

C++: Syllabus & Introduction

Outline

- Course Introduction
- C++ Introduction

Who I am

- Instructor: Keith Perkins
- Office & Office Hours: See Google Meet link on Course webpage and Scholar shell
- Email: keith.perkins@cnu.edu




Who I am

- Instructor: Keith Perkins
- ~~• Office: Luter 201-D~~
- Office Hours: MWF 9-10AM, 2-3PM
TTH 12:15-1PM
- ~~• Office Phone: 594-8425~~
- Email: keith.perkins@cnu.edu
- Collaboration : Piazza





Notes, Lectures, Assignments, Videos ...

- Scholar
 - Lectures
 - Assignments
 - Piazza
 - Course Webpage  Most content here
- Webpage: Note in particular;
 - The Lectures/Readings section
 - You are responsible for everything here
 - The Examples section
 - Please understand these

Assignments


- Read All week 1 readings
- Please install and set up Eclipse CDT
- Please go to projects section of website
 - Complete Project 1 by due date

Syllabus: Prerequisites

- CPSC 250 or equivalent
- Textbook – Any C++ text 
- Suggestions:
 - Absolute C++, Walter Savitch
 - C++ Programming Language, Stroustrup
- References to make you a better programmer
 - Effective C++, Scott Meyers
 - More Effective C++ , Scott Meyers
 - Effective STL, Scott Meyers
 - Effective Modern C++, Scott Meyers 

Syllabus: Major Topics

(Subject to change)

- Week 1 C++ Intro, Market share, Compilation, GIT, Linux introduction
- Week 2 compilation, headers intro, makefiles, Eclipse
- Week 3,4 Headers, functions, Streams, Structs, Enums
- Week 5,6 Standard Library, strings
- Week 7 Standard Library iterators and Lists, Preprocessor directives
- Week 8,9 Pointers, References, Memory
- Week 10 Classes, operators, memory management using RAI 
- Week 11 Exceptions
- Week 12, 13 Inheritance, operator overloading, virtual heierarchy
- Week 14 Registers, Memory, profiling

Syllabus : Evaluation

- 2 Midterm Tests
- 1 Final
- Numerous projects
- See Syllabus for details
- This will be a rigorous course. Please start projects early.

Syllabus: Assignments

- Project 0 – 0 points – IDE
- Project 1 – 100 points – makefile - simple
- Project 2 - 100 points – makefile - harder
- Project 3 - 150 points – File I/O
- Project 4 - 150 points – Modeling a simple system
- Project 5 - 200 points – Static libraries and parsing strings
- Project 6 - 200 points – Polymorphism
- This may change as the semester progresses

Development Environment

- Could use vim, g++, gdb, valgrind, tmux for a command line only dev environment
- Or an IDE, Lots to choose from, Codeblocks, Netbeans, Ms Visual Studio, Eclipse CDT...Clion
- We will use Eclipse CDT

Operating System

- Linux – Ubuntu (18.04)
- Can install yourself or...
- Should be running on the Hunter Creech lab computers.
- Compiler – GNU toolchain
 - G++ version 7.5.0

Development Environments

- Visual Studio 2012

Tools (useful for **advanced debugging**)

- Process Explorer – process relations, kill and find, more powerful than Task Manager
- Dependency Walker – dlls linked etc, old but very good
- Attached Modules (Visual C++, useful to see if incorrect library version is loaded)
- Show includes (Visual C++, what files are being included)

What you will learn

- Standard C++ - to a level of proficiency so you can function professionally, you will not be an expert.
- Some of the C++ syntax
- Coding suggestions and Guidelines to make you a better programmer.
- how to use an IDE, how to use libraries, how to approach and solve programming problems



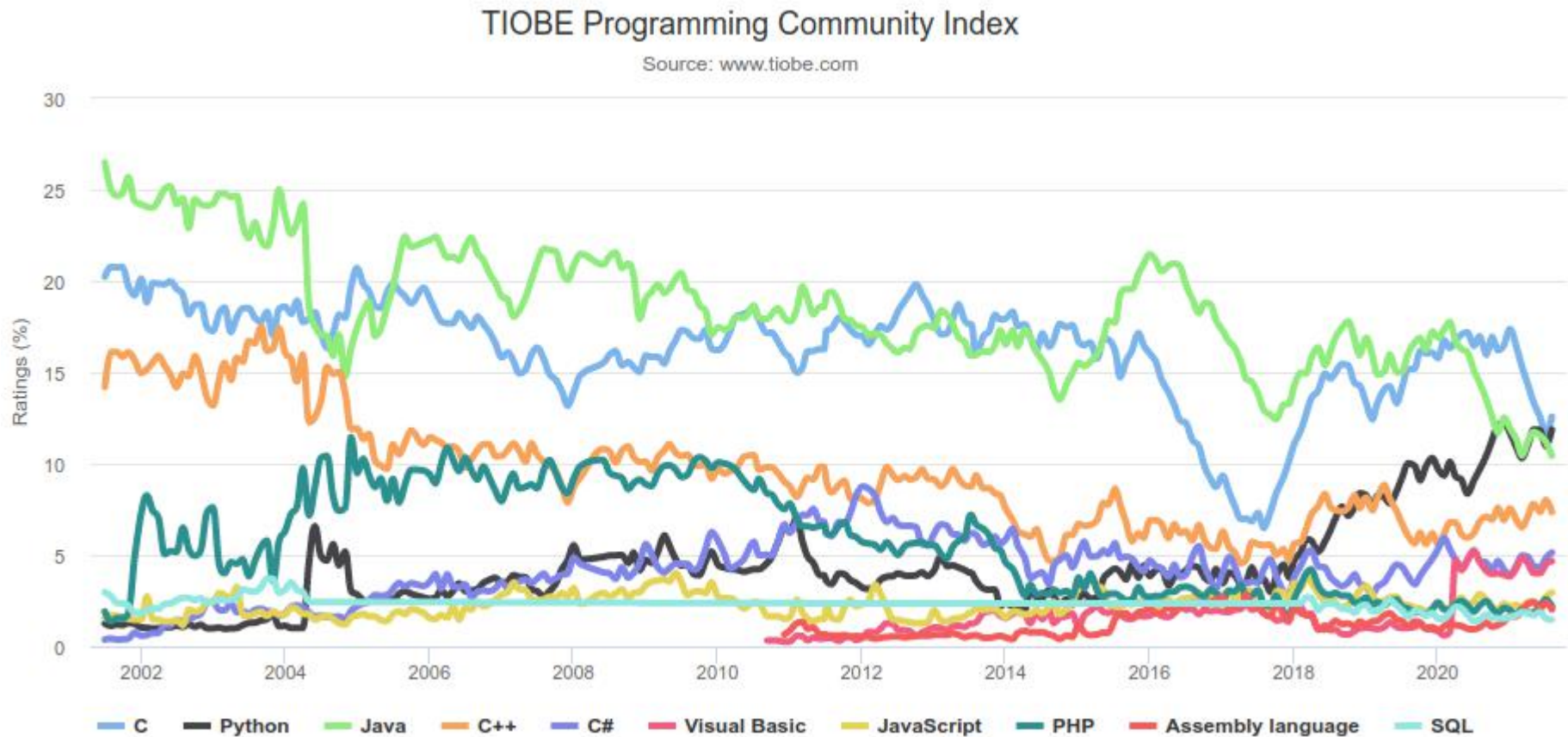
What you will NOT learn

- User Interface (UI), networking– UI is platform dependent, networking is too advanced for intro class (and is MUCH harder in C++ than Java)

Outline

- Course Introduction
- C++ Introduction

C++ Usage

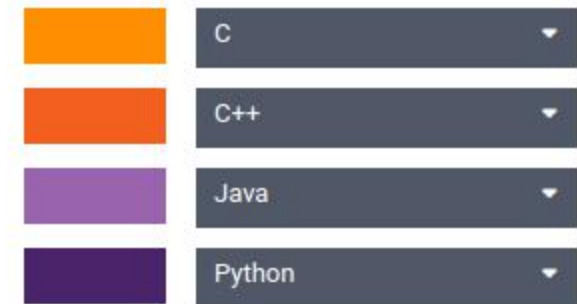
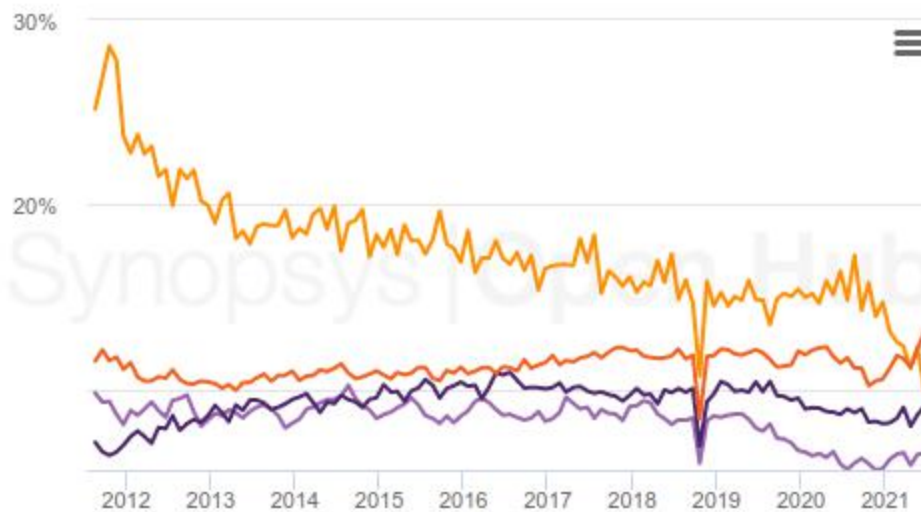


see <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

C++ Usage

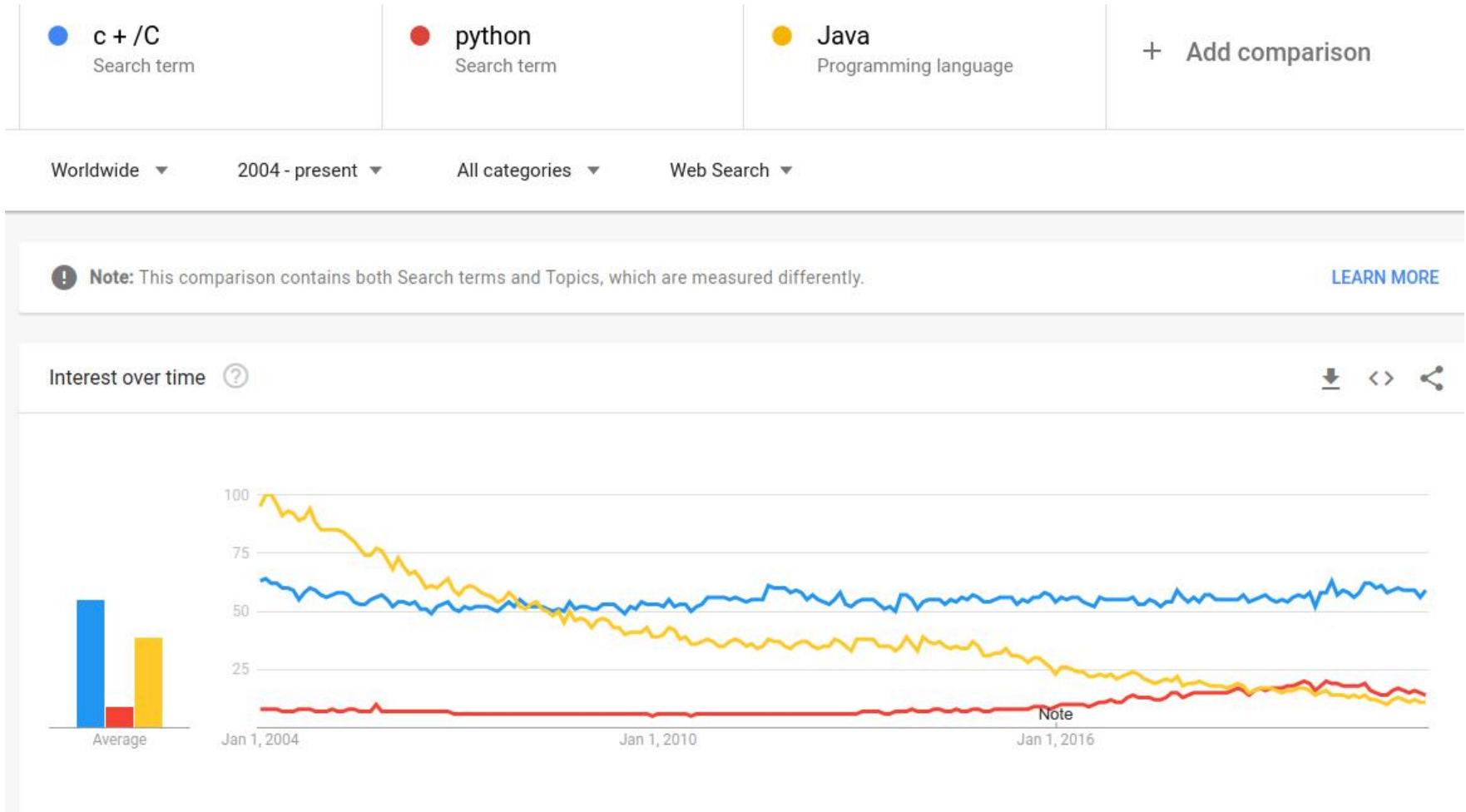
Monthly Commits (Percent of Total)

The lines show the count of monthly commits made by source code developers. Commits including multiple languages are counted once for each language. [More](#)



See <https://www.openhub.net/languages/compare>

C++ Usage



See <https://www.google.com/trends/explore?date=all&q=c%2B%2B%2FC,python,javascript>

C++ ... Why?

- Fast
- You have absolute control over everything
- No need for virtual machine or interpreter
- Elegant when done well
- Only choice for some situations
 - High speed trading
 - Google search
 - Embedded systems
 - Real Time Processing
- Low level control

C++ ... Why not?

- Harder to code than languages that run on a VM (Java, C#)
- No garbage collection, pointers can be (and usually are) a problem
- Must be compiled to target platform, no portable bytecode
- My experience – My Java apps are up and running much faster than my C++ apps.

C++ ... Where is it used?

- Device driver development
- Video Games
- Advanced engines (audio, image processing, etc)
- Telecom
- Embedded software
- Financial - low latency market data feeds
- Google
- Real time video processing

I know Python, why bother?

- Speed
- Software now targets distributed applications
 - Rich user interfaces
 - Cloud storage
 - Mobile Applications
 - Big Data
 - games
- Today, applications require expertise in multiple languages

But... I don't know most of that stuff

- Don't worry, you aren't expected to.
- You learn on the job (while getting paid)