

Pystart.pl

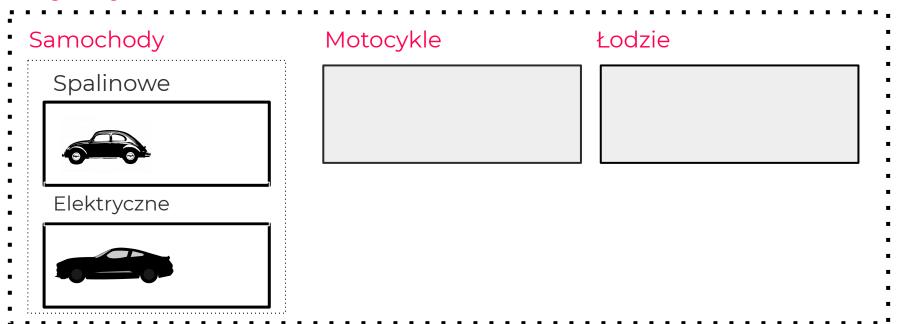
Dziedziczenie

lekcja czterdziesta

O co chodzi?



Pojazdy



O co chodzi?



Pojazdy

właściwości: prędkość maksymalna metody: skręt, przyspiesz, zwolnij



Samochody

właściwości: ilość drzwi, metody: włącz wycieraczki, włącz radio, zmień koło



Spalinowe

właściwości: silnik, metody: włącz silnik,





Jak to wygląda w kodzie?

```
class Vehicle:
class BaseCar(Vehicle):
class Car(BaseCar):
class ElectricCar(BaseCar):
```



→ W nawiasie przekazujemy po jakiej klasie dziedziczymy

→ Dziedziczenie przebiega "kaskadowo"

→ Dziedziczone są zarówno właściwości jak i metody

Przeciążanie metod

```
class Parent:
    def get_type(self):
        return 'parent'
class Child(Parent):
    def get_type(self):
        return 'child'
sample = Child()
print(sample.get_type())
```



→ Przeciążanie metod polega na ich "nadpisywaniu", tzn. w klasie potomnej posiadamy taką samą metodę jak w "rodzicu"

(venv) D:\Trainings\Pystart\week_7>python inheritance.py
child

Przeciążanie metod





A co jeśli potrzebowałbym odebrać wartość z metody "rodzica"?

Super!

Super()

```
class Product:
        def __init__(self, price):
            self.price = price
0
        def get_price(self):
            return self.price
    class DiscountedProduct(Product):
        def get_price(self):
            price = super().get_price()
            return price - 0.1 * price
    product = DiscountedProduct(100)
    print(product.get_price())
```



→ super() wskazuje na klasę z której dziedziczymy, którą przeciążamy.

Kiedy super, a kiedy nie super?

```
class Parent:
           def parent_method(self):
               return 'parent_method'
           def common_method(self):
               return 'common method'
       class Child(Parent):
           def child_method(self):
               return self.parent_method()
13 0
           def common_method(self):
               return super().common_method()
```

- → Jeśli ta sama metoda (common_method) jest dostępna w obu klasach (Parent i Child), to potrzebujemy super()
- → Jeśli metody mają różne nazwy to w Child można wywoływać metody z self. self.parent_method()

Jak zachowa się __init__?

```
class Parent:
    def __init__(self):
        print('Rodzic!')
class Child(Parent):
    pass
sample = Child()
```



- → Init zachowuje się tak samo jak każda inna metoda.
- → Init zadziałał, komunikat się wyświetla.

```
(venv) D:\Trainings\Pystart\week_7>python inheritance.py
Rodzic!
```

Przeciążanie inita

```
class Person:
    def __init__(self, first_name, last_name):
        self.details = f'{first_name} {last_name}'
class Student(Person):
    def __init__(self, first_name, last_name, semester):
        super().__init__(first_name, last_name)
        self.semester = semester
jan = Student('Jan', 'Kowalski', 2)
print(jan.details)
print(jan.semester)
```



- → Przeciążanie inita jest możliwe tak samo jak każdej innej metody.
- → Koniecznie należy wówczas wywołać init klasy po której dziedziczymy.

PyStart #40 Dziedziczenie Zadania dla nabrania wprawy

- Przygotuj klasę BankAccount, która będzie pozwalała wpłacać(deposit)
 i wypłacać(withdraw) środki. Utwórz klasę konta oszczędnościowego
 SavingsAccount, która będzie dziedziczyła po BankAccount i umożliwi
 zwiększenie stanu konta o procent odsetek.
- 2. Przygotuj klasę Employee, która w inicie będzie odbierała imię, nazwisko oraz stawkę godzinową. Przygotuj klasę Manager, która będzie dziedziczyła po klasie Employee, którego każda godzina pracy będzie liczona podwójnie, a dodatkowo będzie możliwość określenia premii managera(add_bonus(amount: int)).

40.1

