

# LS 2 – Programmiertechnik



## Aufgabe 2 - Reflexion Programmierhilfsmittel

- a) Welche Vorteile sollen durch Programmierrichtlinien erzielt werden?

**Antwort:**

- Bessere Übersicht
- Bessere Lesbarkeit
- Einfacher zu ändern und zu erweitern

- b) Zusammen mit Python wurden Python Programmierrichtlinien spezifiziert. Wo sind diese definiert?

**Antwort:**

- Die Python Programmierrichtlinien sind in der **PEP8** definiert.
- PEP – **P**ython **E**nhancement **P**roposal

- c) Welche Konventionen zur Namensgebung sind in den Python Programmierrichtlinien festgelegt?

**Antwort:**

- |                                   |   |            |
|-----------------------------------|---|------------|
| • Variablen, Funktionen, Methoden | → | snake_case |
| • Klassen                         | → | CamelCase  |
| • Konstanten                      | → | UPPERCASE  |

- d) Welche Konventionen zur Einrückung sind in den Python Programmierrichtlinien festgelegt?

**Antwort:**

- 4 Leerzeichen, keine Tabs

- e) Was versteht man in Python unter einem DocString?

**Antwort:**

- DocStrings sind mehrzeilige Kommentare, welche auch zur Dokumentationsgenerierung genutzt werden können.
- DocStrings starten und enden mit drei Gänsefüßchen `""" . . . """`

- f) Welche Aufgabe besitzen Code-Formatierer? Welche Formatierer für Python kennen Sie?

**Antwort:**

- Code-Formatierer formatieren den Programmcode nach bestimmten Vorgaben.
- `black` und `autopep8` sind Code-Formatierer für Python, welche den Programmcode gemäß PEP8 formatieren.

- g) Was versteht man unter einem (Static-)Code-Analyzer? Welche Code-Analyzer für Python kennen Sie? Mit welchem anderen Namen werden Code-Analyzer auch bezeichnet?

**Antwort:**

- Ein Code-Analyzer prüft, ob der Programmcode bestimmte Regeln einhält und zeigt die Ergebnisse an. Er ändert selber den Programmcode nicht.
- `flake8` und `pylint` sind Code-Analyzer für Python.
- Code-Analyzer sind auch unter dem Namen Linter bekannt.

- h) Viele Hilfsmittel stehen in Python in Form von Paketen zur Verfügung. Wie heißt der standardmäßig mit Python ausgelieferte Paket-Manager? Wie lauten die Befehle, um sich die verfügbaren Pakete anzusehen? Wie kann man ein zusätzliches Paket installieren?

**Antwort:**

- `pip - pip installs packages`
- `pip list` zeigt die installierten Pakete und deren Versionen an.
- `pip install paketname` installiert ein Paket.

**Aufgabe 4 - Reflexion Kontrollstrukturen**

- a) Welche drei Kontrollstrukturen kennt die strukturierte Programmierung?

**Antwort:**

- Sequenz
- Verzweigung (Fallunterscheidung)
- Wiederholung (Schleife)

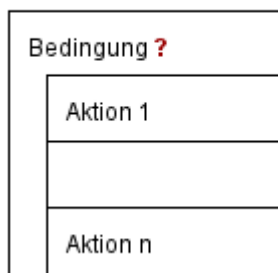
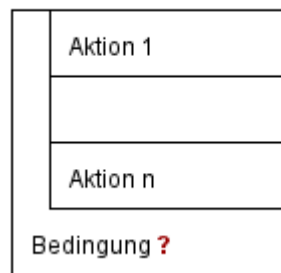
- b) Was ist der Unterschied zwischen einer kopfgesteuerten und einer fußgesteuerten Schleife? Wie werden sie in Struktogrammen dargestellt?

**Antwort:**

Bei einer **kopfgesteuerten Schleife** wird die Bedingung zuerst geprüft und danach der Rumpf durchlaufen

Bei einer **fußgesteuerten Schleife** wird zuerst der Rumpf durchlaufen und dann die Bedingung überprüft, ob ein weiterer Durchlauf stattfindet.

Bei kopfgesteuerten Schleifen wird der Rumpf evtl. nie durchlaufen, bei fußgesteuerten Schleifen wird der Rumpf immer mindestens einmal durchlaufen.

**Kopfgesteuerte Schleife****Fußgesteuerte Schleife**

- c) Wie oft wird eine kopfgesteuerte Schleife mindestens durchlaufen?

Bei kopfgesteuerten Schleifen wird der Rumpf evtl. nie durchlaufen.

- d) Sie wollen die Schleifenvariable `i` in jedem Durchlauf um 10 erhöhen. Wie sieht die entsprechende Python Anweisung aus?

```
i = i + 10
i += 10
```

- e) Wenn der Inhalt der Variablen `i` größer als 50 ist, soll „groß“ ausgegeben werden, ansonsten soll nichts gemacht werden. Wie sieht der entsprechende Python Code aus?

```
if i > 50:  
    print( "groß")
```

- f) Bei einer Fallunterscheidung müssen Sie mehrere Bedingungen verknüpfen. Welche Möglichkeiten (Schlüsselwörter) bietet Python? Geben Sie jeweils ein kleines Beispiel an.

<b>and</b>	(und)	<code>x &lt; 5 and y &gt; 10</code>
<b>or</b>	(oder)	<code>x &lt; 5 or y &gt; 10</code>
<b>not</b>	(nicht)	<code>not x &lt; 5</code>

- g) Was ist der Unterschied zwischen `=` und `==`?

<code>=</code>	→	<b>Zuweisung</b>	→	Das Ergebnis der rechten Seite wird der Variablen auf der linken Seite zugewiesen.
<code>==</code>	→	<b>Vergleich</b> /Prüfung auf Gleichheit	→	ergibt True oder False; Variablen werte werden nicht geändert

- h) Zum Umgang mit Schleifen gibt es in Python die Schlüsselwörter `continue` und `break`. Was bewirken diese Anweisungen?

<b>continue</b>	→	Durchlauf wird abgebrochen und mit neuem Durchlauf angefangen; Sprung an Schleifenanfang
<b>break</b>	→	Schleife wird abgebrochen; Sprung hinter Schleifenende

**Aufgabe 8 - Reflexion Datenstrukturen**

- a) Python kennt standardmäßig 4 Datenstrukturen. Welche?

**Antwort:**

1. Liste
2. Dictionary
3. Tuple
4. Set

- b) Geben Sie ein Beispiel für eine Liste an, welche aus 5 Vornamen besteht.

**Antwort:**

```
namens_liste = ["Albert", "Berta", "Cem", "Daniel", "Emil"]
```

- c) Mit welchem Index wird das erste Listenelement indiziert?

**Antwort:**

Mit dem Index **0**, z.B.: `namens_liste[0]`

- d) Mit welcher Python Funktion kann man bestimmen, wie lang eine Liste ist?

**Antwort:**

Mit der Funktion **len**, z.B.: `len(namens_liste)`

- e) Wodurch unterscheidet sich ein Tuple von einer Liste? Wie würde das Beispiel aus (b) als Tuple aussehen?

**Antwort:**

Ein **Tuple** ist eine Liste, die nicht mehr geändert werden kann.

Das Beispiel aus (b) als Tuple lautet:

```
namens_liste = ("Albert", "Berta", "Cem", "Daniel", "Emil")
```

- f) Python bietet noch weitere Datenstrukturen: **Tuple** und **Set**. Erarbeiten Sie jeweils ein Beispiel, wofür diese Datenstrukturen gut sind und wie man sie nutzen kann.

**Antwort:**

```
""" beispiel_ls2_04 - Beispiel Tuple und Set
    Name, Organisaion:      Markus Breuer, BK-GuT
    Erstellt, Letzte Änderung: 24.11.2020, 18.06.2022
    """

print("Tuple anlegen")
t1 = (1, 2, 3)
print("")

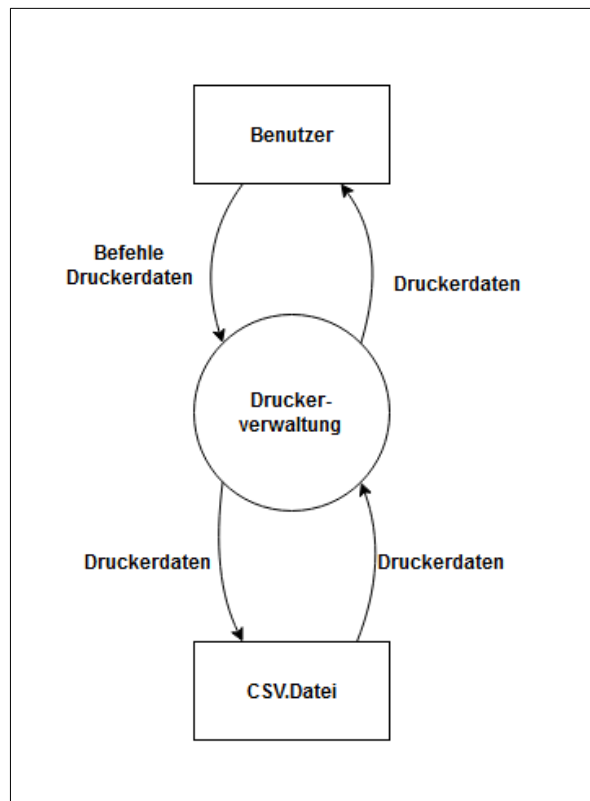
print("Tuple ausgeben")
print(t1)
print("Erstes Element:", t1[0])
print("")

print("Tuple durchlaufen und alle Elemente ausgeben")
for zahl in t1:
    print(zahl)
print("")

print("Mengen anlegen")
m1 = {1, 2, 3}
m2 = {1, 4, 5}
print("")

print("Mengen ausgeben")
print(m1)
print(m2)
print("")

print("Mengenoperationen")
print("Schnittmenge (Und-Verknüpfung):", m1 & m2)
print("Vereinigungsmenge (Oder-Verknüpfung):", m1 | m2)
print("Vereinigungsmenge (Exklusiv-Oder-Verknüpfung):", m1 ^ m2)
print("Differenzmenge:", m1 - m2)
```

**Aufgabe 9 - Druckerverwaltung****Kontextdiagramm**

## Struktogramm

