Bildquelle: https://sitefactory.com.br/marketing-digital/startups-utilizam-marketing-digital-ao-seu-favor/

› PYLINT

Fallstudie Entwicklungswerkzeuge

INHALT

- Allgemeines über Pylint
- Linter
- PEP 8
- Wofür wird Pylint benötigt
- Praxis Teil
- Vor- und Nachteile
- Alternativen/ Konkurenz

ALLGEMEINES ÜBER PYLINT

- Linter für Python
- Statische Codeanalyse
- Prüft auf richtige Formatierung
- Beinhaltet Pyverse
- Verwendung alleinstehend oder in IDE Integriert

- Veröffentlichung: 2001
- Entwickler: Sylvain Thénault
- Geschrieben in Python

WAS SIND LINTER

- Tools zur statischen Code Analyse
- Zeigen Programm-, Formatierungsfehler und verdächtige Strukturen auf.
- Kommt ursprünglich aus Unix

PEP 8

- **Code Layout:**
  - Leerzeichen anstatt Tabulator
  - Klammern außerhalb von Listen
  - Max. Zeilenlänge: 79 Zeichen
  - Leerzeichen richtig setzten

- **Namenskonventionen:**
  - Keine einzelnen Zeichen
  - Variablennamen in snake_case
  - Klassennamen in CamelCase

- **Programmier Empfehlungen:**
  - Kein overload in return statements

WOFÜR WIRD PYLINT BENÖTIGT

- Clean Code
- Code-Qualität erhöhen
- Fehler reduzieren
- Code wird lesbarer und wartbarer
- Einheitlicher Code
- Objektive Messbarkeit der Code Qualität

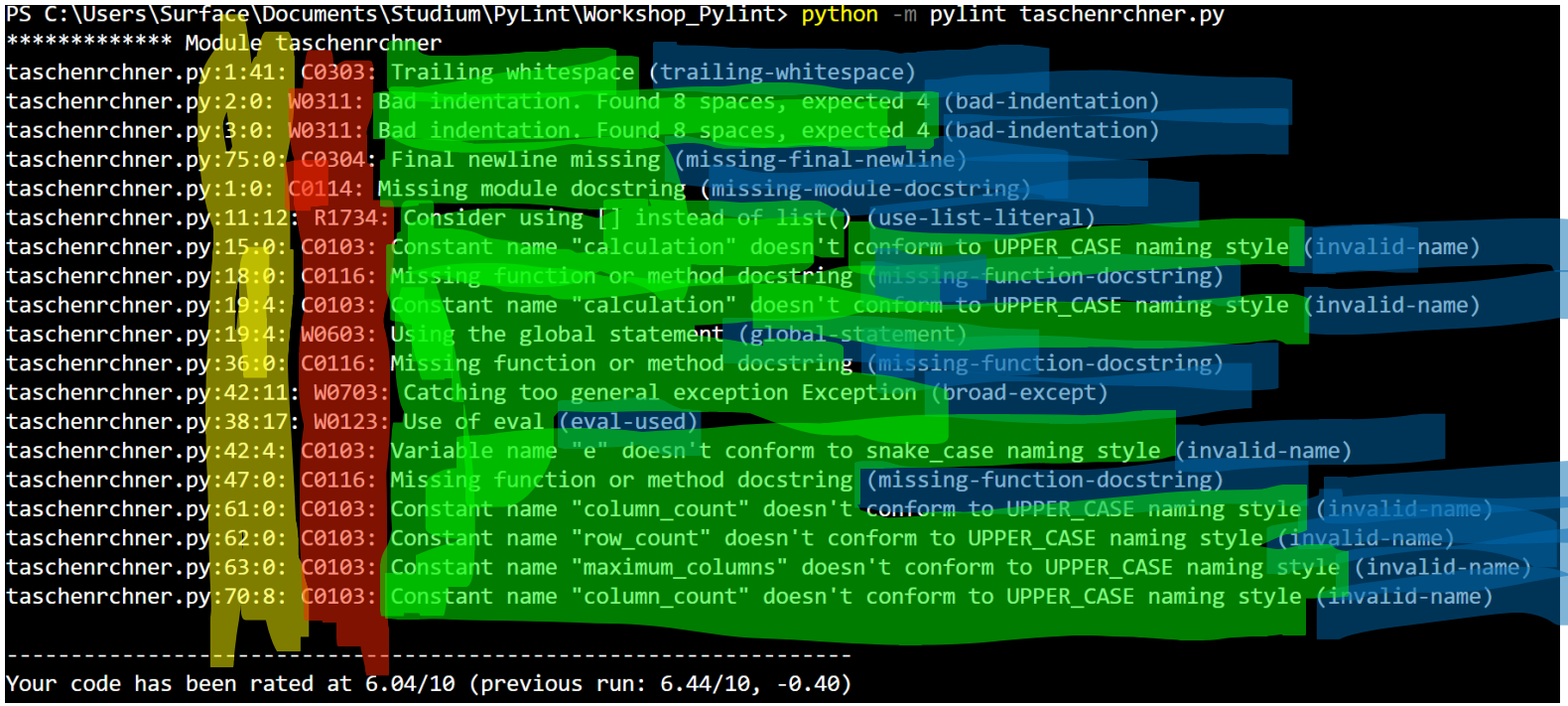
› PRAXIS TEIL 1

PyLint Allgemein
Reports erstellen
Filtern
Hilfe anfragen

AUFGABE 1 – PYLINT AUSFÜHREN

Führt Pylint durch "pylint <dateiname>.py" auf einem Python Code aus, an welchem Ihr selbst gearbeitet habt.

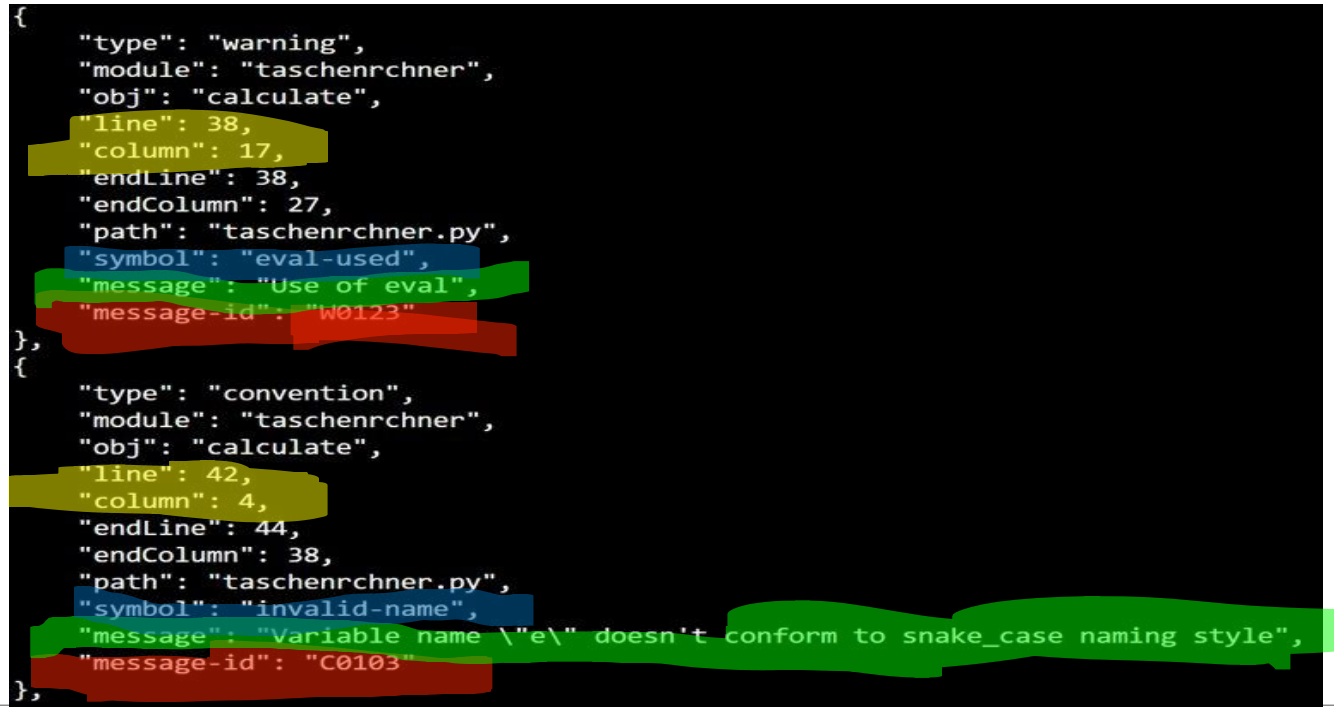
- Zeilen, in denen die Rückmeldung zutrifft
- Message-ID
- Message
- Symbol



AUFGABE 1 – PYLINT AUSFÜHREN

Führt Pylint nochmals auf euren Code aus. Diesmal ändert jedoch das Output-format zu Json mithilfe des Befehls "--output-format=<format>"
--> Symbol, Massage und Massage-id sind die wichtigsten Elemente

- Zeilen, in denen die Rückmeldung zutrifft
- Message-ID
- Message
- Symbol



AUFGABE 1 – PYLINT AUSFÜHREN

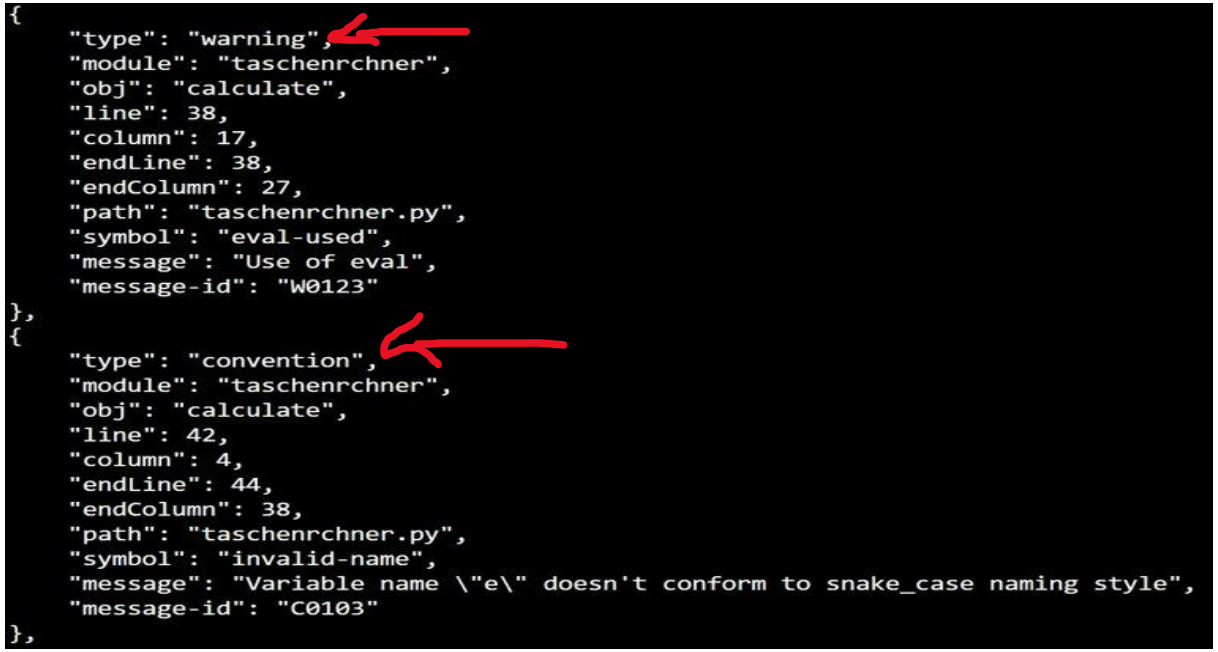
(C) convention, für nicht eingehaltende Standards

(R) refactor, für "bad code smell"

(W) warning, für python-spezifische Probleme

(E) error, für mögliche Bugs im Code

(F) fatal, falls ein Error ensteht, welches die Ausführung von Pylint verhindert



AUFGABE 2 – PYLINT REPORT ERSTELLEN

Wendet Pylint mit dem „–report=y“-Befehl an, um mehr Informationen über eure Analyse zu erhalten. Schreibt raus:

- Anzahl, wie oft jede Kategorie der Meldungen (Conventions, Warnings, Refactors, Error) vorkam
- Die Top 3 Rückmeldungen, die am häufigsten vorkommen (wichtig für später)

AUFGABE 3 – FILTERN & HILFE ANFRAGEN

1. Führt Pylint wieder auf euren code aus, diesmal filtert mithilfe des "--disable=<massage id/category/message>" Befehls.
   --> Filtert alle Rückmeldungen vom Type "Convention" raus
   --> Filtert die meist vorkommende Meldung in eurem Code raus (Top 3 Meldungen)

2. Durch "pylint –help" kann Hilfestellung zu PyLint ausgeführt werden. Das geht auch bei einzelnen Rückmeldungen mithilfe des Befehls "pylint –help=<message-id/message>" Nutze den Befehl, um mehr Informationen zu deinen Top 3 Nachrichten zu erhalten.

› PAUSE
15 MINUTEN

› PRAXIS TEIL 2

Pylint Übungen

AUFGABE 4 – ÜBUNGSAUFGABE CONVENTIONS

1. Führt Pylint auf „Convention.py“ aus und behebt alle Rückmeldungen. Zur Hilfe habt ihr das Cheat Sheet und den –help=<massage-id/message> - Befehl.

AUFGABE 4 – ÜBUNGSAUFGABE CONFIGURATION

1. Erstellt eine Configuration-Datei. Erstellt hierzu eine txt-Datei und nennt sie "myconfig.pylintrc".

   a) Erstellt einen "Checker". Schreibt dazu folgenden Text in die "myconfig.pylintrc":
   [MESSAGES CONTROL]
   --disable=C

AUFGABE 4 – ÜBUNGSAUFGABEN

Führt Pylint auf folgende Dateien aus und behebt die Rückmeldungen:

- Error.py / Behebt nur die Errors! Filtert alle Rückmeldungen raus, welche nicht zum Typ "Error" gehören.
- Refactor.py / Behebt nur die Refactors! Filtert alle Rückmeldungen raus, welche nicht zum Typ "Refactor" gehören.
- Warning.py / Behebt nur die Warnings! Filtert alle Rückmeldungen raus, welche nicht zum Typ "Warning" gehören.
- Tipp: Parallele ausführung durch "pylint <dateiname1>.py, <dateiname2>.py, …"
  Ihr könnt zum filtern –disable=<message (id)> im Terminal oder in der Config-Datei eingeben.

BONUSAUFGABE

Führt Pylint auf der "MaXeRrOrS.py" Datei aus und sorgt für einen Score von 10.

Ihr dürft 3 Sachen in der Config-Datei verändern oder deaktivieren, die euch helfen eure Rückmeldungen zu beheben.

› PRAXIS TEIL 3

Abschlussaufgabe

ABSCHLUSSAUFGABE

Sorgt bei der Datei "taschenrchner.py" für einen Score von 10/10.

Die Konstanten sind gewollt nach dem snake_case naming style benannt worden.

In der Standard Konfiguration wird jedoch "UPPER_CASE" als naming sytle bei Kostanten geprüft.
Setze in der Config-Datei den "--const-naming-style=<naming-style>" - Checker um die Prüfung des naming style zu ändern.

VORTEILE

- Mit Pylint können ohne Probleme aktuelle Quelltexte geprüft werden
- Das Werkzeug wird heute noch weiterentwickelt und gepflegt
- Pylint kann einen Codierungsstil durchsetzen, was dazu beitragen kann, dass der Code leichter zu lesen und zu warten ist.
- Pylint bietet über eine Konfigurationsdatei die Möglichkeit, die Analyse einzuschränken und auf seine Bedürfnisse anzupassen.

NACHTEILE

- Bei Standard Konfiguration kann eine falsch Positive Fehlermeldung kommen
- Pylint kann langsam sein, insbesondere bei vielen Zeilen Code
- Pylint findet nicht alle Arten von Fehlern
- So können Syntaxfehler oder Logikfehler nicht gefunden werden
- Der Code wird nicht ausgeführt, um Laufzeitfehler zu überprüfen

VERGLEICH MIT ANDEREN TOOLS: PYFLAKES

- Pyflakes ist ein Static-Analysis-Werkzeug für Python
- Quellcode auf Syntaxfehler und andere Probleme überprüft
- Es überprüft keine Stilprobleme oder Code-Komplexität
- schnell und einfach zu verwenden

ALTERNATIVE 2: FLAKE 8

- Flake 8 ist ein Static-Analysis-Werkzeug
- Überprüft Quellcode auf Syntaxfehler, Stilprobleme und Code-Komplexität
- Kombiniert mehrere Werkzeuge, darunter Pyflaskes, Pycodestyle und McCabe

Problem	Pylint output	Flake8 output	PyFlakes output
unnecessary semicolon	✓	✓	✗
missing newlines	✓	✓	✗
missing whitespaces	✗	✓	✗
missing docstring	✓	✗	✗
unused arguments	✓	✓	✓
incorrect naming style	✓	✗	✗
selecting element from the list with incorrect index	✗	✗	✗
dividing by zero expression	✗	✗	✗
missing return statement	✓	✗	✗
calling method that doesn't exist	✓	✓	✓
unnecessary pass statement	✓	✗	✗

MENTIMETER

The voting code **7863 9653**

HOCHSCHULE HEILBRONN

VIELEN DANK FÜR IHRE AUFMERKSAMKEIT

Oliver Daub, odaub@stud.hs-heilbronn.de

Dennis Stehle , dstehle@stud.hs-heilbronn.de

Robin Starkl, rstarkl@stud.hs-heilbronn.de