03 - Darstellung und Formatierung

Eine klare und einheitliche Programmierung erleichtert das Lesen eines Sourcecodes.

Programmblöcke

	Die Codezeilen innerhalb eines Programmblocks werden um 4 Stellen eingerückt.

Beispiel

```
if anredeCode == 1:
    print ('Herr')
else:
    print ('Frau')

print ('Ende')
```

Codezeilen

Eine Codezeile sollte immer auf einen Blick erfasst werden können.

	Eine Codezeile ist maximal 120 Zeichen lang. Längere Zeilen werden umgebrochen.
	Ordnen Sie gleiche Elemente wie Klammern, Bezeichner oder Bedingungen untereinander an.

Beispiel

```
if input_value >= minimum or
   input_value <= maximum or
   input_value > 0        and client.client_type.equals('Stammkunde'):
```

```
def __init(self, name, age, gender):
    self.__name   = name
    self.__age    = age
    self.__gender = gender
```

Bemerkung: Die Anordnung von Codezeilen wird i.d.R. nicht durch die automatische Formattierung einer IDE unterstützt. Es muss jedoch damit gerechnet werden, dass automatische Formattierungen durch die Entwickler durchgeführt werden können, welche solche Anordnungen überschreiben.

Leerzeilen

Leerzeilen erleichtern das Erkennen von zusammengehörenden Teilen des Codes.

	Klassen und Funktion werden durch zwei Leerzeilen vom restlichen Code getrennt.
	Vor einer Methode in einer Klasse wird eine Leerzeile eingefügt.
	Logisch zusammengehörende Blöcke in einer Funktion können mit einer Leerzeile getrennt werden. Setze dies sparsam ein.

Beispiel

```
def foobar:
    print ('foo bar')


class Movie:
    def __init__(self, title):
        self._title = title

    @property
    def tile(self):
        return self._title
```

Konstruktoren

Konstruktoren versetzen ein neu erzeugtes Objekt in einen definierten Anfangszustand.

	Die Konstruktor-Methode steht an erster Stelle, um die Übersichtlichkeit zu erhöhen.

Beispiel

```
class Celsius:
    def __init__(self, temperature=0):
        self._temperature = temperature

…
```

Anordnung der Methoden

Durch eine geeignete Reihenfolge fällt es leichter, die richtige Methode effizient zu finden.

	Ähnliche Methoden sollen immer in der gleichen Reihenfolge angeordnet werden.

Beispiel

1. Konstruktor
2. Methoden die eine Verarbeitung auslösen (z.B. save_person)
3. Methoden die auf ein Ereignis reagieren (Listener, Events)
4. Getter/Setter-Methoden