Title: The risks and benefits of teaching purely functional programming in first year

Authors: Manuel M. T. Chakravarty and Gabriele Keller

Published: Journal of Functional Programming, Volume 14, Issue 1, January 2004, Pages 113-123

Link: https://www.cambridge.org/core/journals/journal-of-functional-programming/article/risks-and-benefits-of-teaching-purely-functional-programming-in-first-year/39F929A1793B67BCEF316DBDB717F273

Abstract:

This paper argues that teaching purely functional programming in freshman computer science courses can be detrimental. Instead, the focus should be on teaching elementary programming techniques and essential computing concepts using functional languages as a tool. The authors present an approach they used to teach Haskell to large first year classes and summarize student feedback supporting their claims.

Introduction:

- The authors propose that purely functional programming should not be the focus when teaching freshman programming courses.

- The focus should be on teaching elementary programming techniques, essential computing concepts, analytical thinking and problem solving skills.

- Popular Haskell textbooks for freshmen concentrate on functional programming concepts rather than general computing topics.

- The authors present an approach they used to teach large intro classes in Haskell focusing on general concepts rather than functional programming specifically.

- They summarize student feedback supporting the benefits of their approach.

Goals for Intro Courses:

- The principal aims should be:

1. Conveying elementary programming techniques
2. Introducing essential computing concepts
3. Fostering analytical thinking and problem solving skills

- Good courses integrate these aims as much as possible.

Advantages of Functional Languages:

- Functional languages support the 3 aims well:
  - Clean semantics aid clear discussion of techniques and formal reasoning.
  - Tight theory-practice integration relates aims 1 and 2.
  - High expressiveness allows tackling advanced problems early (aim 3).
- Strong static type discipline is beneficial for all 3 aims.
- Lightweight syntax moves focus to concepts not language.
- Lack of side effects aids discussion of concepts in isolation.
- Conciseness and high-level abstractions allow early complex data structures.
- Clean semantics integrates techniques and theory, e.g. equational reasoning.

Advantages from Not Being Mainstream:

- Most students have no prior Haskell experience so it equalizes knowledge.
- Helps students without prior programming experience not fall behind.
- Allows students with prior programming experience to learn new concepts.
- Feedback showed students appreciated the level playing field.

Environment and Development:

- Conciseness allows developing examples interactively during lectures.
- Students imitate lecturer's development process and style more easily.
- Integrated interactive and batch tools support progressive development.

Survey Results:

- Students found assignments interesting, challenging problem solving tasks.
- Online program development was rated the most helpful lecture technique.

Omissions from Functional Programming Focus:

- Some functional programming topics are omitted:

  - List comprehensions

  - Currying

  - Advanced higher-order functions

  - Lambda expressions

- These are not essential for the 3 aims and cause confusion for beginners.

- Recursion is covered as needed despite being an advanced concept.

Countering Views it is Not Used in Industry:

- Emphasize foundational knowledge over languages.

- Discuss real-world uses of functional languages.

- Highlight practical aspects and connections to other languages/courses.

- Survey comments indicated this helped improve student perspectives.

I/O Programming:

- I/O is considered essential general programming knowledge.

- Functional languages allow precise discussion of state and side effects.

- Use of monads is avoided as too advanced.

- Survey showed I/O considered reasonably difficult but understandable.

Conclusions:

- Functional languages support concentrating on essential computing principles rather than language specifics.

- They facilitate elementary techniques, computing concepts, and analytical thinking aims.

- Avoiding advanced functional programming topics keeps focus on general principles for freshmen.