Propertys nur für spezielle getter und setter, nicht für alle!

LU12d - Property

Dieses Thema ist ein Blick in die Objektorientierte Programmierung im Modul 320.

Mit `@property` und `@attribut.setter` kann eine Klasse den Zugriff auf die Attribute kontrollieren. Dieses Prinzip nennt sich `Data Hiding`.

property und setter

```
...
@property
def firstname(self):
    return self._firstname

@firstname.setter
def firstname(self, value):
    self._firstname = value
...
```

Wenn wir eine Klasse programmieren, wollen wir das Lesen und Schreiben von Attributen kontrollieren. Die Kontrolle ob und wie die Werte der Attribute geändert werden können soll in der Hand der Klasse bleiben. Deshalb schreiben wir zu jedem Attribut eine *Getter*- und eine *Setter*-Methode.

Will ein andere Programmteil den Wert eines Attributs lesen, erfolgt dies über die sogenannte `Getter`-Methode. Diese Methode wird mit dem Decorator `@property` definiert. Im einfachsten Fall gibt diese Methode einfach den Wert des Attributs an den Aufrufer zurück. **Beachten Sie den Underscore vor dem Variablennamen, z.B. _firstname**

In komplexeren Klassen werden einzelne Getter auch eine Verarbeitungslogik beinhalten. Für unsere Programme reichen aber einfache Getter.

Um den Wert eines Attributs zu ändern, stellt unsere Klasse eine `Setter`-Methode bereit. Diese Methode wird mit `@attributname.setter` definiert. Deren Aufgabe ist es, den Wert eines Attributs zu ändern. Auch hier beschränken wir uns vorerst auf die Grundaufgabe: Nimm den `value` und speichere ihn. **Beachten Sie den Underscore vor dem Variablennamen, z.B. _firstname**

Um das Erstellen von `property` und `setter` zu vereinfachen, gibt es in PyCharm sogenannte „Live Templates“. Eine kurze Anleitung zum Erstellen eines solchen Live Templates finden Sie unter https://it.bzz.ch/wikiV2/howto/pycharm/property

Die vollständige Klasse würde dann so aussehen:

member.py

```
from dataclasses import dataclass


@dataclass
class Member:
    """
    a club member
    """
    firstname: str
    lastname: str
    address: str
    place: str
    zip_code: str
    entry_year: int
    birth_year: int
    honorary_member: bool

    @property
    def firstname(self):
        return self._firstname

    @firstname.setter
    def firstname(self, value):
        self._firstname = value

    @property
    def lastname(self):
        return self._lastname

    @lastname.setter
    def lastname(self, value):
        self._lastname = value

    @property
    def address(self):
        return self._address

    @address.setter
```

```
    def address(self, value):
        self._address = value

    @property
    def place(self):
        return self._place

    @place.setter
    def place(self, value):
        self._place = value

    @property
    def zip_code(self):
        return self._zip_code

    @zip_code.setter
    def zip_code(self, value):
        self._zip_code = value

    @property
    def birth_year(self):
        return self._birth_year

    @birth_year.setter
    def birth_year(self, value):
        self._birth_year = value

    @property
    def entry_year(self):
        return self._entry_year

    @entry_year.setter
    def entry_year(self, value):
        self._entry_year = value

    @property
    def honorary_member(self):
        return self._honorary_member

    @honorary_member.setter
    def honorary_member(self, value):
        self._honorary_member = value

if __name__ == '__main__':
    test_person = Person()
    test_person.firstname = 'Hans'
    print(test_person)
```