Programování 1: Třídy a objekty

Martin Mareš
mj@ucw.cz

Katedra Aplikované Matematiky
MFF UK Praha

2019

Definice třídy

```
class Zvire:

    def __init__(self, jmeno, zvuk):
        self.jmeno = jmeno
        self.zvuk = zvuk

    def slysi_na(self, jmeno):
        return self.jmeno == jmeno

    def ozvi_se(self):
        print(self.jmeno, "říká:", self.zvuk)
```

Definujeme nový typ, který má nějaké **atributy** (vlastnosti) a **metody** (funkce, operace).

Objekty

Vytvoříme nový objekt (automaticky zavolá `__init__`):

```
>>> azor = Zvire("Azor", "Haf!")
>>> azor
<Zvire object at 0x7ffff71ce2b0>
```

Atributy objektu:

```
>>> azor.zvuk
'Haf!'

>>> azor.zvuk = "Hafff!"
```

Metody objektu:

```
>>> azor.slysi_na('Příšero')
False

>>> azor.ozvi_se()
Azor říká: Hafff!
```

Identita objektu

```
>>> jezevcik = Zvire("Špagetka", "haf")
>>> bernardyn = Zvire("Bernard", "HAF!!!")
>>> maxipes = bernardyn
>>> maxipes.jmeno = "Fík"
>>> bernardyn.jmeno
'Fík'

>>> type(jezevcik)
<class 'Zvire'>

>>> id(jezevcik), id(bernardyn), id(maxipes)
(737339253592, 737339253704, 737339253704)

>>> bernardyn is maxipes
True

>>> bernardyn is jezevcik
False
```

Protokol pro převod na řetězec

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

    def __str__(self):
        return self.jmeno

    def __repr__(self):
        return f"Zvire({self.jmeno}, {self.zvuk})"

>>> hroch = Zvire("Hippo", "Humpf!")
>>> hroch
Zvire(Hippo, Humpf!)
>>> print(hroch)
Hippo
```

Protokol pro operátory

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

    def __eq__(self, other):
        return self.jmeno == other.jmeno and \
               self.zvuk == other.zvuk
```

```
>>> hroch1 = Zvire("Hippo", "Humpf!")
>>> hroch2 = Zvire("Hippo", "Humpf!")
>>> hroch1 is hroch2
False
>>> hroch1 == hroch2
True
```

Podobně jde předefinovat všechny operátory.

Další protokoly

Další protokoly, které může třída implementovat:

- Konverze na bool, str, int, float
- Indexování: čtení/zápis/mazání *obj*[...], **len(***obj***)**
- Přístup k atributům: čtení/zápis/mazání *obj*.**klíč**
- Volání jako funkce
- Iterátor pro **for** *x* **in** *objekt*

Dokumentační komentáře

```
class Zvire:
    """Vytvoří zvíře s danými vlastnostmi."""

    def __init__(self, jmeno, zvuk):
        self.jmeno = jmeno
        self.zvuk = zvuk

    def slysi_na(self, jmeno):
        """Slyší zvíře na dané jméno?"""
        return self.jmeno == jmeno
```

```
>>> help(Zvire)
```

```
>>> z = Zvire("Lenochod", "Zzz...")
>>> help(z.slysi_na)
```

Dědičnost

```
class Kocka(Zvire):

    def __init__(self, jmeno, zvuk):
        Zvire.__init__(self, jmeno, zvuk)
        self._pocet_zivotu = 9    # interní

    def slysi_na(self, jmeno):
        # Copak kočka slyší na jméno?
        return False
```

```
>>> k = Kocka("Příšerka", "Mňauuu")
>>> k.slysi_na("Příserka")
```
(speciální kočičí verze)

```
False
```

```
>>> k.ozvi_se()
```
(původní zvířecí metoda)

```
Příšerka říká: Mňauuu
```

Dotazy na typy

```
>>> type(k) is Kocka
True

>>> type(k) is Zvire
False

>>> isinstance(k, Kocka)
True

>>> isinstance(k, Zvire)
True

>>> issubclass(Kocka, Zvire)
True
```

Jak to funguje uvnitř

Prostory jmen (namespaces):

- Zabudované funkce (třeba `print`)
- Globální jména (proměnné, funkce)
- Lokální jména uvnitř funkce
- Jména definovaná v třídě
- Jména definovaná v objektu

Obyčejné jméno se hledá ve všech prostorech, které jsou na daném místě v programu „vidět“.

`objekt.jméno` se hledá:

- V prostoru atributů objektu
- V prostoru příslušné třídy
- V prostorech všech nadřazených tříd
- Pozor, třída může mít více přímých předků!

Jak to funguje uvnitř: zabudované typy

Zabudované typy jako int, str apod. jsou rovněž třídy. Volání `int()` nebo `int("1")` je prostě vytvoření objektu dané třídy.

Třída je také objekt:

- Lze psát třeba `Zvire.slysi_na`.
- Uvnitř třídy můžeme přiřazovat do proměnných, ty jsou vidět jako atributy třídy.
- Vnitřek definice `class` je ve skutečnosti normální program, který se provádí uvnitř prostoru jmen třídy.

Modul je také objekt:

- `import math` ho vytvoří
- `math.random` je formálně přístup k atributu

Volání metody:

- `alik.ozvi_se` vytvoří pomocnou funkci, která zavolá `Zvire.ozvi_se` a doplní jako první argument `alik`.