Programovanie v jazyku Python

Úvod do objektovo orientovaného programovania
prednáška 6

Katedra kybernetiky a umelej inteligencie
Technická univerzita v Košiciach
Ing. Ján Magyar, PhD.

Paradigmy programovania

- sekvenčné programovanie
- procedurálne programovanie
- modulárne programovanie

- objektovo orientované programovanie

Objektovo orientované programovanie

- myšlienka zo 70-tych rokov
- náhly rozvoj až po príchode jazyka Java
- program je vnímaný nie ako postupnosť volaní, ale ako spolupráca nezávislých blokov

Prečo objektovo orientované programovanie?

- podpora modularity
- znovupoužitie kódu (code reuse)
- rozšírenie jazyka o vlastné údajové štruktúry a typy

Údajová štruktúra

- všeobecný popis
- implementovaná cez údajové typy
- zvyčajne sa skladá zo skupiny elementárnych hodnôt
- cieľom je zvýšiť efektivitu práce s údajmi
- štandardizované šablóny pre reprezentáciu údajov v počítačoch
- rôzne úrovne abstrakcie
- vieme ich implementovať rôznymi spôsobmi, ale spôsob fungovania sa nemení

Základné údajové štruktúry

- polia
- zoznamy
- zásobníky
- fronty
- hašovacie tabuľky
- stromy

Zásobník (Stack)

- dynamická množina prvkov typu LIFO - last-in, first-out
- základné operácie:
  - inicializácia - CREATE
    - vytvorí prázdny zásobník, môže byť súčasťou PUSH
  - pridanie - PUSH
    - pridá prvok na vrchol zásobníku
  - zmazanie - POP
    - zmaže prvok z vrcholu zásobníku
  - prístup ku vrcholnému prvku - TOP
  - prázdnosť zásobníku - IS_EMPTY

Použitie zásobníkov

- backtracking pri algoritmoch kde hľadáme riešenie princípom pokus-omyl
  - vyhľadávanie do hĺbky
- zásobník volaní

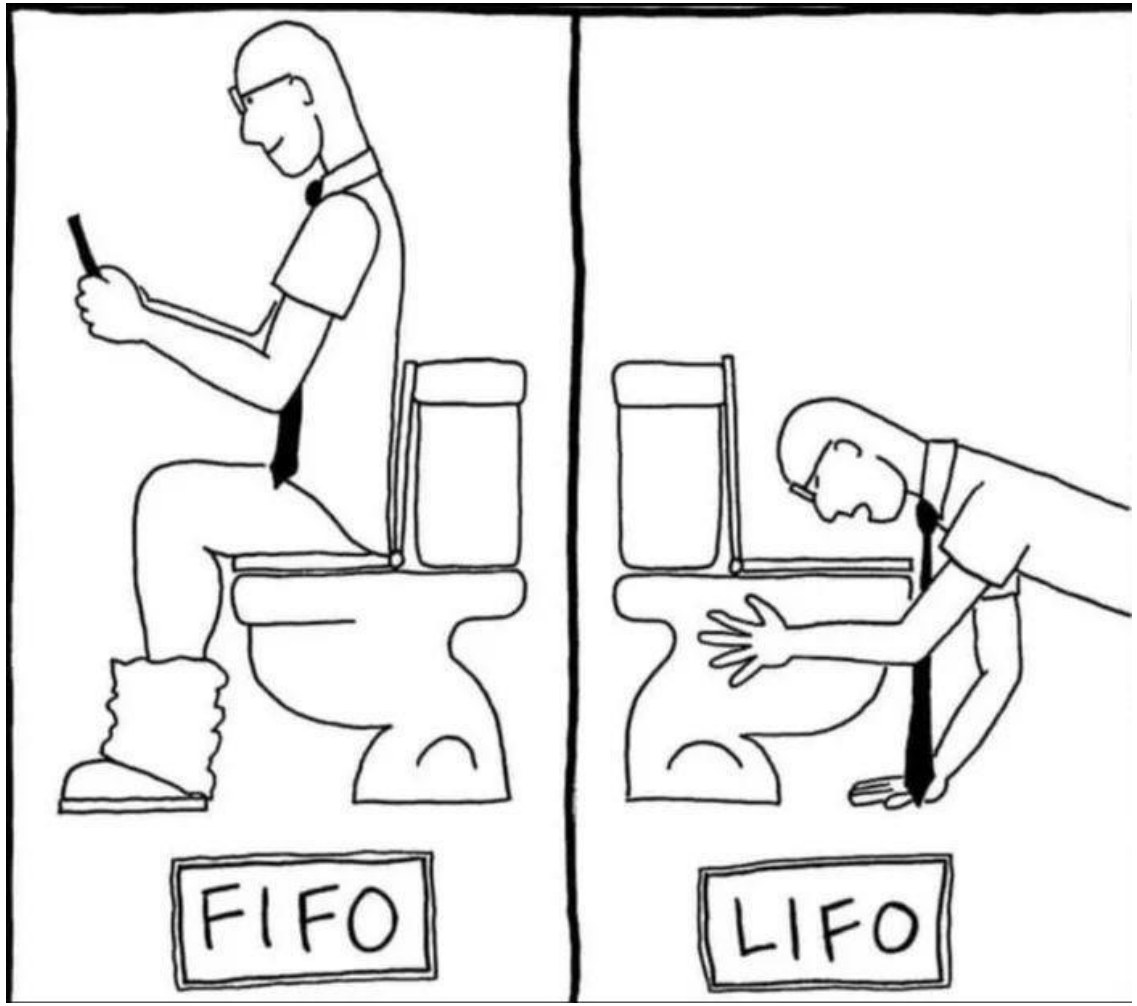
Front (Queue)

- dynamická množina prvkov typu FIFO - first-in, first-out
- základné operácie:
  - inicializácia - CREATE
    - vytvorí prázdny front, môže byť súčasťou ENQUEUE
  - pridanie - ENQUEUE
    - pridá prvok na koniec frontu
  - zmazanie - DEQUEUE
    - vymaže prvok zo začiatku
  - prístup k prvému prvku - HEAD
  - prístup k ostatným prvkom - TAIL
  - prázdnosť frontu - IS_EMPTY

Použitie frontov

- spracovanie požiadaviek
- komunikácia medzi dvoma procesmi
- posielanie správ
- dočasné ukladanie údajov pre neskoršie spracovanie

LIFO/FIFO



Štruktúra objektového riešenia

- kód rozdelíme do modulov
- modul je celok súvisiacich funkcií a hodnôt
- najčastejší prípad použitia modulov sú knižnice
- riešenie je modulárne, ak časti kódu sú rozdelené do rôznych súborov

Moduly v Pythone

- ak chceme pracovať s modulmi, musíme ich importovať

```
import modul_name
```

- možnosť prideliť vlastný názov modulom

```
import modul_name as name
```

- možnosť cieleného importu

```
from modul_name import this, that [as name]
```

Riešenie menných konfliktov

- menný konflikt (name conflict) nastane ak v kontexte vykonávaného kódu existujú dva atribúty (zvyčajne funkcie) s rovnakým názvom
- riešenie - dot notation

```
import numpy as np
import random

np.random.randint(5, 10)
random.randint(5, 10)
```

Základné konštrukty OOP

- trieda a objekt
- trieda je šablóna pre vytvorenie objektov
- objekt je inštanciou triedy – konkrétny príklad

Základné konštrukty OOP - trieda

- **trieda** je špeciálny typ modulov
- slúži na abstrakciu údajov, teda je to **abstraktný dátový typ**
- vzťah k objektom
  - je to šablóna pre vytvorenie objektu
  - trieda je kolekcia objektov s rovnakými vlastnosťami
- z pohľadu interpretera je trieda definíciou vlastného dátového typu
- v Pythone je každá trieda zároveň aj objektom

Základné konštrukty OOP - objekt

- **objekt** je inštanciou triedy - konkrétny príklad/údaj/premenná
- objekt sa skladá z **údajov** a z **funkcií (metód)** - v Pythone sa obe považujú za atribúty
- údaje sú uložené vo vnútorných premenných - existujú iba v rámci objektu

Práca s objektmi v Pythone

```
x = 5
y = 6
x += 1
y -= 2
```

```
my_lst = list()
my_lst.append(6)
my_lst.append(4)
```

Definícia tried v Pythone

```
class Product:
    def __init__(self):
        self.price = 1000

    def set_price(self, new):
        self.price = new

    def sell(self):
        print("Was sold for {}".format(self.price))
```

Vytvorenie objektu v Pythone

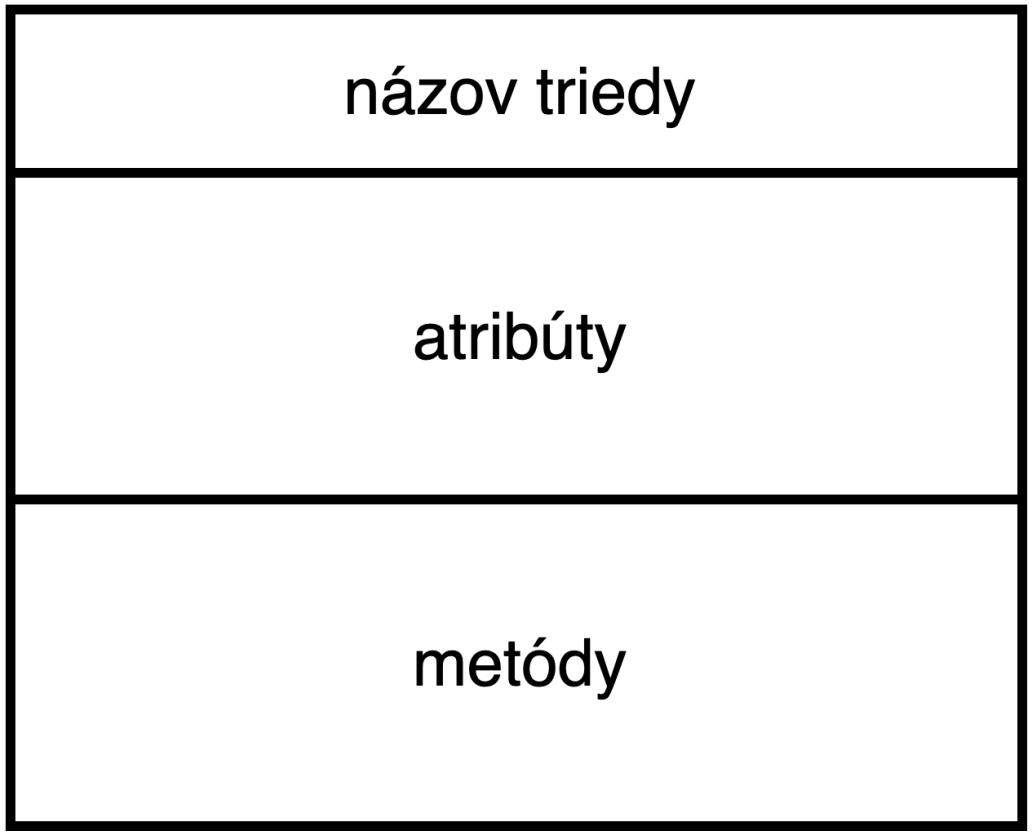
```
t = Product()
t.sell()


l = list()
l.append(5)
```

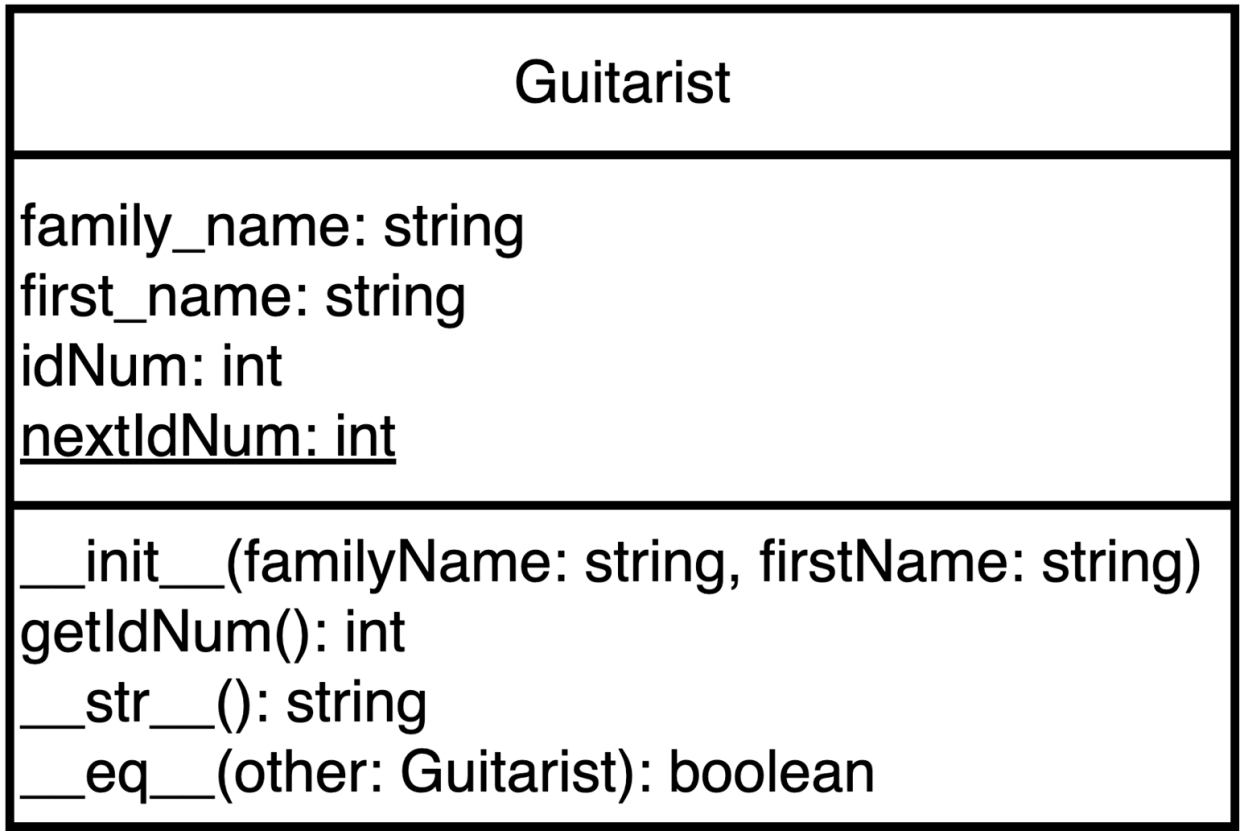
Logický tok programu v OOP

- pri sekvenčnom programovaní môžeme vnímať program ako postupnosť operácií
- v prípade OOP sa program skladá z posielania správ medzi objektmi

```
lst = list()
lst.append(5)
```

Diagram triedy





Enkapsulácia

- v Pythone nemáme príznaky, prístup definujeme priamo v názve premennej
  - `idNum: int` - public premenná
  - `__idNum: int` - private premenná

- premenné triedy sú podčiarknuté
  - `nextIdNum: int`

Zhrnutie

- paradigmy programovania
- objektovo orientované programovanie
- údajové štruktúry
- zásobník a front
- moduly
- trieda a objekt