

CIS 552

Advanced Programming

Fall 2011



Advanced Programming?

- Good programmers get the job done
- **Excellent** programmers
 - write code that other people can read, understand, maintain and modify
 - rewrite code to make it clear and elegant
 - design with *abstractions* (*and have a large toolbox*)
 - design for reuse



Tony Hoare


Turing Award Lecture 1980

*"There are two ways of constructing a software design: One way is to make it so **simple** that there are obviously no deficiencies, and the other way is to make it so **complicated** that there are no obvious deficiencies. The first method is far more difficult."*

Simplicity

The absence of unnecessary elements

simple code is

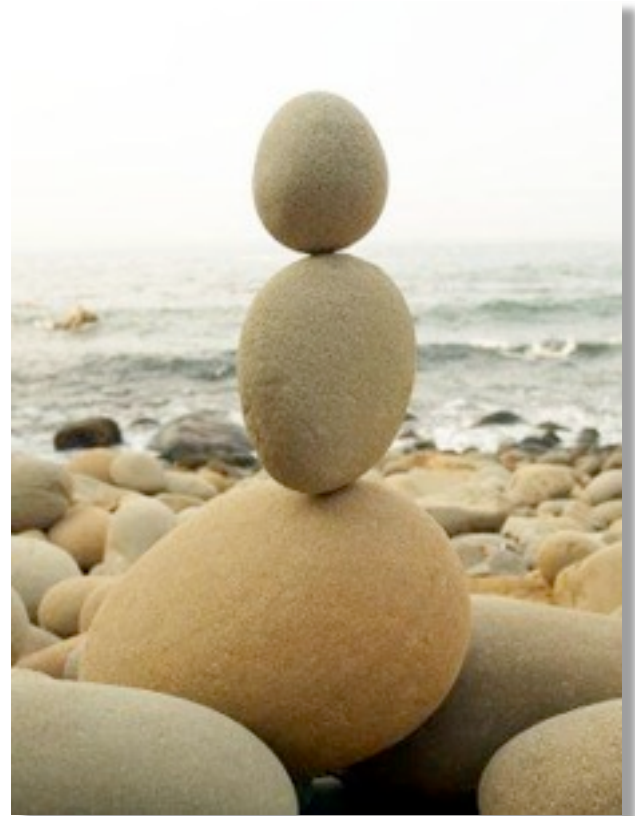


- Readable
- Reusable
- Modifiable
- Predictable
- Checkable

Absence of Unnecessary Elements

- No Mutation
- No Objects
- No Loops

A focus on what code **means**
instead of what it does



Functional Programming

- Readable
- Reusable
- Modifiable
- Predictable
- Checkable

So, Who Uses FP?

- PL Researchers



So, Who Uses FP?



So, Who Uses FP?



Microsoft[®]

So, Who Uses FP?

The image shows the Facebook logo, which consists of the word "facebook" in a white, lowercase, sans-serif font centered within a blue rounded rectangle.

facebook

So, Who Uses FP?



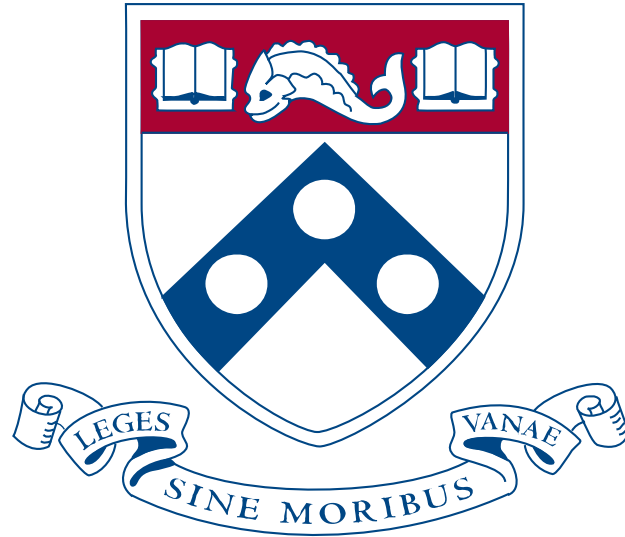
twitter



So, Who Uses FP?



CIS 552



Haskell

Why Haskell?

- Bleeding edge PL technology.

Why Haskell?

- Beautiful.

Why Haskell?

- Blows Your Mind.

Why Haskell?

- Fun.

Plan for the semester

Functional programming

- Black-belt Haskell
- Many small-scale case studies
- Class analysis of design
- Transfer to other languages

Programming in general

- Modular decomposition
- Abstraction
- Testing
- Debugging

What this course is not

- CIS 350/573, Software Engineering
 - Focuses on "Software in the large"
 - Problems that arise in projects that are too large for one person
 - lifecycle models
 - project management
 - design modeling notations (UML)
 - formal specification
- Both courses complement each other

Course staff

Instructor: Dr. Stephanie Weirich
sweirich@seas.upenn.edu

TA: Sam Panzer
panzers@seas.upenn.edu

Fill-in instructor: Chris Casinghino
ccasin@cis.upenn.edu



ICFP 2011

The 16th ACM SIGPLAN International Conference on Functional Programming



Association for
Computing Machinery

Tokyo, Japan

September 19-21, 2011

[Home](#)

[Call for papers](#)

[Call for workshop proposals](#)

[Program](#)

[Registration](#)

[Local information](#)

[Programming contest](#)



ICFP 2011 provides a forum for researchers and developers to hear about the latest work on the design, implementations, principles, and uses of functional programming. The conference covers the entire spectrum of work, from practice to theory, including its peripheries.

Where to go for help

- Course website: (schedule, homework, lecture notes)
- Textbook: Real World Haskell (free online)
- Class forum: piazza.com
- Office hours: Weirich, Wed 1:30-3pm, Levine 510
- Office hours: Panzer, Tues 3-4:30pm, Levine 5th floor

Grading

- 50% Programming assignments
 - graded on correctness and style
 - lowest grade dropped at the end of the semester
- 20 % Final Project (your choice)
- 30 % Class participation
 - questions
 - HW debriefings
 - piazza participation
 - read a good blog post about FP? post about it on piazza!

Audience

- People with strong background in programming and mathematics
- No experience with FP expected
- Priority to CIS undergrads/grad students
- If not registered, see me during office hours after class

First Homework Assignment

- Download from course website
- Due at noon, Saturday, September 18th
- Submit via course website
- Must compile with `-Wall -Werror` to get any credit.
- Late policy
 - 10 point penalty for up to 24 hours late
 - 20 point penalty for up to 48 hours late
 - no credit for assignments submitted after 48 hours

Homework style

- Style guide on course website. Read it.
- Interactive HW discussions
 - Examples of good/bad style
- Revise, revise, revise
 - Passing all the tests is not enough
 - Code must be effective technical communication
 - How can I make this code cleaner, more general, more clear about what it is doing?

Academic Integrity

- I expect you to follow Penn's policies on Academic Integrity. Read it.
- Do not plagiarize or copy code
 - Using a library function is ok
 - Looking up the source of that function and presenting it as your own work is not
- Do not get someone else to do your work for you
 - Ok to ask for help debugging a type error
 - Not ok to ask for the answer
 - Ask high-level questions on piazza

Class Format

- Mostly interactive demos, even when presenting new material
- Lecture notes available after class, try on your own
- Questions strongly encouraged