Geavanceerde Programmeertechnologie

Prof. dr. Kris Luyten

Jo Vermeulen

Wat mag je verwachten?

- Je wordt efficiënter als software ontwikkelaar
- Je kan je weg vinden in nieuwe programmeertalen van verschillende programmeerparadigma´s
- Je beheerst verschillende programmeertechnieken
- Je kan sneller nieuwe talen onder de knie krijgen
- Een hoog tempo met leerstof die van een goede programmeerkennis uitgaat
  - Java, C en C++

Wat verwachten wij?

- Zin om te programmeren en experimenteren
- Zin voor initiatief en zelfstandig werken
- Kunnen is even belangrijk als kennen (maar dat maakt kennen niet minder belangrijk)

Wat moet je doen?

- Aanwezig zijn in de lessen
- **Regelmatig notities nemen tijdens lessen** en responsies
- Taken tijdig inleveren
- Oefenen, oefenen, oefenen – ook in de andere programmeervakken

Wat mag je vooral niet doen?

- Alles laten liggen tot op het einde
- Vragen **alvorens zelf te zoeken**

Overzicht van het vak: thema's

- Programmeren en Programmeerparadigma's
- Functioneel Programmeren (Haskell)
- Logisch Programmeren (Prolog)
- Garbage collection
- Aspect-georiënteerd Programmeren
- Meta-programming (o.a. In Python)
- Software componenten en bibliotheken
- Eventueel: Scala programmeren

Overzicht van het vak: examinering

- Schriftelijk examen: 70%
- Taken (practica): 30%

Communicatie binnen het vak

- Website didactiekinf.uhasselt.be/gpt
- Vragen worden mondeling of *op het forum* gesteld
  - Bij voorkeur **niet** via email
  - Vragen over de leerstof worden via het forum beantwoord
- Hoorcolleges en responsies

Cursusmateriaal

- Wordt ter beschikking gesteld per les
- Van externe bronnen (zoals websites) wordt duidelijk aangegeven wat je er van moet kennen
- Geen aparte studieleidraad. Opdrachten staan in de slides

Doel van deze les

- Overzicht van programmeertalen
- Verschillende programmeerparadigma's kunnen onderscheiden en uitleggen
- Verschillende paradigma's kunnen gebruiken in Python

Programmeren!

- *De basis* van je opleiding
- Goed kunnen programmeren is een vereiste...
  - voor dit vak
  - voor je opleiding
  - voor je job
- Veel oefening helpt je op weg, en *heel veel* oefenen brengt je op niveau

Programmeertalen

http://people.ku.edu/~nkinners/LangList/Extras/classif.htm

http://oreilly.com/news/graphics/prog_lang_poster.pdf

- Talen: Postscript, Oberon, Tcl/Tk, Fortran, Prolog, Pascal, Delphi, Python, Cobol, Modula, Ada, Rexx, ISO C, C#, Javascript, ANSI/ISO C++, Ruby, Self, Eiffel, PHP, Perl, ML, Lisp, Objective-C, VB, Scheme, Haskell, Caml, Smalltalk, Miranda, PL/1, Simula, Java...
- Belangrijkste klassen: imperatief, functioneel, object-georiënteerd en logisch

Programmeertalen

http://people.ku.edu/~nkinners/LangList/Extras/classif.htm

http://oreilly.com/news/graphics/prog_lang_poster.pdf

- Talen: Postscript, Oberon, Tcl/Tk, Fortran, **Prolog**, Pascal, Delphi, Python, Cobol, Modula, Ada, Rexx, ISO C, C#, Javascript, ANSI/ISO C++, Ruby, Self, Eiffel, PHP, Perl, ML, Lisp, Objective-C, VB, Scheme, Haskell, Caml, Smalltalk, Miranda, PL/1, Simula, Java...
- Belangrijkste klassen: imperatief, functioneel, object-georiënteerd en **logisch**

Logisch programmeren

- Programmeren gebaseerd op logica
- Prolog
  - Predicatenlogica
  - Declaratief (*wat* niet *hoe)*
  - Facts (feiten) en clauses (regels)
  - Gebruikt in o.m. expertensystemen

Logisch programmeren: Prolog

facts

```
parent(Frans, Eefje).
parent(Klaar, Eefje).
parent(Eefje, Salammbo).
parent(Eefje,Mattho).
parent(Gustave, Salammbo).
parent(Gustave,Mattho).
```

Clause definieert een relatie

```
grandparent(X,Y) :-
        parent(X,Z),
        parent(Z,Y).
```

Ondervragen van een Prolog programma

```
? – parent(Eefje, Mattho).
        yes
? – grandparent(Frans,Salammbo).
        yes
? – grandparent(Klaar, X).
        X = Salammbo.
        X = Mattho.
```

Logisch programmeren: Prolog

- Verschillende interpreters en compilers: SWI-Prolog, GNU-Prolog, Sicstus Prolog, Amzi! Prolog,…
- Zelf al wat meer leren? “Adventure in Prolog” is een uitstekende resource om mee te starten: http://www.amzi.com/AdventureInProlog/

Programmeertalen

http://people.ku.edu/~nkinners/LangList/Extras/classif.htm

http://oreilly.com/news/graphics/prog_lang_poster.pdf

- Talen: Postscript, Oberon, Tcl/Tk, Fortran, Prolog, Pascal, Delphi, Python, Cobol, Modula, Ada, Rexx, ISO C, C#, Javascript, ANSI/ISO C++, Ruby, Self, Eiffel, PHP, Perl, **ML**, **Lisp**, Objective-C, VB, **Scheme**, **Haskell**, Caml, Smalltalk, **Miranda**, PL/1, Simula, Java...
- Belangrijkste klassen: imperatief, **functioneel**, object-georiënteerd en logisch

Functioneel programmeren

- Gebruikt … functies (maar dan die gebaseerd op Lambda calculus)
- Maar programma heeft geen *state*
- Geen gebruik van “destructive updates” (bijv. variabele updates) – dan = *puur* functioneel
- Functionele programmeertalen zijn bijzonder geschikt om *lijsten* mee te verwerken en bewerken
- Gaan we nog nader bestuderen
  - Haskell (veel veel toffer dan Scheme)

Programmeertalen

http://people.ku.edu/~nkinners/LangList/Extras/classif.htm

http://oreilly.com/news/graphics/prog_lang_poster.pdf

- Talen: Postscript, Oberon, Tcl/Tk, Fortran, Prolog, **Pascal**, Delphi, Python, Cobol, Modula, **Ada**, Rexx, **ISO C**, C#, Javascript, ANSI/ISO C++, Ruby, Self, Eiffel, PHP, Perl, ML, Lisp, Objective-C, VB, Scheme, Haskell, Caml, Smalltalk, Miranda, PL/1, Simula, Java...
- Belangrijkste klassen: **imperatief**, functioneel, object-georiënteerd en logisch

Programmeertalen

http://people.ku.edu/~nkinners/LangList/Extras/classif.htm

http://oreilly.com/news/graphics/prog_lang_poster.pdf

- Talen: Postscript, Oberon, Tcl/Tk, Fortran, Prolog, Pascal, Delphi, Python, Cobol, Modula, Ada, Rexx, ISO C, **C#**, Javascript, **ANSI/ISO C++**, Ruby, Self, Eiffel, PHP, Perl, ML, Lisp, **Objective-C**, VB, Scheme, Haskell, Caml, Smalltalk, Miranda, PL/1, Simula, **Java**...
- Belangrijkste klassen: imperatief, functioneel, **object-georiënteerd** en logisch

Programmeertalen

- Wat is een programmeertaal? Omschrijf wat een programmeertaal is aan de hand van je huidige kennis!
- Wat is een programmeerparadigma?

  Stijl van programmeren die invloed heeft op de manier waarop de uitvoering van de code gebeurt en die getypeerd wordt door de elementen die in de programmeertaal gebruikt worden (wel/geen variabelen, functies, objecten,...)

Multi-Paradigm Programmeertalen

- Programmeertalen die meerdere paradigma’ s ondersteunen
- Combinaties procedureel+OO (bijv. C++) en logisch+functioneel (bijv.) komen veel voor
- **Python** (gebruik Python 3.0)

Opdrachten

Oefeningen op programmeren in meerdere paradigma's in Python

Gegeven de datastructuur (sequence van tuples)

stamboom = [("Frans","Eefje"),("Klaar","Eefje"),("Eefje","Mattho"),("Eefje","Salammbo"),("Gustave","Mattho"),("Gustave","Salammbo")]

Waarbij Frans ouder is van Eefje, Eefje ouder is van Mattho en zoverder.

Opdrachten: Python functioneel

Lees http://docs.python.org/tutorial/datastructures.html

- Gegeven de functie *map* uit sectie 5.1.3. Definieer met *def* een functie *switchplace* die ouder en kind van plaats wijzigt. Pas deze functie op elke element van stamboom toe met *map*.

- Doe hetzelfde maar zonder de *map* functie. Itereer over de elementen *zonder een "tel" variabele hiervoor te gebruiken*. Schrijf in elke iteratie het resultaat van de switchplace functie uit. Tip: sectie 5.6 over looping techniques. Leg uit waarom dit een functionele aanpak is.

- Leg uit en geef een voorbeeld van een *list comprehension* in Python.

Opdrachten: Python OO

Lees http://docs.python.org/tutorial/classes.html

- Maak een class Person. Een Person heeft een naam, en kan één of meerdere kinderen hebben. Itereer over stamboom en creeer class instances op basis van deze tuples. Merk op dat je een boom-structuur van class instances krijgt zo.

- Wat betekent het *self* keyword juist, en waarom wordt het gebruikt?

- Leg uit hoe de scope van variabelen bepaald wordt bij classes in Python.

- Wat is het verschil tussen C++ methods en Python methods? (tip: method objects)