IB113 Úvod do programování a algoritmizace
Přednáška 12
Lehký náhled na zajímavé knihovny v Pythonu
Shrnutí předmětu

Nikola Beneš

4. prosince 2017

Kvízová otázka na úvod

Kdo byl první programátor?

Ada, hraběnka z Lovelace.

Tuto neděli uplyne 202 let
od jejího narození.

Napsala první program pro počítač,
který nikdy nebyl sestrojen
(analytický stroj Charlese Babbage).

Co bude dnes?

Lehký náhled na zajímavé knihovny v Pythonu

- zpracování bitmapových obrázků
- testování
- vizualizace dat

Přehled programovacích jazyků

- rychlý pohled na historii
- klasifikace programovacích jazyků

Shrnutí předmětu

- co jste se (snad) naučili
- co dál, pokud vás programování zaujalo

Zpracování bitmapových obrázků

- knihovna Pillow: `https://pillow.readthedocs.io/`
  - práce s bitmapovými obrázky (různých formátů)
  - bohatá funkcionalita

```
from PIL import Image
```

- objekt Image reprezentuje obrázek
- vytvoření obrázku:
  - `Image.new` – nový prázdný obrázek
  - `Image.open` – ze zadaného souboru
- základní operace
  - `getpixel` – zjištění barvy pixelu
  - `putpixel` – nastavení barvy pixelu
  - `size`, `width`, `height` – velikost obrázku
  - `show`, `save` – zobrazení, uložení
  - `convert` – změna formátu

Vytváření bitmapových obrázků

Příklad: Geometrické útvary

- čtverec
- kruh
- spirála

Transformace bitmapových obrázků

Příklad: Transformace struktury

- převrácení
- zrcadlení
- rozostření

Příklad: Transformace barev

- invertování barev
- odstranění barev
- změna odstínu (nápověda: HSV)

Testování

- spousta různých knihoven, ukážeme si jednu základní
  - `https://docs.python.org/3/library/unittest.html`

```
import unittest
```

- vlastní testovací sada

```
class MyTestCase(unittest):
    def test_something(self):
        # ...
        self.assertEqual(a, b)
        self.assertTrue(a)
        self.assertFalse(b)
        # ...
```

- spuštění testů

```
unittest.main()
```

[příklad]

Testování

- náhodné generování testů: knihovna *hypothesis*
  - `https://hypothesis.readthedocs.io/`

```
from hypothesis import given
from hypothesis.strategies import integers  # příp. jiné...

@given(integers())
def simple_test(n):
    assert n + n == 2 * n
```

- je třeba si zvolit generátor, který bude vytvářet parametry
- v případě, že se najde chyba, snaží se hypothesis najít co nejmenší protipříklad

[příklad]

Vizualizace dat

- práce s většími daty
  - knihovny *numpy*, *scipy*, *pandas* (a jiné)
- vizualizace dat
  - knihovna *matplotlib*

```
import matplotlib.pyplot as plt
```

- příklad: vykreslení grafu četnosti jmen

Kvíz

https://kahoot.it

Přehled programovacích jazyků

Historie

- první počítač (nikdy nesestrojen): 1837 Charles Babbage
  - děrné štítky
- teoretické základy programování: 30.–40. léta
  - Alonzo Church, Alan Turing, ...
- první počítače: 40. léta
  - strojový kód, první programovací jazyky
- počátky vývoje programovacích jazyků: 50.–60. léta
  - FORTRAN, ALGOL, LISP, COBOL
- další vývoj: 70.–80. léta
  - strukturované, objektově-orientované programování
  - vznik jiných druhů programovacích jazyků (logické, funkcionální)
- moderní programovací jazyky
  - stále vznikají nové jazyky
  - staré jazyky se vyvíjejí a mění

Dělení programovacích jazyků (zhruba)

Podle míry abstrakce

- jazyky vyšší/nižší úrovně

Podle způsobu překladu a spuštění

- kompilované/interpretované

Podle paradigmatu

- imperativní/deklarativní

Další druhy

- skriptovací
- objektově-orientované
- paralelní

Většina jazyků kombinuje více přístupů.

Paradigmata programovacích jazyků

Imperativní jazyky

- program je posloupnost instrukcí
- instrukce jsou „rozkazy“, které určují, co má počítač dělat

Deklarativní jazyky

- program je popis toho, *co* se má udělat
- logické programování (Prolog)
  - program je popsán pomocí logických formulí
- funkcionální programování (Haskell)
  - program je popsán pomocí funkcí
  - funkce nemají vedlejší efekty
  - rekurse

Moderní jazyky kombinují více přístupů:

- funkcionální prvky v C++, Javě, Pythonu, ...

Další druhy programovacích jazyků

Skriptovací jazyky

- užitečné pro jednoduché programy, tzv. *skripty*
- usnadnění zdlouhavé manuální činnosti
- Bash, Perl, Python, ...

Objektově-orientované jazyky

- využívají objektového návrhu programu
- většina moderních jazyků obsahuje nějaké prvky OOP
- C++, Java, Python, Objective C, C#, ...

Paralelní výpočty

- počítání na více procesorech / více počítačích zároveň
- získává na popularitě
- podpora v moderních jazycích (např. C++11)
- proč? procesory už moc rychlejší nebudou

Jaký jazyk používat?

Programovacích jazyků je mnoho. Který si mám vybrat?

- špatně položená otázka
- není žádný jeden ideální jazyk použitelný pro všechny druhy úkolů
- záleží na tom, co řešíme
- může být výhodné kombinovat více jazyků/přístupů

Doporučení

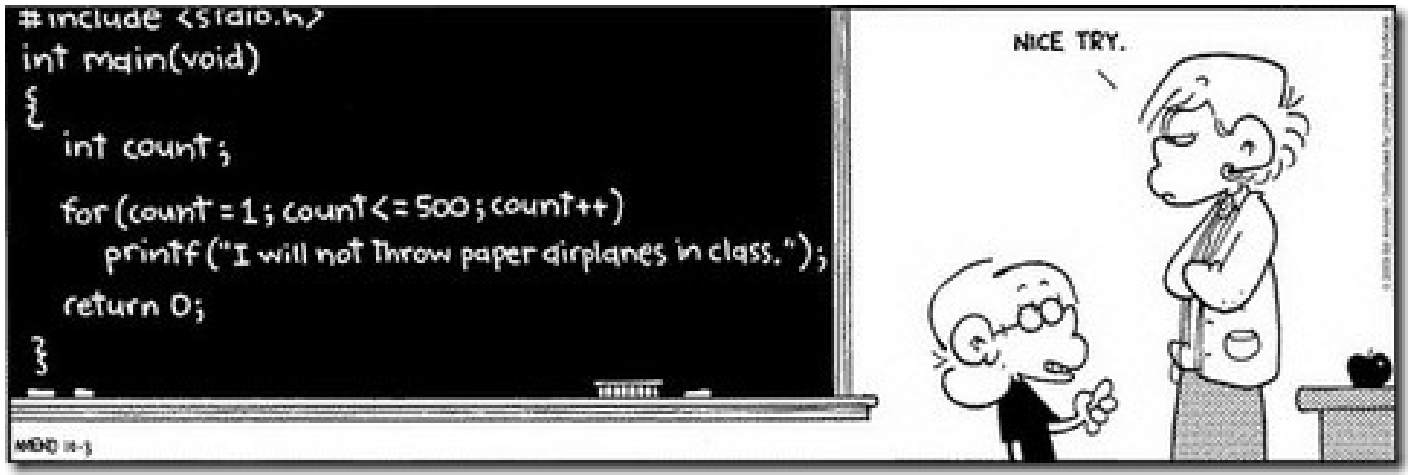
- nefixujte se na jeden jazyk
- různé jazyky s sebou nesou i různé způsoby přemýšlení

„Kolik programovacích jazyků umíš, tolikrát jsi programátorem.“

Shrnutí předmětu

Co jste se (snad) naučili?

- základy programování
  - jednoduché výpočty, zpracování dat
  - jednoduché (náhodnostní) simulace
  - alespoň trochu rozumět kultuře programování
  - dostatečné k tomu, abyste se mohli naučit jiný jazyk
- můžeme donutit počítač, aby dělal to, co chceme
- programování nám může **usnadnit práci**
- programování nám umožňuje dělat zajímavé a zábavné věci



Co dál?

Pokud vás programování zaujalo...

- **IB002** Algoritmy a datové struktury
  - obecně o algoritmech (směrem k větší abstrakci)
- **PB071** Principy nízkoúrovňového programování
  - jak funguje počítač (směrem k nižší abstrakci)
  - programování v C, správa paměti, práce s řetězci
- **PB161**/**PB162** Programování v C++/Javě
  - objektově-orientované programování
- **IB015** Neimperativní programování
  - jiná paradigmata (funkcionální, logické)
- **PV248** Kurz jazyka Python
  - Python podrobněji

A to je vše...

Pěkné svátky, úspěšné zkouškové, ...

... a nezapomeňte programovat.