

# PODSTAWY PROGRAMOWANIA W PYTHON

Dzień 10



# AGENDA

## DAY 10

- klasy c.d.
- przeciążanie metod specjalnych i operatorów
- dziedziczenie

# | metody specjalne

# METODY SPECJALNE

```
class Samochod(object):  
    def __init__(self, marka, model):  
        self.marka = marka  
        self.model = model
```

Określenie ich w klasie umożliwia zdefiniowanie własnych zachowań dla operatorów i metod specjalnych

# INNE METODY SPECJALNE

OPERATORY:

+, -, ==, <, >, len(), print, i in.

\_\_add\_\_(self, other) -> self + other

\_\_sub\_\_(self, other) -> self - other

\_\_eq\_\_(self, other) -> self == other

\_\_lt\_\_(self, other) -> self < other

\_\_len\_\_(self) -> len(self)

\_\_str\_\_(self) -> print(self)

<https://docs.python.org/3/reference/datamodel.html#basic-customization>

# PARADYGMATY OOP

**ABSTRAKCJA**

**DZIEDZICZENIE**

**ENKAPSULACJA**

**POLIMORFIZM**

# dziedziczenie

# definiowanie klas

słowo kluczowe

nazwa

klasa nadrzędna / rodzic

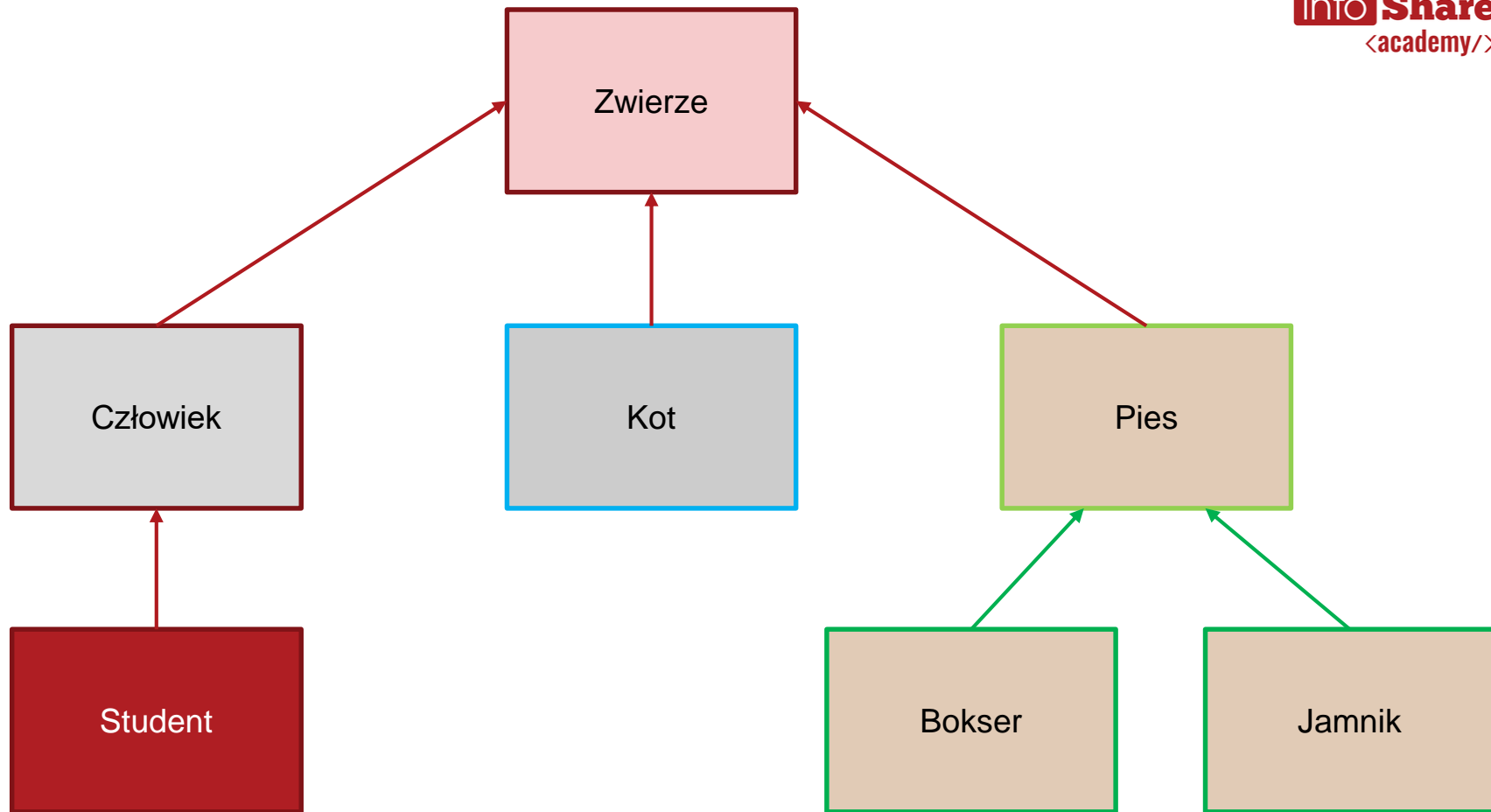
`class Samochod(object):`

`# definicje danych`

`# definicje metod`

- **class** – podobnie jak **def**
- słowo **object** oznacza, że Samochód jest obiektem w Python (object) i **dziedziczy** z niego wszystkie właściwości
  - Samochod jest podklasą object
  - object jest klasą nadrzędną dla Samochod





# definiowanie klas

```
class Zwierze(object):  
    # definicje danych  
    # definicje metod
```

```
class Czlowiek(Zwierze):  
    # definicje danych  
    # definicje metod
```

```
class Student(Czlowiek):  
    # definicje danych  
    # definicje metod
```

Dziedziczenie umożliwia tworzenie klas, które korzystają z atrybutów klas nadrzędnych (superklasa / rodzic).

Klasy dziedziczące (podklasy / dzieci) mogą część atrybutów mieć zdefiniowanych według własnych potrzeb.

# SPRAWDZENIE ZALEŻNOŚCI

**isinstance(obiekt, klasa)** – sprawdza czy dany obiekt jest instancją klasy

**issubclass(klasaA, klasaB)** – sprawdza czy klasaA jest podklasą klasy B



# Thanks!!