PODSTAWY PROGRAMOWANIA W PYTHON

Dzień 11

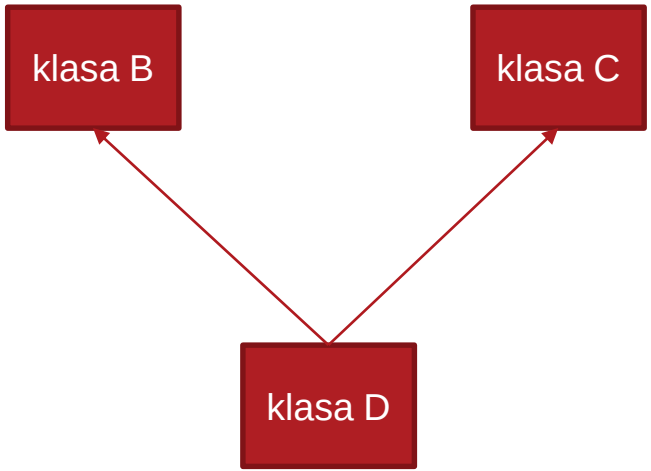
AGENDA
DAY 11

- dziedziczenie diamentowe
- pola klasy
- metody klas
- metody statyczne

dziedziczenie diamentowe

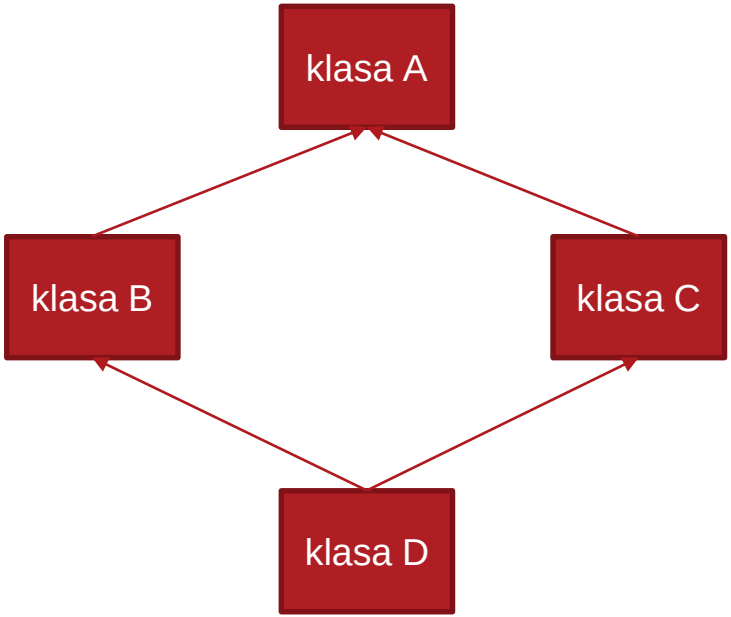
DZIEDZICZENIE OD WIELU RODZICÓW

Klasa może dziedziczyć z wielu klas

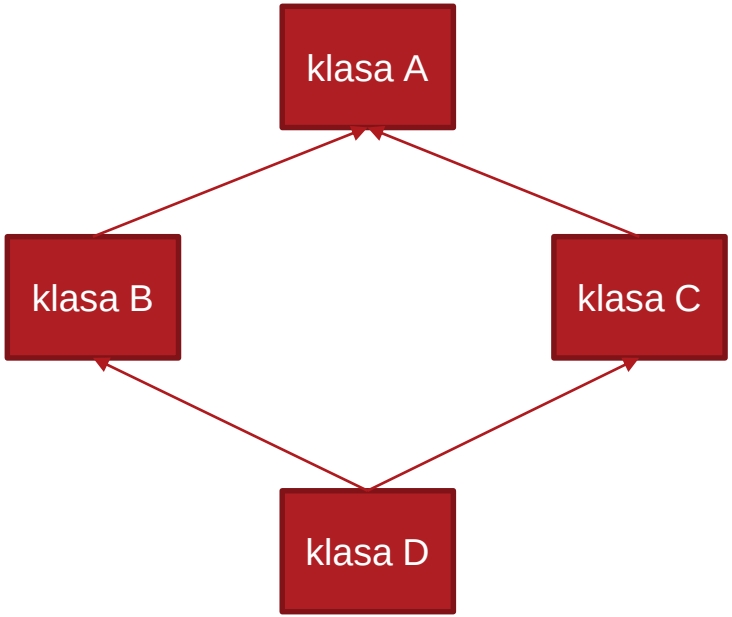


DZIEDZICZENIE DIAMENTOWE

Ale co w przypadku dziedziczenia diamentowego?



DZIEDZICZENIE DIAMENTOWE



Klasa dziecka będzie szukać atrybuty w kolejności od lewej do prawej, z dołu w górę.

W poniższym przykładzie, najpierw poszuka w klasie Horse, a następnie w Donkey

```
class Mule(Horse, Donkey):
    pass
```

super()

nie jest taki super

Musimy uważać jeśli dziedziczymy używając **super()** jako odwołanie do klasy nadrzędnej.

Dlatego lepiej używać jawnie określonych klas.

pola klasy, metody klasy, metody statyczne

POLA KLASY

Zmienne definiowane na poziomie klasy.
Nie używamy słówka **self**

Służą do przechowywania danych niezależnych od instancji

METODY KLASY

Metody, które jako pierwszy argument przyjmują klasę zamiast instancji.

Używamy **dekoratora @classmethod** nad definicją metody.

Pierwszy argument to słowo kluczowe **cls**

Możemy używać jako alternatywne konstruktory

```
@classmethod
def my_class_method(cls):
    pass
```

METODY statyczne

Metody, które nie przyjmują ani instancji ani klasy jako argument.
Wyglądają jak normalne metody

Używamy **dekoratora @staticmethod** nad definicją metody.

Używamy je gdy przekazanie jakiejś informacji nie wymaga tworzenia instancji klasy. (matematyczne)

```
@staticmethod
def my_static_method():
    pass

MyClass.my_static_method()
```

Thanks!!