WELLESLEY

Programming Languages

CS 251

Fall 2021

Carolyn Anderson

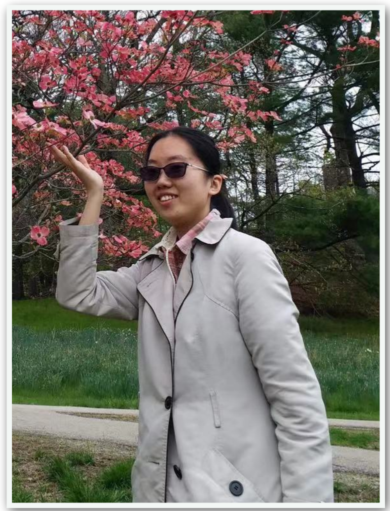
Course Logistics

Staff

Carolyn Anderson (she)
Instructor

Annie Liu (she)
Grader

Anastacia Castro (she)
Tutor



Funing Yang (she)
Tutor

Schedule

- Room: L Wing 043
- Lecture: 11:20-12:35 on Tuesdays and Fridays
- Assignments are due on Mondays at 10 PM

Help Hours

- Tutor hours:
  - Anastacia:
    - Mondays, 4-5pm
    - Thursdays, 1-2pm
  - Funing:
  - Weekends, TBA

- My help hours:
  - Wednesdays, 4-5pm
  - Fridays, 2:15-3:15pm
  - By appointment

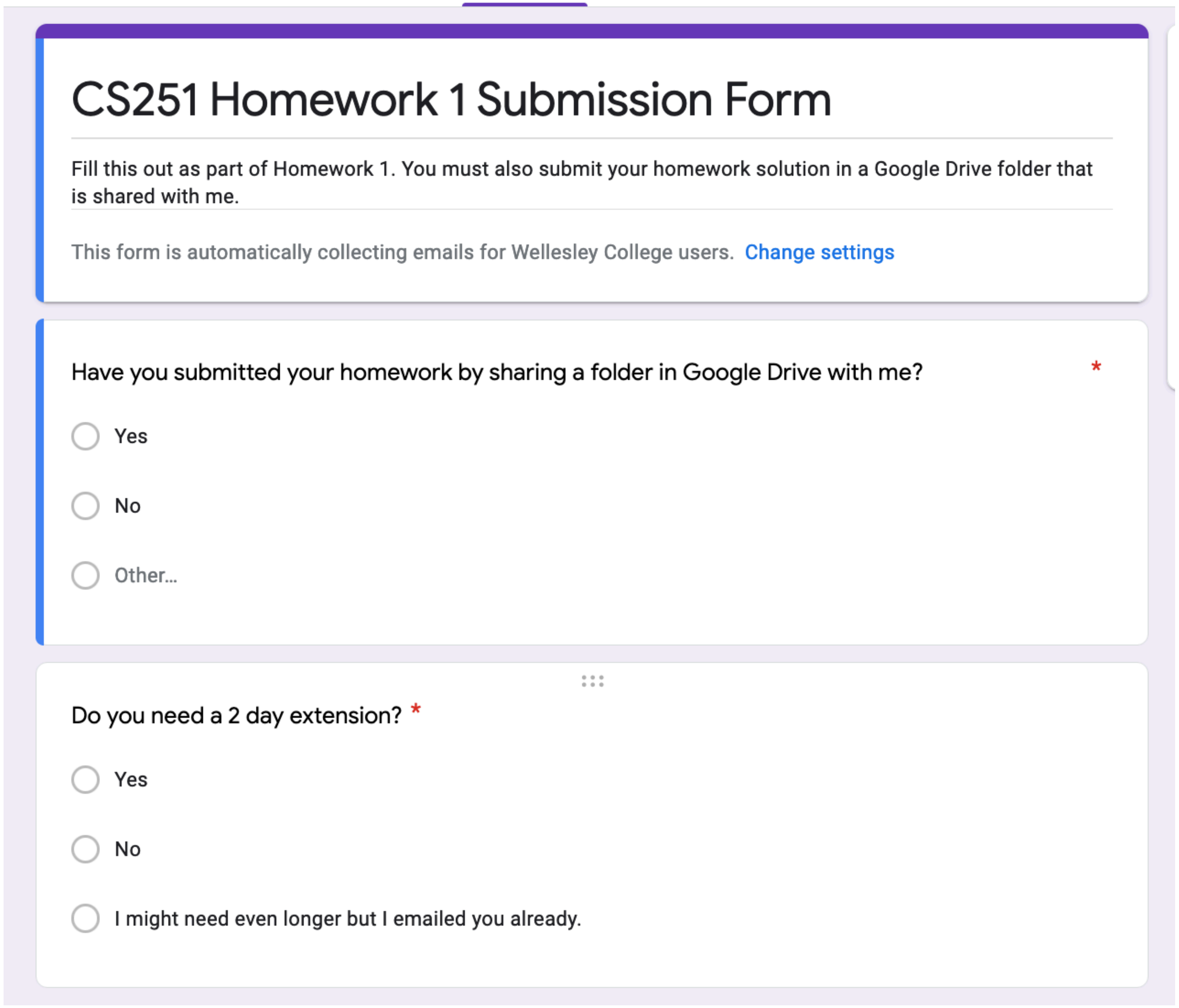
Come to my help hours to …

- Get help with CS251
- Talk about PL
- See pictures of my cats

Assignments

- Assignments are due on **Mondays at 10 PM**
- Homework submission will be through Google Drive.
- You will also submit a Google Form, where you'll mention who you worked with and estimate how long the assignment took you.
- Expect an assignment every week.

Homework Form



CS251 Homework 1 Submission Form

Fill this out as part of Homework 1. You must also submit your homework solution in a Google Drive folder that is shared with me.

This form is automatically collecting emails for Wellesley College users. Change settings

Have you submitted your homework by sharing a folder in Google Drive with me? *

- ◯ Yes
- ◯ No
- ◯ Other...

Do you need a 2 day extension? *

- ◯ Yes
- ◯ No
- ◯ I might need even longer but I emailed you already.

Assignments

- Expect an assignment every week.
- Get help early!
- Late work accepted until **Wednesday**, but **you must submit the form on Monday** and **request an extension**.
- If you won't make the Wednesday deadline, email me **as soon as possible** so that we can make arrangements.

This is so that I can hand work back promptly and discuss any issues with homework problems in class.

Exams

There will be **two** exams:

- The **midterm** will be October 29th
- The **final** will be during finals

Collaboration policy

In this class, you can talk at a high-level with other students about homework assignments, but **you cannot show them your code**.

If you discuss a homework problem with another student, **please note which students** on your assignment when you submit it.

Honor code

Collaboration:

- You may discuss **high-level ideas or strategies with other students**, but you must **report who you worked** with when you submit your assignment.
- You must not communicate detailed algorithms, implementations, code, formulae, or other detailed solution steps.
- **You must not, under any circumstances, view, share, prepare, or accept written solutions or code outside your team.**
- **Wait 30 minutes after discussions** with other students before writing your solution. This helps you know if you actually understand the solution.
- You may not share code from homework problems with other students at any point in the course (**even after the assignment deadline**).

Feedback and Questions

You can submit anonymous feedback or anonymous questions through the **Anonymous Question Form**.

Questions submitted using the form will be answered in the **Q&A document**. Check it regularly for help with problem sets!

If you are submitting feedback about the course rather than a question for the Q&A document, just say that in the form.

Guest Lectures

We're going to have at least 2 **guest lectures**:

- Luna Phipps-Costin from UMass Amherst, talking about **gradual typing**

- Rachit Nigam + Alexa VanHattum from Cornell, talking about **compilers for specialty hardware**

Why study PL?

It gives you ***frameworks*** *for thinking about programming and programming languages*

Understanding PL ***design principles*** *makes it* ***easier to learn new PLs***

It helps you explore ***core concepts in computation****, which is fun in its own right*

Topics

- Semantics
- Functional programming
- Higher-order functions
- First-class functions
- Scope
- Evaluation
- Types and type-checking
- Interpretation and compilation
- Side effects

Topics

- Semantics
- Functional programming
- Higher-order functions
- First-class functions
- Scope
- Evaluation
- Types and type-checking
- Interpretation and compilation
- Side effects

But it's ok if none of these terms sound familiar right now!

What is a Programming Language?

How have you heard people talk about programming languages?

What is a language anyway?

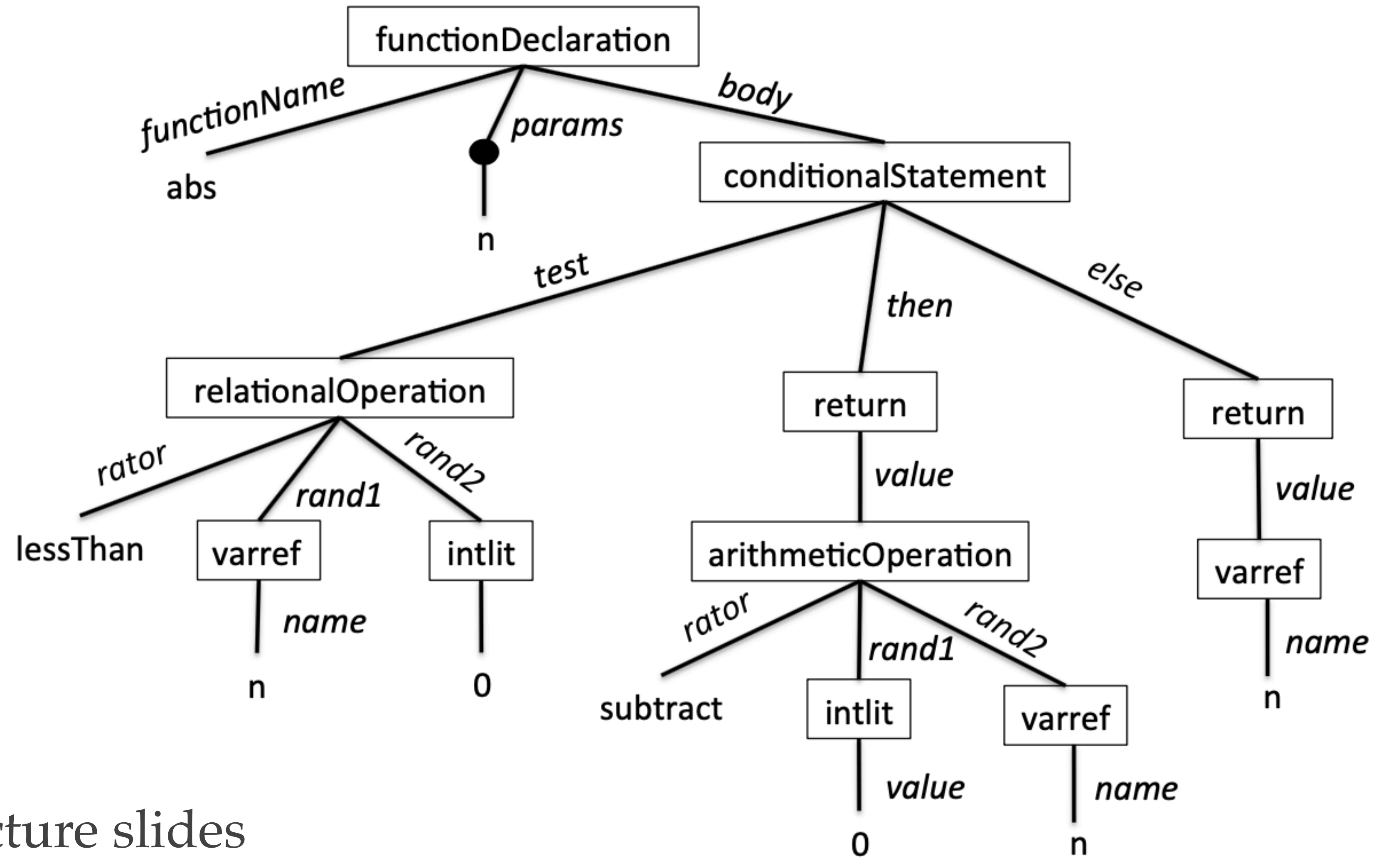
Syntax

- A programming language's syntax defines its **form**
- Syntax tells us:
    - What are the **valid expressions** in the PL?
    - How can we **compose them** to make bigger expressions?
    - Concrete syntax: style choices ({ } versus white space)
    - Abstract syntax: structure only

Abstract versus Concrete Syntax

```
def abs(n):
    if n < 0:
       return –n
    else:
       return n
```

```
public static int abs (int n) {
    if (n < 0) return -n;
    else return n;
}
```



Tree from Lyn Turbak's lecture slides

Semantics

- A programming language's semantics defines its **meaning**
- Syntax tells us:
  - How do we **evaluate** expressions in the PL?
  - How will this program **behave**?
  - **Dynamic semantics**: what happens when a program runs?
    - Does the program **terminate**?
    - What **values** does it produce?
    - What **side-effects** does it have?
  - **Static semantics**: what can we infer about a program without running it?
    - Are expressions **well-typed**?
    - What is the **scope** of a variable?

Sadly, most interesting questions can't be answered statically. 😥

Semantics versus Syntax

colorless green ideas sleep furiously

versus

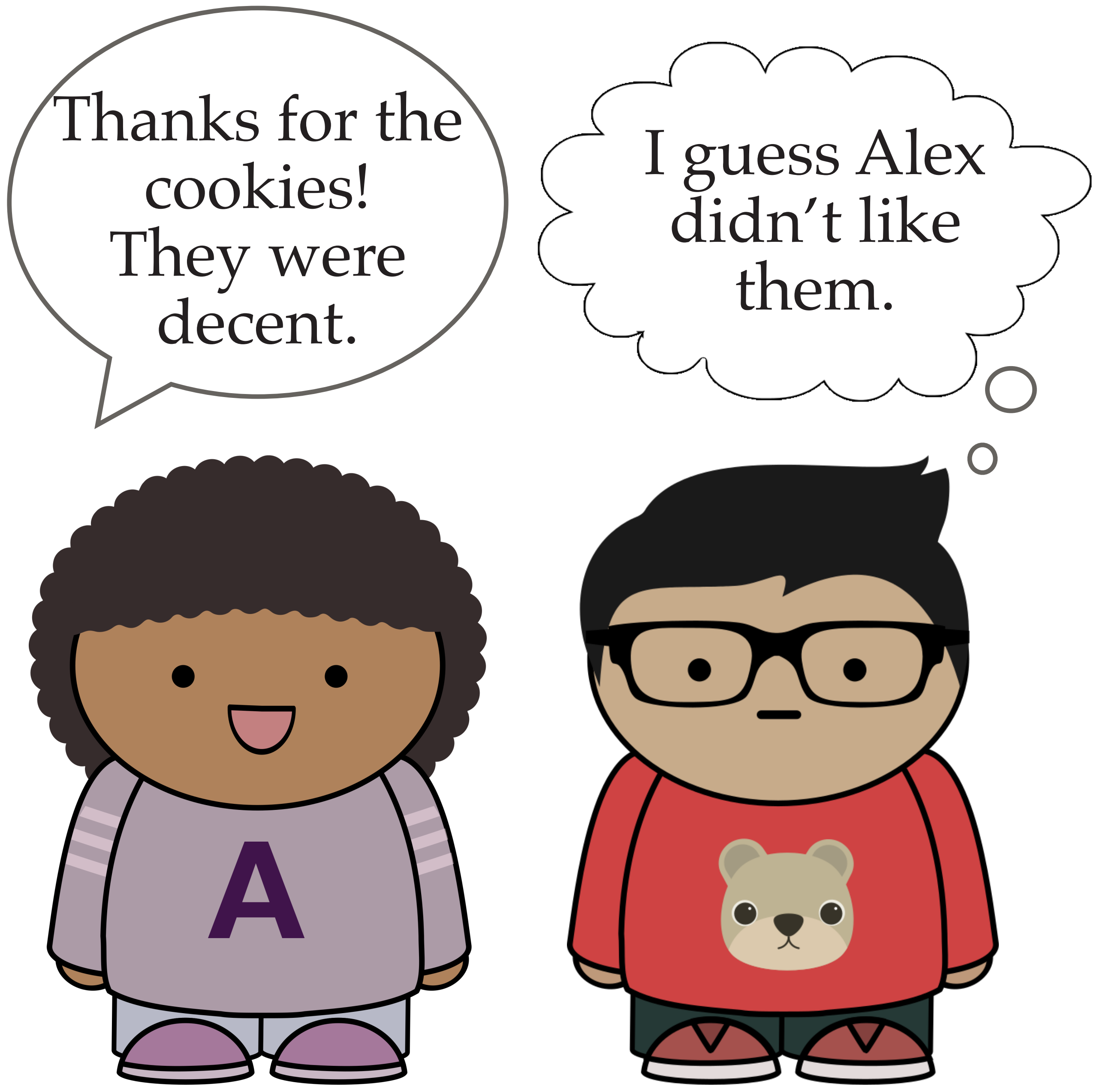
can anyway probably you this read

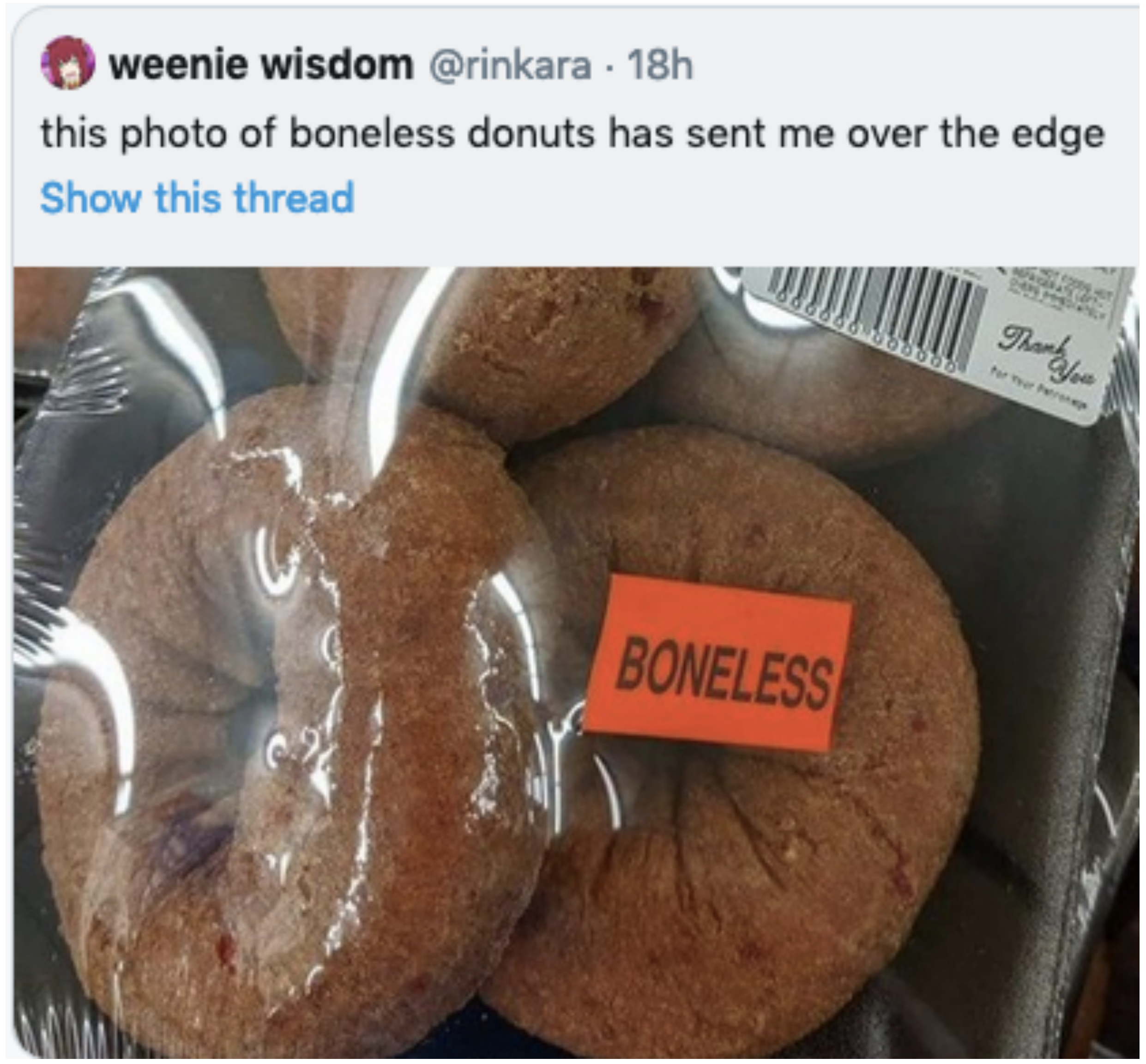
Pragmatics

- A programming language's pragmatics describe its interactions (how it is used)
- Pragmatics is about:
  - How do the semantics of the language **interact** with the programming environment?
    - The physical machine
    - Editors used to program in the language
    - Interfaces with other PLs
  - How do **users** interpret its semantics?
  - What **conventions of use** arise from the community of users?

Semantics versus Pragmatics

[[decent]] = good


Thanks for the cookies! They were decent.
I guess Alex didn't like them.

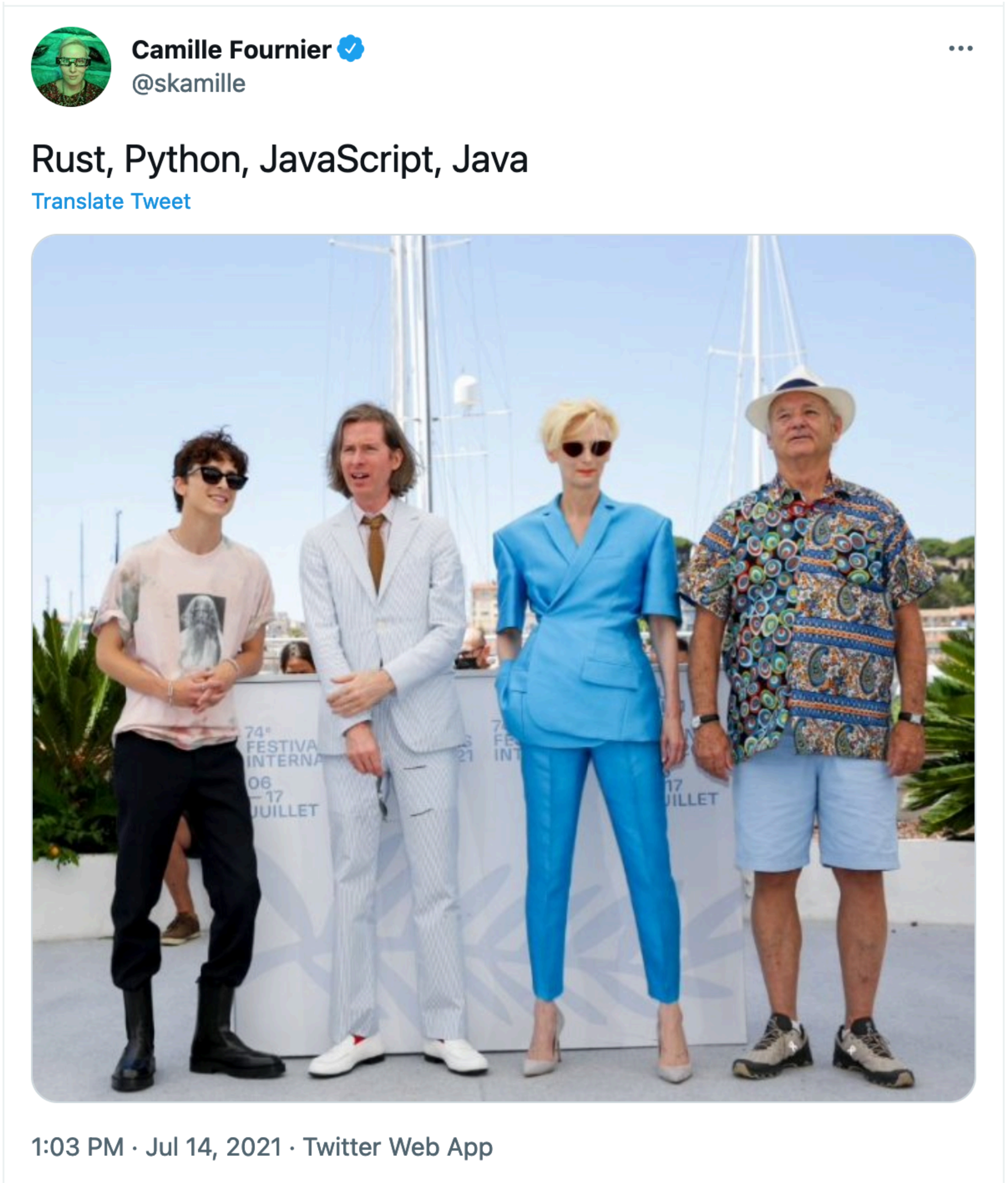


weenie wisdom @rinkara · 18h

this photo of boneless donuts has sent me over the edge

Show this thread

Language attitudes



Camille Fournier
@skamille

Rust, Python, JavaScript, Java

Translate Tweet

1:03 PM · Jul 14, 2021 · Twitter Web App

Which PL is best?

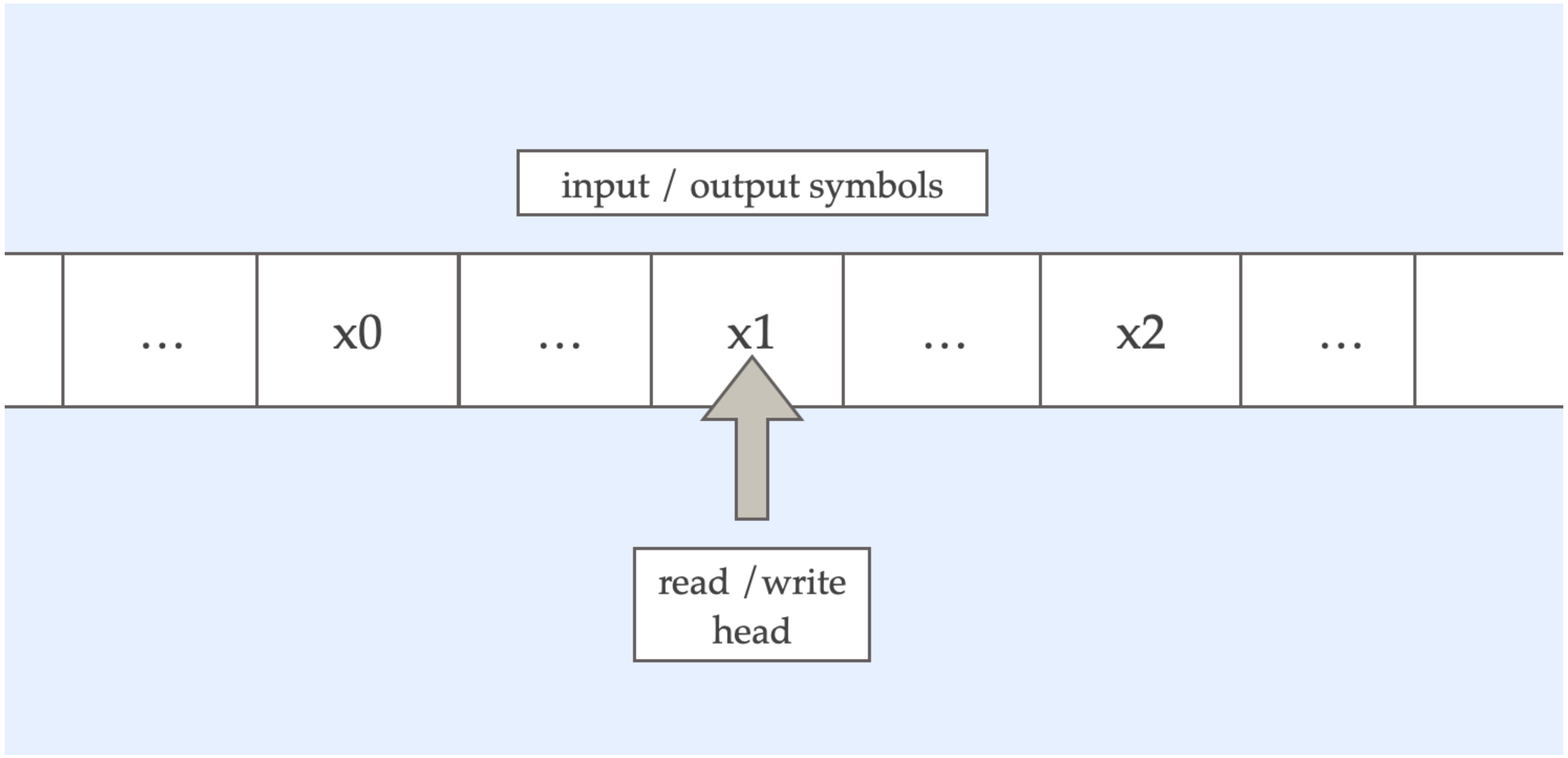
This is a question about language attitudes, disguised as a question about PL semantics!

Power versus expressiveness

- Computational power: what can be expressed in a given language?
- Expressiveness: what is **easy** to express in a given language?

Computational power

Church-Turing Thesis: a function is "effectively calculable" iff it can be computed by a Turing machine.



All languages that are Turing-complete have equal computational power.

How do PL researchers think about programming languages?

Which PL is best *for a given task*?

Abstraction

If all languages are equally powerful, why don't we program in binary?

```
01101000101010101110010101001010001
01101001010111010001101010100101001
00101010101110100011010101001010001
01101000101010101110010101001010001
01101001010111010001101010100101001
00101010101110100011010101001010001
01101000101010101110010101001010001
01101001010111010001101010100101001
00101010101110100011010101001010001
01101000101010101110010101001010001
01101001010111010001101010100101001
```

Domain Specific Languages

Web programming	**CSS**	**HTML**	
Statistics	**Stan**	**R**	**SPSS**
Game development	**UnrealScript**		
Verification	**Coq**	**Dafny**	
Modeling probability	**WebPPL**	**Figaro**	

PL Paradigms

- Functional: computation is expressed by composing functions that manipulate immutable data

Racket
Scala ML
Haskell

- Imperative: computation is seen as updating the program's state

C C++
Python

- Object-oriented: computation is expressed with stateful objects that pass messages to each other

Java
JavaScript

Meta-programming

- **Compilation**: **translate** a program P in a source language S to a program P' in a target language T using a translator written in implementation language I.

  A compiler is just an intermediary; it returns a program but doesn't execute it.

- **Interpretation**: **interpret** a program P in a source language S in terms of an implementation language I.

  An interpreter can execute the code directly, or transform it before executing.

How do PL researchers think about programming languages?

In this class, we will treat programming languages as objects of scientific study. We want to understand how they work, why they work, and how they could be different.