Modifikátory přístupnosti

PVA2 Programování a vývoj aplikací

Adam Fišer

Obsah

Co jsou modifikátory přístupnosti

- Modifikátory přístupnosti jsou klíčová slova, která určují, kdo může přistupovat k danému atributu nebo metodě.
- Python má 4 modifikátory přístupnosti:
  - `public` - veřejný, přístupný zvenčí
  - `protected` - chráněný, přístupný pouze z třídy a dědičných tříd
  - `private` - privátní, přístupný pouze z třídy
  - `static` - statický, přístupný bez vytvoření instance třídy

Modifikátory přístupnosti v Pythonu

- Python nemá striktní modifikátory přístupnosti jako jiné jazyky (např. Java).
- Modifikátory přístupnosti jsou pouze konvencí.
- Python používá tzv. "name mangling" pro privátní atributy.
- Modifikátory přístupnosti se zapisují pomocí podtržítek:
  - `public` - `atribut`
  - `protected` - `_atribut`
  - `private` - `__atribut`
  - `static` - `atribut`

Public

- Veřejný modifikátor přístupnosti.
- Atribut nebo metoda je přístupná zvenčí.
- Přístupná z jakékoliv části programu.
- Syntaxe: `atribut` nebo `metoda()`

```
class Auto:
    znacka = "Škoda"
    def vypis_znacku(self):
        print(self.znacka)
```

Protected

- Chráněný modifikátor přístupnosti.
- Atribut nebo metoda je přístupná pouze z třídy a dědičných tříd.
- Zapisuje se pomocí prefixu `_` před začátkem názvu atributu nebo metody.
- Syntaxe: `_atribut` nebo `_metoda()`

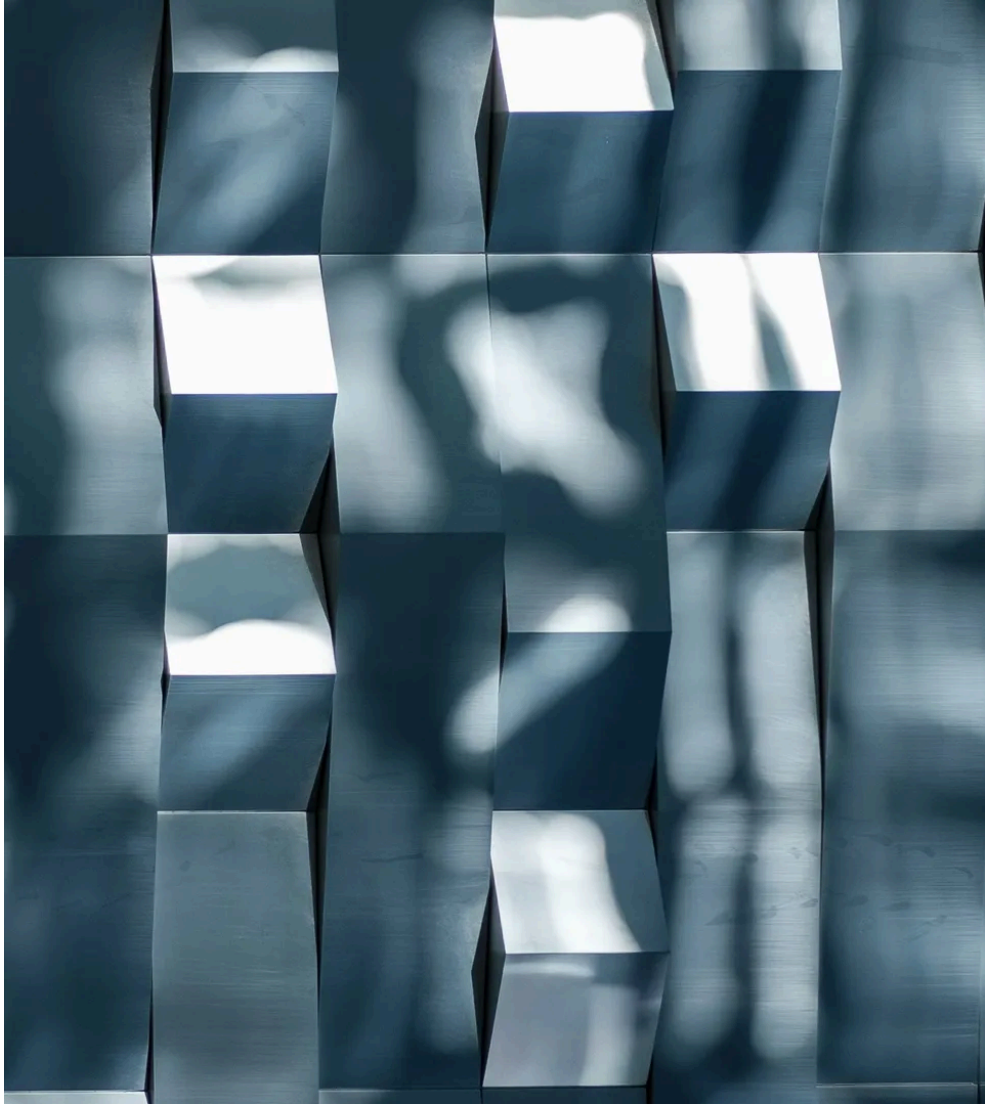
```
1   class Auto:
2       _model = "Octavia"
3       def _vypis_model(self):
4           print(self._model)
```

Private

- Privátní modifikátor přístupnosti.
- Atribut nebo metoda je přístupná pouze z třídy.
- Zapisuje se pomocí prefixu `__` před začátkem názvu atributu nebo metody.
- Syntaxe: `__atribut` nebo `__metoda()`

```
class Auto:
    __vin = "1234567890"
    def __vypis_vin(self):
        print(self.__vin)
```

Getter a Setter



Gettery a Settery

- Gettery a settery jsou metody třídy, které slouží k čtení a zápisu hodnot atributů.
- Výhoda je možnost kontroly hodnot při zápisu.
- Čitelnější kód a snadnější údržba.
- Gettery a settery se používají minimálně pro privátní atributy.
- Způsob zápisu závisí na konvencích v Python např
  - Zapisují se s prefixem `get_` a `set_` před název atributu.
  - V python alternativně lze použít dekorátory `@property` a `@atribut.setter`.

Příklad

```python
class Osoba:
    def __init__(self):
        self._vek = 0

    def get_vek(self):
        return self._vek

    # Setter
    def set_vek(self, vek):
        if vek < 18:
            raise ValueError("Omlouváme se, že váš věk je nižší než kritéria způsobilosti.")
        self._vek = vek
```

```python
otakar = Osoba()
otakar.set_vek(19) # Volání setteru nikoli atributu
print(otakar.get_vek()) # Volání getteru nikoli atributu
```

Alternativní zápis s dekorátory

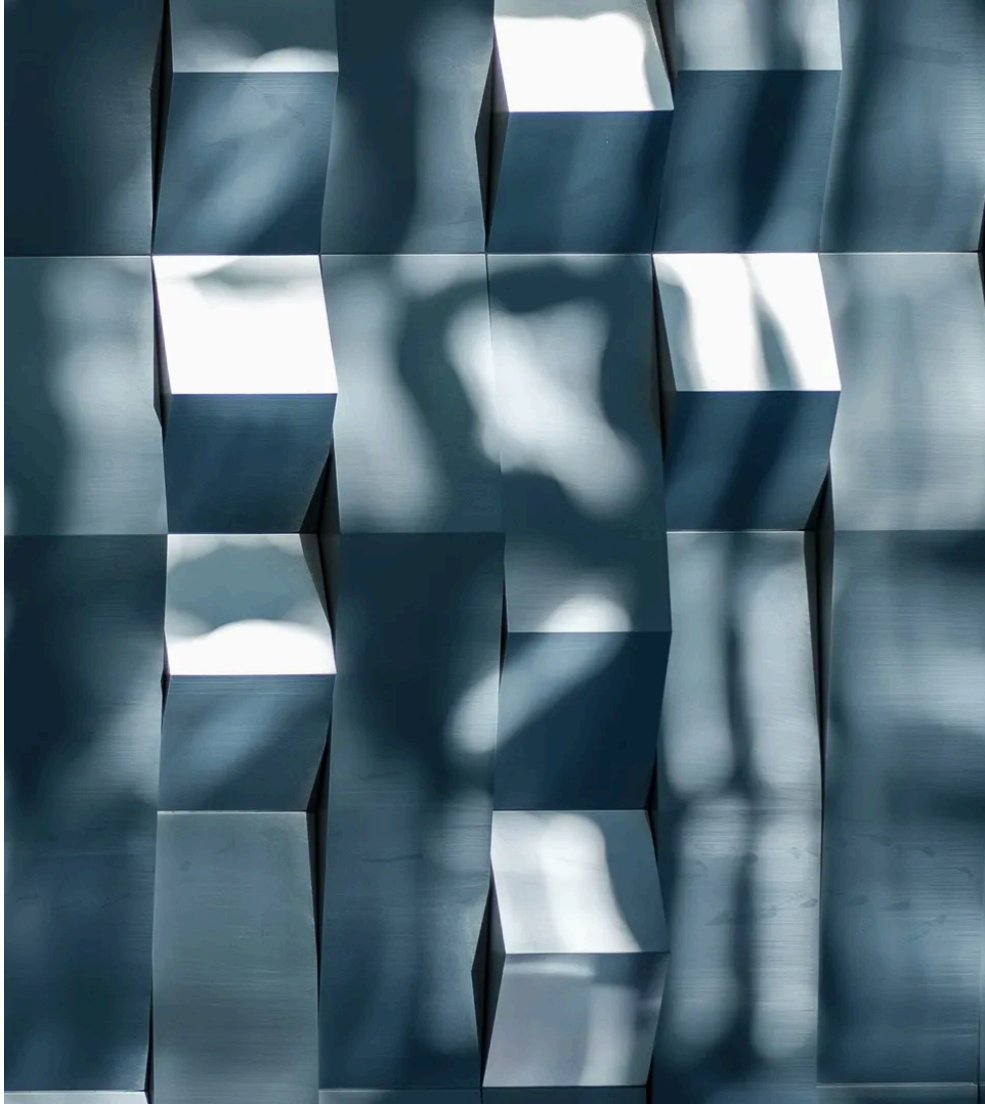
```python
class Osoba:
    def __init__(self):
        self._vek = 0

    @property
    def vek(self):
        return self._vek

    @vek.setter
    def vek(self, vek):
        if vek < 18:
            raise ValueError("Omlouváme se, že váš věk je nižší než kritéria způsobilosti.")
        self._vek = vek
```

```python
otakar = Osoba()
otakar.vek = 19 # Volání setteru nikoli atributu
print(otakar.vek) # Volání getteru nikoli atributu
```

Příklad



Třída `Thermometer`

```
15            if scale == "C":
16                self._temperature_celsius = value
17            elif scale == "K":
18                if value < 0:
19                    raise ValueError("Teplota v Kelvinech nemůže být záporná!")
20                self._temperature_celsius = value - 273.15
21            elif scale == "F":
22                self._temperature_celsius = (value - 32) * 5 / 9
23            else:
24                raise ValueError("Neplatná jednotka! Použijte 'C', 'K' nebo 'F'.")
25
26        def get_temperature(self):
27            """Getter pro aktuální teplotu v °C"""
28            return self._temperature_celsius
29
30        def get_temperature_fahrenheit(self):
31            """Vrací teplotu ve stupních Fahrenheit"""
32            return self._temperature_celsius * 9 / 5 + 32
33
34        def get_temperature_kelvin(self):
35            """Vrací teplotu v Kelvinech"""
36            return self._temperature_celsius + 273.15
```

Použití třídy `Thermometer`

```python
from thermometer import Thermometer

# Vytvoření instance třídy Thermometer
thermometer = Thermometer(25)

# Výpis teploty ve stupních Celsia, Fahrenheita a Kelvinech
print("Teplota v °C:", thermometer.get_temperature())
print("Teplota v °F:", thermometer.get_temperature_fahrenheit())
print("Teplota v K:", thermometer.get_temperature_kelvin())

# Nastavení teploty v Kelvinech
thermometer.set_temperature(300, "K")
print("\nPo nastavení na 300 K:")
print("Teplota v °C:", thermometer.get_temperature())
print("Teplota v °F:", thermometer.get_temperature_fahrenheit())
print("Teplota v K:", thermometer.get_temperature_kelvin())

# Pokus o nastavení záporné hodnoty v Kelvinech (vyvolá chybu)
try:
    thermometer.set_temperature(-10, "K")
except ValueError as e:
    print("\nChyba:", e)
```

Děkuji za pozornost

Otázky?

Repository / Prezentace