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
- Getting Started With C

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/46162/discussion/

Today's thread:
https://edstem.org/us/courses/46162/discussion/3504473

Guiding Principles For In-Person Class

- Under normal circumstances, in-person instruction is ideal for learning material. We're close to normal, but some pre-pandemic practices and expectations are clearly not returning.
- Prefer to watch lecture remotely or even later on? We completely understand.
- We'll do virtually anything we can to support you. We've designed CS107 to maximize flexibility without sacrificing rigor and post conditions.
- We 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 arise. We're reasonable people and will do whatever we can to help.
- Some still prefer masks and social distance even though it's not required. CS107 staff members respect all preferences, and we hope do as well.

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 so we can understand the **how** and **why**:

- **How** is program data represented on the hardware?
- **How** does the heap work and **how** is it implemented?
- **How** does a computer know how to run code?
- **How** does an executable map onto the components of computer systems?
- **Why** is my program doing one thing when I expected it to do something else?

Understanding computing at this level demystifies how complex systems work and helps us become better software developers. We're rewarded by being curious about why a broken system is behaving the way it does.

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

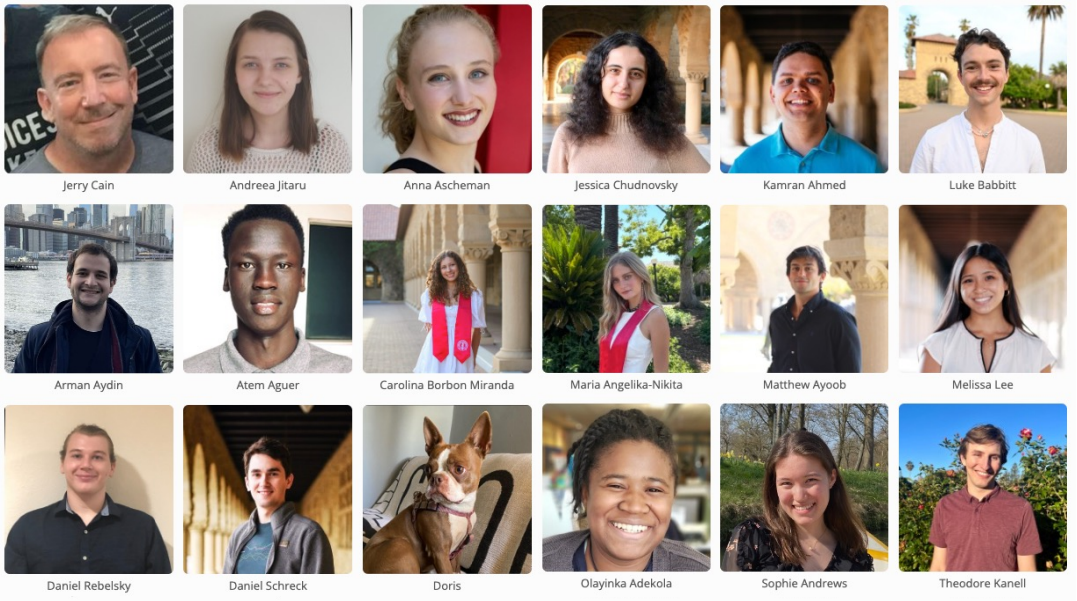
and, to gain some **exposure** to

- 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: CS107ACE

- **CS107ACE** is an extra 1-unit "ACE" section providing additional support, practice and instruction.
- Class meets on Tuesdays and Thursdays from 10:30 – 11:20am in Lathrop 180.
- Admission by application, and details provided during tomorrow's class, which is open to all interested in applying.

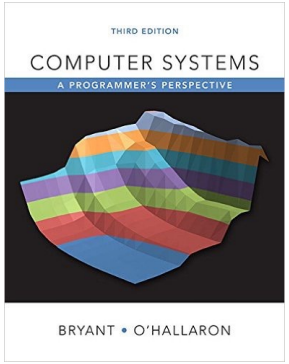
Eric Bear

Plan For Today

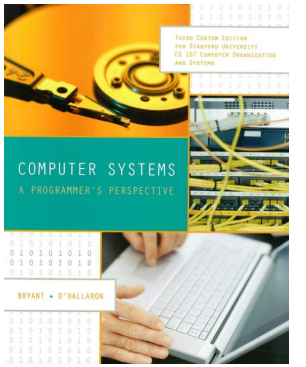
- Introduction
- CS107 Course Policies
- Unix and the Command Line
- Getting Started With C

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 the prior week's material

- **Extra Lecture**: Tuesday, October 3, 4:30 – 5:20 pm in Skilling Auditorium
- **assign0**: due next Wednesday (covers today's and Friday's lectures)

Grading

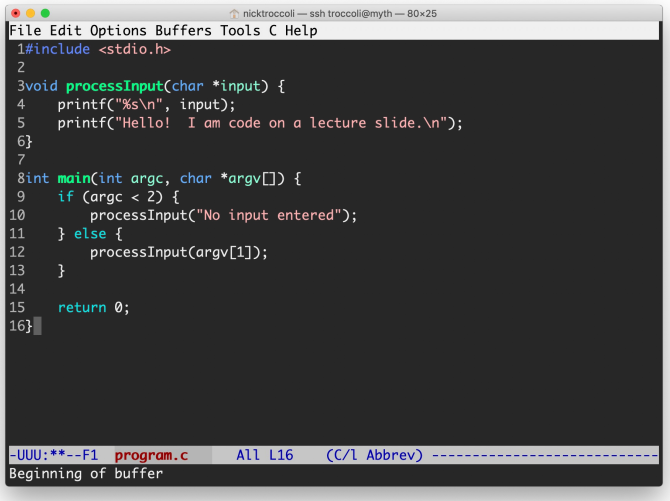
*****	55%	Assignments
**	15%	Lab Participation
*	10%	Midterm Exam
**	20%	Final Exam

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 mostly 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



The Style Bucket System

+	An outstanding job: could be used as example code on good style.
ok	A good job: solid effort, but 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.
- **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 directly submitted to and approved by Jerry.

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 9/27—that's later today—at 5PM PST** and **are not first-come first-serve**, so take your time finalizing your schedule before submitting. You may submit your preferences anytime until **Sunday 10/1 at 5PM PST**. Sign up on the course website.

Grading

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

Exams

- **Midterm exam** – Thursday, November 2, 7:00-9:00PM outside of class
  Thursday, November 2, 3:30-5:30PM if 7:00PM can't work
  - Contact the course staff by 11:59PM on Friday, October 27th if you have an academic or University conflict with both times and absolutely cannot make either
- **Final exam** – Monday, December 11, 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 7:00 – 10:00pm
- 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 as old-school, pencil-and-paper.

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 student 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 **Office Hours**
  - Chat about course topics or just hang out
  - Sign up in a queue for 1:1 CA help; schedule will be posted on course website tomorrow
  - Mix of in-person and online helper hours
  - Best for **group work**, **coding/debugging questions (only with CAs, though)** or **longer discussions about course material**
- **Email** the Course Staff
  - cs107-aut2324-staff@lists.stanford.edu or individual CAs/instructor
  - Best for **private matters** (e.g., grading questions).

Updated Stanford Honor Code

- The **Honor Code** is an undertaking of the Stanford academic community, individually and collectively. Its purpose is to uphold a culture of academic honesty.

  - Students will support this culture of academic honesty by neither giving nor accepting unpermitted academic aid in any work that serves as a component of grading or evaluation, including assignments, examinations, and research.
  - Instructors will support this culture of academic honesty by providing clear guidance, both in their course syllabi and in response to student questions, on what constitutes permitted and unpermitted aid. Instructors will also not take unusual or unreasonable precautions to prevent academic dishonesty.
  - Students and instructors will also cultivate an environment conducive to academic integrity. While instructors alone set academic requirements, the Honor Code is a community undertaking that requires students and instructors to work together to ensure conditions that support academic integrity.

above drawn verbatim from: https://communitystandards.stanford.edu/policies-guidance/honor-code

It is your responsibility to ensure you have read and are familiar with the honor code guidelines shared via the main page of the CS107 course website. Please read them and come speak with 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 regularly examined with help of software tools.
- If you need help, please contact us and we will do what we can.
  - We do not want you to feel any pressure to violate the Honor Code because you need to comfortably arrive at an outcome that you're happy with and think will impress others.
  - If you realize you've made a mistake, you may retract one or more assignment submissions at any time, no questions asked, up to the time the final exam begins.

https://cs107.stanford.edu/collaboration

Question Break

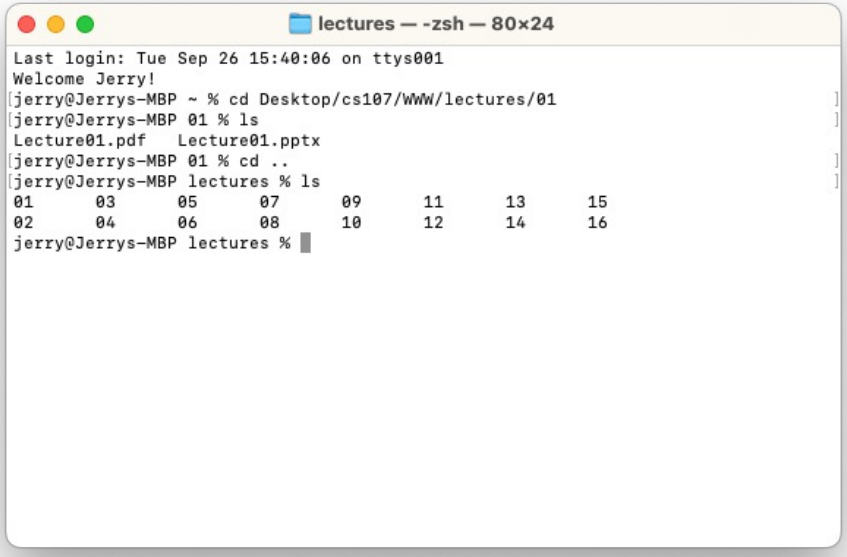
What questions do you have about course support or the honor code?

Plan For Today

- Introduction
- CS107 Course Policies
- Unix and the Command Line
- Getting Started With C

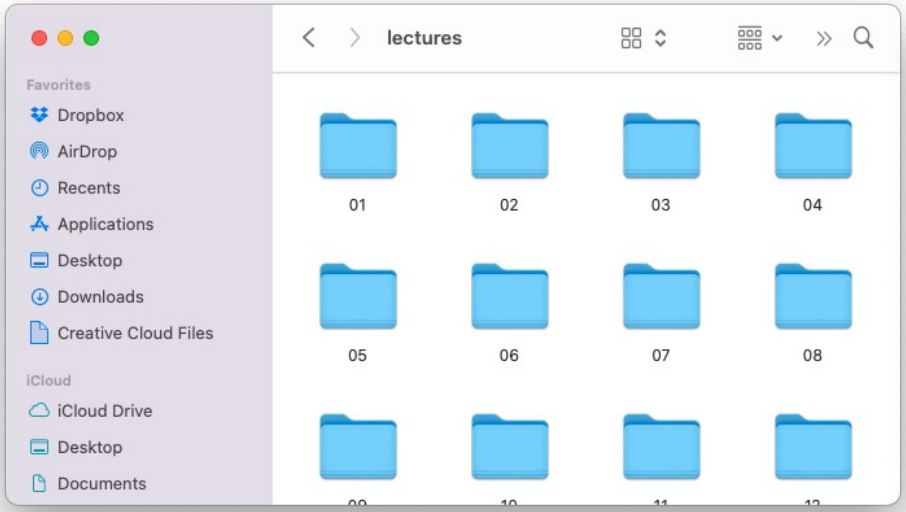
What is Unix?

- **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

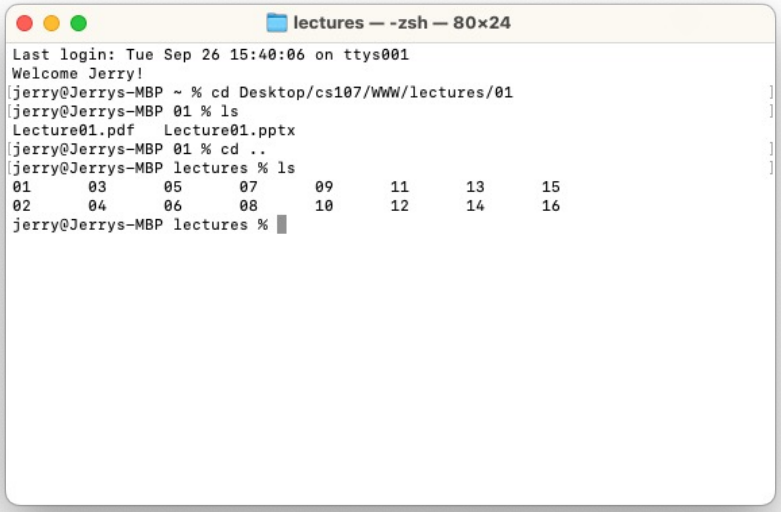


What is the Command Line?

- 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

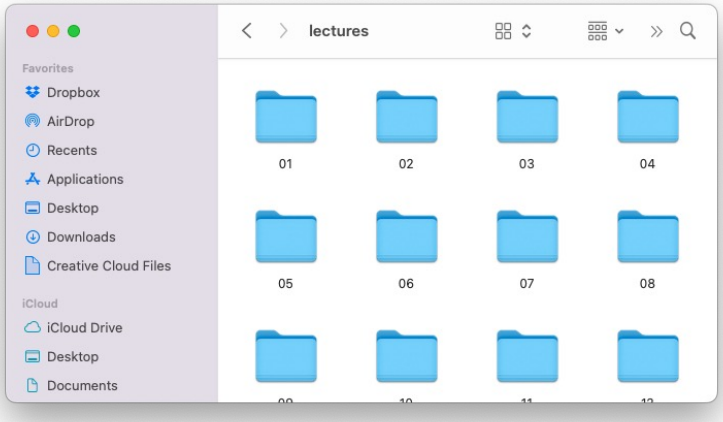


Command-line interface

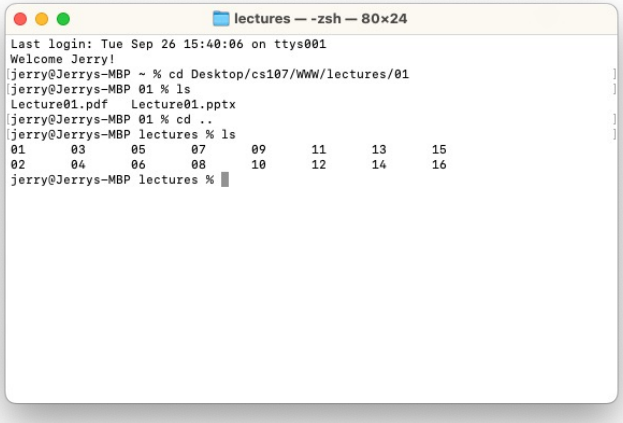
Command Line Vs. GUI

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



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/getting-started.html

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 repeatedly 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/getting-started.html

Plan For Today

- Introduction
- CS107 Course Policies
- Unix and the Command Line
- Getting Started With C

The C Language

C was created around 1970 to make writing Unix and Unix tools easier.

- Part of the C/C++/Java family of languages (C++ and Java were created later)
- Design principles:
  - Small, simple abstractions over 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 virtually zero runtime checks. This is a double-edged sword.

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 still write functions, and define new variable types with classes, and call methods on objects.

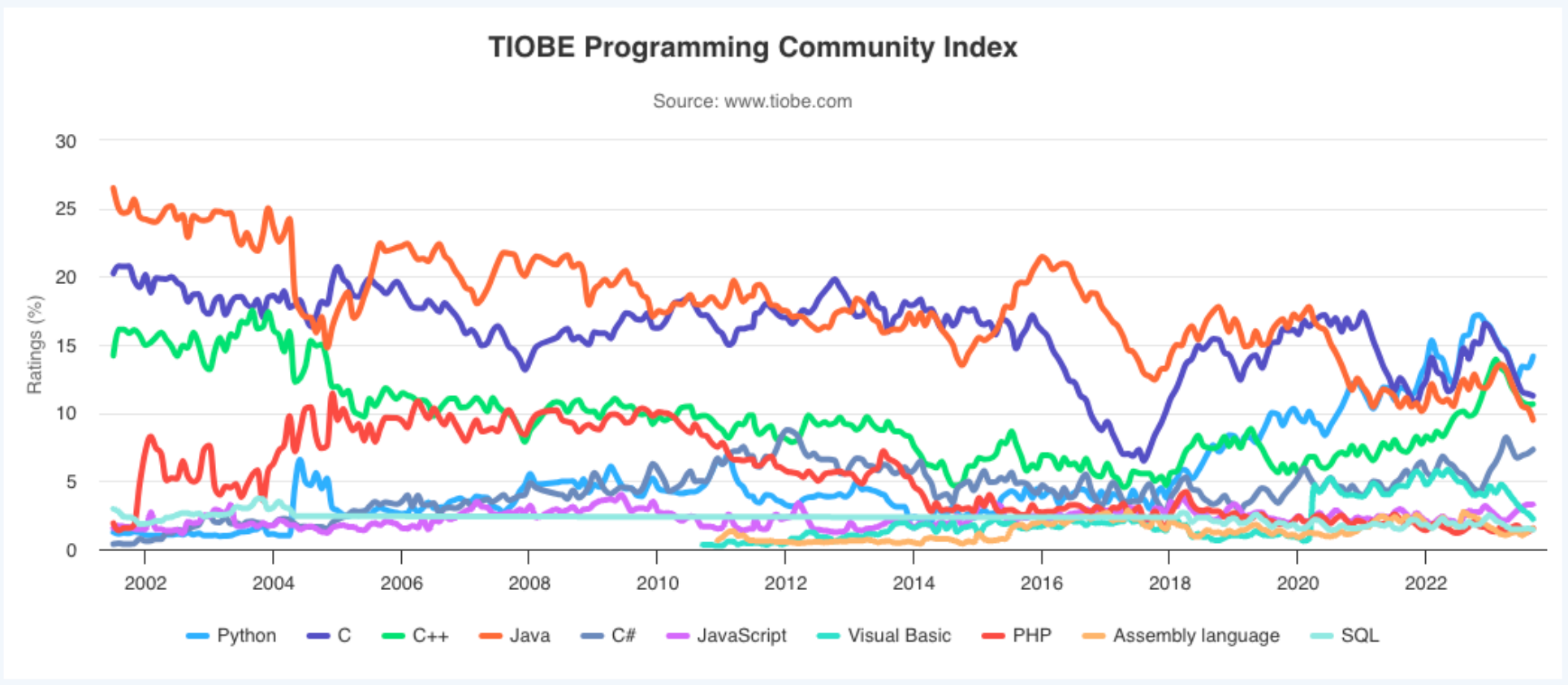
Python is also procedural, but dynamically typed: you still write functions and call methods on objects, but traditionally omit data types when coding. The development process is very different.

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 written using C.
- C is a 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 understand and even manipulate the underlying system.
- Modern alternatives to C are emerging (e.g., Rust), but they're more complicated and not quite ready for those new to systems programming

Programming Language Popularity



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

Our First C Program

```c
/*
 * hello.c
 * This program prints a welcome message
 * to the user.
 */
#include <stdio.h>   // for printf

int main(int argc, char *argv[]) {
    printf("Hello, world!\n");
    return 0;
}
```

Our First C Program

```c
/*
 * hello.c
 * This program prints a welcome message
 * to the user.
 */
#include <stdio.h>   // for printf

int main(int argc, char *argv[]) {
    printf("Hello, world!\n");
    return 0;
}
```

Program comments
You can write block or inline comments.

Our First C Program

```c
/*
 * hello.c
 * This program prints a welcome message
 * to the user.
 */
#include <stdio.h>   // for printf

int main(int argc, char *argv[]) {
    printf("Hello, world!\n");
    return 0;
}
```

Import statements
C libraries are written with angle brackets.
Local libraries use quotes instead:
`#include "wordle-utils.h"`

Our First C Program

```c
/*
 * hello.c
 * This program prints a welcome message
 * to the user.
 */
#include <stdio.h>   // for printf

int main(int argc, char *argv[]) {
    printf("Hello, world!\n");
    return 0;
}
```

Main function – entry point for the program
Should always return a small integer (0 = success)

Our First C Program

```
/*
 * hello.c
 * This program prints a welcome message
 * to the user.
 */
#include <stdio.h>   // for printf

int main(int argc, char *argv[]) {
    printf("Hello, world!\n");
    return 0;
}
```

Main parameters – **main** takes two parameters, both relating to the *command line arguments* used to execute the program.

argc is the *number* of arguments in **argv**
argv is an *array of arguments* ***(char * is C string)***

Our First C Program

```
/*
 * hello.c
 * This program prints a welcome message
 * to the user.
 */
#include <stdio.h>   // for printf

int main(int argc, char *argv[]) {
    printf("Hello, world!\n");
    return 0;
}
```

printf – prints output to the screen

Console Output: printf

`printf(`***`text, arg1, arg2, arg3,...`***`);`

`printf` makes it easy to print out the values of variables or expressions.

If you include *placeholders* in your printed text, `printf` will replace each placeholder *in order* with the values of the parameters passed after the text.

`%s (string)` `%d (integer)` `%f (double)`

```
// Example
char *classPrefix = "CS";
int classNumber = 107;
printf("You are in %s%d", classPrefix, classNumber); // You are in CS107
```

Familiar Syntax

```
int x = 42 + 7 * -5;                  // variables, types
double pi = 3.14159;
char c = 'Q';                         /* two comment styles */

for (int i = 0; i < 10; i++) {        // for loops
    if (i % 2 == 0) {                 // if statements
        x += i;
    }
}

while (x > 0 && c == 'Q' || b) {      // while loops, logic
    x = x / 2;
    if (x == 42) {
        return 0;
    }
}

binky(x, 17, c);                      // function call
```

Boolean Variables

To declare Booleans, (e.g. `bool b = ____`), you must include `stdbool.h`:

```
#include <stdio.h>     // for printf
#include <stdbool.h>   // for bool

int main(int argc, char *argv[]) {
    bool x = argc > 2 && argv[argc - 1][0] != 'A';
    if (x) {
        printf("Hello, world!\n");
    } else {
        printf("Howdy, world!\n");
    }
    return 0;
}
```

Boolean Expressions

C treats a nonzero value as true, and a zero value as false:

```
#include <stdio.h>

int main(int argc, char *argv[]) {
    int x = 5;
    if (x) {   // true
       printf("Hello, world!\n");
    } else {
       printf("Howdy, world!\n");
    }
    return 0;
}
```

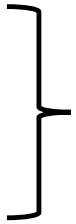
Question Break

Writing, Debugging and Compiling

We will use:

- the **emacs** text editor to write our C programs
- the **make** tool to compile our C programs
- the **gdb** debugger to debug our programs
- the **valgrind** tools to debug memory errors and measure program efficiency

Now



Next week

Working On C Programs

- **ssh** – remotely log in to **`myth`** computers
- **emacs** – text editor to write and edit C programs
  - Use the mouse to position cursor, scroll, and highlight text
  - Ctl-x Ctl-s to save, Ctl-x Ctl-c to quit
- **make** – compile program using provided Makefile
- **./myprogram** – run executable program (optionally with arguments)
- **make clean** – remove executables and other compiler files
- Lecture code is accessible at **/afs/ir/class/cs107/lecture-code/lect[N]**
  - Make your own copy: **cp -r /afs/ir/class/cs107/lecture-code/lect[N] lect[N]**
  - See the website for even more commands, and a complete reference.

Demo: Compiling And Running A C Program

Get up and running with our guide:
http://cs107.stanford.edu/getting-started.html

Question Break

Get up and running with our guide:
http://cs107.stanford.edu/getting-started.html

Assign0

Assignment 0 (Intro to Unix and C) is due in one week on **Wed. 10/4 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

Recap

- CS107 is a programming class in C that teaches you about what goes on under the hood of programming languages and software.
- We'll use Unix and command line tools to write, debug and run our programs.
- Please visit the course website, cs107.stanford.edu, where you can read the General Information page, information about the Honor Code in CS107, and more about CS107 course policies and logistics.

We're looking forward to an awesome quarter!

Preview: Next Time

- Make sure to reboot Boeing Dreamliners every 248 days
- Comair/Delta airline had to cancel thousands of flights days before Christmas
- Many operating systems may have issues storing timestamp values beginning on Jan 19, 2038
- Reported vulnerability CVE-2019-3857 in **`libssh2`** may allow a hacker to remotely execute code

Next time: *How can a computer represent integer numbers? What are the limitations?*