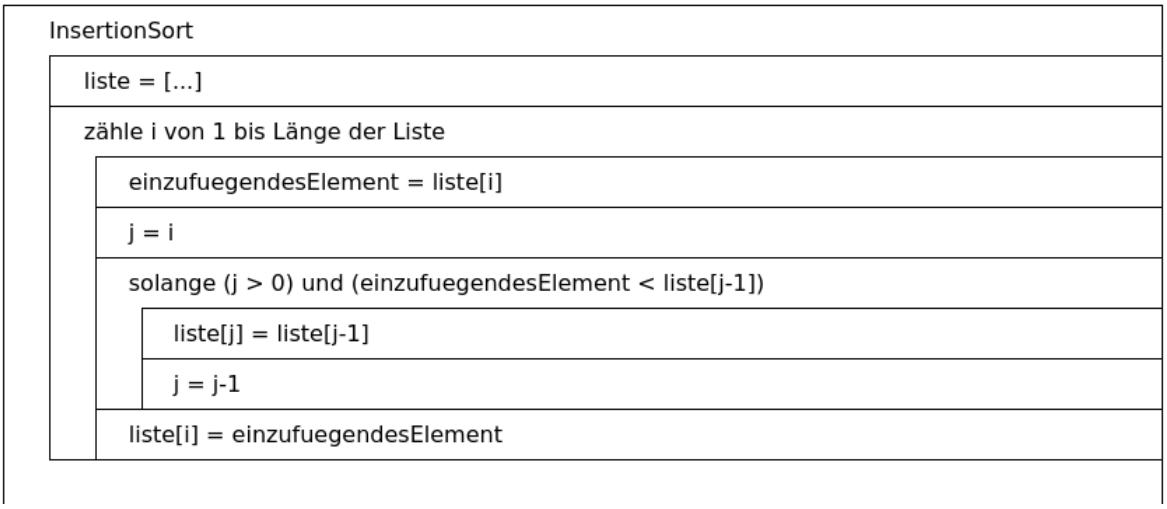
Insertion-Sort: Pseudo-Code

```
liste = […]
zähle i von 1 … Länge der Liste
     einzufuegendesElement = liste[i]
     j = i
     solange (j > 0) und (einzufuegendes Element < liste[j-1])
          liste[j] = liste[j-1]
          j = j - 1
     liste[j] = einzufuegendesElement
```

Übertragen Sie diesen Pseudo-Code in eine Python-Funktion insertionSort(liste) und testen Sie ihn.

Spielen Sie den Code mit den Karten durch.

Insertion-Sort: Struktogramm



Führen Sie mit einem kurzen Array:
test = [20, 10, 30, 5, 10]
einen Trockentest durch.