Veřejné a privátní atributy a metody

jsou v Pythonu dělány jmennou konvencí

- V Pythonu je vše veřejné.
- Chceme-li udělat položku privátní, zahájíme její název dvěma podtržítky.
- V ostatních jazycích je systematická opora pro ochranu objektu,
- uvidíme v létě v C#.

Konstruktor

je funkce jmenující se `__init__`

- Objekt obvykle chceme netriviálně zinicializovat (při vzniku).
- K tomu slouží konstruktor, tedy funkce jménem `__init__`.
- Tato funkce je zavolána implicitně při tvorbě objektu:
- `jan_novak=student()`
- Platí stejná pravidla jako pro metody.

Příklad

konstruktoru

```
def __init__(this,jmeno="Jan Novak",
hodnoceni=4,zapocet='N'):
    this.jmeno=jmeno
    this.hodnoceni=hodnoceni
    this.zapocet=zapocet
    print('Zde konstruktor!')
```

Dědičnost

aneb třídy jako šablony

- Za jméno třídy do závorek napíšeme jméno rodiče.
- Pak zdědíme jeho atributy a metody:
- `class matfyzak(student):....`
- Další významný prostředek dekompozice programu (a eliminace zbytečného opakování kódu).
- Chceme-li přistoupit k rodiči: `super`,
- `super(matfyzak,this).__init__(...)`
- Při inicializaci potomka je třeba inicializovat i rodiče!

Příklad

dědičnosti

```
class matfyzak(student):
    __init__(this,jmeno='AB'
,hodnoceni=1,zapocet='Y',iq=300):
        print('Konstruktor matfyzacky')
        super(matfyzak,this).__init__(jmeno,hodnoceni,zapoce
        this.iq=iq
...
ab=matfyzak()
```

Statické a třídní metody

- Některé metody má smysl definovat pro třídu (ne každý objekt zvlášť),
- těm říkáme *statické*.
- V Pythonu se jim říká `classmethod`:
- `@classmethod def f(...):`
- volání: `jmenotridy.f(...);`
- `@staticmethod def f(...):`
- Statická metoda neví o atributech třídy.

Statické atributy

anebo všechno je jinak

```
class a:
    x=1 #statický atribut!
    def f(this):
        this.x=10 #zalozi nestaticky atribut x
    @classmethod
    def sf(this):
        this.x=100 #modifikuje staticky atribut
```

Neexistuje-li nestatický atribut, objekty přistupují k atributu statickému!

Násobná dědičnost

je k vidění poměrně zřídka

- Můžeme chtít dědit od více předků:
- `class a:..., class b:...,`
- rodiče oddělíme čárkou:
- `class c(a,b):...`
- Tím zdědíme od všech jmenovaných předků.

Diamantový problém

já bych ho česky označil jinak... a byl by malér

- `class a:...,`
- `class b(a):..., class c(a):...,`
- `class d(b,c):...` – a třídu a zdědíme dvakrát.
- Python má důmyslný algoritmus co s tím (v každé verzi jiný).

Soubory

se ovládají ve většině jazyků podobně

- Funkce `open`, metody `read`, `readline`, `write`, `close`.
- Soubor otevřeme, přečteme, přečteme řádek, zapíšeme, zavřeme.
- `open` bere dva parametry (stringy): název, režim.
- Režimy: `r`, `w`, `a`, `x`, symbol `+`, přepínače binarity `t`, `b`.
- Načtení jednoho znaku: `f.read(1)`.

Příklad

práce se souborem

```
f=open('soubor.txt', 'r')
f.read()
f.close()
```