IB113 Úvod do programování a algoritmizace
Přednáška 1

Organizace předmětu
Základy Pythonu

Nikola Beneš

18. září 2017

Úvodní dotazník

`https://kahoot.it`

Cíle předmětu

Cíle

- zvládnutí základních programátorských konstrukcí (proměnné, funkce, větvení, cykly)
- obecné principy použitelné v mnoha programovacích jazycích
- seznámení se s programátorským/algoritmickým stylem myšlení

Po absolvování byste měli

- umět psát programy pro jednoduché výpočty
- ovládat základy programového zpracovávání dat
- být schopni vytvářet jednoduché (náhodnostní) simulace
- alespoň trochu rozumět kultuře programování
- mít dostatečný základ k tomu, naučit se jiný programovací jazyk

Programy by měly být nejen korektní, ale i „pěkné“.

Programovací jazyk Python

Python

- populární programovací jazyk
- ceněný pro jednoduchost učení
- důraz kladem na obecné koncepty
- záměrně neprobíráme specifika Pythonu

Používáme Python verze 3, neplést s Pythonem verze 2!

Organizace předmětu

Přednášky

- nepovinné, ale doporučené (měly by se nahrávat)
- slajdy a ukázky budou k dispozici ve studijních materiálech

Cvičení

- povinná (max. dvě neomluvené neúčasti), pět skupin
- součástí cvičení jsou domácí úkoly a zápočtový test

Studijní materiály

- interaktivní osnova v ISu

Komunikace

- primárně s cvičícími
- raději osobně než elektronicky
- diskusní fórum v ISu spíše než e-mail

Problémy a nestandardní situace je možné řešit, ale je třeba se vždy včas ozvat.

Hodnocení předmětu

Domácí úkoly (150 bodů)

- šest za semestr
- bodování: 20, 20, 20, 30, 30, 30
- nutné minimum: 100 bodů + alespoň 1 bod z každé

Zápočtový test (100 bodů)

- programování na cvičení (poslední týden semestru)
- nutné minimum: 50 bodů
- opravný termín začátkem týdne po semestru

Zkouška (150 bodů)

- testové otázky, programování na papír
- zkouší se pochopení principů a základních pojmů
- nutné minimum 75 bodů

Domácí úkoly

- v kompetenci cvičících, jedna z úloh bude „centrální“
- pokud nezvládnete úlohu úplně, zkuste alespoň část (za méně bodů)
  - okomentuje „známé nedostatky“
  - jasně vyznačte, jak jste si zjednodušili zadání

- **pracujte samostatně**, neopisujte
  - záporné body, disciplinární komise
  - neřešíme, kdo opisoval – **nesdílejte svá řešení**

- pokud možno zkuste na řešení přijít sami, než začnete hledat na webu
  - připravíte se o „Aha moment“ a ve výsledku se nic nenaučíte
  - jasně vyznačte, kterou část z webu jste převzali

Sbírka příkladů

`http://www.fi.muni.cz/IB111/sbirka/`

- sdílíme ji s předmětem IB111
- interaktivní webová stránka
- obsahuje i části mimo záběr tohoto předmětu
  - cvičící upřesní, které části jsou pro nás relevantní
- procvičení nad rámec cvičení

Doplňkové zdroje

- doporučená literatura v sylabu předmětu v ISu
- Učíme se programovat v jazyce Python `http://howto.py.cz/`
- `http://interactivepython.org/` – interaktivní učebnice
- dokumentace k Pythonu `https://docs.python.org/`
- `https://www.hackerrank.com/` – příklady s automatickým vyhodnocování v prohlížeči
- `http://pythontutor.com/` – vizualizace toho, co se děje za běhu programu
- `https://tutor.fi.muni.cz/` – programovací, logické, matematické i jiné úlohy

- a mnohé další...
  - máte nějaký užitečný? sdílejte v diskusním fóru předmětu na ISu

Předpoklady

- základní počítačová gramotnost
- středoškolská matematika (faktoriál, prvočíslo, logaritmus, dělení se zbytkem)
- logické spojky (konjunkce `and`, disjunkce `or`, negace `not`)
- angličtina (alespoň pasivně)

`https://kahoot.it`

Algoritmus

Co to je algoritmus?

- návod/postup, jak „mechanicky“ vyřešit určitý typ úlohy/problému
- posloupnost instrukcí

Příklady algoritmů

- kuchařský recept
- notový zápis hudby
- Euklidův algoritmus (největší společný dělitel dvou čísel)
- rozklad přirozených čísel na součin prvočísel
- vyřešit Sudoku
- vygenerovat zadání Sudoku tak, aby mělo jen jedno řešení
- řešit nějakou logickou úlohu

`http://www.fi.muni.cz/~xpelanek/IB111/vkz/`

Žádoucí vlastnosti algoritmů

- jasný vstup a výstup
- obecnost
  - není jen pro omezenou sadu instancí
  - „spočítat 7×5“ vs. „spočítat součin dvou zadaných čísel“
- determinovanost
  - každý krok je jasně a přesně definován
  - v každé situaci jednoznačné, jak dál postupovat
  - není vždy nutné (pravděpodobnostní algoritmy)
- konečnost
  - skončí po konečném počtu kroků
- elementárnost kroků
  - každý krok je dostatečně jednoduchý

- efektivita
  - časová náročnost, prostorová náročnost, ...

Zápis algoritmu pro počítač

- musí být opravdu přesný
  - počítače jsou „hloupé“
  - „přiměřeně osolte“ vs. „nasypte do hrnce 1,7 g soli“
- používáme programovací jazyky
  - ty definují sadu elementárních instrukcí, které můžeme kombinovat
  - více o různých druzích jazyků v samostatné přednášce koncem semestru

Programování: motivace

Proč pořádně zvládnout základy programování?

- základ pro další studium
- užitečnost
  - profesní
  - občasná
- kreativita
- způsob myšlení

K čemu využít programování?

- vědecké výpočty
- zpracování dat (statistiky, grafy, ...)
- programování pro web
- skriptování (automatizace „nudných“ činností)
- vestavěné systémy
- (mobilní) aplikace

Programování

Jak se učit programování?

- tréninkem
- programování je dovednost; dovednost se získává praxí
- ukážeme vám cestu, ale jít po ní už musíte sami
  - přednášky, cvičení i domácí úkoly „ukazují cestu“
  - je vhodné si občas jen tak zkusit něco naprogramovat
  - k dispozici ukázky z přednášek – zkoušejte modifikovat

Základy Pythonu

Jak a kam psát programy

- program je textový soubor („plain text“)
  - Python: přípona `.py`
- spuštění programu:
  - Python je *interpretovaný* jazyk (více později)
  - Python postupně čte řádky programu a provádí je

[ukázka použití Pythonu z příkazové řádky]

- existují tzv. *vývojová prostředí* (IDE), která usnadňují psaní programů
  - automatické doplňování názvu funkcí apod. („napovídání“)
  - snadné spouštění programů klávesovou zkratkou / z menu
  - zvýrazňování syntaxe (barvy)

- IDE pro Python
  - základní součást instalace Pythonu: IDLE
  - existují i lepší, např. komerční PyCharm, který má verzi zdarma
  - doporučujeme vyzkoušet PyCharm

[ukázka použití PyCharm]

Syntaxe Pythonu

Obecné poznámky k syntaxi

- „case sensitive“ – na velikosti písmen záleží
  - `print` není totéž, co `PRINT` nebo `Print`
- komentáře – cokoli od znaku # po konec řádku
  - Python je ignoruje
  - užitečné pro dokumentaci, poznámky k fungování programu
- písmena s diakritikou
  - Python 3 umožňuje
  - raději nepoužívat v názvech proměnných, funkcí apod.
  - obecně raději preferovat angličtinu
    (proč? jak byste rádi četli kód po někom, kdo si proměnné pojmenoval finsky?)

Základní konstrukce

Výstup

- výpis textu, čísla, ...
- standardní výstup: obrazovka (terminál, konzole), dá se přesměrovat do souboru
- funkce `print(...)`

```
print("Ahoj. Toto je můj první program v Pythonu.")

print("Šest krát sedm je", 6 * 7)
```

Základní konstrukce

Proměnné

- místa v paměti, do kterých ukládáme hodnoty (čísla, řetězce, ...)
- hodnota se může měnit (proto *proměnné*)
- v Pythonu nemají proměnné (na rozdíl od hodnot) datový typ (více o typech příště)
- přiřazení hodnoty do proměnné: `proměnná = hodnota`
- v Pythonu se proměnná vytváří prvním přiřazením

```
x = 6 * 7
print(x)

x = "Tohle je text."
print(x)
```

Doporučení: Názvy proměnných volíme tak, aby z nich bylo jasné, co reprezentují.

Základní konstrukce

Cykly

- provedení části programu opakovaně
- procházení složitějších datových struktur

Cyklus s daným počtem opakování `for`

```
print("Před cyklem")
for i in range(10):
    print("Číslo", i)
print("Po cyklu")
```

- všimněte si, že „tělo“ cyklu v Pythonu vyznačujeme odsazením
  - standardně 4 mezery, IDE typicky odsazuje automaticky nebo smíme použít Tab

Základní konstrukce

Cyklus s podmínkou `while`

```
x = 10000
while x > 0:
    print("x je", x)
    x = x // 10
print("Na konci programu je v x hodnota", x)
```

Poznámka: `//` je v Pythonu 3 *celočíselné* dělení, `/` je *reálné dělení* (podrobněji příště)

Základní konstrukce

Větvení

- provedení části programu jen v případě, že platí určitá podmínka

```
x = 6 * 7
if x == 42:
    print("Vesmír je v pořádku.")
```

Poznámka: Všimněte si rozdílu mezi = a ==.

- jednoduché = je přiřazení (ukládáme hodnotu do proměnné)
- dvojité == je porovnání (ptáme se, zda jsou dvě hodnoty stejné)

Základní konstrukce

Složitější větvení

```
name = "James Bond"
if name == "James Bond":
    print("Hello, Mr. Bond!")
else:
    print("Hello, stranger!")
```

- je-li podmínka pravdivá, provede se blok příkazů za `if`
- není-li pravdivá, provede se blok příkazů za `else`

Ještě složitější větvení

```
x = -17
if x > 0:
    print("x je kladné")
elif x < 0:
    print("x je záporné")
else:
    print("x není ani kladné, ani záporné")
```

- podmínky se vyhodnocují postupně shora
- provede se blok příkazů u první podmínky, která je pravdivá

Základní konstrukce

Funkce

- způsob, jak rozložit program na menší logické celky
- funkce má parametry a může vracet hodnotu
  (to uvidíme příště)

```
def say_hello(name):
    print("Hello,", name, ", nice to meet you.")

say_hello("Tony Stark")
say_hello("Pepper Potts")
```

Proč chceme rozkládat programy do funkcí?

- vyhneme se opakování stejného/podobného kódu
- modularita (Lego), znovupoužitelnost
- snazší přemýšlení o problému, „dělba práce“

K úvodnímu procvičení

Želví grafika

- příklady na prvním a částečně i druhém cvičení
- vyzkoušení některých základních konstrukcí,
- virtuální „želva“, která chodí po obrazovce a zanechává za sebou stopu
- želva má polohu a orientaci (natočení)
- příkazy „pohni se vpřed o danou vzdálenost“, „otoč se o daný úhel“

K úvodnímu procvičení

```
from turtle import Turtle, done
leonardo = Turtle()

leonardo.forward(100)
leonardo.right(120)
leonardo.forward(100)
leonardo.right(120)
leonardo.forward(100)

done()
```

- zvládli byste tento program napsat lépe?

Závěrečný kvíz

`https://kahoot.it`

Závěr

Co jsme se dnes dozvěděli?

- organizační informace k předmětu
- podmínky hodnocení
- doporučené zdroje
- co je to algoritmus a program
- k čemu je programování
- základní konstrukce v jazyce Python

Co bude příště?

- zopakování základních konstrukcí s více podrobnostmi
- datové typy v Pythonu
- zásady psaní „pěkného“ kódu, PEP8
- příklady jednoduchých programů