Introductie en basis

Nascholing Programming Paradigms

COEN CROMBACH / FONTYS – KEES HUIZING TU EINDHOVEN - 10 JUNI 2020

Functioneel paradigma

- Zit in alle imperatieve programmeertalen sinds FORTRAN
- denk in functies *zonder side effects (pure functions, immutability)*
  - dus geen assignments
- recursie is het principe principe om herhaling te krijgen
- recursie is principe om datastrucutren te maken (lists, abstract data types)
- functies zijn *full class citizens:* kunnen als parameter optreden
  - vergelijk: variabele - waarde/expressie
    x — 3 + y * 4
    functie - definitie / lambda expressie
    f — \a -> a * a
- op basis hiervan: bibliotheek van krachtige flexibele functies

- Imperatief:
  FETCH x
  ADD 2
  MUL 5
- n:=0; s:=0;
  while n < N {
  s := s + a[n];
  n := n + 1;
  }

- Functioneel:
  (x + 2) * 5
- sum(a, n) {
  if n == a.length
  return 0;
  else
  return a[n] +
  sum(a, n+1);
  }

Waarom functioneel?

- Andere positie innemen
- Belangrijk denkprincipe, je ziet het steeds meer in imperatieve talen
  - lambda's in Python:
    students is een dictionary van lists van cijfers
    welke student heeft het hoogste gemiddelde cijfer?

    max(students.keys(), default=' ', key=lambda k: sum(students[k]) / len(students[k]) if students[k] else 0)
  - Java Streams API:
    functioneel opereren op (grote) sequenties van data
    met *filter, map, reduce*
    - functioneel, dus geen side effects
      mede daardoor is optimalisatie van performance mogelijk door parallel processing

Jupyter notebook

- Documenten met tekst en executeerbare code
- Beschikbaar voor vele programmeertalen
- Georganiseerd in *cellen*
  - markdown: voor tekst (indeling, uitleg, opdrachten)
  - code: voor code in een bepaalde programmeertaal
    - met ctrl-Enter wordt de cell uitgevoerd en de output getoond
- Draait in de browser
- Geen installatie nodig, draait op onze server
  - Toegang via **github**-account (even maken, indien nodig!)

Two types of notebooks

- Elm REPL
  levert een read-evaluate-print-loop, als een shell
  - handig om de taal te leren en de basisconcepten
- Elm
  levert de volledige elm-functionaliteit met interfacing naar HTML
  - nodig voor toepassingen

- Nodig voor onze elm-port naar de Jupyter notebooks

Elm

- lijkt op Haskell (maar niet hezelfde! o.a. geen list comprehensions)
  - geen haakjes bij functie-applicatie, maar ‘niks’ (spatie); binding is van links naar rechts
  - currying: laat argument weg en je hebt een functie
    - stel je hebt een vermenigvuldigingsfunctie gedefinieerd (heb je trouwens al als (*))
      times a b = a * b
      dan is times 2 een functie met één parameter, een verdubbelfunctie;
      voorbeeld:
      List.map (times 2) [1, 2, 3]
      verdubbelt alle elemenen van de lijst → [2, 4, 6]
  - typing is impliciet, maar wordt wel gecheckt
- alle functies zijn *totaal:* leveren altijd een waarde op (desnoods *Nothing*). Errors en excepctions zitten in de data

Elm bis

- Het bijzondere van elm is dat je het kunt koppelen aan een webpagina (of -site)
- Daarmee is visualisatie en interactie mogelijk
- Volgens het *paradigma* van model-view-update
- Goede foutmeldingen (soms iets té veel van het goede)

Vervolgens

- We gaan langs wat basisideeën van het functionele paradigma
- Aan de hand van elm
- Zelf doen in een notebook!
- Coen volgt de chat
- Volgende keer o.a. toepassingen