PEPs - Python Enhancement Proposals

Proseminar: Python im Hochleistungsrechnen

Noah Fuhst

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

2019-29-04

Gliederung

1. Einleitung
   - Die Entwicklung von Python
   - Warum sind die PEPs wichtig?
2. Hauptteil
   - Was sind PEPs?
   - Welche Arten von PEPs gibt es?
   - Wichtige PEPs
   - Tools zum Einhalten der PEPs
   - Die Erstellung und Aufnahme neuer PEPs
3. Zusammenfassung
4. Literatur

Crashkurs Geschichte

- Entwickelt von Guido van Rossum (bis 2018 "wohlwollender Diktator auf Lebenszeit")
- Python 0.9 1991
- Python 2.1 2001 (unter PSF)
- Alle Versionen Open Source

Abbildung: Python Logo [1]

Abbildung: Guido van Rossum [2]

Warum sind die PEPs wichtig?

- Python ist Open-Source $\Rightarrow$ Viele Entwickler
- Kommunikation unter den Entwicklern
- Vorschlagen neuer Features
- Feedback / Dokumentation

Was sind PEPs?

- "PEP stands for Python Enhancement Proposal. A PEP is a design document providing Information to the Python community, or describing a new feature for Python or its processes or environment"[Cog00]
- Informationen über Python in verschiedenen Bereichen
- Beantworten generelle Fragen
- Helfen bei der Programmierung
- Nutzung oft freigestellt, aber empfohlen

Welche Arten von PEPs gibt es?

Nach PEP-1:

1. Neue Features oder Implementationen in Python
   - Patchnotes
   - Bugfixes
   - Vorschläge
2. Informierende PEPs
   - Einhaltung freiwillig
   - Anwendung bereits existierender Implementationen
3. Prozess-PEPs
   - Prozesse um Python herum
   - Einhaltung verpflichtend
   - META-PEPs
   - z.B. PEP 8: Style Guide for Python Code

Wichtige PEPs

1. PEP 0: Index of Python Enhancement Proposals (PEPs)
2. PEP 1: PEP Purpose and Guidelines
3. PEP 8: Style Guide for Python Code
4. PEP 20: The Zen of Python

PEP 0: Index of Python Enhancement Proposals (PEPs)

- Liste aller PEPs und Autoren
- Verschiedene Implementationszustände
- Auch abgelehnte PEPs
- PEP-History ebenfalls abrufbar

PEP 1: PEP Purpose and Guidelines

- Was sind PEPs?
- Adressaten
- PEP Erstellungs- und Approval-Prozess
- PEP Wartung und Aktualisierung
- Tipps für erfolgreiche PEPs
- Formatierung etc.

PEP 8: Style Guide for Python Code

- Wahrscheinlich wichtigstes PEP
- Quelltextkonventionen für Python Code
- Soll für lesbaren Code sorgen
- Generellere Hinweise in PEP 20

PEP 8 Crashkurs I[1]

- Einrückung in Python nicht optional!
- 4 Leerzeichen pro Einrücklevel (Keine Tabs!)

```
1 def method(self):
2     a = 1
```

- Maximal 79 Zeichen pro Zeile

```
1 def method(self, variable_1, variable_2, variable_3, va
```

- Binäre Operanden nach Zeilenumbrüchen
- Vor Klassen zwei freie Zeilen, vor Methoden eine

[1] Quelle [8]

PEP 8 Crashkurs II[2]

- Konstante Werte groß schreiben mit Unterstrichen

```
1 KONSTANTER_WERT = 30
```

- Imports in eigenen Zeilen

```
1 import math
2 import cmath
```

- Lücken um Vergleichs- oder Zuweiseoperationen

```
1 a = 1
2 if(b == True)...
3 j and k
```

- Kommentare auf Englisch

```
1 # Comments are to be written in English
```

[2] Quelle [8]

PEP 8 Crashkurs III[3]

- Blockkommentare > Zeilenkommentare

```
1 # This is better than the one below
2 def method(self):
3     a = 1 # increments a
```

- CamelCase Nameskonvention für Klassen

```
1 class NeueKlasse
```

- Variablen kleingeschrieben mit Unterstrichen

```
1 neue_variable = "Neu"
```

[3] Quelle [8]

Vergleich 1: Falsch

```
import Konto, math

max_number=100 #constant!
class New:
    def method(self, variable_1, variable_2, variable_3, variable_4, variable_5, variable_6):
        x=1
    def berechnemonatlichekosten(self, salary, rent, car_insurance, food, internet, interest, g
        Health_Insurance=90
        gelduebrig=(salary-#takes salary
                    rent-#deducts rent
                    car_insurance+#dedurcts car insurance
                    food-#deducts food costs
                    internet+#deducts internet costs
                    interest-#adds bank account interest
                    gasoline)#deducts gasoline
        if gelduebrig<0and self.method(1, 2, 3, 4, 5, 6):
            salary=0
            Konto.kontoschliessen()
class New2:
    def method2(self):
        a=1
```

Vergleich 2: Richtig

```python
import Konto
import math

MAX_NUMBER = 100


class New:

    def method(self, variable_1, variable_2, variable_3, variable_4,
               variable_5, variable_6):
        x = 1

    # Calculates the monthly costs of living alone. That is, if you even get
    # interest in the current financial situation.
    def berechnemonatlichekosten(self, salary, rent, car_insurance, food,
                                 internet, interest, gasoline):
        health_insurance=90
        geld_uebrig=(salary
                     - rent
                     - car_insurance
                     - food
                     - internet
                     + interest
                     - gasoline)
        if geld_uebrig < 0 and self.method(1, 2, 3, 4, 5, 6):
            salary = 0
            Konto.kontoschliessen()


class New2:

```

PEP 20

- Etwas anderes PEP
- Grundlegende Codedesigntipps
- Muss nicht eingehalten werden
- Stellt die Philosophie hinter der Sprache dar
- Zusammenspiel mit PEP 8

Link

Tools zum Einhalten der PEPs

- Tools zum Einhalten von gewissen PEPs
- Z.B. flake8, pylint, autopep8
- Teilweise in IDEs eingebaut
- Nicht nur Überprüfung von PEPs

Die Erstellung und Aufnahme neuer PEPs

- Jeder Python-Entwickler kann PEPs vorschlagen
- Erstellung eines Dokuments im PEP GitHub
- Prüfung durch die PEP-Autoren bzgl. Formatierung
- Feedback von der Community einholen
- Endgültige Prüfung von den "Core Developersöder dem Steering Council
- Abschließend Implementation des Vorschlags

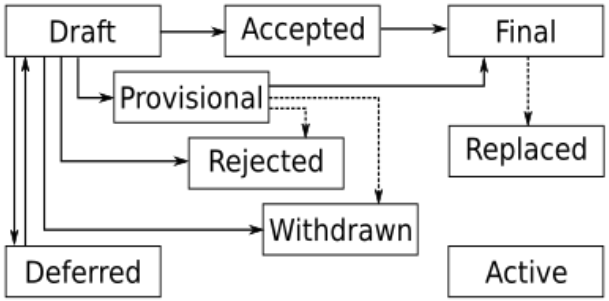

Abbildung: Übersicht der Zustände von PEPs [3]

Zusammenfassung

- PEPs = Regeln und Richtlinien für alles was Python betrifft, FAQs
- Nutzung von Informationellen PEPs freigestellt, META-PEPs und Prozess-PEPs verpflichtend
- Wichtigstes PEP: PEP 8 - Quelltextkonventionen
- Tools zum Einhalten von PEP 8, wie flake8, pylint, autopep8
- PEP 20 zur Philosophie von Python (Gut lesbarer Code)

Literatur

[Cog00] Barry Warsaw; Jeremy Hylton; David Goodger; Nick Coghlan. PEP 1 – PEP Purpose and Guidelines. 06 2000. Letzter Zugriff: 14.04.2019.

Liste aller verwendeten Quellen

[1] Barry Warsaw, Jeremy Hylton, David Goodger, Nick Coghlan: "PEP 1 – PEP Purpose and Guidelines", https://www.python.org/dev/peps/pep-0001/, Letzter Zugriff: 21.04.2019

[2] Python-Dev: "PEP 0 – Index of Python Enhancement Proposals", https://www.python.org/dev/peps/, Letzter Zugriff: 14.04.2019

[3] Python Software Foundation: "History and License – Python 3.7.3 Documentation", https://docs.python.org/3/license.html, Letzter Zugriff: 14.04.2019

[4] Hideo Hattori: 'autopep8 1.4.4', https://pypi.org/project/autopep8/, Letzter Zugriff: 14.04.2019

Liste aller verwendeten Quellen II

[5] Python Software Foundation: "General Python FAQ", https://docs.python.org/3/faq/general.html, Letzter Zugriff: 14.04.2019

[6] Python Software Foundation: "What is Python? Executive Summary", https://www.python.org/doc/essays/blurb/, Letzter Zugriff: 14.04.2019

[7] "Guido van Rossum", https://de.wikipedia.org/wiki/Guido_van_Rossum, Letzter Zugriff: 14.04.2019

Liste aller verwendeten Quellen III

[8] Guido van Rossum, Barry Warsaw, Nick Coghlan: "PEP 8 – Style Guide for Python Code", https://www.python.org/dev/peps/pep-0008/, Letzter Zugriff: 14.04.2019

[9] "Benevolent Dictator for Life", https://de.wikipedia.org/wiki/Benevolent_Dictator_for_Life, Letzter Zugriff: 14.04.2019

[10] Aahz, Anthony Baxter: "PEP 6 – Bug Fix Releases", https://www.python.org/dev/peps/pep-0006/, Letzter Zugriff: 14.04.2019

Liste aller verwendeten Quellen IV

[11] "Pylint - code Analysis for Python", https://www.pylint.org/, Letzter Zugriff: 14.04.2019

[12] "Flake8: Your Tool For Style Guide Enforcement", http://flake8.pycqa.org/en/latest/index.html, Letzter Zugriff: 14.04.2019

[13] Python Software Foundation: "Python 3.0 Release", https://www.python.org/download/releases/3.0/, Letzter Zugriff: 14.04.2019

Liste aller verwendeten Quellen V

[14] Barry Warsaw: "PEP 10 – Voting Guidelines", https://www.python.org/dev/peps/pep-0010/, Letzter Zugriff: 14.04.2019

[15] David Goodger; Guido van Rossum: "PEP 257 – Docstring Conversions", https://www.python.org/dev/peps/pep-0257/, Letzter Zugriff: 14.04.2019

[16] Tim Peters: "PEP 20 – The Zen of Python", https://www.python.org/dev/peps/pep-0020/, Letzter Zugriff: 21.04.2019

Abbildungsverzeichnis

[1] Python Software Foundation: "The Python Logo"https://www.python.org/community/logos/, Letzter Zugriff: 21.04.2019

[2] Doc Searls: "File: Guido van Rossum OSCON 2006.jpg", https://commons.wikimedia.org/w/index.php?curid=4974869, Letzter Zugriff: 21.04.2019

[3] Barry Warsaw, Jeremy Hylton, David Goodger, Nick Coghlan: "PEP 1 – PEP Purpose and Guidelines", Letzter Zugriff: 21.04.2019