Datenstrukturen

Liste:

["Element"]

- Kann nachträglich erweitert werden
- Reihenfolge wird beibehalten
- Elemente dürfen mehrfach vorkommen

Dictionary:

{"key": "value"}

- Speichert eine Zuordnung von einem Schlüssel (key) zu einem Wert (value)
- Reihenfolge wird nicht beibehalten
- Jeder Schlüssel darf aber nur 1x existieren

DefaultDict:

defaultdict(object)

- `from collections import defaultdict`
- defaultdict kann einen Standardwert zum Key eines Dictionaries hinzufügen
- kann sehr einfach Elemente in einer Liste zählen

Tupel:

("Element1", "Element2")

- Ähnlich wie eine Liste
- Kann aber nachträglich nicht erweitert werden
- Ermöglichst die Ausgabe von mehreren Return-Werten

Set:

{"Element1"}

- Kann nachträglich erweitert werden
- Reihenfolge wird nicht beibehalten
- Index - Schreibweise s[15] wird nicht unterstützt
- Jedes Element kommt nur 1x vor
-

➢ Wenn du garantierten möchtest, dass jedes Element nur 1x drinnen vorkommt - sehr viel effizienter als eine Liste, da ein Set hierauf optimiert ist

Warteschlange:

Queue()

- Elemente werden nacheinander eingefügt
- Können in der vorgegebenen Reihenfolge wieder abgerufen werden
- Direkter Zugriff auf die Elemente nicht möglich

Prioritätswarteschlange:

queue.PriorityQueue()

- Optimierte Datenstruktur, um Elemente nach einer Priorität zu sortieren
- Elemente werden quasi automatisch nach ihrer Priorität sortiert
- Direkter Zugriff auf die Elemente nicht möglich
- Reicht also für eine Warteschlange nach Priorität

➢ Braucht man nicht so häufig wie eine Liste / Set, aber wenn man sie für einen Einsatzzweck gebrauchen kann ist sie für solche Einsatzzwecke oft signifikant schneller als eine Liste