Gruppe D
Quick-Sort

Vorgehen:
Erarbeiten Sie zusammen das Verfahren „Quick-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. Bereiten Sie einen Kurzvortrag vor, in dem Sie das Verfahren vorstellen.

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.

Dieser Algorithmus gehört zu denen, die das Prinzip „devide and conquer“ (teile und herrsche) umsetzen. Überlegen Sie, was mit diesem Prinzip gemeint sein könnte.

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

Quick-Sort: Textbeschreibung

Bei Quicksort wird ein zufälliges Element des Arrays ausgewählt, für unser Beispiel nehmen wir das Element, für dessen Index i gilt i = len(Array)//2. Dieses wird als „Pivot-Element“ (abgeleitet vom französischen Wort für Dreh-/Angelpunkt) bezeichnet. Dann werden alle übrigen Elemente des Arrays mit dem Pivot-Element verglichen. Alle Elemente, die größer sind als das Pivot-Element werden in einem neuen Array gespeichert (größer). Alle anderen Elemente werden in einem zweiten Array gespeichert (kg (für kleinergleich)).

18	10	38	11	23	24	45	11

Array größer:

38	24	45

Array kg:

18	10	11	11

Mit den so entstehenden Arrays wird in gleicher Weise verfahren. So wird das Array größer so verarbeitet. Pivot-Element ist die 24.

38	24	45

Die beiden entstehenden Arrays sind größer-größer = [38,45] und das leere Array größer-kg=[]. Das Array größer-größer wird nun in gleicher Weise bearbeitet. Als Pivot-Element dient die 45. Das Array größer-größer-größer=[], das Array größer-größer-kg = [38]. Da es nur ein Element hat ist es auch sortiert. Jetzt gehen wir wieder rückwärts:

- Für das Array größer-größer fügen wir dem sortierten Array größer-größer-kg das Pivot-Element 45 und das leere Array größer-größer-größer. So erhalten wir die sortierte Liste größer-größer = [38, 45].
- Für das Array größer fügen wir dem leeren Array größer-kg das Pivot-Element 24 an und die sortierte Liste größer-größer an und erhalten das sortierte Array größer = [24, 38, 45].

Auf gleiche Weise wird mit dem Array kg=[18,10,11,11] verfahren. Für dieses Array wählen wir das Pivot-Element 11 und teilen das Array in kg-größer=[18] und kg-kg=[10,11] auf. Da kg-größer nur ein Element hat, muss nur kg-kg weiter sortiert werden. Das Pivot-Element hier ist erneut die 11. Es entsteht das leere Array kg-kg-größer=[] und das einelementige Array kg-kg-kg=[10]. Nun können wir wieder rückwärts gehen:

- Für das Array kg-kg wird dem Array kg-kg-kg das Pivot-Element für kg-kg (11) sowie das leere Array kg-kg-größer. Es entsteht so das sortierte Array kg-kg = [10,11].
- Für das Array kg wird dem sortierten Array kg-kg das Pivot-Element für kg (11) und das einelementige (und damit sortierte) Array kg-größer=[18] angehängt. Somit erhalten wir das sortierte Array kg = [10,11,11,18]

Im letzten Rückwärtsschritt wird dem sortierten Array kg = [10,11,11,18] das Pivot-Element unseres ersten Schrittes 23 und das sortierte Array größer = [24, 38, 45] angehängt. Es entsteht so das sortierte Array [10, 11, 11, 18, 23, 24, 38, 45].

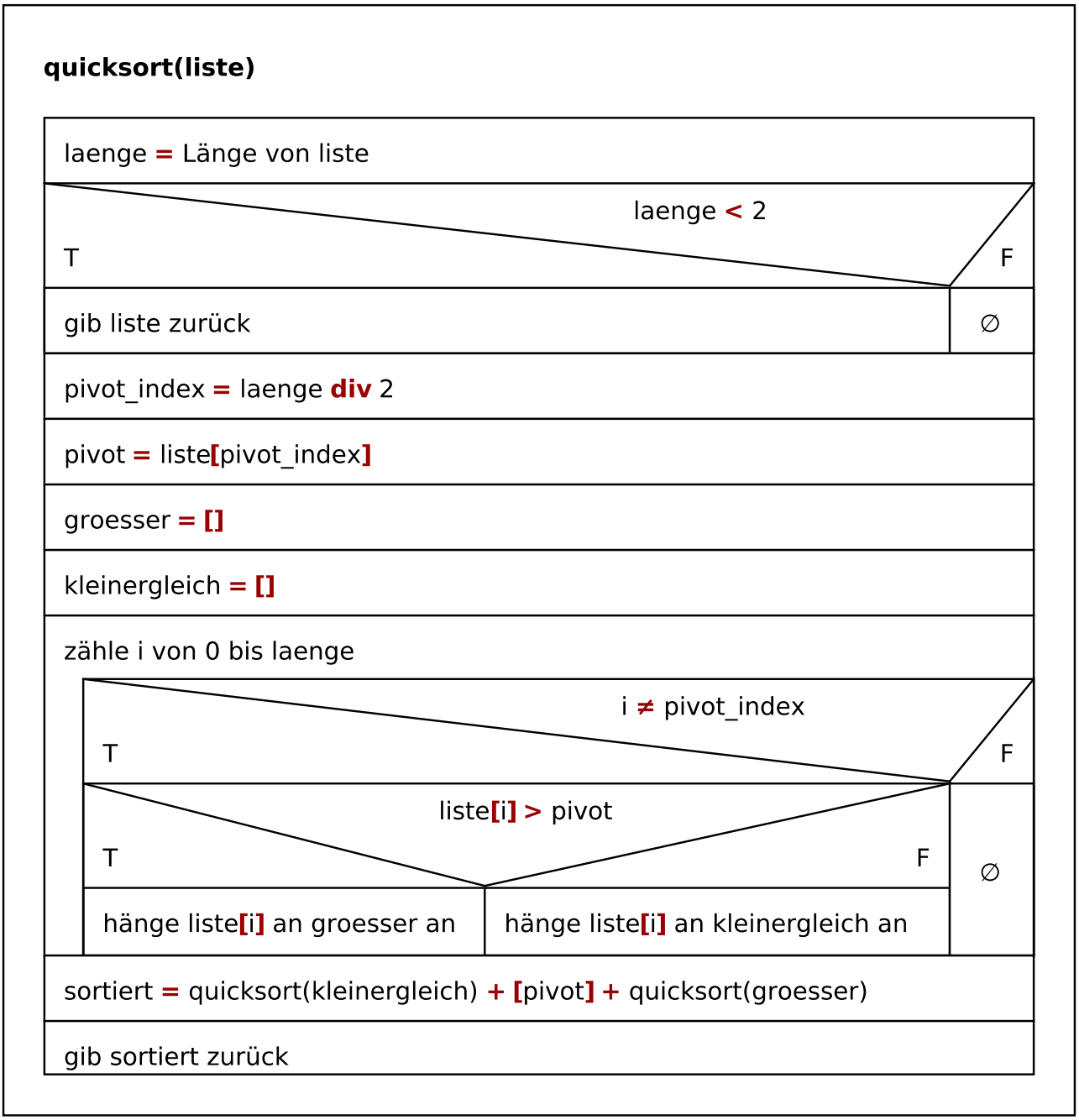
Quick-Sort: Pseudo-Code

```
funktion quicksort(liste)
     laenge = Länge der Liste
     wenn lange < 2
          gib liste zurück
     pivot_index = laenge div 2
     pivot = liste[pivot_index]
     groesser = []
     kleinergleich = []
     zähle i von 0 bis laenge
          wenn i != pivot_index:
               wenn liste[i] > pivot:
                    hänge liste[i] an groesser an
               sonst:
                    hänge liste[i] an kleinergleich an
     sortiert = quicksort(kleinergleich) + [pivot] +
quicksort(groesser)
     gib sortiert zurück
```

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

Spielen Sie den Code mit den Karten durch.

Quick-Sort: Struktogramm



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