Přednáška 2

Python: Úvod do objektově orientovaného programovaní

Základní koncepty OOP — pokračování

155YOBP Objektové programování, zimní semestr 2022

Koncepty OOP
- Vícenásobná dědičnost
- Abstraktní třídy
- Polymorfismus

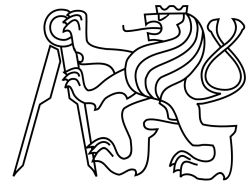
Související témata
- Magické metody
- Statické proměnné, metody
- Vlastnosti

Martin Landa

martin.landa@fsv.cvut.cz

Fakulta stavební ČVUT v Praze
Katedra geomatiky

http://geo.fsv.cvut.cz/gwiki/155YOBP



Koncepty OOP
Vícenásobná dědičnost
Abstraktní třídy
Polymorfismus
Související témata
Magické metody
Statické proměnné, metody
Vlastnosti

Obsah přednášky

1 **Koncepty OOP**
- Vícenásobná dědičnost
- Abstraktní třídy
- Polymorfismus

2 **Související témata**
- Magické metody
- Statické proměnné, metody
- Vlastnosti

Koncepty OOP
Vícenásobná dědičnost
Abstraktní třídy
Polymorfismus
Související témata
Magické metody
Statické proměnné, metody
Vlastnosti

Vícenásobná dědičnost

Priorita

- Záleží na pořadí rodičovských tříd
- Method Resolution Order `__mro__`
- Volá se první nalezený konstruktor dle MRO

```
class Pes:
    def __init__(self, jmeno):
        self.jmeno = jmeno

class Vlk:
    def __init__(self):
        self.pocet_zubu = 42

class Krizenec(Vlk, Pes):
    pass # prázdný příkaz

v = Krizenec()
v.pocet_zubu # 42
v.jmeno      # NELZE
```


Koncepty OOP
Vícenásobná dědičnost
Abstraktní třídy
Polymorfismus
Související témata
Magické metody
Statické proměnné, metody
Vlastnosti

Vícenásobná dědičnost

Priorita

- Záleží na pořadí rodičovských tříd

```
class Pes:
    def __init__(self, jmeno):
        self.jmeno = jmeno
    def mam_jmeno(self):
        return True

class Vlk:
    def __init__(self):
        self.pocet_zubu = 42
    def mam_jmeno(self):
        return False

class Krizenec1(Vlk, Pes):
    pass

v1 = Krizenec1()
v1.mam_jmeno()
```


Koncepty OOP
Vícenásobná dědičnost
Abstraktní třídy
Polymorfismus
Související témata
Magické metody
Statické proměnné, metody
Vlastnosti

Vícenásobná dědičnost

Priorita

- Záleží na pořadí rodičovských tříd

```
class Pes:
    def __init__(self, jmeno):
        self.jmeno = jmeno
    def mam_jmeno(self):
        return True

class Vlk:
    def __init__(self):
        self.pocet_zubu = 42
    def mam_jmeno(self):
        return False

class Krizenec2(Pes, Vlk):
    pass

v2 = Krizenec2('Bella') # konstruktor tridy Pes
v2.mam_jmeno()
```


Koncepty OOP
Vícenásobná dědičnost
Abstraktní třídy
Polymorfismus
Související témata
Magické metody
Statické proměnné, metody
Vlastnosti

Vícenásobná dědičnost

Koncepty OOP
Vícenásobná dědičnost
Abstraktní třídy
Polymorfismus
Související témata
Magické metody
Statické proměnné, metody
Vlastnosti

Priorita

- Volá se první nalezený konstruktor dle MRO

```
class Krizenec3(Pes, Vlk):
    def __init__(self, jmeno):
        # super().__init__(jmeno)
        Pes.__init__(self, jmeno)
        Vlk.__init__(self)

v3 = Krizenec3('Bella')
v3.jmeno
v3.pocet_zubu
```

Abstraktní třídy, metody

Abstract Base Classes

- Modul abc

```
from abc import ABC, abstractmethod

class Pes(ABC):
    def __init__(self, jmeno):
        self.jmeno = jmeno

    @abstractmethod
    def rad_plavu(self):
        pass

class LabradorskyRetrivr(Pes):
    def rad_plavu(self):
        return True

p = Pes('Bára')                    # NELZE
p = LabradorskyRetrivr('Bára')
p.rad_plavu()
```


Koncepty OOP
Vícenásobná dědičnost
Abstraktní třídy
Polymorfismus
Související témata
Magické metody
Statické proměnné, metody
Vlastnosti

Polymorfismus

Koncepty OOP
Vícenásobná dědičnost
Abstraktní třídy
Polymorfismus
Související témata
Magické metody
Statické proměnné, metody
Vlastnosti

- Schopnost zaujímat více forem v různých významech
- Objekt se chová podle toho, jaké je třídy instancí
- Lze volat jednu metodu (stejné jméno) s různými parametry (různá implementace)
- Přetěžování operátorů (rozdílná operace v závislosti na typu operandů)
- Statický polymorfismus (šablony v C++)
- Dynamicky typované jazyky (Python) používají dynamický polymorfismus

Koncepty OOP
Vícenásobná dědičnost
Abstraktní třídy
Polymorfismus
Související témata
Magické metody
Statické proměnné, metody
Vlastnosti

Operátor součtu (+) se chová podle toho jakými instancemi třídy jsou jednotlivé operandy

```
c = 1 + 1
type(c)

s = 'ahoj' + ' svete'
type(s)

cs = 1 + ' svete' # NELZE
```

Funkce `len()` se chová podle toho jaká instance třídy je zadána jako argument

```
len("haf")
len(["haf", "haf"])
len({"jmeno": "Gulliver", "rasa": "krizenec"})
```

Koncepty OOP
Vícenásobná dědičnost
Abstraktní třídy
Polymorfismus
Související témata
Magické metody
Statické proměnné, metody
Vlastnosti

Dědičnost a přepsání metody:

```
class Pes:
    def rad_plavu(self):
        return None

class LabradorskyRetrivr(Pes):
    def rad_plavu(self):
        return True

p = Pes()
p.rad_plavu()

l = LabradorskyRetrivr()
l.rad_plavu()
```

Polymorfismus | Přetížení operátoru

Koncepty OOP
Vícenásobná dědičnost
Abstraktní třídy
Polymorfismus
Související témata
Magické metody
Statické proměnné, metody
Vlastnosti

```
class Pes:
    def __init__(self, rasa):
        self.rasa = rasa
    def __add__(self, b):
        if self.rasa == 'labradorský retrívr' \
          and b.rasa == 'pudl':
            return Pes('Labradoodle')
        return None

same = Pes('labradorský retrívr')
sami = Pes('pudl')
sam = same + sami
sam.rasa
```

Obsah přednášky

1 Koncepty OOP
- Vícenásobná dědičnost
- Abstraktní třídy
- Polymorfismus

2 Související témata
- Magické metody
- Statické proměnné, metody
- Vlastnosti

Koncepty OOP
Vícenásobná dědičnost
Abstraktní třídy
Polymorfismus
Související témata
Magické metody
Statické proměnné, metody
Vlastnosti

Koncepty OOP
Vícenásobná dědičnost
Abstraktní třídy
Polymorfismus
Související témata
Magické metody
Statické proměnné, metody
Vlastnosti

Magické metody

- Začínají a končí dvěma podtržítky __
- Mají specifický význam

Připomenutí:

- `__init__()` (konstruktor)
- `__del__()` (destruktor)
- `__add__()` (operátor součtu)
- ...

https://diveintopython3.net/special-method-names.html

Magické metody | Příklad

Koncepty OOP
Vícenásobná dědičnost
Abstraktní třídy
Polymorfismus
Související témata
Magické metody
Statické proměnné, metody
Vlastnosti

```
class Pes:
    def __str__(self):
        return "pes"

    def __getattr__(self, key):
        if key == 'rasa':
            return 'neznámá'
        return None

    def __eq__(self, b):
        return str(self) == str(b)

p1 = Pes()
print(p1)
p1.rasa
p1.jmeno

p2 = Pes()
print(p1 == p2)
```

Statické proměnné

```
class Pes:
    rasa = 'neznámá'

    def zvol_rasu(self, rasa):
        self.rasa = rasa

class Pes1:
    def __init__(self, rasa):
        self.rasa = rasa

Pes.rasa
Pes1.rasa           # NELZE

p = Pes()
p.rasa
p.zvol_rasu('jezevčík')
p.rasa

p1 = Pes1('pudl')
p1.rasa
```


Koncepty OOP
Vícenásobná dědičnost
Abstraktní třídy
Polymorfismus
Související témata
Magické metody
Statické proměnné, metody
Vlastnosti

Statické metody

Nemají přístup k atributům ani metodám objektu

```
class Pes:
    def __init__(self, jmeno):
        self.jmeno = jmeno

    def prvni(self):
        return self.jmeno[0]

    @staticmethod
    def haf():
        return 'haf'

p = Pes('Gulliver')
p.prvni()
p.haf()
```

Koncepty OOP
Vícenásobná dědičnost
Abstraktní třídy
Polymorfismus
Související témata
Magické metody
Statické proměnné, metody
Vlastnosti

Properties

Metody getter/setter

- Getter – získání hodnoty atributu objektu
- Setter – nastavení hodnoty atributu objektu

```
class Pes:
    def __init__(self, jmeno, hmotnost_kg):
        self.jmeno = jmeno
        self.hmotnost_kg = hmotnost_kg
    def ziskej_hmotnost_lb(self):
        # getter
        return self.hmotnost_kg * 2.205
    def nastav_hmotnost_lb(self, hmotnost):
        # setter
        self.hmotnost_kg = hmotnost * 0.454

p = Pes('Gulliver', 6.0)
p.ziskej_hmotnost_lb()
p.nastav_hmotnost_lb(13.4)
```


Koncepty OOP
Vícenásobná dědičnost
Abstraktní třídy
Polymorfismus
Související témata
Magické metody
Statické proměnné, metody
Vlastnosti

Properties

Metody getter/setter

- Getter – dekorátor `@property`
- Setter – dekorátor `@<getter>.setter`

```
class Pes:
    def __init__(self, jmeno, hmotnost_kg):
        self.jmeno = jmeno
        self.hmotnost_kg = hmotnost_kg
    @property
    def hmotnost_lb(self):
        return self.hmotnost_kg * 2.205
    @hmotnost_lb.setter
    def hmotnost_lb(self, hmotnost):
        self.hmotnost_kg = hmotnost * 0.454

p = Pes('Gulliver', 6.0)
p.hmotnost_lb
p.hmotnost_lb = 13.4
```


Koncepty OOP
Vícenásobná dědičnost
Abstraktní třídy
Polymorfismus
Související témata
Magické metody
Statické proměnné, metody
Vlastnosti