Lecture 1: Welcome to CS106L!

CS106L, Winter 2025

Today’s Agenda

- Introductions!
- The Pitch 🦈 🦈
- Course Logistics

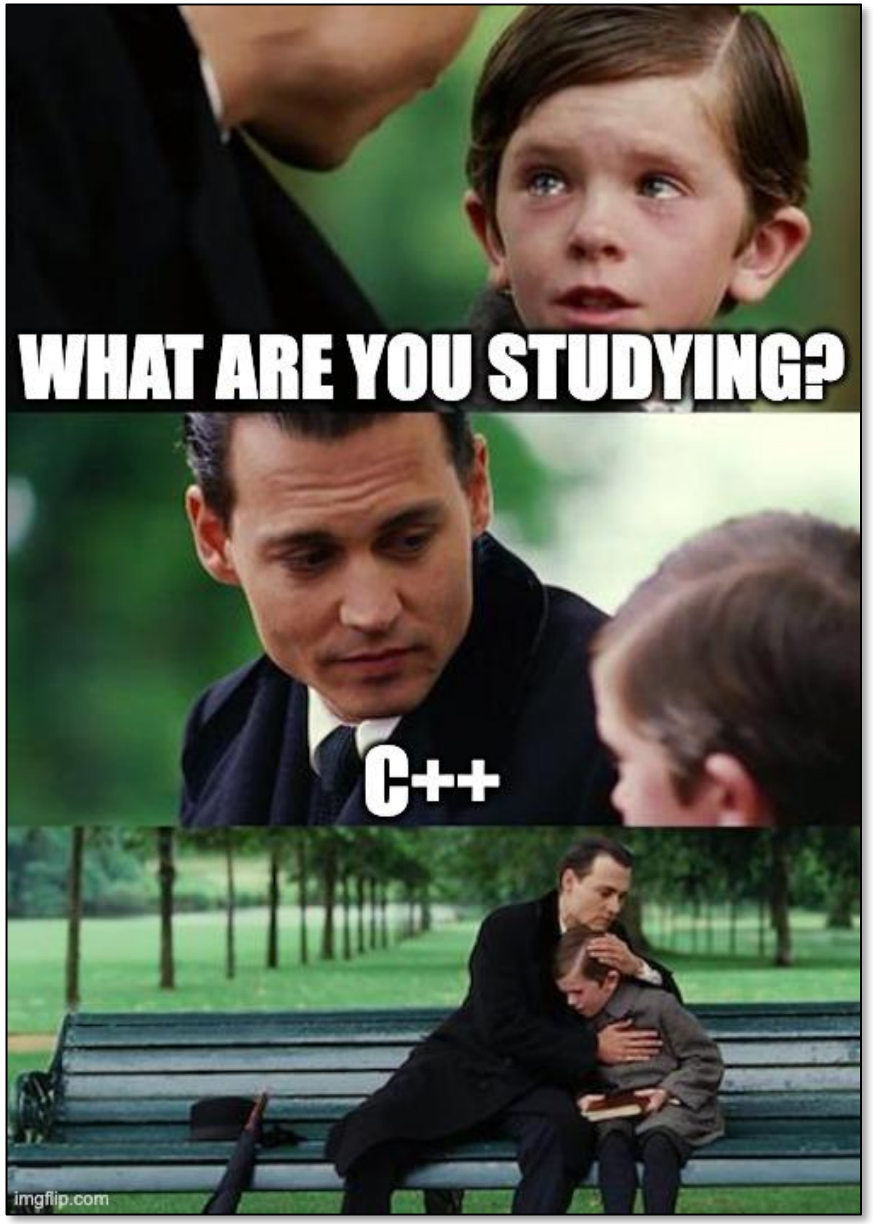
Introductions

STANFORD
MEN'S BASKETBALL

Now you can meet (some of) each other!

- Turn to the people next to you and introduce yourselves!
- **Potential Conversation Topics:**
  - What's something you're into and not into?
  - Why do you want to take this class?

The Pitch 🦈🦈


WHAT ARE YOU STUDYING?
C++
imgflip.com

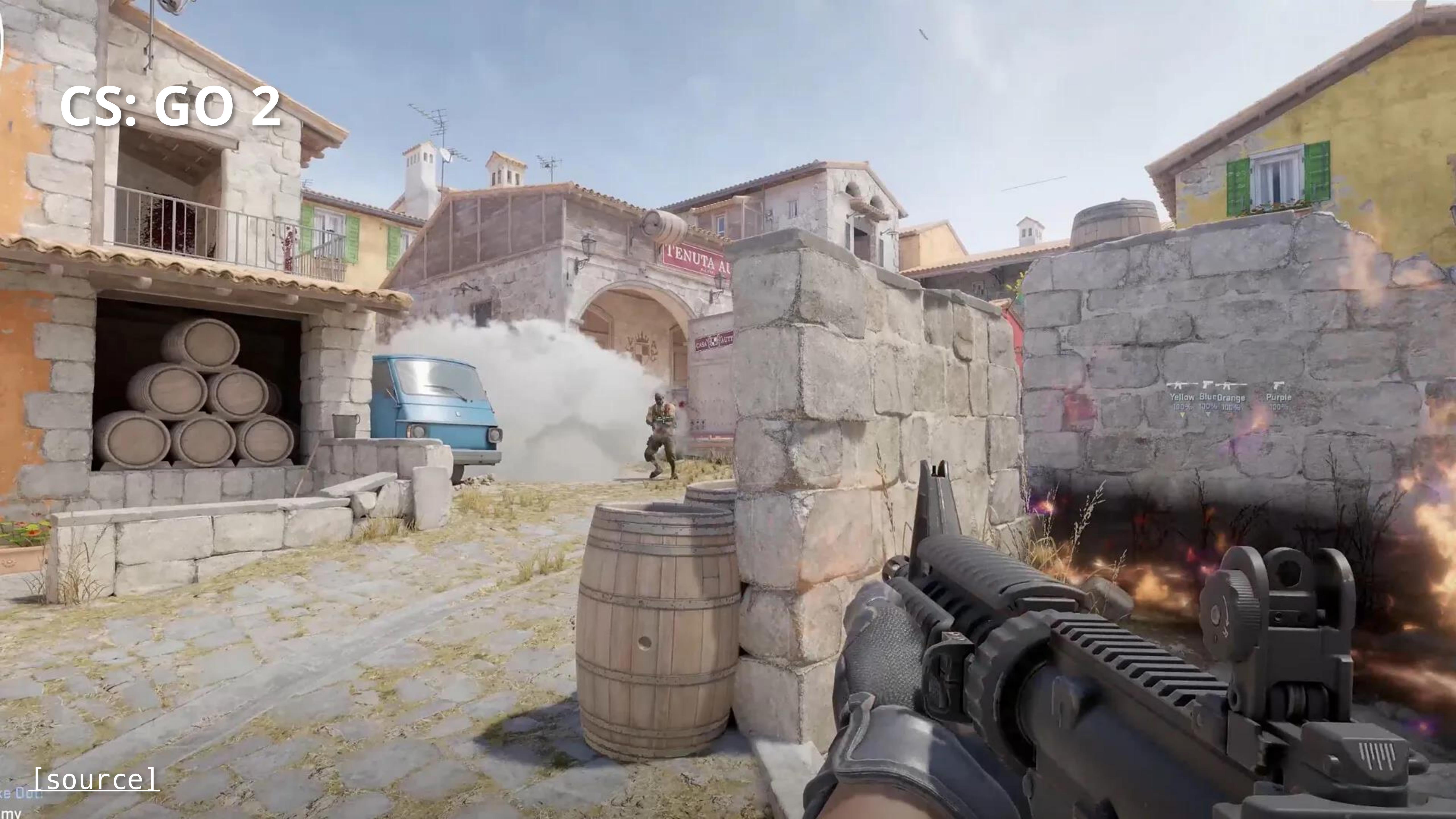
Why C++?

“The invisible foundation of everything”

Valorant

[source]

CS: GO 2



[source]

...and many more!

Fallout

grand theft auto FIVE V

SKYRIM

ASSASSIN'S CREED ™

PLAYERUNKNOWN'S BATTLEGROUNDS

FORTNITE

High Frequency Trading

Optiver

CITADEL | Securities

HRT

[source]

Self Driving

[source]

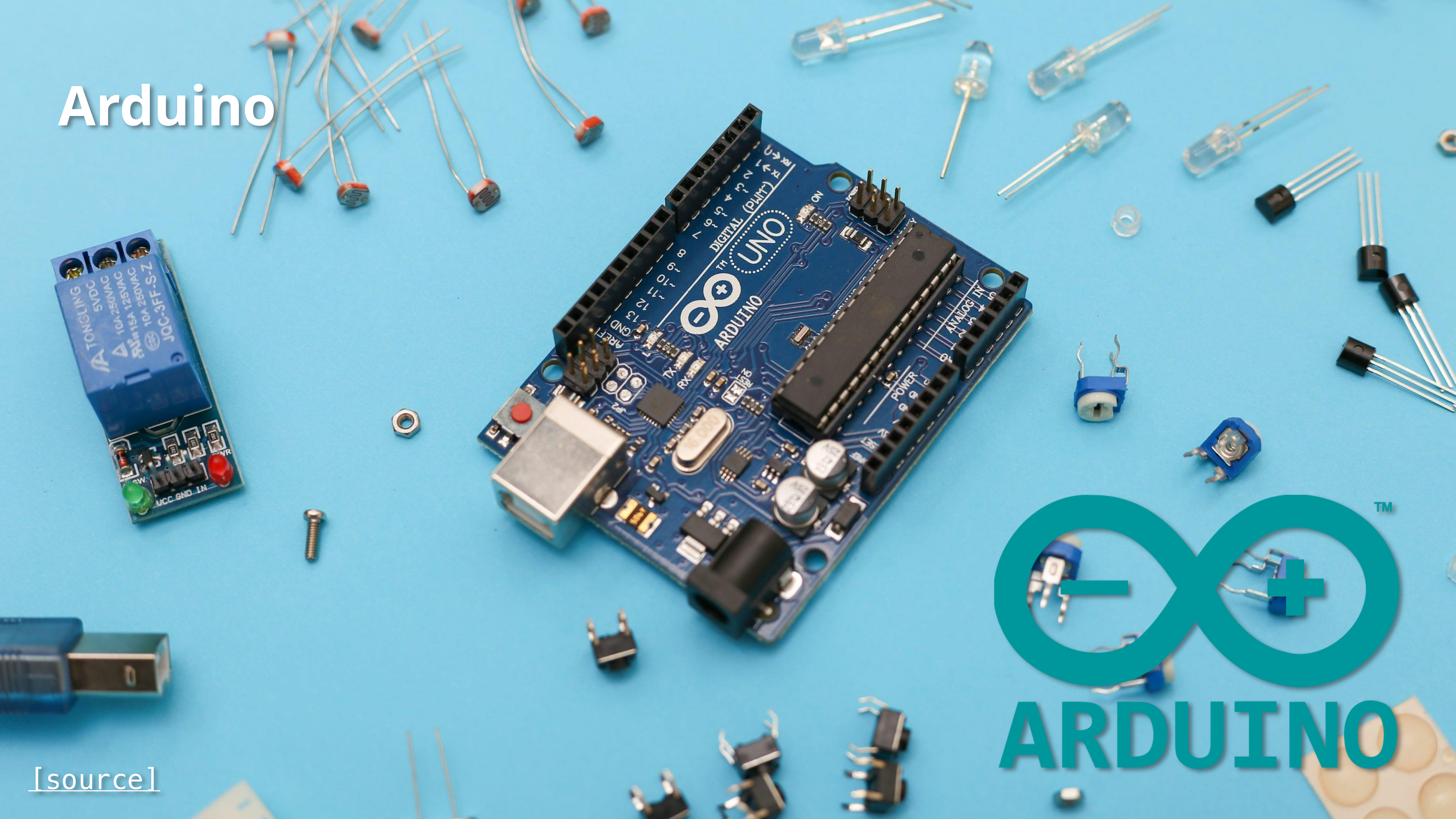
GPU Programming

NVIDIA

CUDA

[source]

Arduino



ARDUINO

[source]

And much, much more!

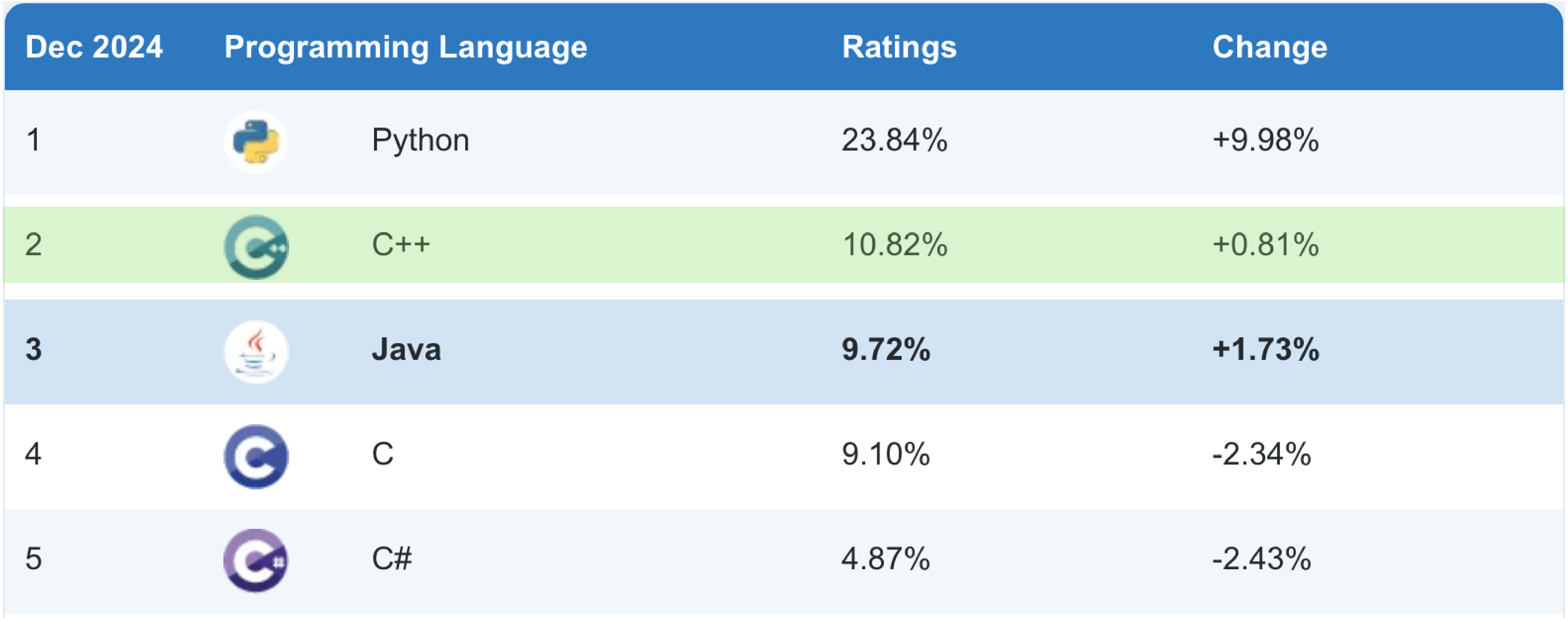
- Databases (MySQL, MongoDB)
- Web Browsers (Chrome, Safari, Edge)
- Virtual Reality (Quest)
- Low level ML (PyTorch, TensorFlow, OpenAI)
- Compilers, virtual machines (JVM, LLVM, GCC)
- Operating Systems (Windows)

“The invisible foundation of everything”

C++ is great for…

- Handling lots of data
- And handling it very efficiently
- And doing it in an elegant, readable way

C++ was created in 1983, still #2!



Dec 2024	Programming Language		Ratings	Change
1		Python	23.84%	+9.98%
2		C++	10.82%	+0.81%
3		**Java**	**9.72%**	**+1.73%**
4		C	9.10%	-2.34%
5		C#	4.87%	-2.43%

[TIOBE Index, December 2024]

C++ in Industry

amazon.com®

Meta

OpenAI

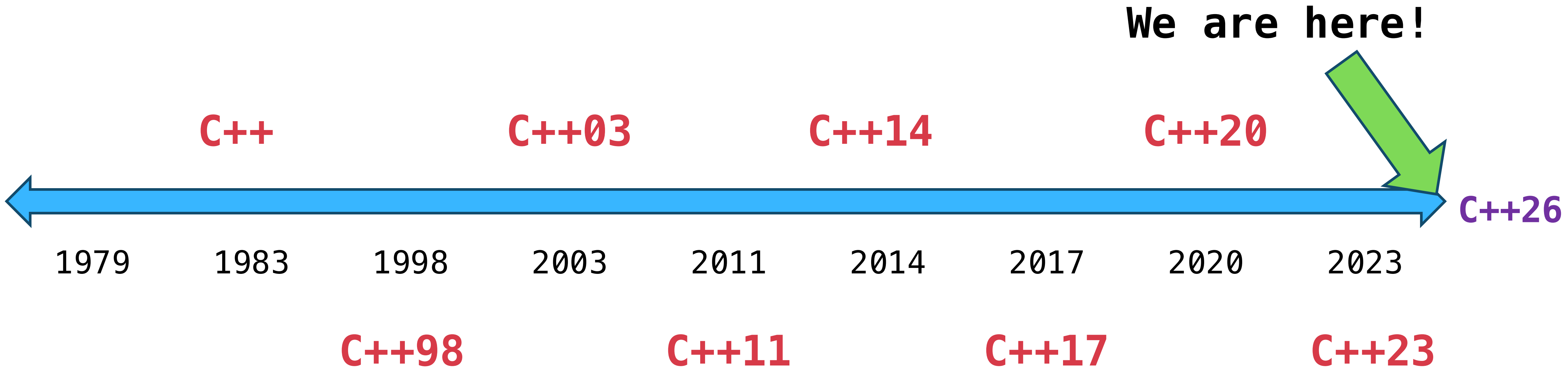
Google

CppCon 2018

[source]

The C++ Community

- C++ has a **MASSIVE** user base
- C++ Standard continues to be revised every three years



What is C++?

A valid C++ program

```cpp
#include <iostream>
#include <string>

int main() {
   auto str = std::make_unique<std::string>("Hello World!");
   std::cout << *str << std::endl;
   return 0;
}

// Prints "Hello World!"
```

Also a valid C++ program

```
#include "stdio.h"
#include "stdlib.h"

int main(int argc, char *argv) {
   printf("%s", "Hello, world!\n");
   // ^a C function!
   return EXIT_SUCCESS;
}
```

C++ is backwards compatible with C. Neat!

Also a valid C++ program

```
#include "stdio.h"
#include "stdlib.h"

int main(int argc, char *argv) {
    asm(".LC0:\n\t"
           ".string \"Hello, world!\"\n\t"
       "main:\n\t"
           "push rbp\n\t"
           "mov rbp, rsp\n\t"
           "sub rsp, 16\n\t"
           "mov DWORD PTR [rbp-4], edi\n\t"
           "mov QWORD PTR [rbp-16], rsi\n\t'
           "mov edi, OFFSET FLAT:.LC0\n\t"
           "call puts\n\t");
    return EXIT_SUCCESS;
}
```

C++ History: Assembly

```
section .text
global _start                     ;must be declared for linker (ld)
_start:                           ;tell linker entry point
   mov edx,len                    ;message length
   mov ecx,msg                    ;message to write
   mov ebx, 1                     ;file descriptor (stdout)
   mov eax, 4                     ;system call number (sys_write)
   int 0x80                       ;call kernel
   mov eax, 1                     ;system call number (sys_exit)
   int 0x80                       ;call kernel
section .data
   msg db 'Hello, world!' ,0xa    ;our dear string
   len equ $ - msg                ;length of our dear string
```

C++ History: Assembly

- ✅ Unbelievably **simple** instructions
- ✅ Extremely **fast** (when well-written)
- ✅ Complete **control** over your program

Why don’t we always use assembly?

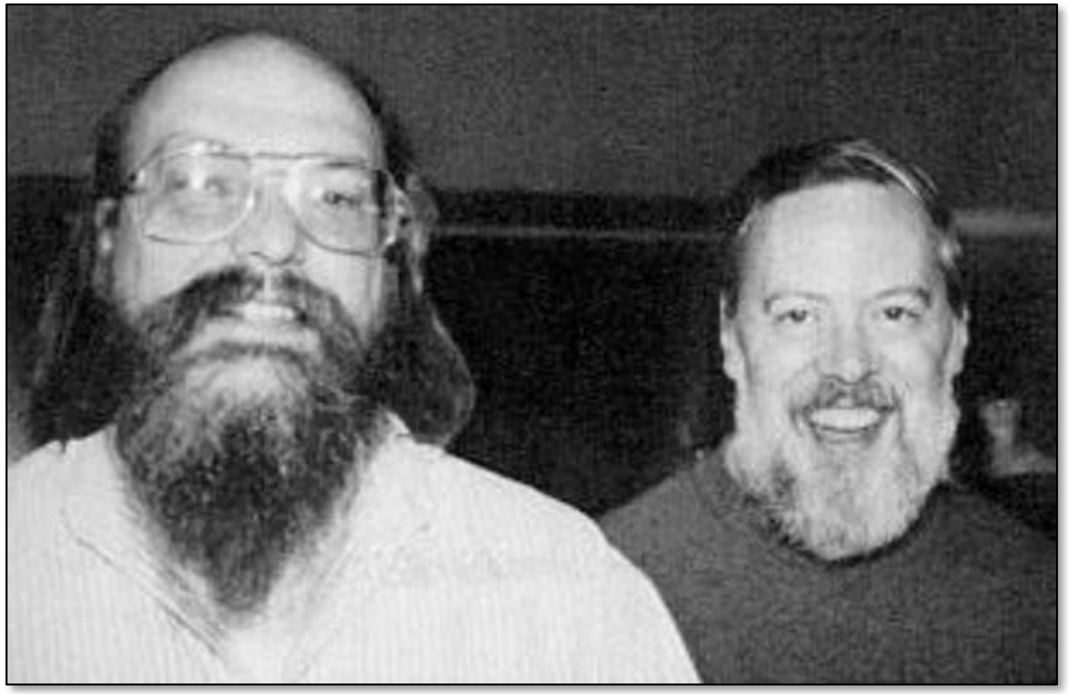
C++ History: Assembly

- ✅ Unbelievably **simple** instructions
- ✅ Extremely **fast** (when well-written)
- ✅ Complete **control** over your program

- ❌ A **lot of code** (even for simple tasks)
- ❌ Very **hard to understand**
- ❌ Extremely **unportable**

C++ History: Invention of C

- Dennis Ritchie created C in 1972 to much praise.
- C made it easy to write code that was:
  - Fast
  - Simple
  - Cross platform
    - **Compilers!** Source Code → Assembly
- Learn to love it in **CS107**!



Ken Thompson and Dennis Ritchie, creators of the C language

C++ History: Invention of C

- C was popular because it was simple
    - *“When I read C I know what the output Assembly is going to look like”*

      —Linus Torvalds, creator of Linux
- However, C has some weaknesses:
    - No **objects** or classes
    - Difficult to write **generic or templated** code
    - **Tedious** to write large programs

C++ History: Welcome to C++!

- In 1983, the beginnings of C++ were created by Danish computer scientist Bjarne Stroustrup
- He wanted a language that was
  - Fast
  - Simple to use
  - Cross-platform
  - **Had high level features**

Bjarne Stroustrup, the man himself ☺

C++ Design Philosophy

- Express ideas and intent directly in code.
- Enforce safety at compile time whenever possible.
- Do not waste time or space.
- Compartmentalize messy constructs.
- Allow the programmer full control, responsibility, and choice.

"Code should be elegant **and** efficient; I hate to have to choose between those"

—Bjarne Stroustrup

C++ Design Philosophy (Summarized)

- Readable
- Safety
- Efficiency
- Abstraction
- Multi-paradigm

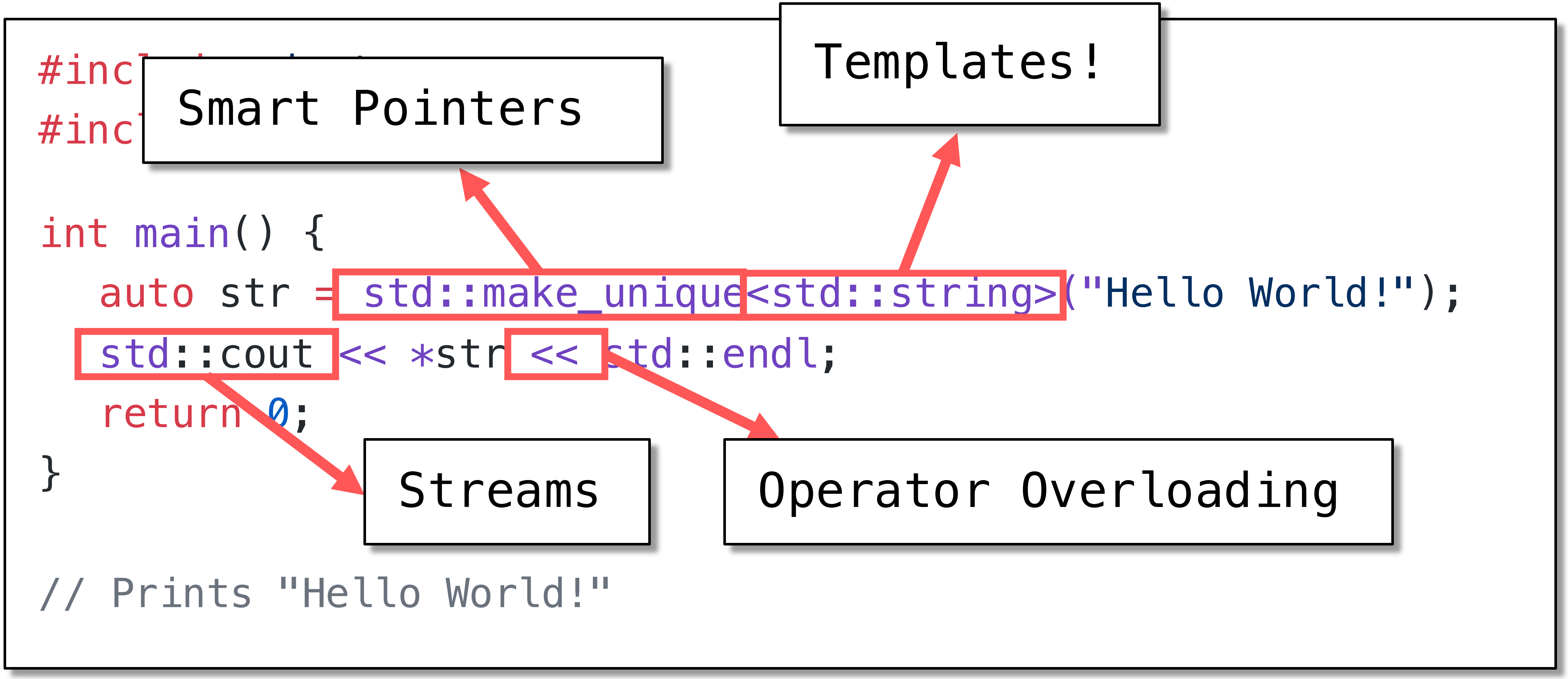
A valid C++ program

```
#include "stdio.h"
#include "stdlib.h"

int main(int argc, char *argv) {
    asm(".LC0:\n\t"
            ".string \"Hello, world!\"\n\t"
        "main:\n\t"
            "push rbp\n\t"
            "mov rbp, rsp\n\t"
            "sub rsp, 16\n\t"
            "mov DWORD PTR [rbp-4], edi\n\t"
            "mov QWORD PTR [rbp-16], rsi\n\t"
            "mov edi, OFFSET FLAT:.LC0\n\t"
            "call puts\n\t");
    return EXIT_SUCCESS;
}
```

A valid C++ program

```
#include "stdio.h"
#include "stdlib.h"

int main(int argc, char *argv) {
  printf("%s", "Hello, world!\n");
  // ^a C function!
  return EXIT_SUCCESS;
}
```

A valid C++ program



```
#incl
#incl

int main() {
   auto str = std::make_unique<std::string>("Hello World!");
   std::cout << *str << std::endl;
   return 0;
}

// Prints "Hello World!"
```

Topics We’ll Cover

Week 1	Welcome	Types & Structs
Week 2	Initialization & References	Streams
Week 3	Containers	Iterators & Pointers
Week 4	Classes	Template Classes
Week 5	Template Functions	Functions & Lambdas
Week 6	Operator Overloading	Special Member Functions
Week 7	Move Semantics	std::optional and Type Safety
Week 8	RAII, Smart Pointers, C++ Projects	

Why take CS106L?

CS106B vs. CS106L

CS106B

- Focus on **concepts** like abstractions, recursion, pointers etc.
- **Bare minimum** C++ in order to use these concepts

CS106L

- Focus is on **code**: what makes it good, what powerful and elegant code looks like
- The real deal: No Stanford libraries, only STL
- Understand **how** and **why** C++ was made

When might you use C++?

- In one of Stanford’s classes
  - **CS 111:** Operating Systems Principles
  - **CME 213:** Introduction to parallel computing using MPI, openMP, and CUDA
  - **CS 143:** Compilers
  - **CS 144:** Introduction to Computer Networking
  - **CS 248A:** Computer Graphics: Rendering, Geometry, and Image Manipulation
  - **MUSIC 256A:** Music, Computing, Design: The Art of Design
  - …and more!
- And in real life!

“Nobody should call themselves a professional if they only know one language” —Bjarne Stroustrup

C++ helps develop good coding hygiene

- Am I using objects the way they're meant to be used?
  - Type checking, type safety
- Am I using memory efficiently?
  - Reference/copy semantics, move semantics
- Am I modifiying something I'm not supposed to?
  - `const` and const correctness
- Other languages relax these restrictions

Magnus vs. Me

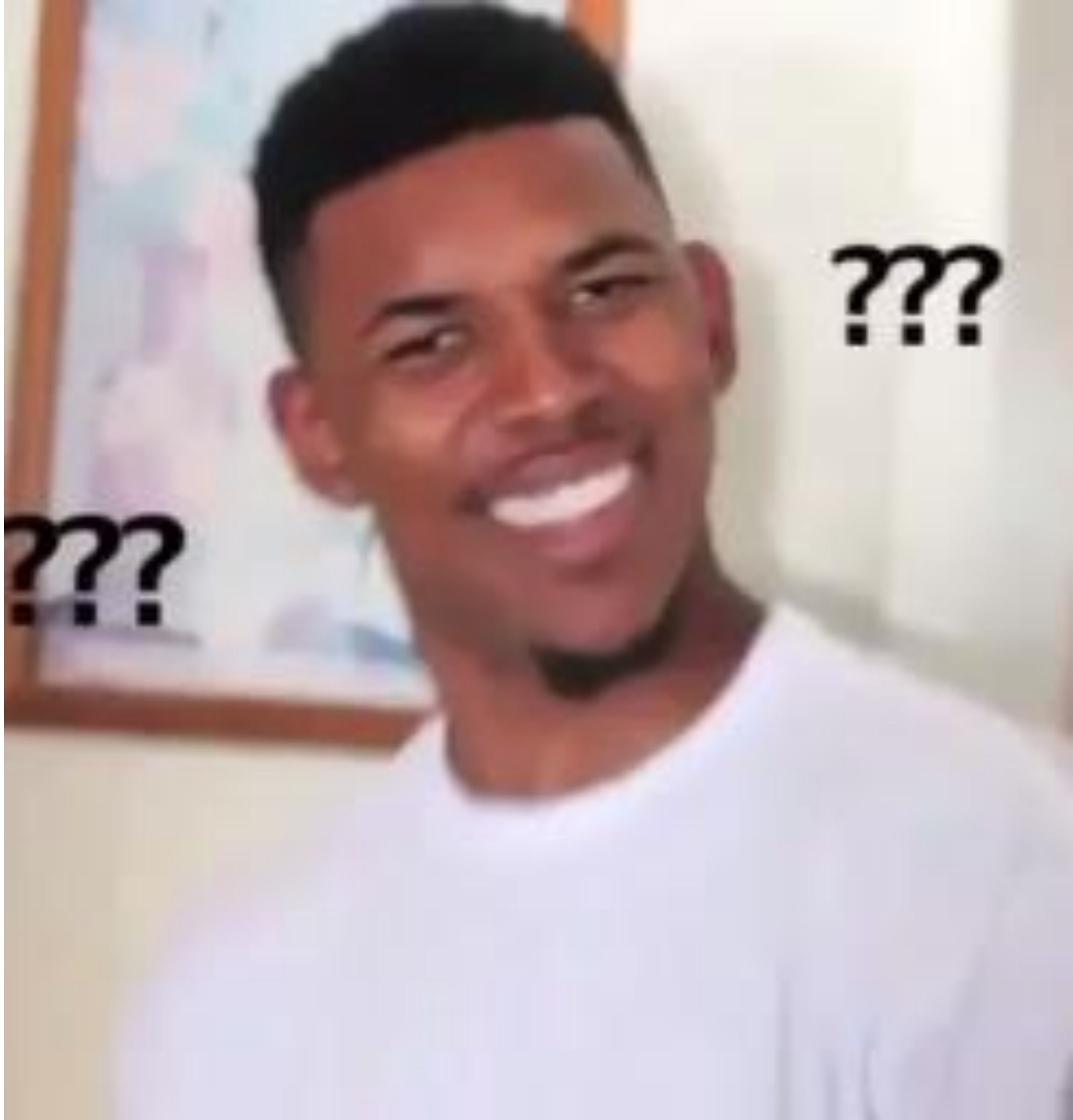
What questions do you have?

bjarne_about_to_raise_hand

Course Logistics

Asking Questions

- We welcome questions!
- Feel free to raise your hand at ***<u>any time</u>*** with a question
- We'll also pause periodically to solicit questions and check understanding



Access and Accommodations

- Disabled students are a valued and essential part of the Stanford community. We welcome you to our class!

- Please work with OAE but also let us know if there's anything we can do to make the course more accessible to you.

- Don't be shy about asking for accommodations if problems arise. We're very reasonable people and will do whatever we can to help.

Community Norms

- Shame-free zone
- Treat your peers and instructors with kindness and respect
- Be curious
- Communication is key!
- Recognized we are all in-process (humility, question posing, avoid perfectionism)

Guiding Principles

- We will do everything we can to support you. We want to provide flexibility to the best of our ability!
- We want to hear your feedback so we can ensure the class is going as smoothly as possible for everyone
- Please communicate with us if any personal circumstances or issues arise! We are here to support you :)

What questions do you have?

`bjarne_about_to_raise_hand`

Lecture

- Held **Tuesdays** and **Thursdays** from 3:00pm – 4:20pm in Turing Auditorium

- Lecture is not recorded.

- **Attendance is required.** Short participations quizzes (1-2 questions) will be given at the beginning of lecture starting in week 2. **All students are given 2 free absences**.

Illness

- If you are sick, for the wellbeing of yourself and others **please stay home**, take care of yourself, and reach out to us – **we never want** you to feel that you must attend class if you are not feeling well!

- Similarly, if you have an emergency or exceptional circumstance, **please reach out to us** so that we can help!

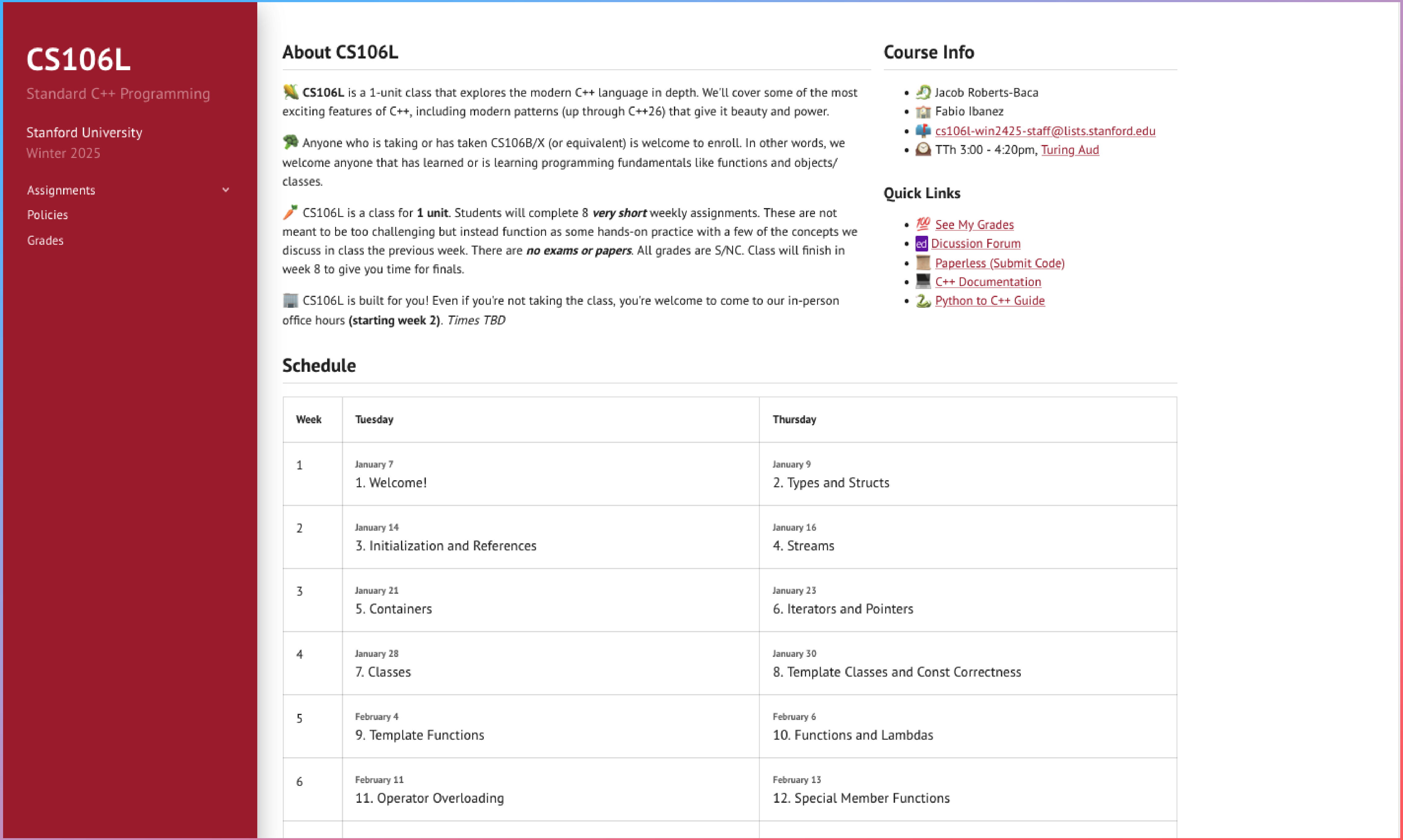
Office Hours

- OH times are TBD and will be in person
    - These will be settled by week 2 (before the first assignment)

- We want to talk to you! Come talk!

- Extra OH weeks 9 – 10!

- Watch the course website (cs106l.stanford.edu) and Ed for more info.



CS106L

Standard C++ Programming

Stanford University
Winter 2025

- Assignments
- Policies
- Grades

About CS106L

🌽 **CS106L** is a 1-unit class that explores the modern C++ language in depth. We'll cover some of the most exciting features of C++, including modern patterns (up through C++26) that give it beauty and power.

🥦 Anyone who is taking or has taken CS106B/X (or equivalent) is welcome to enroll. In other words, we welcome anyone that has learned or is learning programming fundamentals like functions and objects/classes.

🥕 CS106L is a class for **1 unit**. Students will complete 8 ***very short*** weekly assignments. These are not meant to be too challenging but instead function as some hands-on practice with a few of the concepts we discuss in class the previous week. There are ***no exams or papers***. All grades are S/NC. Class will finish in week 8 to give you time for finals.

🏢 CS106L is built for you! Even if you're not taking the class, you're welcome to come to our in-person office hours **(starting week 2)**. *Times TBD*

Course Info

- 🐉 Jacob Roberts-Baca
- 🏫 Fabio Ibanez
- 📫 cs106l-win2425-staff@lists.stanford.edu
- 🕰️ TTh 3:00 - 4:20pm, Turing Aud

Quick Links

- 💯 See My Grades
- ed Dicussion Forum
- 📜 Paperless (Submit Code)
- 💻 C++ Documentation
- 🐍 Python to C++ Guide

Schedule

Week	Tuesday	Thursday
1	January 7 1. Welcome!	January 9 2. Types and Structs
2	January 14 3. Initialization and References	January 16 4. Streams
3	January 21 5. Containers	January 23 6. Iterators and Pointers
4	January 28 7. Classes	January 30 8. Template Classes and Const Correctness
5	February 4 9. Template Functions	February 6 10. Functions and Lambdas
6	February 11 11. Operator Overloading	February 13 12. Special Member Functions

Assignments

- There will be 8 short weekly assignments (typically will take 1 hour at most depending on experience)
  - Submissions will be on paperless as directed on the assignment handout!

- Assignments will be released on Fridays and due in one week (the following Friday)
  - All students have three free late days.

Grading

- Grading is S/NC. We expect everyone to get an S!

- **How do you get an S?**
  - Attend **11 of the 13** lectures between Week 2 and Week 9
  - Successful completion of **6 out of 8** weekly assignments

Get in touch with us!

- Here are the best ways to communicate with us!

- Email us: **cs106l-win2425-staff@lists.stanford.edu**
  - Please use this email and not our individual emails so we both receive the message!

- Public or private post on Ed

- After class or in our office hours

What questions do you have?

bjarne_about_to_raise_hand