



# Pystart.pl

## Dziedziczenie

lekcja czterdziesta

# PyStart #40 Dziedziczenie

O co chodzi?

Pojazdy

Samochody

Spalinowe



Elektryczne



Motocykle



Łodzie



# PyStart #40 Dziedziczenie

## 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,



# PyStart #40 Dziedziczenie

## Jak to wygląda w kodzie?

```
1 class Vehicle:
2     pass
3
4
5 class BaseCar(Vehicle):
6     pass
7
8
9 class Car(BaseCar):
10    pass
11
12
13 class ElectricCar(BaseCar):
14    pass
15
16
```

- 
- W nawiasie przekazujemy po jakiej klasie dziedziczymy
  - Dziedziczenie przebiega “kaskadowo”
  - Dziedziczone są zarówno właściwości jak i metody

# PyStart #40 Dziedziczenie

## Przeciążanie metod

```
1  class Parent:
2      def get_type(self):
3          return 'parent'
4
5
6  class Child(Parent):
7      def get_type(self):
8          return 'child'
9
10
11  sample = Child()
12  print(sample.get_type())
13
```

→ 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
```

# PyStart #40 Dziedziczenie

## Przeciążanie metod



A co jeśli potrzebowalbyśmy odebrać wartość z metody “rodzica” ?

**Super!**

# PyStart #40 Dziedziczenie

## Super()



```
1 class Product:
2     def __init__(self, price):
3         self.price = price
4
5     def get_price(self):
6         return self.price
7
8
9 class DiscountedProduct(Product):
10    def get_price(self):
11        price = super().get_price()
12        return price - 0.1 * price
13
14
15 product = DiscountedProduct(100)
16 print(product.get_price())
```

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

# PyStart #40 Dziedziczenie

## Kiedy super, a kiedy nie super ?



```
1 class Parent:
2     def parent_method(self):
3         return 'parent_method'
4
5     def common_method(self):
6         return 'common method'
7
8
9 class Child(Parent):
10    def child_method(self):
11        return self.parent_method()
12
13    def common_method(self):
14        return super().common_method()
15
```

- 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()**



# PyStart #40 Dziedziczenie

## Jak zachowa się `__init__` ?



```
1  class Parent:
2      def __init__(self):
3          print('Rodzic!')
4
5
6  class Child(Parent):
7      pass
8
9
```

```
10 sample = Child()
11
```

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

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

# PyStart #40 Dziedziczenie

## Przeciążanie inita



```
1 class Person:
2     def __init__(self, first_name, last_name):
3         self.details = f'{first_name} {last_name}'
4
5
6 class Student(Person):
7     def __init__(self, first_name, last_name, semester):
8         super().__init__(first_name, last_name)
9         self.semester = semester
10
11
12 jan = Student('Jan', 'Kowalski', 2)
13 print(jan.details)
14 print(jan.semester)
15
```

- 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

1. 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

