ECE326

PROGRAMMING LANGUAGES

Lecture 1 : Course Introduction

Kuei (Jack) Sun
ECE
University of Toronto
Fall 2020

Course Instructor

- Kuei (Jack) Sun
- Contact Information
  - Use Piazza
    - Send me (Kuei Sun) a private post
    - http://piazza.com/utoronto.ca/fall2020/ece326
      - Sign up to the course to get access
  - By E-mail
    - kuei.sun@mail.utoronto.ca
- Research Interests
  - Systems programming, software optimization, etc.

Course Information

- Course Website
  - http://fs.csl.toronto.edu/~sunk/ece326.html
  - Lecture and tutorial notes/videos, assignment handouts
- Quercus
  - Grade posting, online exams, lab group sign-up!
- Piazza
  - Course announcement, course discussion
  - Assignment discussion
    - Lab TAs will read and answer relevant posts periodically

Course Information

- No required textbook
  - Exam questions will come from lectures, tutorials, and assignments
  - See course website for suggested textbooks
- Lab sessions
  - Get help from a TA with your assignments
- Tutorials
  - Cover supplementary materials not in lectures
  - Go through sample problems that *may* appear on exams

Background

- Programming Languages
  - A formal language consisting of a instructions to implement algorithms and perform tasks on computers
  - Thousands exist, more being made
- This course
  - General-purpose programming languages
    - Meant for use across different application domain
- Programming Paradigms
  - A way to categorize languages by style and features

Level of Abstraction

- more abstraction
- easier to write

- more direct hardware access
- better performance

High-Level Languages	Python, Ruby Java, Kotlin, Scala, Clojure Haskell, Racket Visual Basic, C#
Systems Languages	C/C++ Ada, D, Rust Swift (by Apple, for iOS apps)
Low-Level Languages	Assembly Languages Machine Languages

Level of Abstraction

- Systems programming languages
  - Designed for performance
  - Allows some level of hardware awareness
    - Optimization hints (e.g. `restrict`, `volatile`, …etc)
    - Inline assembly
  - Still provides some high-level concepts
    - At a trade-off of longer compilation time

Compiler Optimization

- Example: C
  - restrict keyword
    - Informs compiler that a pointer has no alias

```
void add2(int * a, int * b, int * restrict c) {
      *a += *c;
      // normally, *c must be reloaded because a may
      // point to same address as c, which means that
      // *c could change after “*a += *c” is executed
      *b += *c;
}
```

    - Compiler can thus be more aggressive in optimizing this function

Programming Style

- Imperative Programming
  - Writing commands and statements, changing program state
  - Concerns with *how* a program operates
  - E.g. procedural programming, object-oriented programming
- Declarative Programming
  - Writing expressions and desired result
  - Concerns with *what* a program should achieve
  - E.g. SQL queries, functional programming

```
SELECT firstName, LastName FROM Customers WHERE city=“Toronto”;
```

Course Outline

- Procedural Programming
  - Writing procedures and passing variables to them (APS105)
- Object-Oriented Programming
  - Objects interacting with one another (ECE244)
- Metaprogramming
  - Writing code that generates more code (CSC324)
- Concurrent Programming
  - Multiple tasks overlapping in their executions (ECE344, CSC367)
- Functional Programming
  - Programming in the style of evaluating mathematical functions (CSC324)

Course Objective

- Learn different ways of writing computer programs
  - Law of the instrument
    - “To the man who only has a hammer, everything he encounters begins to look like a nail.” – Abraham Maslow
  - One style may work better than others for a given problem
    - E.g. recursion vs. iteration
- Analyze programming languages and their features
  - Semantic: more expressive power
  - Syntactic: easier to read/write
  - Optimization: improves performance and efficiency

Syntactic Sugar

- Example: Java
  - For Loop

```
String[] fruits = { “Apple”, “Banana”, “Strawberry” };
for (int i = 0; i < fruits.length; i++) {
      system.out.println(fruits[i]);
}
```

  - For-Each Loop

```
for (String f : fruits) {
      system.out.println(f);
}
```

  - Same bytecode generated, but easier to read

Course Outline

- 3 Programming languages
  - Python 3
    - Extremely popular high-level programming language
  - Rust
    - Fairly new systems programming language
    - Focuses on performance and safety
  - C++11
    - A newer version of C++ compared to what's learned in ECE244
- There will be at least one assignment for each language
  - More information on course website

Course Objective

- Learn programming languages *that matters*
  - Popularity – large community of active developers
    - Easy to find help
    - Easy to find libraries or packages
    - Easy to find jobs

Popularity

- By job posting
  - Java: 65,986 jobs
    - Mainly used by web application developers
  - Python: 61,818 jobs
    - Known for high productivity
  - JavaScript: 38,018 jobs
    - The de facto language for modern browsers
  - C++: 36,798 jobs
    - Favored by game developers, large application software, …etc.
  - C#, PHP, Perl…

Source: https://www.codingdojo.com/blog/the-7-most-in-demand-programming-languages-of-2019

Popularity

- By contribution on GitHub
  - JavaScript
  - Java
  - Python
  - PHP
  - C++
  - C#, TypeScript, Shell, C, Ruby, …etc
- Correlates well with popularity by job postings

Source: https://github.blog/2018-11-15-state-of-the-octoverse-top-programming-languages/

Popularity

- By Google search trends
  - JavaScript
  - Python
  - Java
  - Go
    - Google’s programming language
    - Aims to simplify programming in multicore, networking environment
  - Elixir
  - Ruby, Kotlin, TypeScript, Scala, Clojure, ..etc.

Source: https://codeburst.io/10-top-programming-languages-in-2019-for-developers-a2921798d652

Popularity

- By growth in community
  - Kotlin
  - HCL
  - TypeScript
  - PowerShell
  - **Rust**
  - CMake
  - Go
  - Python, Groovy, SQLPL

Source: https://github.blog/2018-11-15-state-of-the-octoverse-top-programming-languages/

Closing Notes

- Immediate TODOs:

- Get a group number and/or a partner ASAP
- Check course website for important dates and grading scheme
- Figure out how to remote access lab machines