

DSCB 230 - Aufgabenblatt 12

Die Aufgabenblätter werden in dieser Veranstaltung in Jupyter Notebooks veröffentlicht und bearbeitet. Diese finden Sie in der Github Organisation für Data Science 2 unter dem Repository *dscb230-tutorial* (<https://github.com/hka-mmvm/dscb230-tutorial>). Die Musterlösung wird ebenfalls in Form eines Jupyter Notebook in Github hochgeladen.

Linting im Terminal und in VS Code

Um „Code-Konventionen“ einzuhalten, kann man auf bestimmte Module wie z.B. pylint zugreifen. Sie unterstützen beim Programmieren, sodass z.B. Konventionen bei der Variablenbenennung oder Whitespace zur besseren Lesbarkeit des Codes eingehalten werden.

Es gibt grundsätzlich zwei Möglichkeiten, wie man Linter benutzen kann. Zum einen lassen sie sich über das Terminal/Konsole ansteuern, zum anderen gibt es in VS Code Extensions, die auch eine visuelle Unterstützung ermöglichen.

In Folgendem wird anhand des Moduls „pylint“ (<https://pypi.org/project/pylint/>) gezeigt, welche Möglichkeiten es gibt, solche Module zu installieren und zu verwenden.

Verwendung über das Terminal/Konsole

Installation

Möchte man den Linter über das Terminal/Konsole verwenden, kann man sich das Modul über pip/conda/etc. herunterladen.

```
$ pip install pylint
```

Verwendung

Betrachten wir nun folgenden Code-Ausschnitt. Dieser soll mithilfe von pylint „aufgeräumt“ werden. Lege dir eine Datei mit diesem Code an, um den Umgang mit pylint zu üben.

```
import datetime

class dog:
    def __init__(self, name, age, position= "standing"):
        self Name = name
        self age = age
        self position = position

    def sit(self)
        self.position="sitting"

    def __str__(self):
        return f"Dog {self Name}"
```

Wie man sehen kann, werden Variablen hier nicht einheitlich benannt, es fehle Docstrings und Der Import von datetime ist unnötig.

Navigiere im Terminal/Konsole nun zu dem Ordner, der deine Datei beinhaltet und führe folgenden Befehl aus („pylint_demo.py“ ist hierbei durch deinen Dateinamen zu ersetzen).

```
$ pylint pylint_demo.py
```

Du solltest nun folgenden Output bekommen.

```
***** Module pylint_demo
pylint_demo.py:17:0: C0304: Final newline missing (missing-final-newline)
pylint_demo.py:1:0: C0114: Missing module docstring (missing-module-docstring)
pylint_demo.py:3:0: C0103: Class name "dog" doesn't conform to PascalCase naming style (invalid-name)
pylint_demo.py:5:8: C0103: Attribute name "Name" doesn't conform to snake_case naming style (invalid-name)
pylint_demo.py:3:0: C0115: Missing class docstring (missing-class-docstring)
pylint_demo.py:9:4: C0116: Missing function or method docstring (missing-function-docstring)
pylint_demo.py:1:0: W0611: Unused import datetime (unused-import)
```

gibt die Zeile
an, in der der
Fehler steht

Fehlertyp

Beschreibung

Außerdem wird der Code bewertet und bekommt ein Rating.

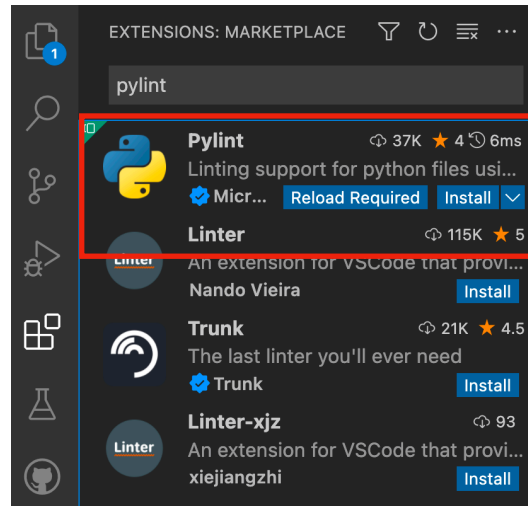
```
-----
Your code has been rated at 4.17/10 (previous run: 4.17/10, +0.00)
```

Man kann nun anhand des Outputs seinen Code verbessern.

Verwendung mit VS Code

Einrichtung

Suche in den Extensions nach pylint und installiere die Extension.



Falls dein Code jetzt noch nicht wie folgt farblich markiert wird, muss noch ein Schritt durchgeführt werden.

```
import datetime

class dog:
    def __init__(self, name, age, position= "standing"):
        self.Name = name
        self.age = age
        self.position = position

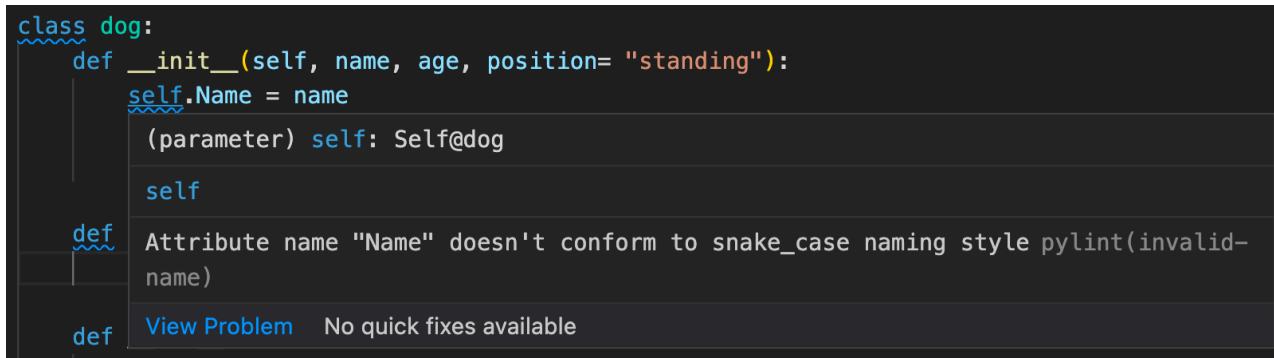
    def sit(self):
        self.position="sitting"

    def __str__(self):
        return f"Dog {self.Name}"
```

Suche in den Einstellungen von VS Code nach „Terminal“, klicke auf „Edit in settings.json“ und füge folgende Zeile hinzu:

```
13     "jupyter.askForKernelRestart": false,
14     "jupyter.widgetScriptSources": [
15         "jsdelivr.com",
16         "unpkg.com"
17     ],
18
19     "python.linting.pylintEnabled": true,
20
21 ]
```

Wenn man nun in der Datei über eine Markierung hovert, wird angezeigt, was man verbessern sollte.



```
class dog:
    def __init__(self, name, age, position= "standing"):
        self.Name = name
        (parameter) self: Self@dog
        self
    def
        Attribute name "Name" doesn't conform to snake_case naming style pylint(invalid-
        name)
    def
        View Problem No quick fixes available
```

In diesem Fall wird z.B. angemerkt, dass die Variable „Name“ nicht den Konventionen (hier snake_case) entspricht.

Jedes mal, wenn man die Datei abspeichert, wird pylint erneut ausgeführt und zeigt aktuelle Konventions-Verletzungen an.

Aufgaben:

1. Verwende das Terminal, um den Code-Ausschnitt den Konventionen entsprechend zu schreiben.
2. Verwende die Extension in VS Code, um den Code-Ausschnitt den Konventionen entsprechend zu schreiben.