Programování 1.10: Třídy a objekty

Petr Čermák
cermak@mag.mff.cuni.cz

Katedra Fyziky Kondenzovaných Látek
MFF UK Praha

2020
forked from https://gitlab.kam.mff.cuni.cz/mj/prm1

Definice třídy

```
1  class Zvire:
2
3      def __init__(self, jmeno, zvuk):
4          self.jmeno = jmeno
5          self.zvuk = zvuk
6
7      def slysi_na(self, jmeno):
8          return self.jmeno == jmeno
9
10     def ozvi_se(self):
11         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

```
1  class Zvire:
2      def __init__(self, jmeno, zvuk):
3          self.jmeno = jmeno
4          self.zvuk = zvuk
5
6      def __str__(self):
7          return self.jmeno
8
9      def __repr__(self):
10         return f"Zvire({self.jmeno}, {self.zvuk})"
```

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

Protokol pro operátory

```
1  class Zvire:
2      def __init__(self, jmeno, zvuk):
3          self.jmeno = jmeno
4          self.zvuk = zvuk
5      def __eq__(self, other):
6          return self.jmeno == other.jmeno and \
7                 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

`__bool__`, `__str__`, `__int__`, `__float__`

- Indexování: čtení/zápis/mazání,obj[...], len(obj)

`__contains__`, `__setitem__`, `__getitem_`

- Přístup k atributům: čtení/zápis/mazání obj.klíč

`__getattr__`, `__setattr__`

- Volání jako funkce

`__call__`

- Iterátor pro for x in objekt

`__iter__`, `__next__`

Dokumentační komentáře

```
1  class Zvire:
2      """Vytvoří zvíře s danými vlastnostmi."""
3
4      def __init__(self, jmeno, zvuk):
5          self.jmeno = jmeno
6          self.zvuk = zvuk
7
8      def slysi_na(self, jmeno):
9          """Slyší zvíře na dané jméno?"""
10         return self.jmeno == jmeno
```

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

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

Dědičnost

```
1  class Kocka(Zvire):
2
3      def __init__(self, jmeno, zvuk):
4          Zvire.__init__(self, jmeno, zvuk)
5          self._pocet_zivotu = 9    # interní
6
7      def slysi_na(self, jmeno):
8          # Copak kočka slyší na jméno?
9          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`.

Úkoly na hodinu

- Naučte každé zvíře slyšet navíc na jméno potvůrka.
- Doplňte atribut pozice a zařiďte, aby zvíře slyšelo na jméno jenom tehdy, když pozice je doma.
- Vytvořte odvozenou třídu Pes, jehož metoda ozvi_se na každé druhé zavolání zaštěká a jinak zavrčí.