Python Datenstrukturen - Ausführliches Handout mit Erklärungen und Beispielen

Was bedeutet „geordnet“?

„Geordnet“ bedeutet, dass die Elemente in der Reihenfolge gespeichert und gelesen werden, in der sie eingefügt wurden. Der Zugriff erfolgt über einen Index (bei Listen und Tupeln) oder über Schlüssel (bei Dictionaries).

Beispiel für geordnete Struktur (Liste):

```
zahlen = [100, 200, 300]
print(zahlen[0])  # Ausgabe: 100
print(zahlen[1])  # Ausgabe: 200
```

Beispiel für ungeordnete Struktur (Set):

```
werte = {"a", "b", "c"}
for w in werte:
    print(w)  # Reihenfolge kann beliebig sein
```

---

1. Liste (List)

Listen sind geordnete und veränderbare Sammlungen von Elementen. Sie erlauben doppelte Werte und verschiedene Datentypen.

```
# Liste erstellen
obst = ["Apfel", "Banane", "Kirsche"]

# Zugriff auf Elemente
print(obst[0])      # Apfel
print(obst[-1])     # Kirsche

# List Unpacking
a, b, c = obst
print(a, b, c)
```

2. Iteration über Listen

```
for frucht in obst:
    print(frucht)
```

```
# Mit Index
for i in range(len(obst)):
    print(f"Index {i}: {obst[i]}")
```

3. Bearbeiten von Listen

```
obst.append("Orange")        # Am Ende hinzufügen
obst.insert(1, "Mango")      # An Position 1 einfügen
obst.remove("Banane")        # Ein Element löschen
del obst[0]                  # Nach Index löschen
obst[0] = "Erdbeere"         # Ersetzen eines Elements
obst.clear()                 # Liste leeren
```

4. Element in Liste finden

```
obst = ["Apfel", "Banane", "Kirsche"]
if "Banane" in obst:
    print("Gefunden!")

index = obst.index("Kirsche")
print(f"Kirsche bei Index {index}")
```

5. Funktionen mit Listen

```
def ausgabe(liste):
    for eintrag in liste:
        print(eintrag)

# Funktionsaufruf
ausgabe(obst)
```

6. Listen sortieren

```
zahlen = [4, 2, 9, 1]
zahlen.sort()                # Aufsteigend
zahlen.sort(reverse=True)    # Absteigend
neue_liste = sorted(zahlen) # Kopie sortiert
```

7. Tupel (Tuple)

Tupel sind geordnete, aber unveränderliche Sammlungen.

```
farben = ("rot", "grün", "blau")
print(farben[1])
```

8. Dictionary (dict)

Dictionaries sind Sammlungen von Schlüssel-Wert-Paaren. Sie sind geordnet (ab Python 3.7), veränderbar und lassen keine doppelten Schlüssel zu.

```
person = {
    "name": "Tom",
    "alter": 30,
    "stadt": "Hamburg"
}

# Zugriff
print(person["name"])

# Hinzufügen/Ändern
person["beruf"] = "Ingenieur"

# Iteration
for schluessel, wert in person.items():
    print(f"{schluessel}: {wert}")
```

9. Mengen (Set)

Sets sind ungeordnete, unveränderliche Sammlungen, die keine Duplikate enthalten.

```
primzahlen = {2, 3, 5, 7, 11, 2, 3}  # Duplikate werden entfernt
print(primzahlen)

primzahlen.add(13)
primzahlen.remove(5)

A = {1, 2, 3}
B = {3, 4, 5}

print(A.union(B))        # Vereinigung
print(A.intersection(B)) # Schnittmenge
print(A.difference(B))   # Differenz
```

Zusammenfassungstabelle

Struktur	Geordnet	Veränderbar	Duplikate erlaubt	Besonderheit
Liste	Ja	Ja	Ja	Standard-Datenstruktur

Struktur	Geordnet	Veränderbar	Duplikate erlaubt	Besonderheit
Tupel	Ja	Nein	Ja	Gut für feste Gruppen
Dictionary	Ja	Ja	Nein (Schlüssel)	Schlüssel-Wert-Zuordnung
Set	Nein	Ja	Nein	Schnelle Mengenoperationen

Dieses Handout dient als Referenz für die gängigsten Python-Datenstrukturen mit typischen Anwendungsbeispielen und Erklärungen.