CSE 114A: Fall 2023

Foundations of Programming Languages

Lecture 1: Course Overview

Owen Arden
UC Santa Cruz

A Programming Language

- Two variables
  - x, y
- Three operations
  - x++
  - x--
  - (x=0)? L1:L2;

```
L1: x++;
    y--;
    (y=0)?L2:L1
L2: …
```

Fact: This is “equivalent to” to every PL!

Good luck writing quicksort

… or Windows, Google, Spotify!

So why study PL ?

Programming language
shapes
Programming thought

So why study PL ?

Language affects how:

- Ideas are expressed
- Computation is expressed

Course Goals

“Free your mind”

-Morpheus

Learn New Languages/Constructs

New ways to:

- describe
- organize
- think about

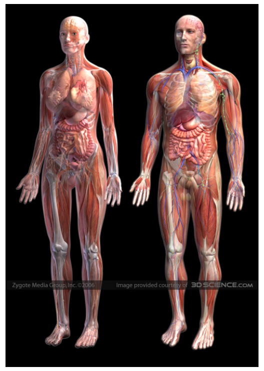
computation

Goal: Enable you to Program

- Readable
- Correct
- Extendable
- Modifiable
- Reusable

Goal: How to learn new PLs

No Java (C#) 15 (10) years ago
AJAX? Python? Ruby? Erlang? F#?...



Learn the anatomy of a PL

- Fundamental building blocks
- Different guises in different PLs

Re-learn the PLs you already know

Goal: How to design new PLs

…“who, me ?”

Buried in every extensible system is a PL

- Emacs, Android: Lisp
- Word, Powerpoint: Macros, VBScript
- Unreal: UnrealScript (Game Scripting)
- Facebook: FBML, FBJS
- SQL, Renderman, LaTeX, XML …

Enables you to choose right PL

"…but isn't that decided by
- libraries,
- standards,
- and my boss ?"

Yes.

My goal: educate tomorrow's tech leaders & bosses, so you'll make informed choices

Speaking of Right and Wrong...

Imperative Programming

$x = x+1$

WTF?

$x = x+1$

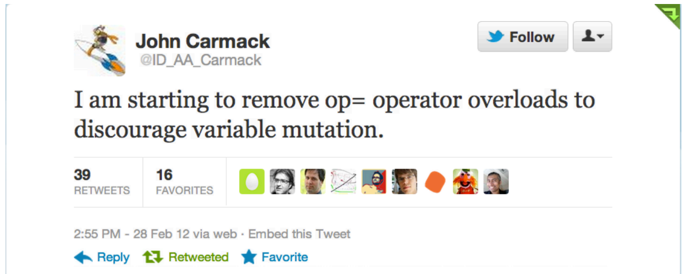
Imperative = Mutation

Imperative = Mutation

Bad!

Don’t take my word for it

John Carmack
Creator of FPS: Doom, Quake,...



John Carmack
@ID_AA_Carmack

I am starting to remove op= operator overloads to discourage variable mutation.

39 RETWEETS 16 FAVORITES

2:55 PM - 28 Feb 12 via web · Embed this Tweet

Don’t take my word for it

Tim Sweeney (Epic, Creator of UNREAL)

*“In a concurrent world,
imperative is the wrong default”*

Functional Programming

Functional Programming ?

No Assignment.

No Mutation.

No Loops.

OMG! Who uses FP?!

So, Who Uses FP ?

Google

MapReduce

So, Who Uses FP ?

Microsoft®

Linq, F#

So, Who Uses FP ?

facebook

Erlang

So, Who Uses FP ?

Scala

So, Who Uses FP ?

Wall Street

(all of the above)

So, Who Uses FP ?

…CSE 114A

Course Mechanics and Logistics

Logistics

Course website:

https://ucsc-cse-114a.github.io/fall23/

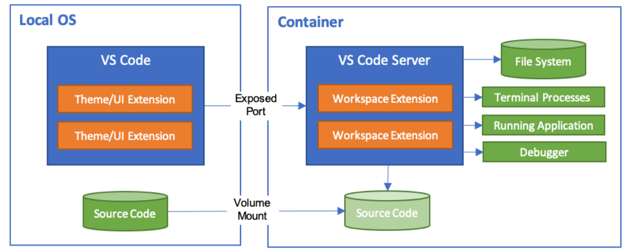
Resources

Course texts (optional):

- An Introduction to Functional Programming Through Lambda Calculus by Greg Michaelson. Free pre-print.
- Thinking Functionally with Haskell by Richard Bird. Available online (free via library).
- Programming in Haskell (2nd ed.) by Graham Hutton.
- Real World Haskell by Bryan O'Sullivan. Available online (free via library).
- Learn You a Haskell for Great Good by Miran Lipovača. Available free online
- Write You a Haskell by Stephen Diehl. (incomplete, but useful) Available free online

Resources

Haskell Dev Container

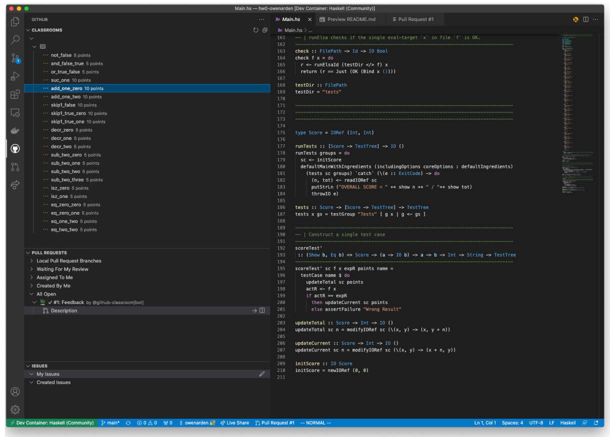


- https://github.com/UCSC-CSE-114A/cs114a-devcontainer

Recommended IDE: VS Code

- Legit IDE setup for Haskell!
  - Devcontainer: A Haskell dev environment is built in a container and VS Code automatically mounts the container volume
  - Codespaces: devcontainers in the cloud!!
  - Also some integrations with Git and GitHub Classroom

VS Code



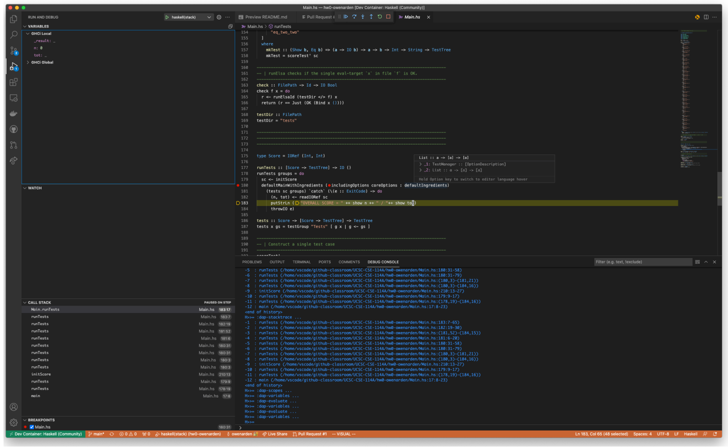
not_false 5 points
and_false_true 5 points
or_true_false 5 points
suc_one 10 points
add_one_zero 10 points
add_one_two 10 points
skip1_false 10 points
skip1_true_zero 10 points
skip1_true_one 10 points
decr_zero 5 points
decr_one 5 points
decr_two 5 points
sub_two_zero 5 points
sub_two_one 5 points
sub_two_two 5 points
sub_two_three 5 points
isz_zero 5 points
isz_one 5 points
eq_zero_zero 5 points
eq_zero_one 5 points
eq_one_two 5 points
eq_two_two 5 points

PULL REQUESTS
Local Pull Request Branches
Waiting For My Review
Assigned To Me
Created By Me
All Open
✓ #1: Feedback by @github-classroom[bot]
Description

ISSUES
My Issues
Created Issues

```haskell
-- | runElsa checks if the single eval-target `x` in file `f` is OK.
--------------------------------------------------------------------------------
check :: FilePath -> Id -> IO Bool
check f x = do
  r <- runElsaId (testDir </> f) x
  return (r == Just (OK (Bind x ())))

testDir :: FilePath
testDir = "tests"

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

type Score = IORef (Int, Int)

runTests :: [Score -> TestTree] -> IO ()
runTests groups = do
  sc <- initScore
  defaultMainWithIngredients (includingOptions coreOptions : defaultIngredients)
    (tests sc groups) `catch` (\(e :: ExitCode) -> do
      (n, tot) <- readIORef sc
      putStrLn ("OVERALL SCORE = " ++ show n ++ " / "++ show tot)
      throwIO e)

tests :: Score -> [Score -> TestTree] -> TestTree
tests x gs = testGroup "Tests" [ g x | g <- gs ]

--------------------------------------------------------------------------------
-- | Construct a single test case
--------------------------------------------------------------------------------
scoreTest'
 :: (Show b, Eq b) => Score -> (a -> IO b) -> a -> b -> Int -> String -> TestTree
--------------------------------------------------------------------------------
scoreTest' sc f x expR points name =
  testCase name $ do
    updateTotal sc points
    actR <- f x
    if actR == expR
      then updateCurrent sc points
      else assertFailure "Wrong Result"

updateTotal :: Score -> Int -> IO ()
updateTotal sc n = modifyIORef sc (\(x, y) -> (x, y + n))

updateCurrent :: Score -> Int -> IO ()
updateCurrent sc n = modifyIORef sc (\(x, y) -> (x + n, y))

initScore :: IO Score
initScore = newIORef (0, 0)
```

VS Code



```
_result: _
n: 0
tot: _
```

```
Main.runTests   Main.hs 183:17
runTests        Main.hs 183:7
runTests        Main.hs 182:19
runTests        Main.hs 181:52
runTests        Main.hs 181:6
runTests        Main.hs 180:31
runTests        Main.hs 180:31
runTests        Main.hs 180:3
runTests        Main.hs 180:3
initScore       Main.hs 210:13
runTests        Main.hs 179:9
runTests        Main.hs 178:19
main            Main.hs 17:8
```

```
Main.hs 180:31
```

```haskell
        "eq_two_two"
    ]
  where
    mkTest :: (Show b, Eq b) => (a -> IO b) -> a -> b -> Int -> String -> TestTree
    mkTest = scoreTest' sc

--------------------------------------------------------------------------------
-- | runElsa checks if the single eval-target `x` in file `f` is OK.
--------------------------------------------------------------------------------
check :: FilePath -> Id -> IO Bool
check f x = do
  r <- runElsaId (testDir </> f) x
  return (r == Just (OK (Bind x ())))

testDir :: FilePath
testDir = "tests"

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

type Score = IORef (Int, Int)

runTests :: [Score -> TestTree] -> IO ()
runTests groups = do
  sc <- initScore
  defaultMainWithIngredients (includingOptions coreOptions : defaultIngredients)
    (tests sc groups) `catch` (\(e :: ExitCode) -> do
      (n, tot) <- readIORef sc
      putStrLn ("OVERALL SCORE = " ++ show n ++ " / "++ show tot)
      throwIO e)

tests :: Score -> [Score -> TestTree] -> TestTree
tests x gs = testGroup "Tests" [ g x | g <- gs ]

--------------------------------------------------------------------------------
-- | Construct a single test case
--------------------------------------------------------------------------------
scoreTest'
```

```
List :: a -> [a] -> [a]
> _1: TestManager :: [OptionDescription]
> _2: List :: a -> [a] -> [a]
Hold Option key to switch to editor language hover
```

```
-5  : runTests (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:180:31-58)
-6  : runTests (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:180:31-79)
-7  : runTests (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:(180,3)-(181,21))
-8  : runTests (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:(180,3)-(184,16))
-9  : initScore (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:210:13-27)
-10 : runTests (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:179:9-17)
-11 : runTests (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:(178,19)-(184,16))
-12 : main (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:17:8-23)
<end of history>
H>>= :dap-stacktrace ...
-1  : runTests (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:183:7-65)
-2  : runTests (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:182:19-30)
-3  : runTests (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:(181,52)-(184,15))
-4  : runTests (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:181:6-20)
-5  : runTests (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:180:31-58)
-6  : runTests (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:180:31-79)
-7  : runTests (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:(180,3)-(181,21))
-8  : runTests (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:(180,3)-(184,16))
-9  : initScore (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:210:13-27)
-10 : runTests (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:179:9-17)
-11 : runTests (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:(178,19)-(184,16))
-12 : main (/home/vscode/github-classroom/UCSC-CSE-114A/hw0-owenarden/Main.hs:17:8-23)
<end of history>
H>>= :dap-scopes ...
H>>= :dap-variables ...
H>>= :dap-evaluate ...
H>>= :dap-variables ...
H>>= :dap-evaluate ...
H>>= :dap-variables ...
H>>=
```

Peer Instruction (ish)

Peer Instruction

- Make class interactive
  - Help YOU and ME understand whats tricky
- Respond to in-class quizzes
  - 5% of your grade
  - Respond to 75% questions
- Bring laptop/phone if you have one

In Class Exercises

1. Solo Vote: Think for yourself, select answer

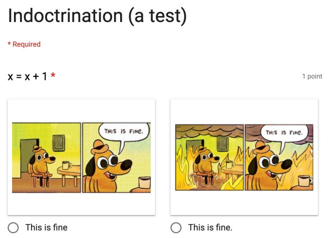
2. Discuss: Analyze Problem with neighbors
   - Practice analyzing, talking about tricky notions
   - Reach consensus
   - Have questions, raise your hand!

3. Group Vote: Everyone in group votes

4. Class-wide Discussion:
   - What did you find easy/hard?
   - Questions from here show up in exams

In Class Exercises

Let’s try it out (if you have a device):



Indoctrination (a test)

* Required

x = x + 1 * 1 point

○ This is fine

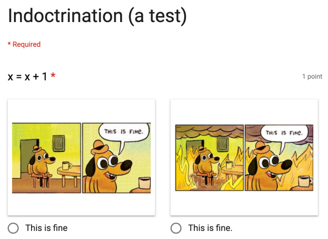
○ This is fine.

http://tiny.cc/cse116-trial

Make your individual choice

In Class Exercises

Let’s try it out (if you have a device):



Indoctrination (a test)

* Required

x = x + 1 * 1 point

○ This is fine

○ This is fine.

http://tiny.cc/cse116-trial

Now “confer” with a neighbor and agree on a choice for your group

Requirements and Grading

In-Class Exercises:	5%
Midterm:	30%
Programming Assignments (6):	30%
Final:	35%

Two hints/rumors:

1. Lots of work
2. Don’t worry (too much) about grade

Note: Regrades must be requested *within two weeks of receiving grade*

Resources

- Online lecture notes
- Readings and exercises
- Lecture capture available in Yuja
- Pay attention to lecture and section!
- Do assignments yourself (+partner)!

Ask for help!

- Lots of help available, will be adding more soon. (watch website)
- Goal: course staff available ***every day of the week***
- Discussion sections with TAs to help with lecture concepts and get unstuck on assignments

Programming Assignments

All assignments are managed through GitHub Classroom (link on course page).

- **You must *push* your submitted code.**

Deadline Extension:

- Four “late days”, used as “whole unit”
- 5 mins late = 1 late day
- Plan ahead, no other extensions

See course webpage for HW deadlines

Programming Assignments

Unfamiliar languages
+ Unfamiliar environments

Start Early!

Weekly Programming Assignments

Scoring = Test suite

No Compile, No Score

Weekly Programming Assignments

Forget Java, C, C++ …
… other 20th century PLs

Don’t complain
… that Haskell is hard
… that Haskell is @!%@#

Immerse yourself in new language

It is not.

(You can do it)

Immerse yourself in new language

Word from our sponsor …

- Programming Assignments done ALONE or in **(official)** groups of two (as permitted)
- We use plagiarism detection software
  - MOSS is fantastic, plagiarize at your own risk
- Zero Tolerance
  - offenders punished ruthlessly
- Please see academic integrity statement:
  - https://ue.ucsc.edu/academic-misconduct.html