Lekce 5

Algoritmy a složitost

Funkce, lambda funkce a rekurze. Základy složitosti a algoritmizace. Příklady základních algoritmů: největší společný dělitel, prvočísla.

C2184 Úvod do programování v Pythonu
podzim 2014

Mgr. Stanislav Geidl
Národní centrum pro výzkum biomolekul
Masarykova univerzita

Funkce

- funkce je **část programu** (je podprogram, angl. subroutine), která lze volat opakovaně z různých části programu, příkladem může být funkce print()
- každá funkce má svůj **identifikátor** (podobně jako proměnná), pomocí kterého ji můžeme volat
  `moje_funkce()`
- požadavky na funkci můžeme specifikovat pomocí **parametrů**
  `moje_funkce(parametr1, parametr2)`
- výsledek funkce může být předán jako **návratová hodnota**
  `navratova_hodnota = moje_funkce()`

Definice funkce

- pomocí klíčového slova `def` a bloku (: a odsazení)

```
def moje_funkce(parametr):
    pass
```

- návratovou hodnotu definujeme klíčovým slovem `return`, POZOR: po jejím zavolání se již v provádění funkce nepokračuje

```
def moje_funkce(parametr):
    print()
    return True
    print() # nedosažitelný kód
```

- pomocí `return` můžeme navracet hodnotu, hodnotu proměnné nebo také nic (`None`)

```
def moje_funkce(parametr):
    return
```

- defaultní hodnoty a názvy parametrů

Proměnné a funkce

- **globální proměnné** - existují mimo funkci, ale můžeme ji v rámci funkce používat a upravovat

```
a = 1
b = 2
def moje_funkce(parametr):
    print(a)
    print(b)
    b = 20
print(b)
```

- **parametry**

```
a = 1
def moje_funkce(parametr):
    print(a)
    print(parametr)
```

- **lokální proměnné** - existují pouze v rámci konkretního volání funkce a má přednost před globální proměnnou

```
a = 1
def moje_funkce(parametr):
    a = 10
    b = 20
print(b)
```

Rekurze a rekurzivní volání funkce

- volání sama sebe

```
def moje_funkce():
    moje_funkce()
```

- ukázka výpočtu faktoriálu

```
def faktorial(a):
    if a < 2:
        return 1
    else:
        return a * faktorial(a - 1)
```

Anonymní lambda funkce

- v případě, kdy potřebuje funkci volat "jednorázově"

```
square = lambda x: x**2
square(5) # 25
```

- příklad vlastního porovnávače ve funkci sort:

```
list.sort(cmp=lambda x, y: cmp(abs(x), abs(y)))
```

Algoritmus

- Algoritmus je schematický postup pro řešení určitého druhu problémů, který je prováděn pomocí konečného množství přesně definovaných kroků.
- V běžném životě se s algoritmy setkáváme ve formě kuchyňských receptů, návodů a postupů ...
- Algoritmus má tyto vlastnosti:
  1. konečnost - má konečný počet kroků
  2. určitost - všechny kroky jsou přesně definovány
  3. koreknost - pro správné zadání skončí se správným výsledkem
  4. obecnost - řeší všechny úlohy a vstupy, pro které je navržen
- Algoritmy dělíme na interaktivní a rekurzivní. Interaktivní používá cykly - opakuje konkretní bloky či podprogramy. Rekurzivní algoritmy používájí rekurzivní funkce - volají samy sebe.
- Asymptotická složitost algoritmu charakterizuje počet provedených operací v závislosti na velikosti dat (například počet prvků v seznamu). Například pokud procházíme pole, pak složitost bude lineární (na každý prvek připadá konstantní množství operací). Pokud ale budeme potřebovat pole seřadit, složitost algoritmu poroste.

Formy zápisu algoritmů

- graficky
  - Vývojové diagramy
  - Strukturogramy
  - UML diagramy (aktivitní diagramy)
- textově
  - Přirozený jazyk
  - Pseudokód
  - Formální jazyk a programovací jazyk

Ukázka vývojového diagramu

ukázka

Obecné zásady psaní kódu - proč?

proč?

Obecné zásady psaní kódu - proč?

- čitelnost
- udržitelnost - udržba a rozšiřování
- zdroj hotových funkcí či jiných částí programu -> tvorba univerzálnějších knihoven

Obecné zásady psaní kódu

Obecné zásady psaní kódu

- odsazení řádků musí odpovídat vnořeným blokům (v Pythonu automaticky)
- Jeden řádek - jeden příkaz (značka) (v Pythonu automaticky)
- logické celky odděleny mezerou (code style)
- dodržujte jednotné názvy objektů
- identifikátory by měly co nejpřesněji popisovat význam
- dodržovat formát pro konstrukce
- každá funkčnost naprogramovaná pouze jednou (DRY - don't repeate yourself)
- kód nesmí obsahovat magická čísla
- POUŽÍVAT KOMENTÁŘE

Obecné zásady psaní komentářů

- komentáře nesmí duplikovat kód
- komentář musí jednoznačně osvětlit popisovanou část kódu
- komentuje se hlavička každého souboru, se kterým se pracuje
- komentuje se hlavička všech objetků, funkcí a konstrukcí procedur
- komentují se všechny netriviální konstrukce

Jak na to

- na internetu můžeme najít "hotový kód"
- vymyslet to:
  - snažit se problem rozložit na jednodušší úkony - funkce
  - nakreslit vývojový diagram
  - ...

Více

http://www.algoritmy.net/