Pystart.pl

Dekorator property

lekcja czterdziesta pierwsza

PyStart #41 Dekorator property

Dlaczego? Pamiętacie metody dostępowe?

```python
class Person:
    def __init__(self, first_name, last_name):
        self._first_name = first_name
        self._last_name = last_name

    def first_name(self):
        return self._first_name

    def last_name(self):
        return self._last_name

    def details(self):
        return f'{self._first_name} {self._last_name}'
```

- ➔ Te metody przypominają bardziej właściwości
- ➔ Natomiast details() po prostu zwraca dynamiczne właściwości.

PyStart #41 Dekorator property

Dlaczego? Pamiętacie metody dostępowe?

```
15
16      person = Person(first_name='Jan', last_name='Kowalski')
17      print(person.first_name())
18      print(person.last_name())
19      print(person.details())
20
```

➔ Te puste metody wyglądają brzydko, dlatego zrobimy tak..

```
12      @property
13      def details(self):
14          return f'{self._first_name} {self._last_name}'
```

PyStart #41 Dekorator property

Dlaczego? Pamiętacie metody dostępowe?

```python
class Person:
    def __init__(self, first_name, last_name):
        self._first_name = first_name
        self._last_name = last_name

    @property
    def first_name(self):
        return self._first_name

    @property
    def last_name(self):
        return self._last_

    @property
    def details(self):
        return f'{self._fi
```

```python
person = Person(first_name='Jan', last_name='Kowalski')
print(person.first_name)
print(person.last_name)
print(person.details)
```

➔ Dodanie tego dekoratora pozwala usunąć nawiasy :)

PyStart #41 Dekorator property

To wszystko?

Co jeszcze można zrobić z tym dekoratorem?

Ano.. pozostałe metody dostępowe

PyStart #41 Dekorator property

Dlaczego? Pamiętacie metody dostępowe?

```python
class Product:
    def __init__(self, price):
        self.price = price
        self._discount = 0.1

    @property
    def discounted_price(self):
        return self.price - self.price * self._discount

    @property
    def discount(self):
        return self._discount

```

➔ getter - tak jak poprzednio mogę pobierać wartości bez nawiasów korzystając z dekoratora property

PyStart #41 Dekorator property

Dlaczego? Pamiętacie metody dostępowe?

```python
class Product:
    def __init__(self, price):
        self.price = price
        self._discount = 0.1

    @property
    def discounted_price(self):
        return self.price - sel

    @property
    def discount(self):
        return self._discount
```

```python
class Product:
    def __init__(self, price):
        self.price = price
        self._discount = 0.1

    @property
    def discount(self):
        return self._discount

    @discount.setter
    def discount(self, new_discount):
        if 0 < new_discount < 1:
            self._discount = new_discount

    @discount.deleter
    def discount(self):
        del self._discount
```

PyStart #41 Dekorator property

Dlaczego? Pamiętacie metody dostępowe?

```
10      @property
11      def discount(self):
12          return self._discount
13
14      @discount.setter
15      def discount(self, new_discount):
16          if 0 < new_discount < 1:
17              self._discount = new_discount
```

```
24  car = Product(1000)
25  print(car.discounted_price)
26  car.discount = 0.2
27  print(car.discounted_price)
```

PyStart #41 Dekorator property

Dlaczego? Pamiętacie metody dostępowe?

```
10      @property
11      def discount(self):
12          return self._discount
13
14      @discount.deleter
15      def discount(self):
16          self._discount = 0
17
```

```
18
19      car = Product(1000)
20      print(car.discounted_price)
21      del car.discount
22      print(car.discounted_price)
```

PyStart #41 Dekorator property

Zadania dla nabrania wprawy

Przygotuj mini program do zarządzania Twoimi oszczędnościami. Starasz się wpłacać pieniądze z różnych źródeł, chcesz na bieżąco wiedzieć ile udało CI się już zaoszczędzić.

41.1

a) Przygotuj klasę **Saving**, która w inicie odbiera datę powstania oszczędności oraz jej wartość. Oba pola są prywatne. Wartość musi być zawsze wartością dodatnią.

b) Przygotuj klasę **Savings**, która będzie posiadała metodę add_saving(saving: Saving), służącą do dodawania nowych obiektów do listy.

c) Za pomocą dekoratora property stwórz metodę total służące do wyświetlania łącznej wartości oszczędności.