CODE REVIEW 1

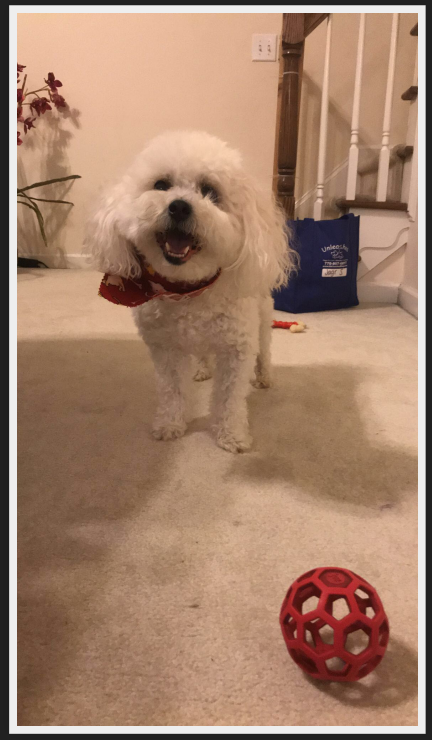
JOSH SEIDES

TOPIC OUTLINE

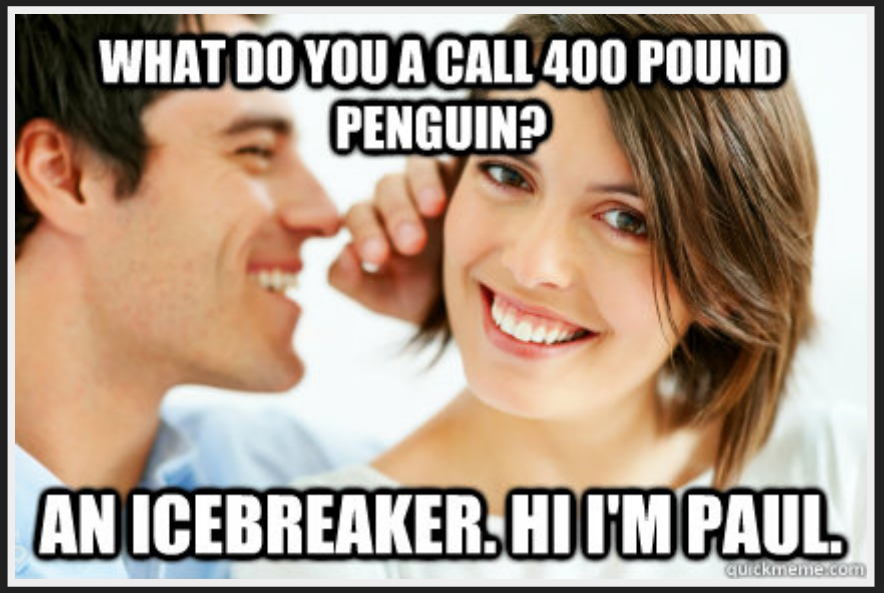
- Introduction
- Concrete vs. abstract syntax
- Type inference
- First-order functional programming
- Aside on Git and command line
- Testing

INTRODUCTION

ABOUT ME



- Sophomore
- Atlanta, GA
- Mather <3
- Computer Science, Economics
- HSA and HCCG
- New York...Rangers, Giants, Yankees <3<3


WHAT DO YOU A CALL 400 POUND PENGUIN?
AN ICEBREAKER. HI I'M PAUL.
quickmeme.com

LOGISTICS

- Every Friday 12-1pm @ Science Center 222
- Important concepts, practice, and lab
- Pset buddies?! No.
- Individual meetings
- Questions, questions, questions

jseides@college.harvard.edu

Kevin Zhu

to Josh

12:50 AM View details

I haven't even met you yet and I already know you're the worst TF and the reason my parents don't love me. Screw you and you're sweet tea drinking, sweetgreen munching a-double-hockey-sticks.

WHY OCAML?

WHY OCAML?

- No side effects
- Better reasoning
- Concurrency
- Other paradigms

CONCRETE VS. ABSTRACT SYNTAX

DEFINITION

- **Concrete syntax**: content and order of actual characters written
- **Abstract syntax**: structure of syntax
  - Order of operations
  - Tree diagrams

EXAMPLES

```
- 3 - 5
- (3 - 5)
(- 3) - 5
```

WHO CARES?

NO ONE. JK...

- Compilers
- Abstraction
- Code maintenance

QUESTIONS 1-3

TYPE INFERENCE

WHO CARES?

WHO CARES?

- Error-catching
- Writing functions

DEFINITION

- Statically-typed
- Atomic types
- User-defined types
- Compound types
  - Different types
  - Type vs. value constructors

EXAMPLES

```
1 (* int *)
2.5 (* float *)
true (* bool *)
"mather" (* string *)
() (* unit *)
```

WRITING TYPE EXPRESSIONS

GENERAL FORM

- Non-functions
- Functions
- Inline typing vs. type expressions

QUESTIONS 4-8

FIRST-ORDER FUNCTIONAL PROGRAMMING

DEFINITION

- **First-order**: inputs and outputs are atomic, user-defined, or compound
- **Higher-order**: inputs and/or outputs are functions
- Functions as **first-class citizens**

GENERAL APPROACH FOR LIST PROBLEMS

1. Define type expression
2. (Usually) add rec keyword
3. Match the list
4. Base case: isolate
5. Recursive case: assume and solve

EXAMPLES

```
square_all [1; 2; 3] ;; (* [1; 4; 9] *)
```

QUESTIONS 9-10

COMMAND LINE AND GIT

COMMANDS

- cd
- ls
- mkdir
- tab completion

GIT

- See worksheet
- git add .
- git commit -am "message"
- git push

TESTING

WHY?

- The autograder is not your friend
- About that pset...

EXAMPLES

```
assert (max_list [1; 1; 1] = 1) ;;
assert (max_list [1; 4; 2] = 4) ;;
assert (max_list [4; 2; 1] = 4) ;;
assert (max_list [1; 2; 4] = 4) ;;
```