PEP8 Merkblatt

www.datamics.com

Einführung

In Python ist nicht nur eine saubere Programmierung wichtig, sondern auch die Formatierung des Codes. Gut lesbarer Code fördert die gemeinschaftliche Arbeit an diesem. Python ist eine der wichtigsten Programmiersprachen in Open-Source-Projekten. Dies ist auch ein Resultat der hohen Verständlichkeit professionell entwickelter Python-Programme.
Der allgemein akzeptierte Standard für das Layout von Python-Code ist PEP8 (Python Enhancement Proposal 8, siehe `https://www.python.org/dev/peps/pep-0008/`).

Code Layout

Einrückung

Einrückung sollte immer mit vier Leerzeichen pro Codeblock erfolgen und niemals mit Tabulator.

Zeichen pro Zeile

Eine einzelne Zeile Code sollte nie mehr als 80 Zeichen haben. Viele Editoren können eine visuelle Markierung an dieser Grenze anzeigen.

Leerzeilen

Vor und nach Funktionen und Klassen der obersten Ebene sollten immer zwei Leerzeilen stehen. Möglichst selten kann eine einzelne Leerzeile zur logischen Trennung von Codeböcken verwendet werden.

Importe

Importe von Modulen sollten alphabetisch absteigend in einzelnen Zeilen erfolgen. Importe von Modulfunktionen mit **from** können in einer einzigen Zeile alphabetisch geordnet erfolgen. Praktisch ist auch der import von Standardmodulen, alphabetisch geordnet in einer einzigen Zeile, gebräuchlich.

Programmaufbau

Üblicher Aufbau:
1. Deklarationen (Interpreter, Encoding)
2. Programmdokumentation, Autor, Copyright
3. Importe
4. Konstanten
(5. ggf. globale Variablen)
6. Funktionen
7. Klassen
(8. __main__-Codeblock)

Namenskonventionen

Konstanten

Konstanten sollten durchgängig mit Grossbuchstaben und dem Unterstrich _ als Trennzeichen zwischen Komponenten geschrieben werden.

```
PIN_NUMBER = 17
```

Variablen und Funktionen

Für Variablen und Funktionen gilt:

- Alle Zeichen werden klein geschrieben
- Komponenten werden mit _ getrennt
- Bezeichner mit nur einem Buchstaben sind zu vermeiden (Ausnahmen z.B.: **i** und **j** für Schleifeniteration, **x** und **y** in mathematischen Funktionen)
- Insbesondere die Bezeichner **l** (kleines El), **O** (grosses Oh) und **I** (grosses Ih) sollten vermieden werden
- Sprechende Bezeichner sind wichtig (beispielsweise **ergebnisse** statt **tmp**)
- Zu lange Bezeichner sind zu vermeiden
- Sonderzeichen ausser _, auch deutsche Umlaute (ä, ö, ü), sind zu vermeiden
- Python-Schlüsselwörter wie beispielsweise **str** sollten nicht überschrieben werden

```
def run_gui(user_language="ENG"):
```

Klassennamen

Klassennamen werden ohne Trennzeichen, aber mit einem Grossbuchstaben zu Beginn jeder Komponente geschrieben.

```
class WidgetController:
```

Klassen- und Instanzmethoden

Der erste Parameter von Instanzmethoden sollte **self** und der erste Parameter (statischer) Klassenmethoden **cls** genannt werden.

```
class Test:
    @classmethod
    def method1(cls): pass

    def method2(self): pass
```

Private Instanzmethoden und -variablen

Private Instanzmethoden und -variablen sollten mit einem vorangehenden _ gekennzeichnet werden.

```
class Quadrat:
    def _hoehe(self): pass
```

Docstrings und Kommentare

Docstrings

Docstrings sind Anmerkungen zur Funktionsweise und Ausführung von Modulen, Klassen, Methoden und Funktionen. Sie werden direkt in der Zeile nach der Klassen-, Methoden- oder Funktionsdefinition gesetzt und von drei mal doppelten Anführungszeichen eingeschlossen. Bei mehrzeiligen Docstrings stehen die abschliessenden dreifachen doppelten Anführungszeichen in einer eigenen Zeile.

```
class Quadrat:
    "Klasse für Quadrate."

    def flaecheninhalt(self):
    """Mit dieser Methode
    wird der Flaecheninhalt berechnet.
    """
```

Docstrings ausgeben

Der Docstring jeder Klasse oder Funktion in Python kann mit **help(object)** oder **object.__doc__** ausgegeben werden, mit **object** als dem Namen der Klasse oder Funktion.

Kommentare

Python-Code sollte so weit wie möglich selbsterklärend sein und daher mit so wenigen Kommentaren wie möglich auskommen. Kommentare in einer eigenen Zeile sind nach dem Muster **# Leerzeichen Kommentar** zu definieren. Kommentare nach Code folgen dem Muster **Leerzeichen Leerzeichen # Leerzeichen Kommentar**.

```
# Debug-Flag wird als Konstante gesetzt
DEBUG = True
test = 10   # Refactor in Production!
```

Verschiedenes

Aufzählungen

Aufzählungen (Listen, Dictionary-Schlüssel-Wert-Paare, Basisklassen, multiple Variablenzuweisungen, ...) in der Form **Element1 Komma Leerzeichen Element2** ohne Leerzeichen vor dem ersten und nach dem letzten Element. Die Schlüssel-Wert-Paare von Dictionaries werden nach dem Muster **Schlüssel Doppelpunkt Leerzeichen Wert** formatiert. Zwischen dem Bezeichner der Aufzählung und den Klammern sollte kein Leerzeichen stehen.

```
dictionary = {1: 'a', 2: 'c'}
def funktion(parameter1, parameter2):
```

Variablenzuweisung

Variablenzuweisungen sollten nach dem folgenden Muster vorgenommen werden: **Variablenname Leerzeichen Gleichzeichen Leerzeichen Wert**. Bei der Zuweisung von Standardwerten zu Funktionsparametern sind keine Leerzeichen erforderlich.

```
variable = 1
def test(wert=1):
```

Explizit None testen

Wenn festgestellt werden soll, ob eine Variable **x** den Wert **None** hat, so ist **if x is not None** unbedingt **if x** vorzuziehen, da manche leere Datentypen zu **False** evaluieren. Ausserdem sollte zur Überprüfung von **None** kein Gleichheitszeichen verwendet werden.

```
if x is None: pass
```

PEP8 Checker

Es gibt eine Vielzahl von Programmen, die ein Script auf PEP8-Formatierung überprüfen. Ein einfaches Beispiel dafür ist das Pip-Modul **pep8**, welches leider nur einen Teil der PEP8-Konventionsverletzungen im Code erkennt.

Installation

Im Terminal:

```
pip install pep8
```

PEP8-Prüfung

Im Terminal mit **SCRIPT** als Name des Programmes:

```
pep8 –first SCRIPT.py
```

DATAMICS
machine intelligence consulting services