CS 313 Lecture 1: Introduction
Goals of this course

1. Learn about principles of programming languages

• 4 paradigms:
  • procedural
  • object-oriented (OO)
  • functional
  • logic
Goals of this course

1. Learn about principles of programming languages
2. Survey of programming languages (+ history)

• 4 paradigms:
  • procedural
  • object-oriented (OO)
  • functional
  • logic

  C, Pascal
  Smalltalk, Ruby
  Scheme, ML
  Prolog

others? (HW 0)
Why study PLs?

1. as **users**
   - know how to choose appropriate language
   - know useful programming constructs

2. as **designers:**
   - want to understand motivation for different PL constructs & facilities

3. as **implementers** (compiler writers):
   - need to know cost for PL facilities
   - take CS 433!
Theoretical vs practical power

• all programming languages have *universal computing power*
  • equivalent to Turing machine
  • any program in one language can be rewritten in another
    → *theoretically equivalent*

• but: interested in practical power
  • how easily / elegantly can a problem be solved?
  • how fast?
  • how easy to debug / maintain / change?
Programming Linguistics

• CS 313 resembles linguistics (rather than language) course

• difference: we can design language, linguists can’t

• Sapir-Whorf hypothesis:
  • “the structure of a language defines the boundaries of thought”

• applied to PLs:
  • a given PL influences the types of solutions we’re likely to see
Trade-offs

• expressive power vs. safety
  • assembly, C: allow unchecked type conversion
    + more expressive
    - harder to debug
  • Pascal is “safe” language:
    + harder to make mistakes
    - less expressive

• static vs dynamic typing:
  • static (e.g. Pascal, Java): all types known at compile time
  • dynamic (e.g. Python): type checking happens at run time
Trade-offs

• implementation vs. usefulness
  what features to build in?

  • many (C++, LISP)
    + easy to use    + shorter programs
    - hard to learn  - hard to write compiler
  
  • few (Pascal, Scheme)
    + easy to learn    + easy to write compiler
    - longer programs

  • compromise: small(ish) language + libraries
    • e.g. Python, Java
Trade-offs

• Concise vs. clean syntax

{ ... } begin ... end
x++ x := x + 1
a + b (+ a b)
overloading +
Summary

• Organize programming languages by paradigm
• Get historical perspective from survey of PLs
• Relationship to Linguistics
• Understand trade-offs why features are present/absent in PLs
Homework

• do HW 0 on course page
• read Sethi Ch 1 (optional – background material for lecture 1)

Before next class:
• watch Lecture 2 videos
• take Quiz 2
• submit a question about the material