

CS107, Lecture 1

Welcome to CS107!

reading:

[Course Syllabus](#)

Bryant & O'Hallaron, Ch. 1 (skim)

[Honor Code and Collaboration Page](#)

CS107 To-do List

- Introduction
- CS107 Course Policies
- Unix and the Command Line

Guiding Principles For In-Person Class

- We've not recovered from the stress of the past two years. Under normal circumstances, in-person instruction is ideal for learning material.
- We're closer to normal, but we're not there yet. If you prefer to watch live lecture remotely or watch videos of lectures later, I completely understand.
- We'll do everything we can to support you. We have designed the course to provide flexibility.
- We will listen and constantly reevaluate to ensure the class is going as smoothly as possible for everyone.
- Don't be shy asking for accommodations if problems come up. We're very reasonable people and will do whatever we can to help.
- Some prefer to wear masks and social distance even when not required. CS107 staff members will respect your preferences, and we hope you will as well.

Asking Questions

- Feel free to raise your hand at any time with a question
- If you prefer, you can anonymously post your question in the Ed forum thread for each day's lecture
- We'll monitor the thread throughout the lecture for questions



Visit Ed (or access via Canvas):

<https://edstem.org/us/courses/28214/discussion/>

Today's thread:

<https://edstem.org/us/courses/28214/discussion/1820101>

What question best summarizes CS107?

How / why?

CS107: How/Why?

The CS106 series taught you how to solve problems as a programmer. CS107 goes deeper to understand the **how** and **why**:

- **How** is data in our program represented?
- **How** does the heap work?
- **How** does a computer know how to run code?
- **How** does a program map onto the components of computer systems?
- **Why** is my program doing one thing when I expected it to do another?

Understanding computing at this level demystifies how complex systems work and helps us become better software developers.

CS107 and Programming Experience

- CS107 will further develop your programming experience and provide additional mileage with coding.
- CS107 focuses heavily on **debugging** and getting to the root of why something works the way it does.
- For the duration of the quarter, we'll emphasize how to become a better debugger, how to write better code, and how to further refine your software development acumen.

CS107 Learning Goals

The goals for CS107 are for students

to acquire a **fluency** with

- pointers and memory and how to make use of them when writing C code
- an executable's address space and runtime behavior

to acquire **competency** with

- the translation of C to and from assembly code
- the implementation of programs that respect the limits of computer arithmetic
- the ability to identify bottlenecks and improve runtime performance
- the ability to navigate your own Unix development environment
- the ethical frameworks you need to better design and implement software

and to have some **exposure** to

- the basics of computer architecture

Course Overview

1. **Bits and Bytes** - *How can a computer represent integer numbers?*
2. **Characters and C Strings** - *How can a computer represent and manipulate more complex data types like text?*
3. **Pointers, Stack Memory and Heap Memory** – *How can we effectively manage all forms of memory in our programs?*
4. **Generics** - *How can we tap our knowledge of computer memory and data representation to write code that works with any data type?*
5. **Assembly** - *How does a computer compile, interpret, and execute C programs? And what does assembly code look like?*
6. **Heap Allocators** - *How do core memory-allocation operations like `malloc` and `free` work? Are the built-in versions always good enough?*

Teaching Team



Jerry Cain biography (jerry@cs.stanford.edu):

- Chemistry undergrad MIT, originally a chemistry Ph.D. student here, switched to CS in 1994
- Taught CS107 for the first time in Autumn 1999, have taught it 25 times now!

Companion Class: CS107A

- **CS107A** is an extra 1-unit "ACE" section with additional course support, practice and instruction.
- Meets for an additional weekly section and has additional review sessions
- Admission by application: see course website for details: **cs107a.stanford.edu**



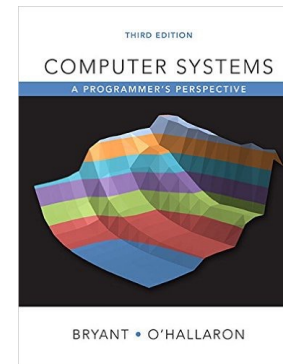
Frankie Cerkenik

Plan For Today

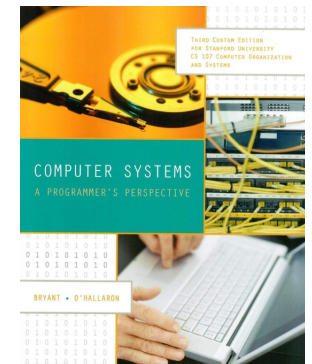
- Introduction
- **CS107 Course Policies**
- Unix and the Command Line

Textbook(s)

- *Computer Systems: A Programmer's Perspective* by Bryant & O'Hallaron, **3rd Edition**
 - **3rd edition matters** – important updates to content
 - Stanford Library has generously scanned **all** readings for CS107 under "fair use" (private study, scholarship, research). [[Canvas -> Files](#)]. Please do not distribute.
 - If you want more context, you may want to purchase a full copy
- A C programming reference of your choice
 - *The C Programming Language* by Kernighan and Ritchie (free link on course website Resources page)
 - Other C programming books, websites, or reference sheets



Full textbook



CS107 full chapters




canvas

CS107-specific readings

The textbook (and C programming references) are **excellent** resources for the course, especially post-midterm!

Course Structure

- Lectures: understand concepts, see demos
- Labs: learn tools, study code, discuss with peers  Great preview of homework!
- Assignments: build programming skills, synthesize lecture/lab content

	Monday	Wednesday	Friday
Week N		Lecture: part A	Lecture: part B
Week N + 1	Lecture: part C		

CS107 labs will meet at various times on Wednesdays and Thursday to exercise prior week's material

- **assign0**: due next Wednesday (covers today's lecture and part of Wednesday's)

Grading

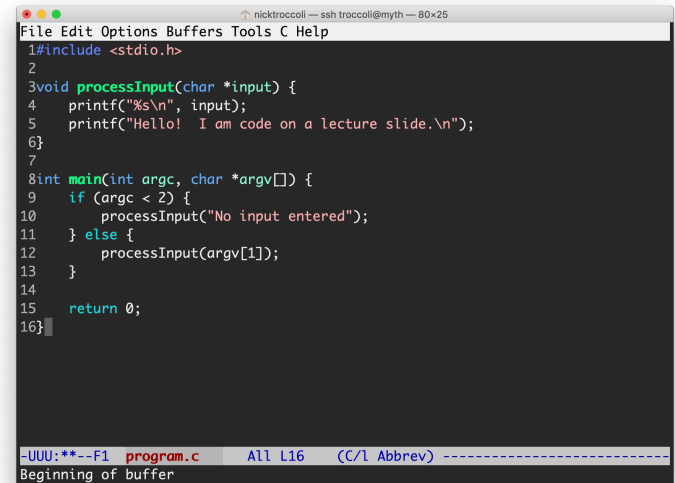
*****	55%	Assignments
**	15%	Lab Participation
*	10%	Midterm Exam
**	20%	Final Exam

Grading

*****	55%	Assignments
**	15%	Lab Participation
*	10%	Midterm Exam
**	20%	Final Exam

Assignments

- 7 programming assignments completed individually using **Unix command line tools**
 - Free software, already installed on Myth machines / available on course website
 - We supply starter projects for each assignment
- Graded on **functionality** (behavior) and **style** (elegance)
 - Functionality graded using *automated tools*, given as point score – no TA review
 - Style graded via TA code review, with *occasional automated tests*, given as bucket score
 - Grades published via email and CS107 website



```
File Edit Options Buffers Tools C Help
1#include <stdio.h>
2
3void processInput(char *input) {
4    printf("%s\n", input);
5    printf("Hello! I am code on a lecture slide.\n");
6}
7
8int main(int argc, char *argv[]) {
9    if (argc < 2) {
10        processInput("No input entered");
11    } else {
12        processInput(argv[1]);
13    }
14
15    return 0;
16}
-UUU:**--F1 program.c All L16 (C/l Abbrev) -----
Beginning of buffer
```

The Style Bucket System

+	An outstanding job: could be used as example code on good style.
ok	A good job: solid effort, but also opportunities for improvement.
-	Conveys some effort and understanding but has larger problems that would need to be addressed before checking into a professional code base.
--	Suffers from many issues and represents minimally passing work.
0	No work submitted, or barely any changes from the starter assignment.

Assignment Late Policy

- **Start out with 5 "free late days"**: each late day allows you to submit an assignment up to 24 additional hours late without penalty. (No late days permitted for the first or last assignments)
- **Hard deadline 48 hours** after original due date
- Penalty per day after late days are exhausted (1 day: 90% cap; 2 days: 80% cap)
- Late days are "pre-granted extensions" – additional extensions for exceptional circumstances must be approved by the **instructor**.



Question Break

What questions do you have about the overall course goals,
textbook or assignments?

Grading

*****	55%	Assignments
**	15%	Lab Participation
*	10%	Midterm Exam
**	20%	Final Exam

Lab Sections

- Weekly 1-hour 30-minute in-person labs led by a CA, starting *next* week, offered on Wednesdays and Thursdays.
- Hands-on practice in small groups with lecture material and course concepts.
- Graded on attendance + participation
- SCPD students complete work remotely (more info in [SCPD Handout](#))
- Lab preference submissions open **Wednesday 9/28 at 5PM PST** and **are not first-come first-serve**. You may submit your preferences anytime until **Sunday 10/2 at 5PM PST**. Sign up on the course website.

Grading

*****	55%	Assignments
**	15%	Lab Participation
*	10%	Midterm Exam
**	20%	Final Exam

Exams

- **Midterm exam** – Tuesday, November 1, 7:00-9:00PM outside of class
 - Contact the course staff by 11:59PM on Wednesday, October 26 if you have an academic or University conflict with this time, and absolutely cannot make the regularly scheduled midterm.
- **Final exam** – Monday, December 12, 3:30PM-6:30PM
 - If and only if you hold a conflict with this time because of a competing final exam—that is, you're taking two MWF 10:30am classes—then you can take the final on the same day, from 12:15 – 3:15pm.
- Both exams are closed-book, closed-notes, although you can bring one double-sided page of notes. You will also be provided with a syntax reference sheet.
- SCPD students have 24hr window during which to take the exams.
- Exams are administered electronically.

Grading

*****	55%	Assignments
**	15%	Lab Participation
*	10%	Midterm Exam
**	20%	Final Exam

Read our full course policies document:
<https://cs107.stanford.edu/syllabus.html>



Question Break

What questions do you have about labs or exams?

Getting Help

- Post on the **Discussion Forum**
 - Online discussion forum for students; post questions, answer other students' questions
 - Best for course material discussions, course policy questions, short debugging questions or general assignment questions. Don't post assignment code under any circumstances.
- Visit **Helper Hours**
 - Chat about course topics or just hang out
 - Sign up in a queue for 1:1 TA help; schedule posted on course website by Wednesday.
 - Mix of in-person and online helper hours
 - Best for **group work, coding/debugging questions (with TAs only!) or longer course material discussions**
- **Email the Course Staff**
 - cs107-aut22-staff@lists.stanford.edu or individual graders/instructor
 - Best for **private matters** (e.g. grading questions, OAE accommodations).

Stanford Honor Code

- The **Honor Code** is an undertaking of the students, individually and collectively:
 - that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;
 - that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.
- The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.
- While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

see also: <http://honorcode.stanford.edu/>

It is your responsibility to ensure you have read and are familiar with the honor code guidelines posted on the main page of the CS107 course website. Please read them and come talk to us if you have any questions or concerns.

Honor Code and CS107

- Please help us ensure academic integrity:
 - Indicate any assistance received on HW (books, friends, etc.).
 - Do not look at other people's solution code or answers
 - Do not give your solutions to others or post them on the web or our Ed forum.
 - Report any inappropriate activity you see performed by others.
- Assignments are checked regularly for similarity with help of software tools.
- If you need help, please contact us and we will help you.
 - We do not want you to feel any pressure to violate the Honor Code in order to succeed in this course.
 - If you realize that you have made a mistake, you may retract your submission to any assignment at any time, no questions asked, up to the start of the final exam.

<https://cs107.stanford.edu/collaboration>



Question Break

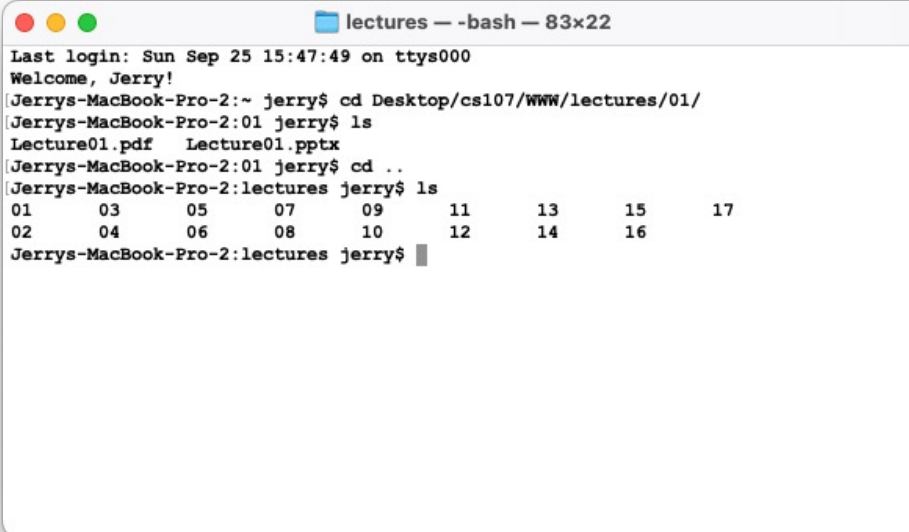
What questions do you have about course support or the honor code?

Plan For Today

- Introduction
- CS107 Course Policies
- **Unix and the Command Line**

What is Unix?

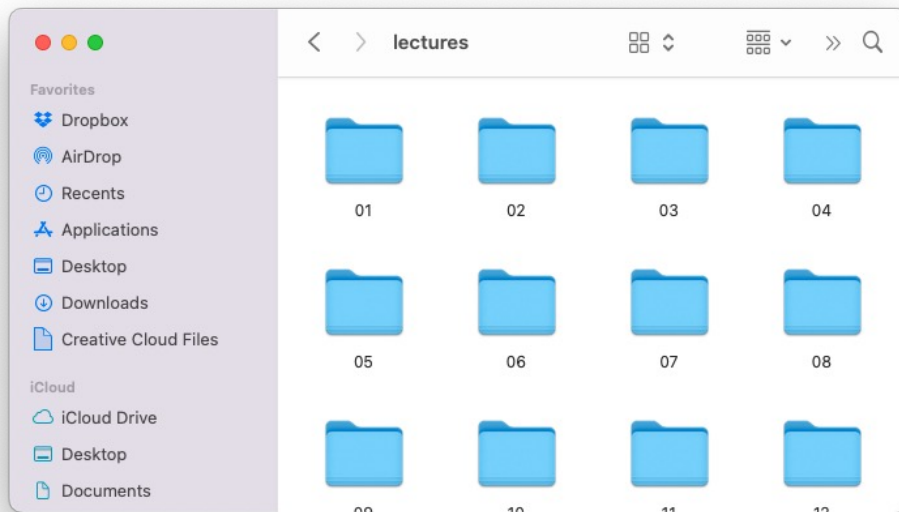
- **Unix**: a set of standards and tools commonly used in software development.
 - **macOS** and **Linux** are operating systems built on top of Unix
- You can navigate a Unix system using the **command line** (“terminal”)
- Every Unix system works with the same tools and commands



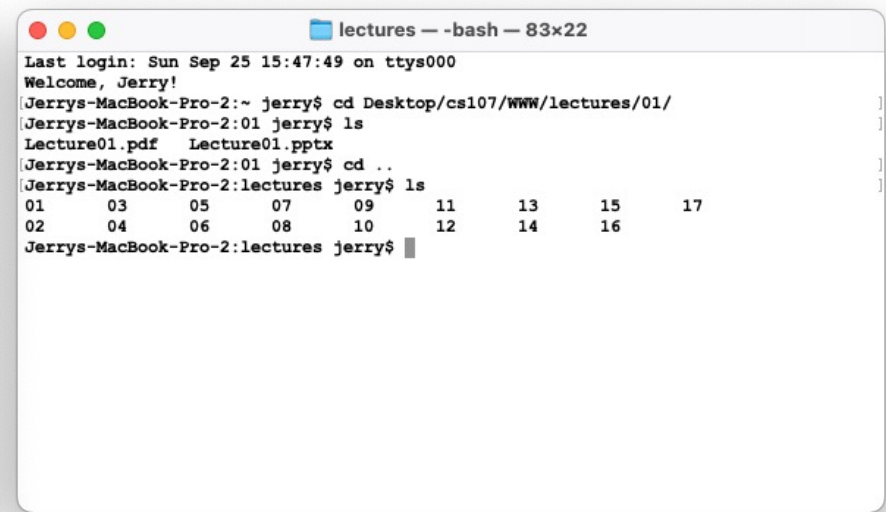
```
lectures --bash -- 83x22
Last login: Sun Sep 25 15:47:49 on ttys000
Welcome, Jerry!
[Jerrys-MacBook-Pro-2:~ jerry$ cd Desktop/cs107/www/lectures/01/
[Jerrys-MacBook-Pro-2:01 jerry$ ls
Lecture01.pdf  Lecture01.pptx
[Jerrys-MacBook-Pro-2:01 jerry$ cd ..
[Jerrys-MacBook-Pro-2:lectures jerry$ ls
01      03      05      07      09      11      13      15      17
02      04      06      08      10      12      14      16
[Jerrys-MacBook-Pro-2:lectures jerry$
```


What is the Command Line?

- The **command-line** is a text-based interface (i.e., **terminal** interface) to navigate a computer, instead of a Graphical User Interface (GUI).



Graphical User Interface

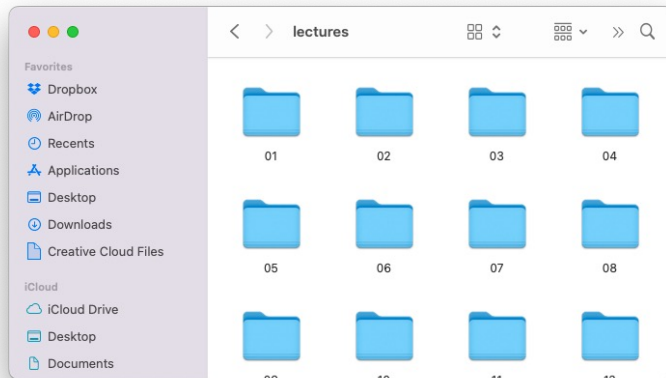


Text-based interface

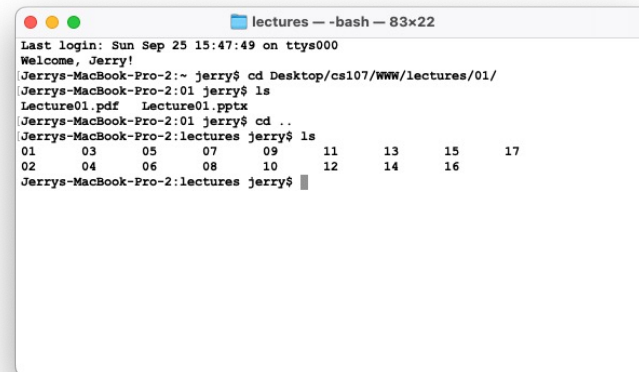
Command Line Vs. GUI

Just like a GUI file explorer interface, a terminal interface:

- shows you a **specific place** on your computer at any given time.
- lets you go **into folders** and **out of folders**.
- lets you **create new** files and **edit** files.
- lets you **execute programs**.



Graphical User Interface

A screenshot of a terminal window titled 'lectures -- bash -- 83x22'. The terminal shows the following output:

```
Last login: Sun Sep 25 15:47:49 on ttys000
Welcome, Jerry!
Jerry-MacBook-Pro-2:~ jerry$ cd Desktop/cs107/WWW/lectures/01/
Jerry-MacBook-Pro-2:01 jerry$ ls
Lecture01.pdf  Lecture01.pptx
Jerry-MacBook-Pro-2:01 jerry$ cd ..
Jerry-MacBook-Pro-2:lectures jerry$ ls
01  03  05  07  09  11  13  15  17
02  04  06  08  10  12  14  16
Jerry-MacBook-Pro-2:lectures jerry$
```

Command-line interface

Why Use Unix / the Command Line?

- You can navigate almost any device using the same tools and commands:
 - Servers
 - Laptops and desktops
 - Embedded devices (Raspberry Pi, etc.)
 - Mobile Devices (Android, etc.)
- Used frequently by software engineers:
 - **Web development:** running servers and web tools on servers
 - **Machine learning:** processing data on servers, running algorithms
 - **Systems:** writing operating systems, networking code and embedded software
 - **Mobile Development:** running tools, managing libraries
 - And more...
- We'll use Unix and the command line to implement and execute our programs.

Unix Commands To Try

- **cd** – change directories (..)
- **ls** – list directory contents
- **mkdir** – make directory
- **emacs** – open text editor
- **rm** – remove file or folder
- **man** – view manual pages

See the course website for more commands and a complete reference.

Demo: Using Unix and the Command Line



Get up and running with our guide:

<http://cs107.stanford.edu/resources/getting-started.html>

Unix Commands Recap

- **cd** – change directories (..)
- **ls** – list directory contents
- **mkdir** – make directory
- **emacs** – open text editor
- **rm** – remove file or folder
- **man** – view manual pages

See the course website for more commands and a complete reference.

Learning Unix and the Command Line

- Using Unix and the command line can be intimidating at first:
 - It looks retro!
 - How do I know what to type?
- It's like learning a new language:
 - At first, you may have to constantly look things up (**resources** on course website!)
 - It's important to spend as much time as possible (during labs and assignments) building muscle memory with the tools



Question Break

Get up and running with our guide:

<http://cs107.stanford.edu/resources/getting-started.html>

The C Language

C was created around 1970 to make implementing Unix and Unix built-ins easier.

- Part of the C/C++/Java family of languages (C++ and Java were created later)
- Design principles:
 - Small, simple abstractions of hardware
 - Minimalist aesthetic
 - Prioritizes efficiency and simplicity over safety and high-level abstractions

C vs. C++ and Java

They all share:

- Syntax
- Basic data types
- Arithmetic, relational, and logical operators

C limitations:

- No advanced features like operator overloading, default arguments, pass by reference, classes and objects, ADTs, etc.
- No extensive libraries (no graphics, networking, etc.) – Small language footprint means not much to learn 😊
- Weak compiler and almost no runtime checks (this may cause security vulnerabilities!)

Programming Language Philosophies

C is procedural: you write functions, rather than define new variable types with classes and call methods on objects. **C is small, fast and efficient.**

C++ is procedural, with objects: you write functions, and define new variable types with classes, and call methods on objects.

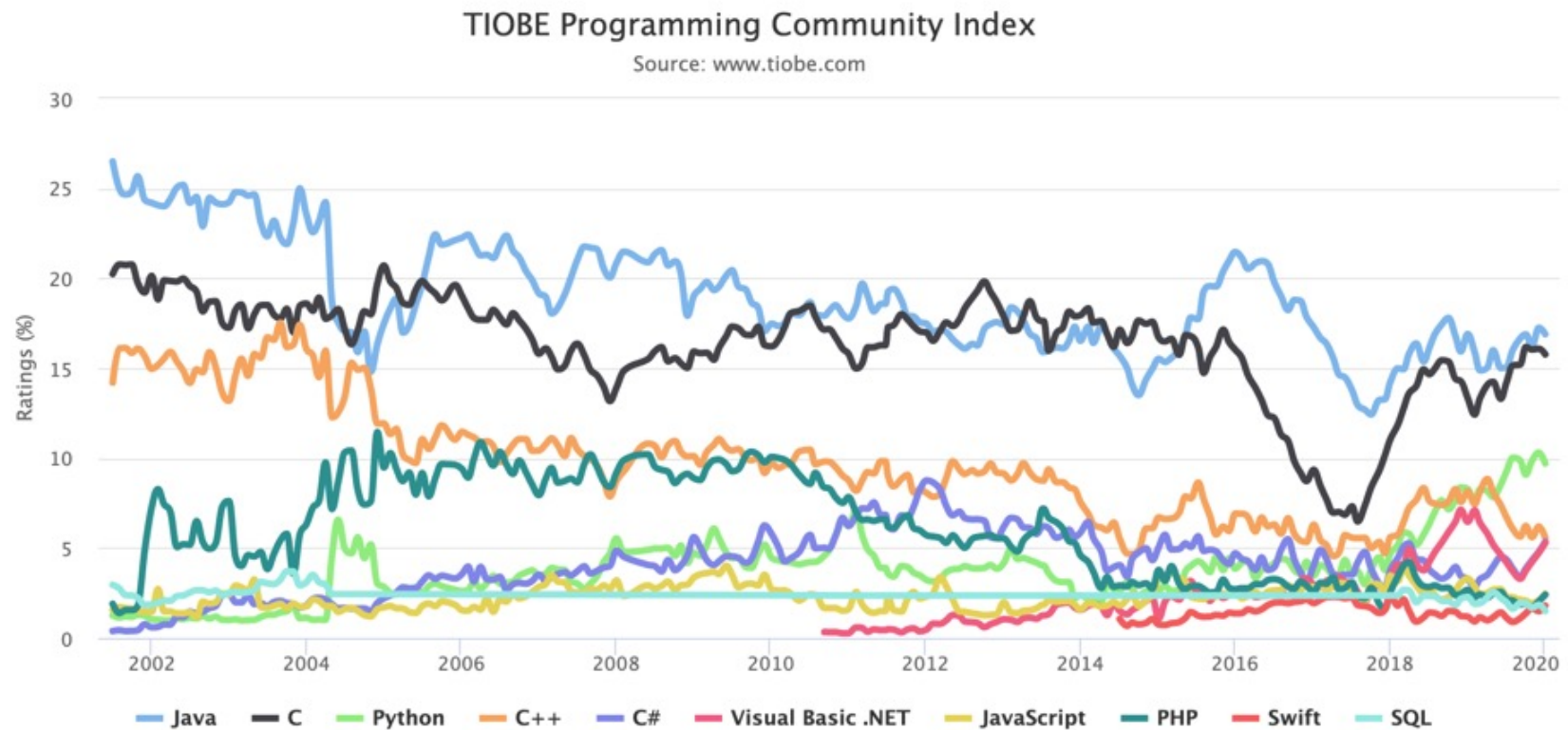
Python is also procedural, but dynamically typed: you still write functions and call methods on objects, but the development process is very different.

Java is object-oriented: virtually everything is an object, and everything you write needs to conform to the object-oriented design pattern.

Why C?

- Many tools (and even other languages, like Python!) are built with C.
- C is the language of choice for fast, highly efficient programs.
- C is popular for systems programming (operating systems, networking, etc.)
- C lets you work at a lower level to manipulate and understand the underlying system.
- Modern alternatives to C are emerging, but they're more complicated

Programming Language Popularity



<https://www.tiobe.com/tiobe-index/>

Assign0

Assignment 0 (Intro to Unix and C) is due in a week from Wednesday on **10/5 at 11:59PM PDT**.

There are **5** parts to the assignment, which is meant to get you comfortable using the command line, and editing/compiling/running C programs:

- Visit the website resources to become familiar with different Unix commands
- **Clone** the assign0 starter project
- **Answer** several questions in `readme.txt`
- **Compile** a provided C program and **modify** it
- **Submit** the assignment