Lecture 1: Introduction

- welcome!
- what is *theoretical* computer science?
- course themes
- course logistics

Pravesh Kothari
Instructor

Pedro Paredes
Instructor

Matt Weinberg
Instructor

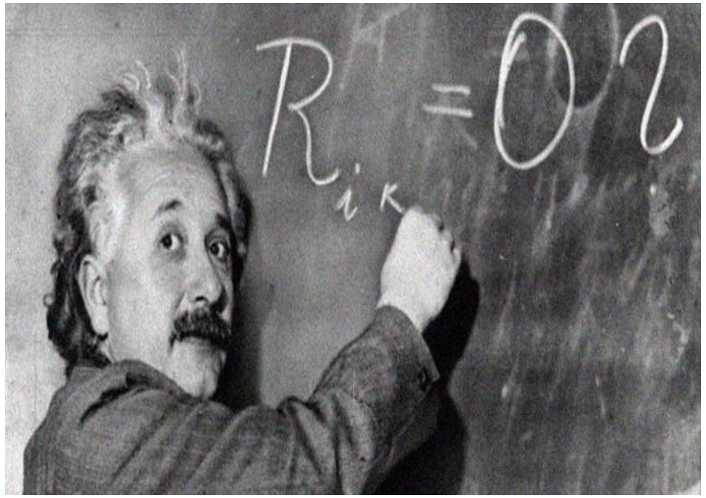
Daniel Braga
Graduate TA

Ilya Maier
Graduate TA

Fangqi Dong
Graduate TA

TBD
UCA

TBD
UCA



Lecture 1: Introduction

- welcome!
- what is *theoretical* computer science?
- course themes
- course logistics

Theoretical Computer Science

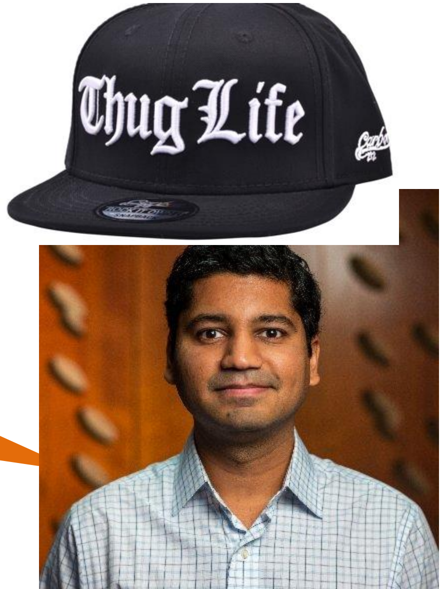
Computer Science

Science

Writing Python programs to solve problems?

Computer Science

The science of computation



Computer science is no more about computers than astronomy is about telescopes

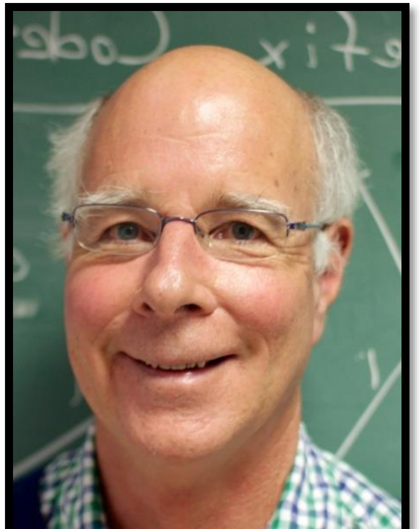
-Edsger Dijkstra

Computer science is no more about computers than astronomy is about telescopes

-Edsger Dijkstra

Computer science is no more about computers than astronomy is about telescopes

--Mike Fellows



Writing Python programs to solve problems?

Computer Science

The science of computation

rigorous study

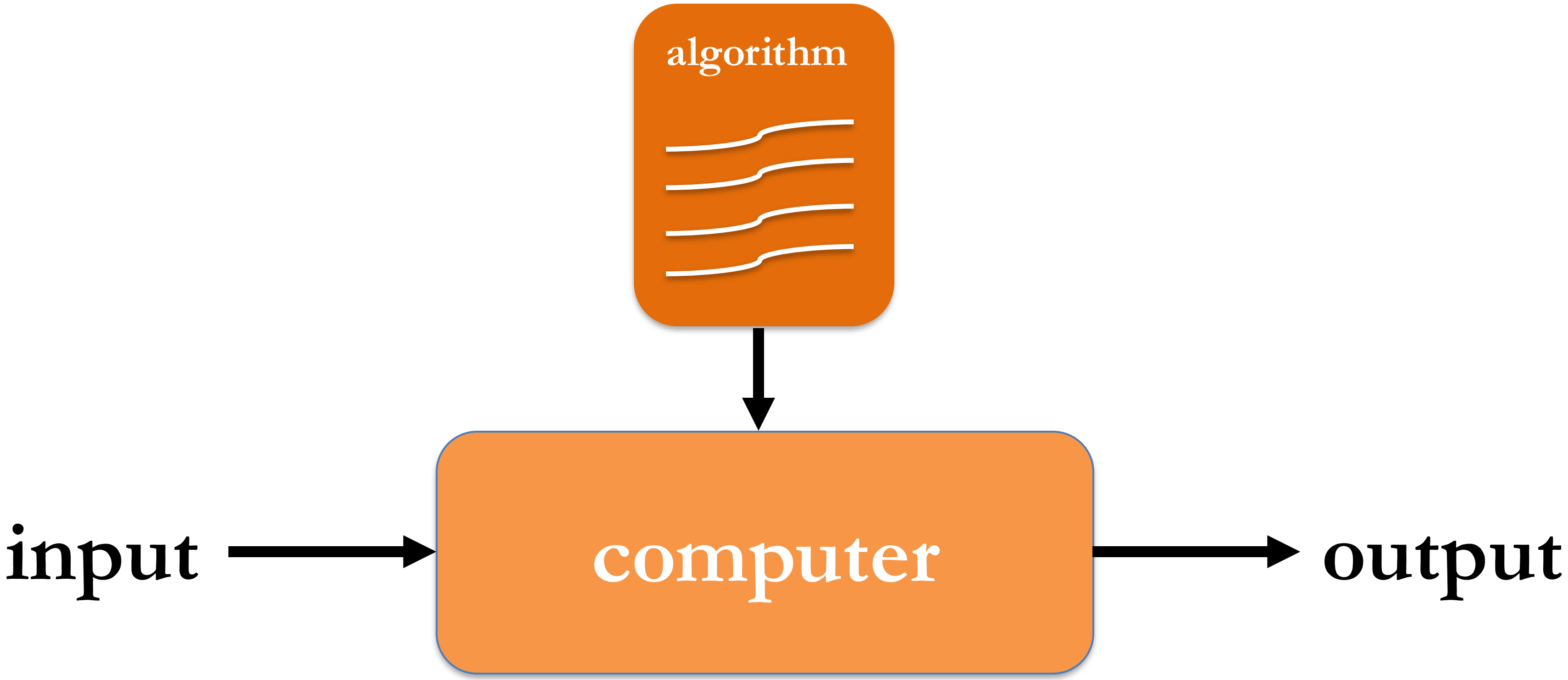
manipulation of information/data

Computer science is the study of computation—the processes that transform information. It asks which problems are solvable, how to solve them, and at what cost. It builds abstractions (algorithms, data, languages, systems) to automate tasks reliably at scale. It also examines the limits, ethics, and impacts of computing on people and society.

Computer science is the study of computation—the processes that **transform information**. It asks which problems are solvable, how to solve them, and at what cost. It builds abstractions (**algorithms**, data, languages, systems) to automate tasks reliably at scale. It also examines the limits, ethics, and impacts of computing on people and society.

Computation: manipulation of information/data

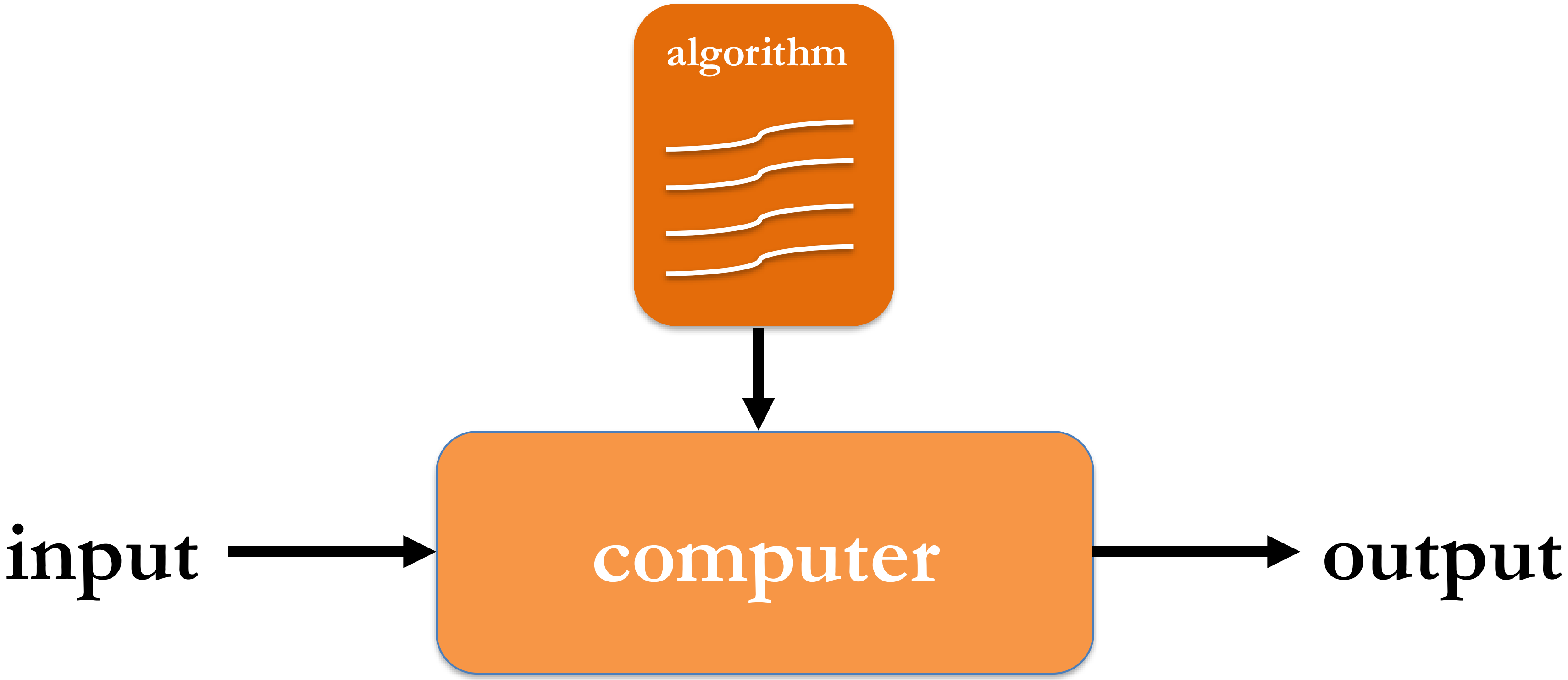
Algorithm: rigorous description of *how* the data is manipulated



Computation: manipulation of information/data

Algorithm: rigorous description of *how* the data is manipulated

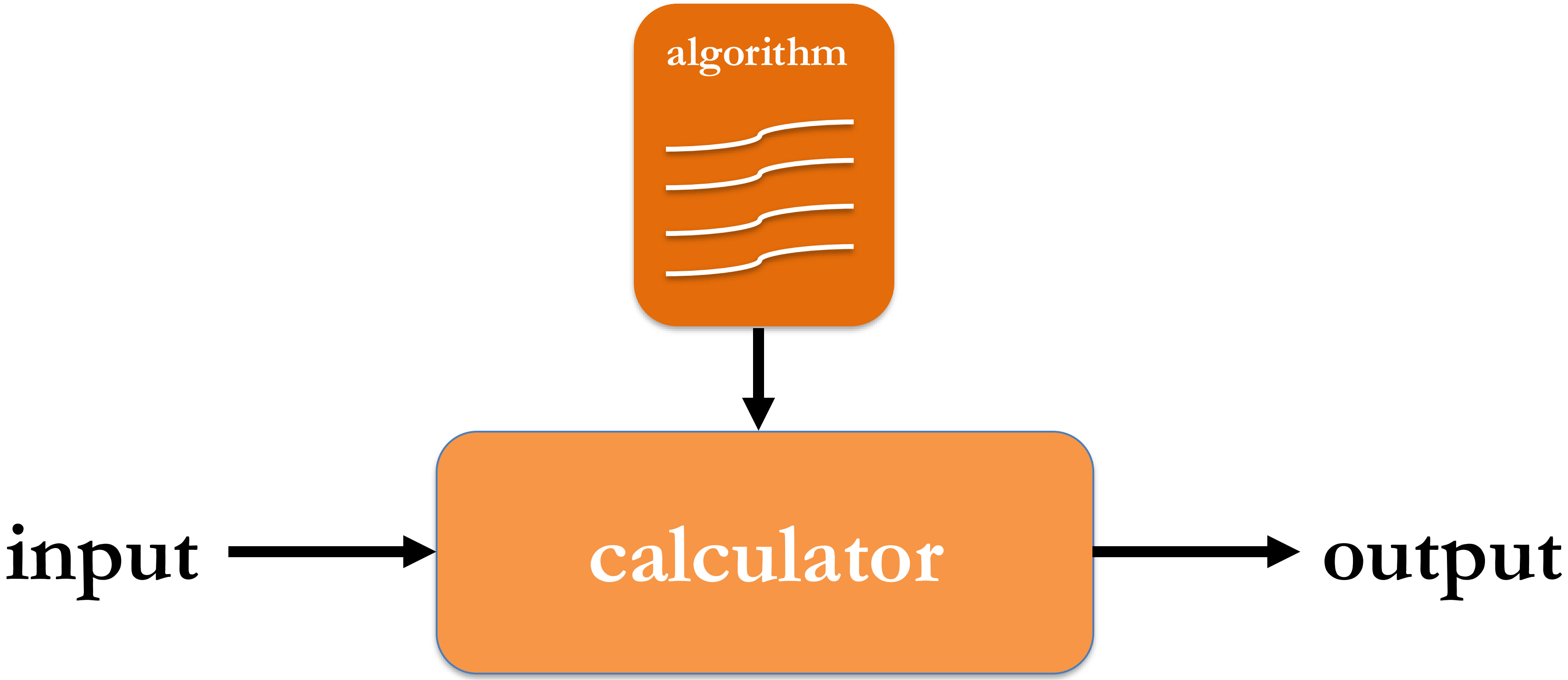
Computational problem: the input/output pairs



Computation: manipulation of information/data

Algorithm: rigorous description of *how* the data is manipulated

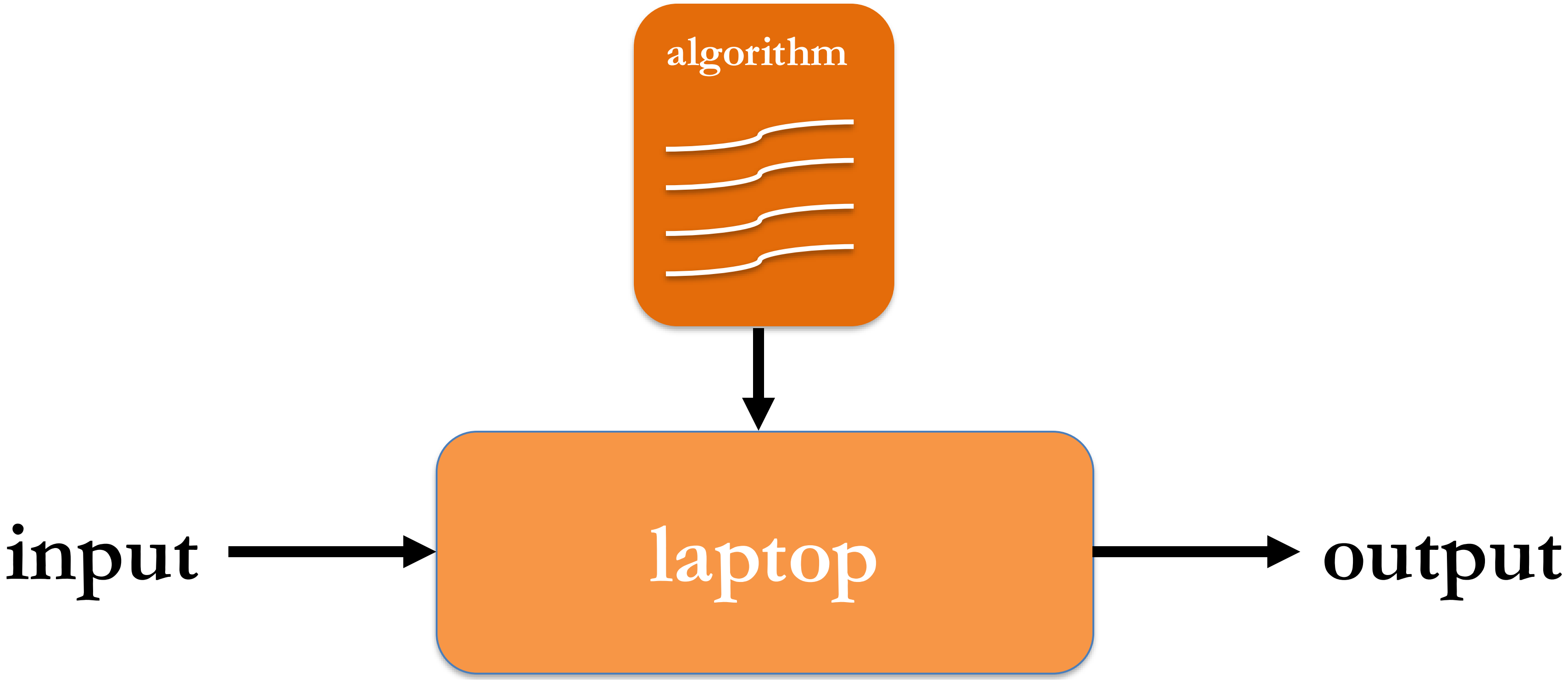
Computational problem: the input/output pairs



Computation: manipulation of information/data

Algorithm: rigorous description of *how* the data is manipulated

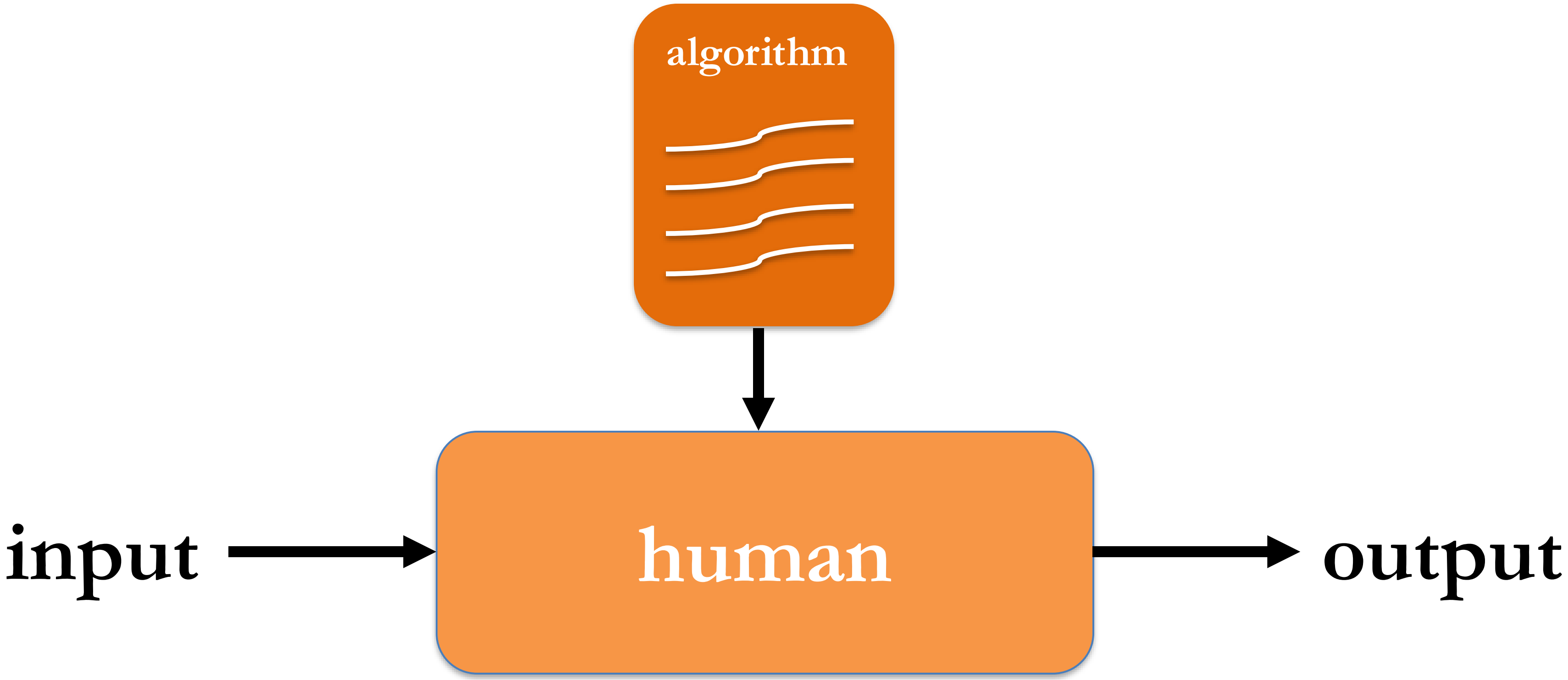
Computational problem: the input/output pairs



Computation: manipulation of information/data

Algorithm: rigorous description of *how* the data is manipulated

Computational problem: the input/output pairs



com·put·er:

1. a programmable machine which performs computations and calculations

2. a human employed to perform computations and calculations

(usage 1: roughly 1897–present)

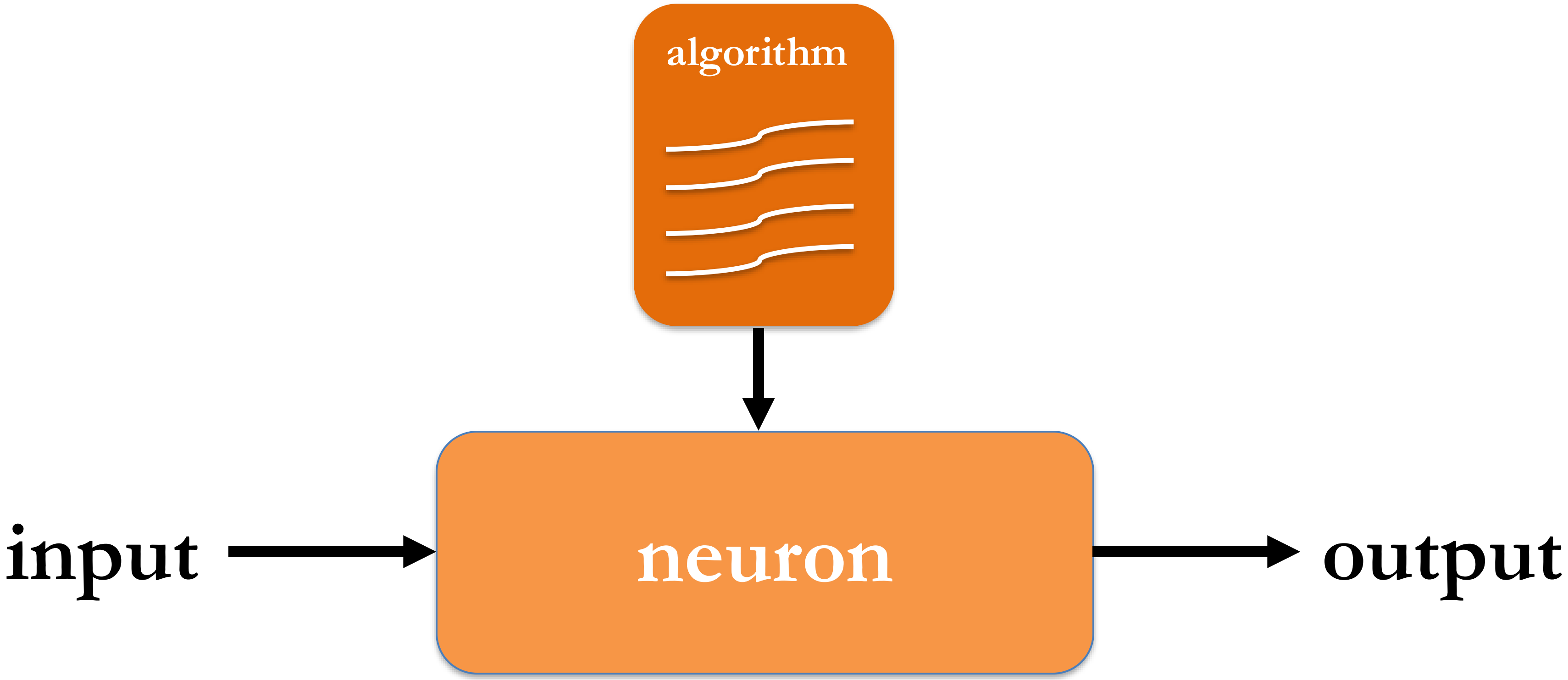
(usage 2: roughly 1613–1945)


FRIDEN

Computation: manipulation of information/data

Algorithm: rigorous description of *how* the data is manipulated

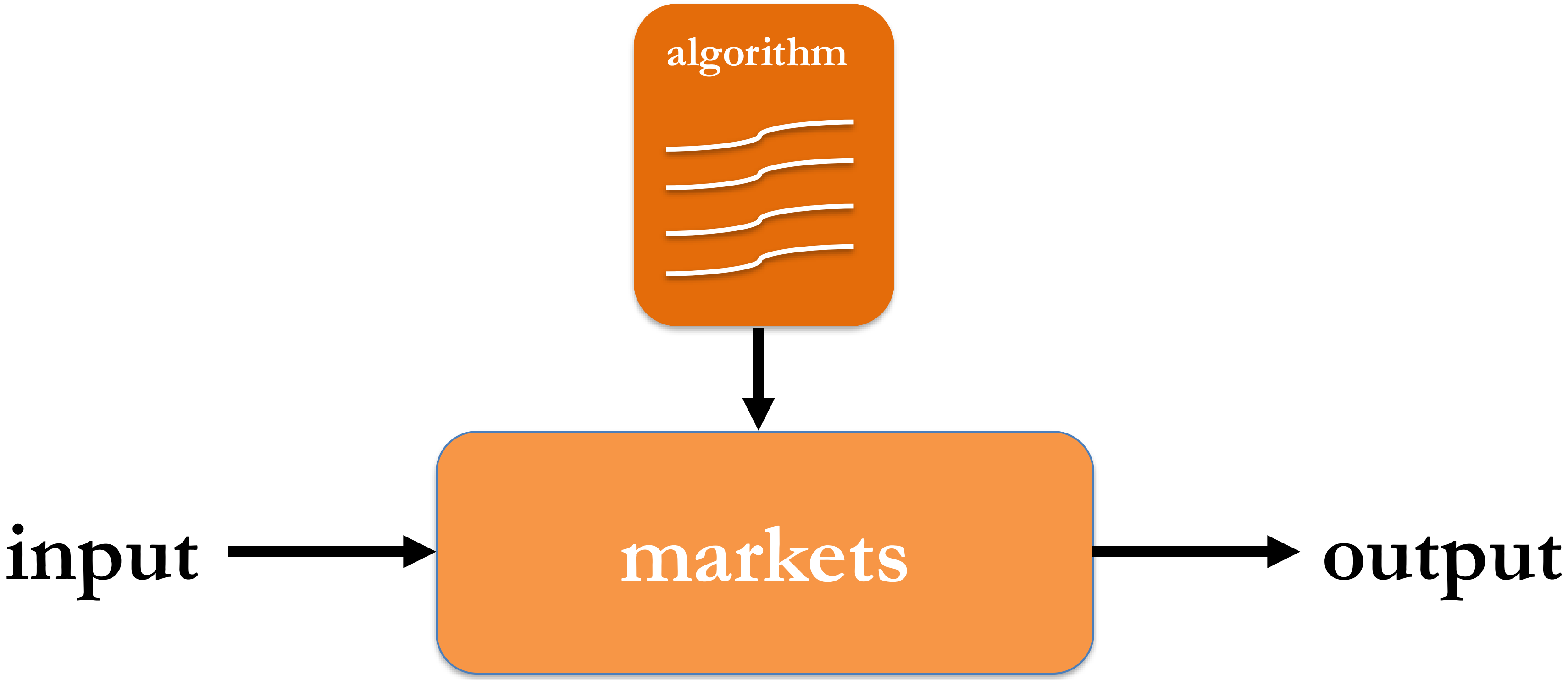
Computational problem: the input/output pairs



Computation: manipulation of information/data

Algorithm: rigorous description of *how* the data is manipulated

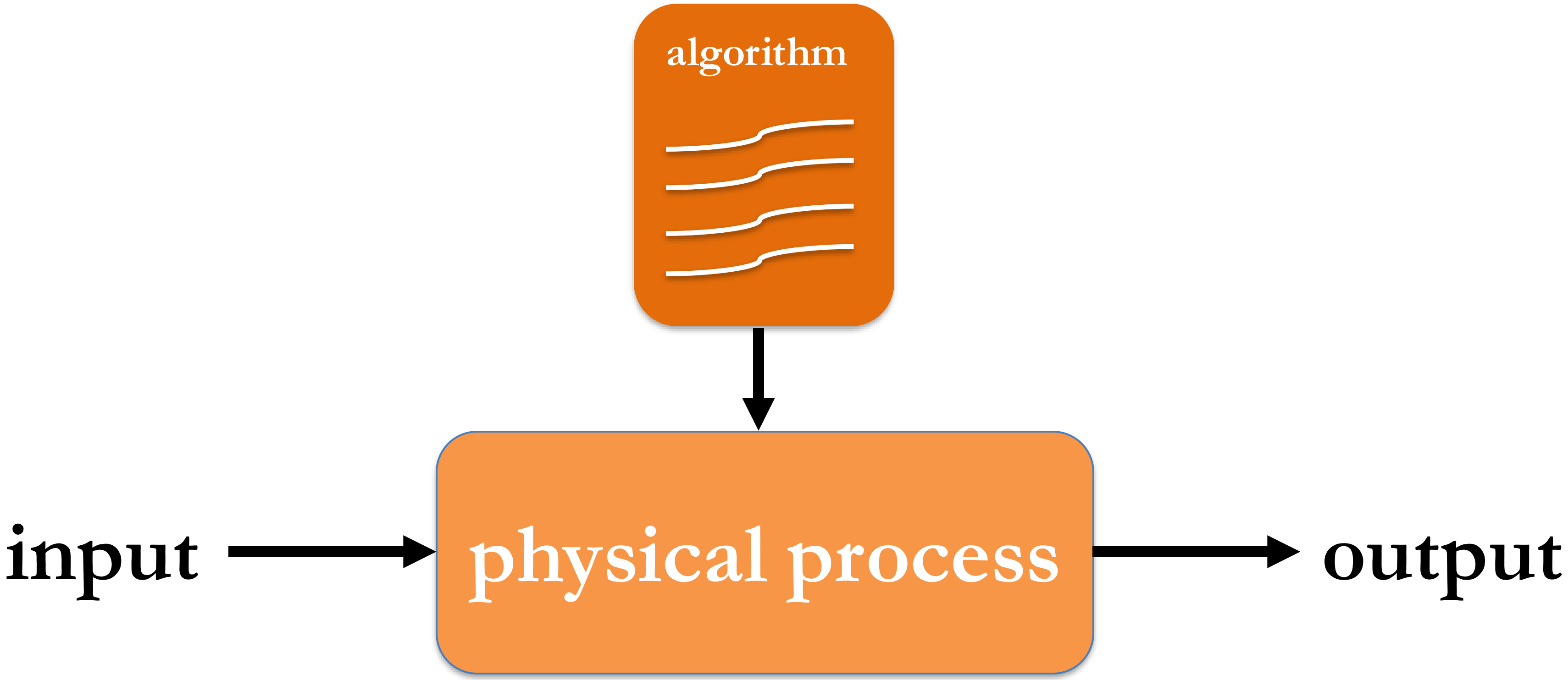
Computational problem: the input/output pairs



Computation: manipulation of information/data

Algorithm: rigorous description of *how* the data is manipulated

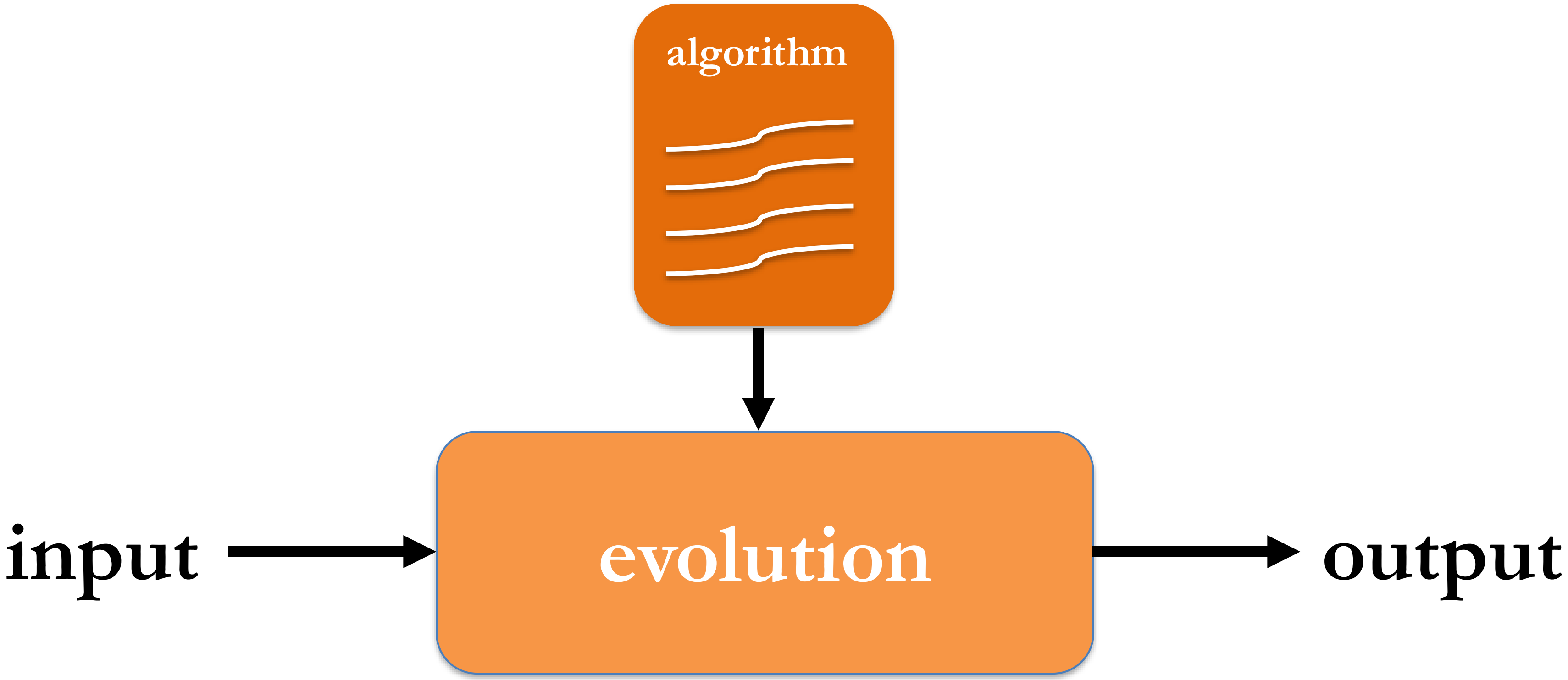
Computational problem: the input/output pairs



Computation: manipulation of information/data

Algorithm: rigorous description of *how* the data is manipulated

Computational problem: the input/output pairs



computational lens

Computational biology

Computational economics

Computational neuroscience

Computational astrophysics

Computational chemistry

Computational finance

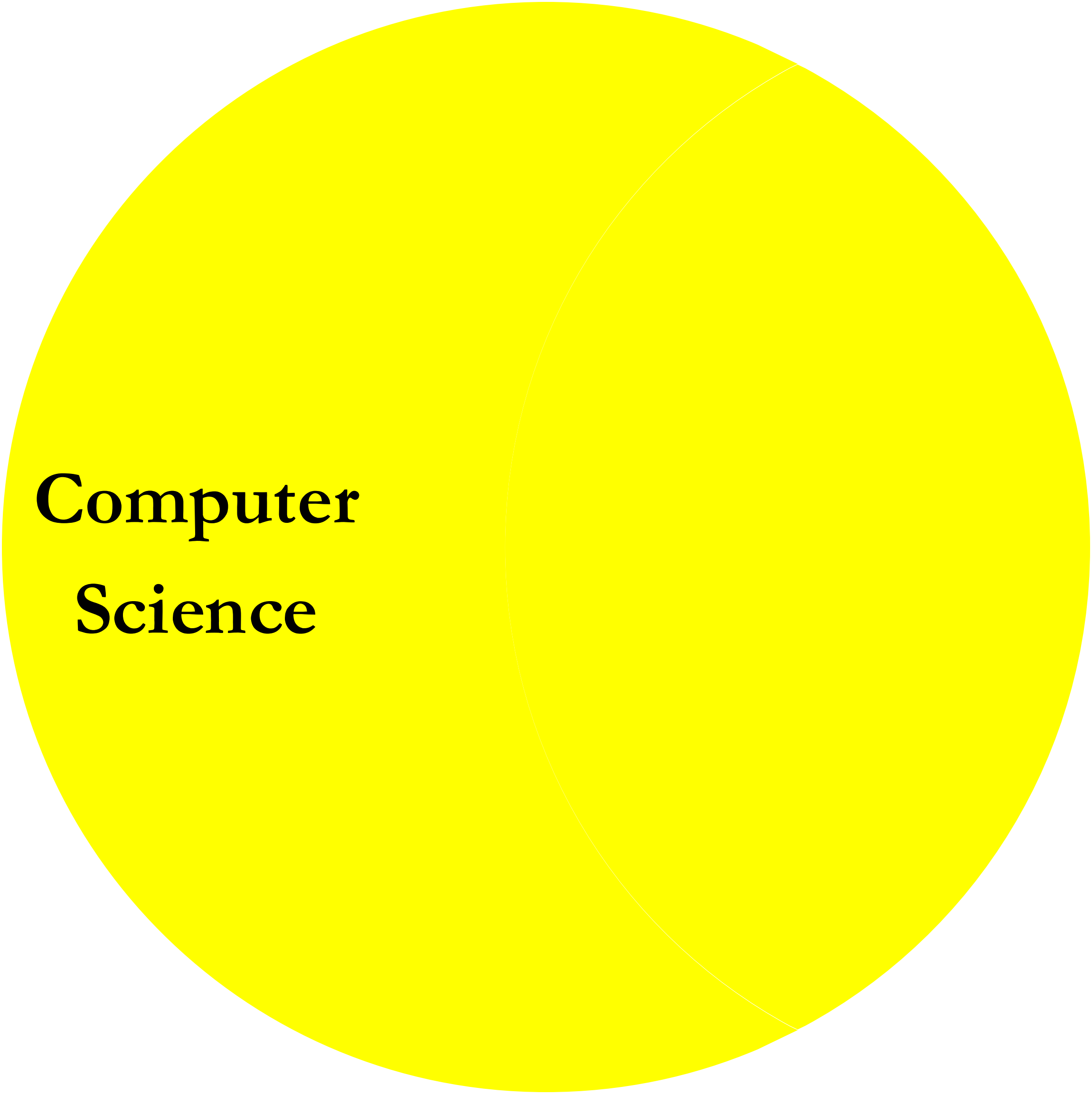
Computational linguistics

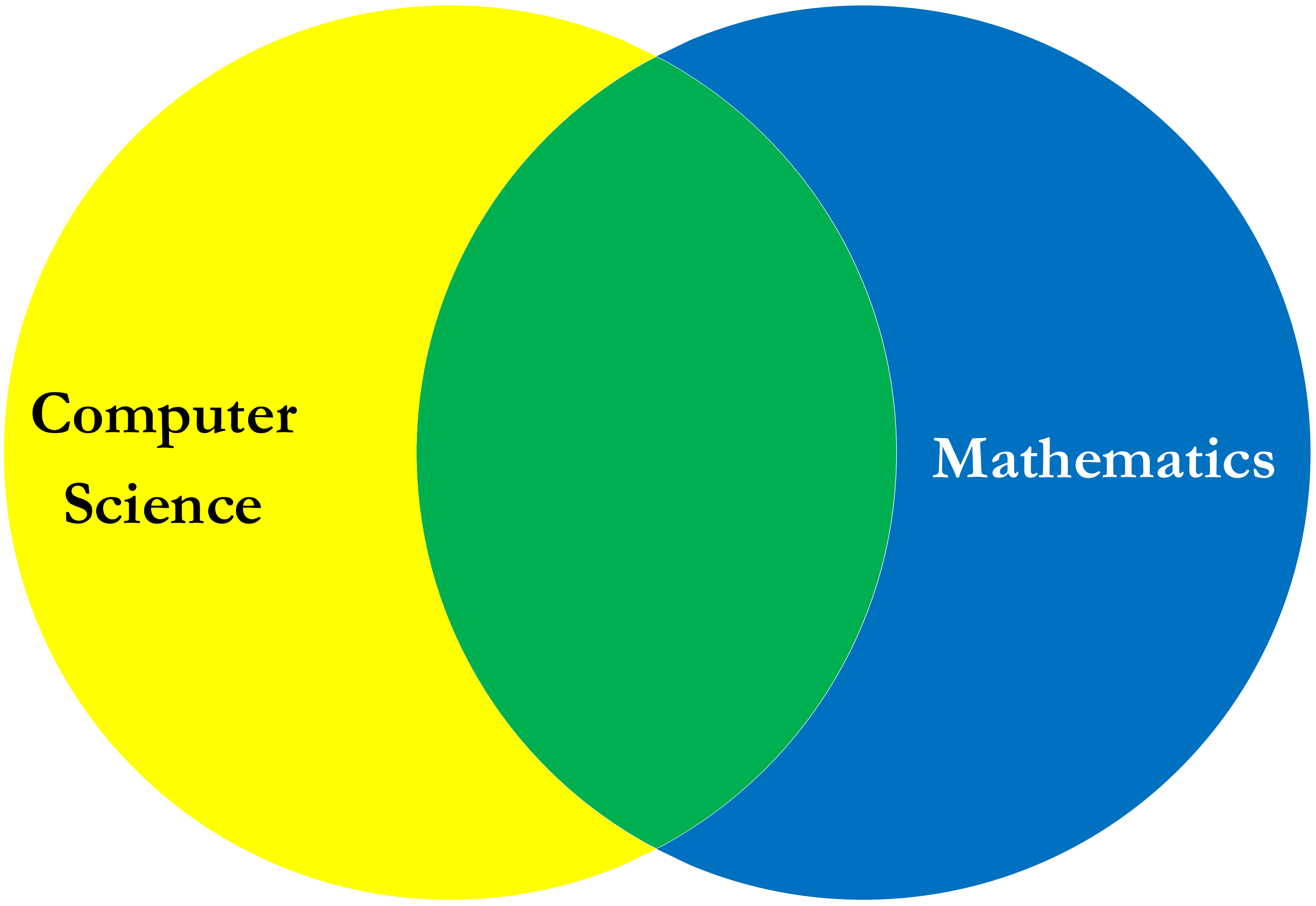
Computational quantum mechanics

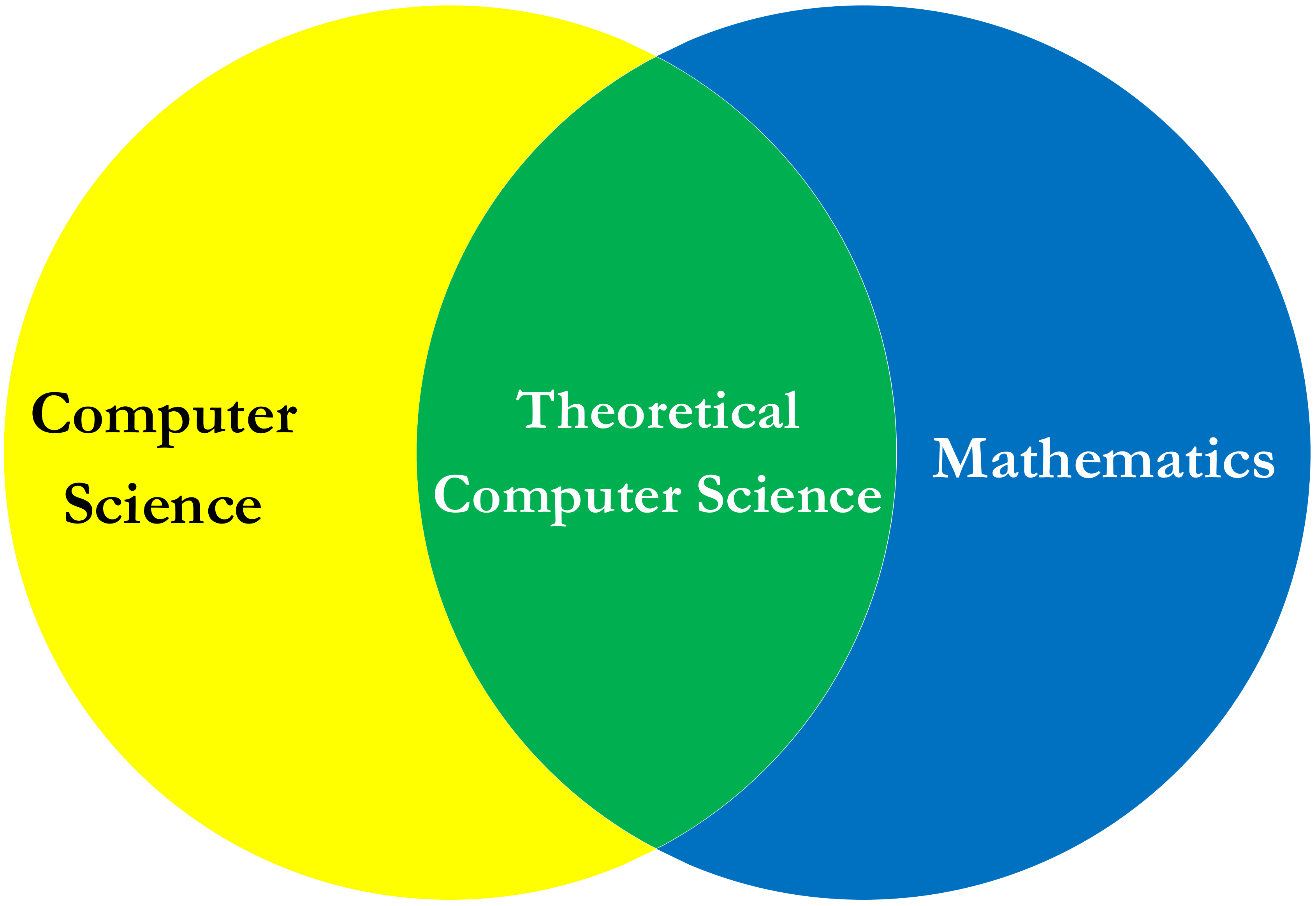
Computational statistics

. . .

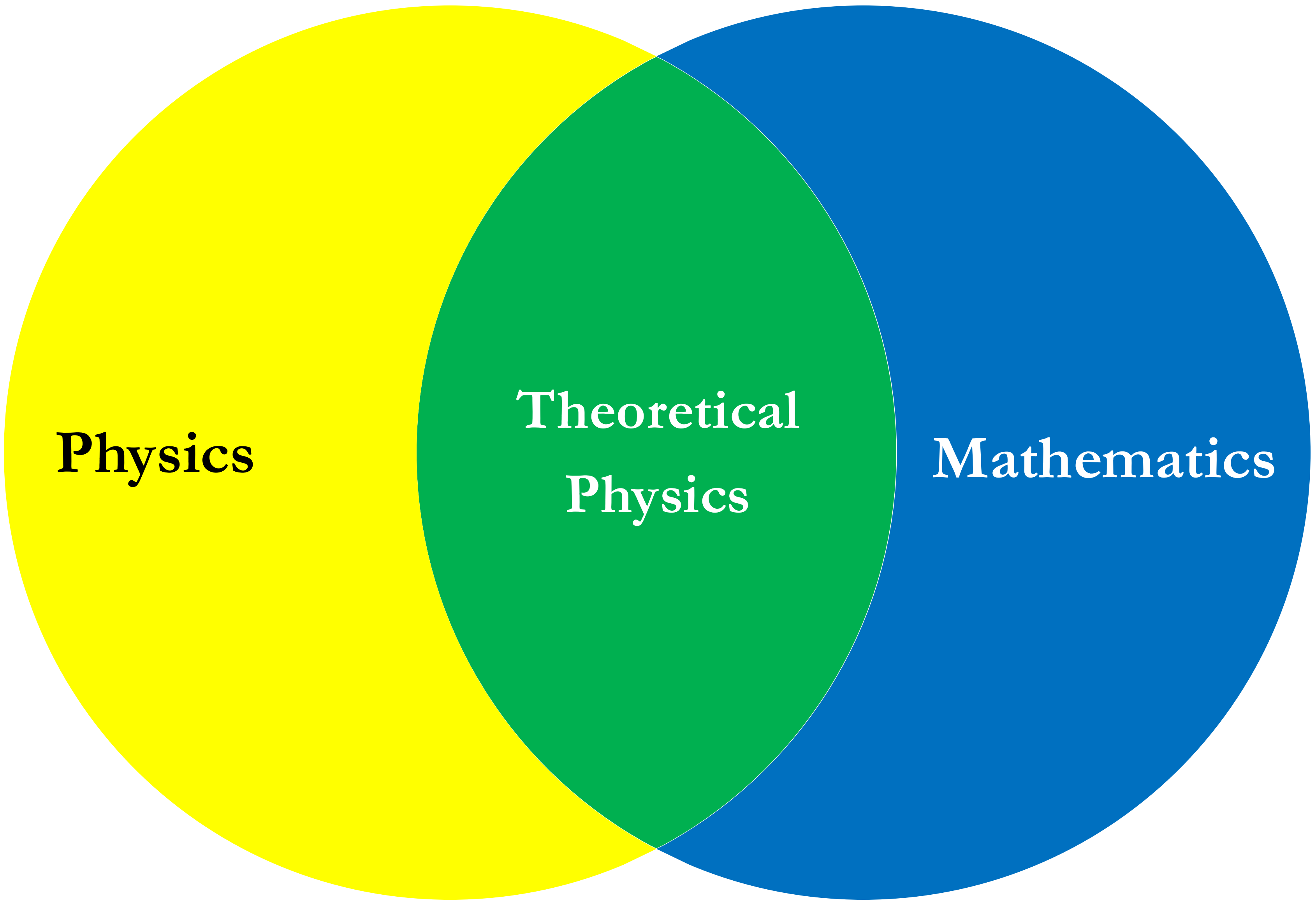
Theoretical Computer Science

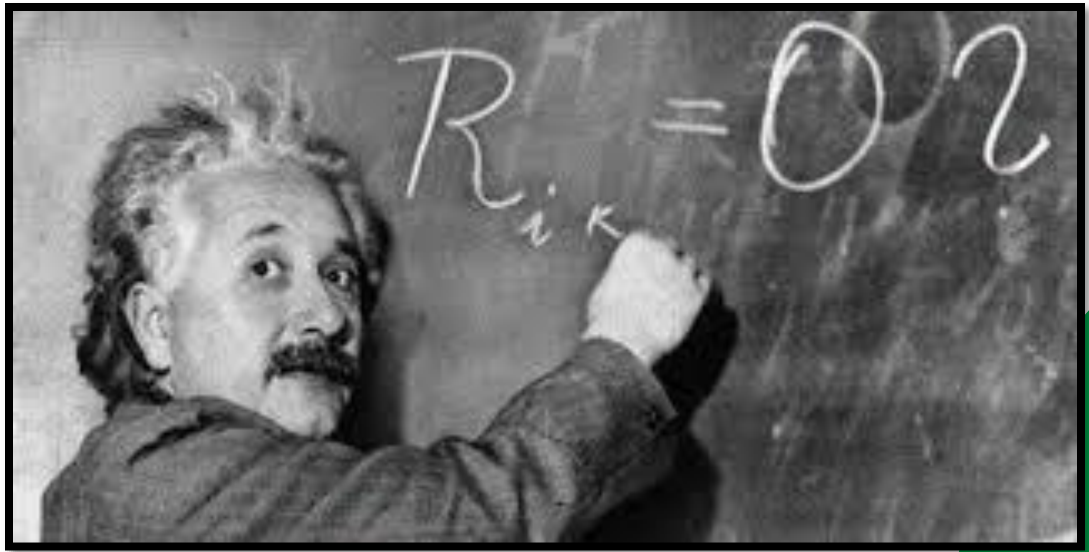

Computer
Science


Computer
Science
Mathematics

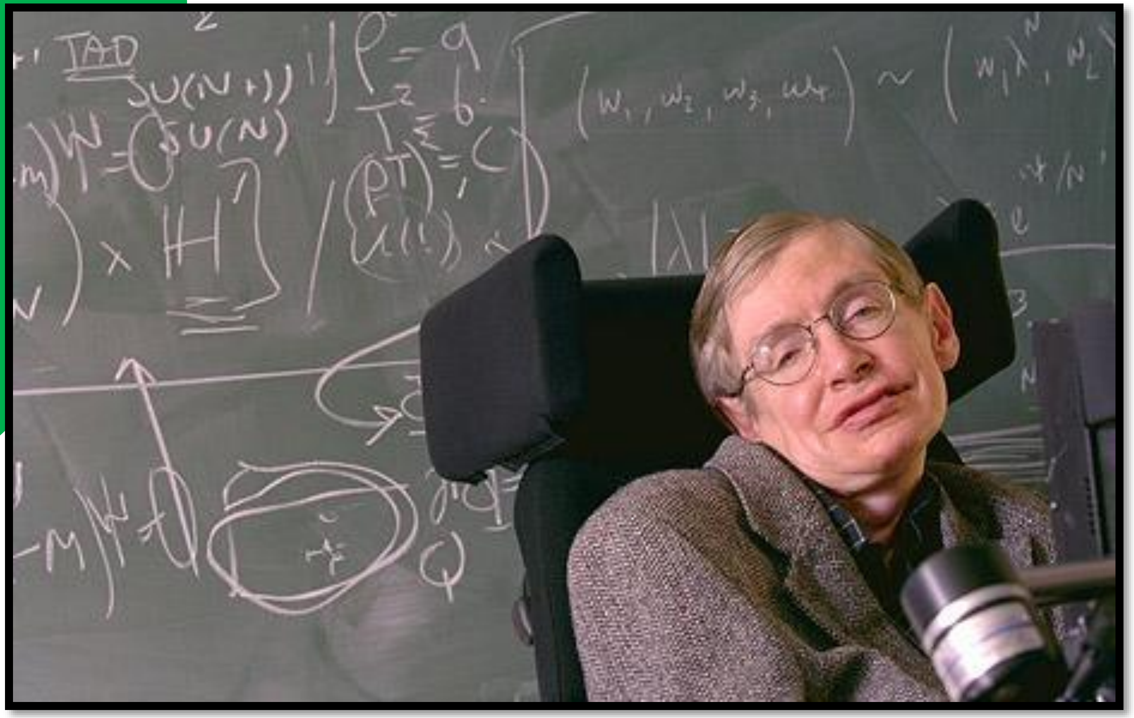

Computer
Science
Theoretical
Computer Science
Mathematics

An analogy with physics may help


Physics
Theoretical Physics
Mathematics



Theoretical Physics



Okay, we don't have everybody

Princeton prof

Theoretical Computer Science

Princeton grad + IAS prof

youtube.com/watch?v=6FDIn4fN2v8

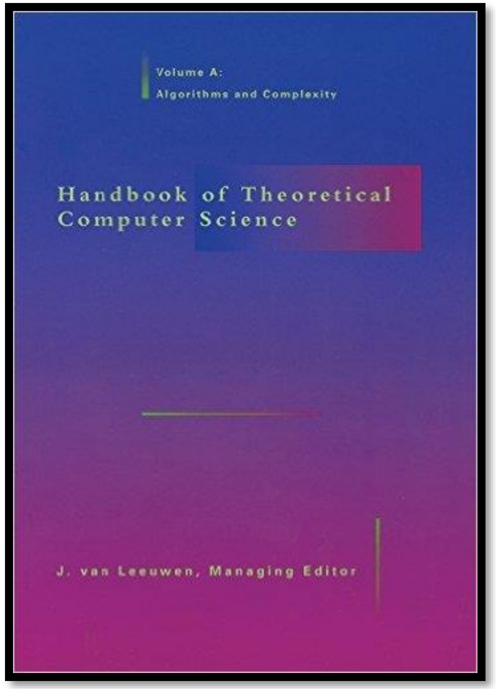
youtube.com/watch?v=6e2k0339lkI

youtube.com/watch?v=TK_vD-VnsFw

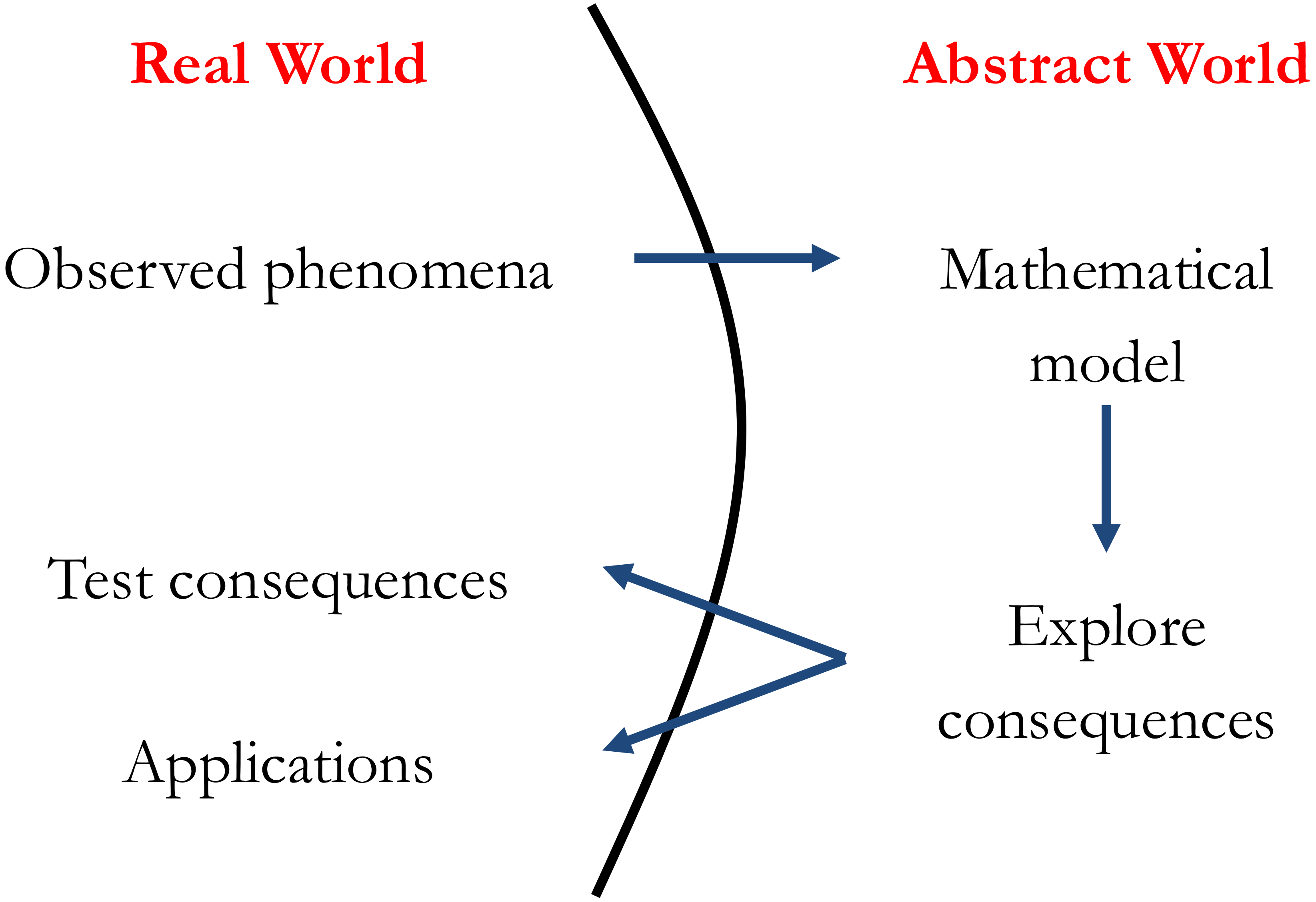
"Theory A": algorithms & complexity

"Theory B": formal methods, logic, PL

for the purposes of this course



the role of theoretical physics



the role of theoretical computer science

Real World

Abstract W

computation

recent!

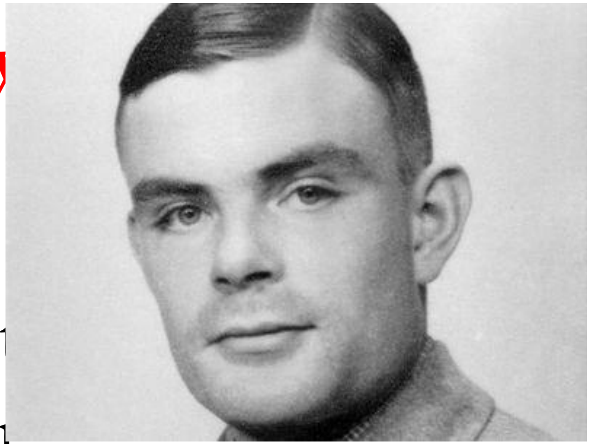
mathemat

model

explore

consequences

applications

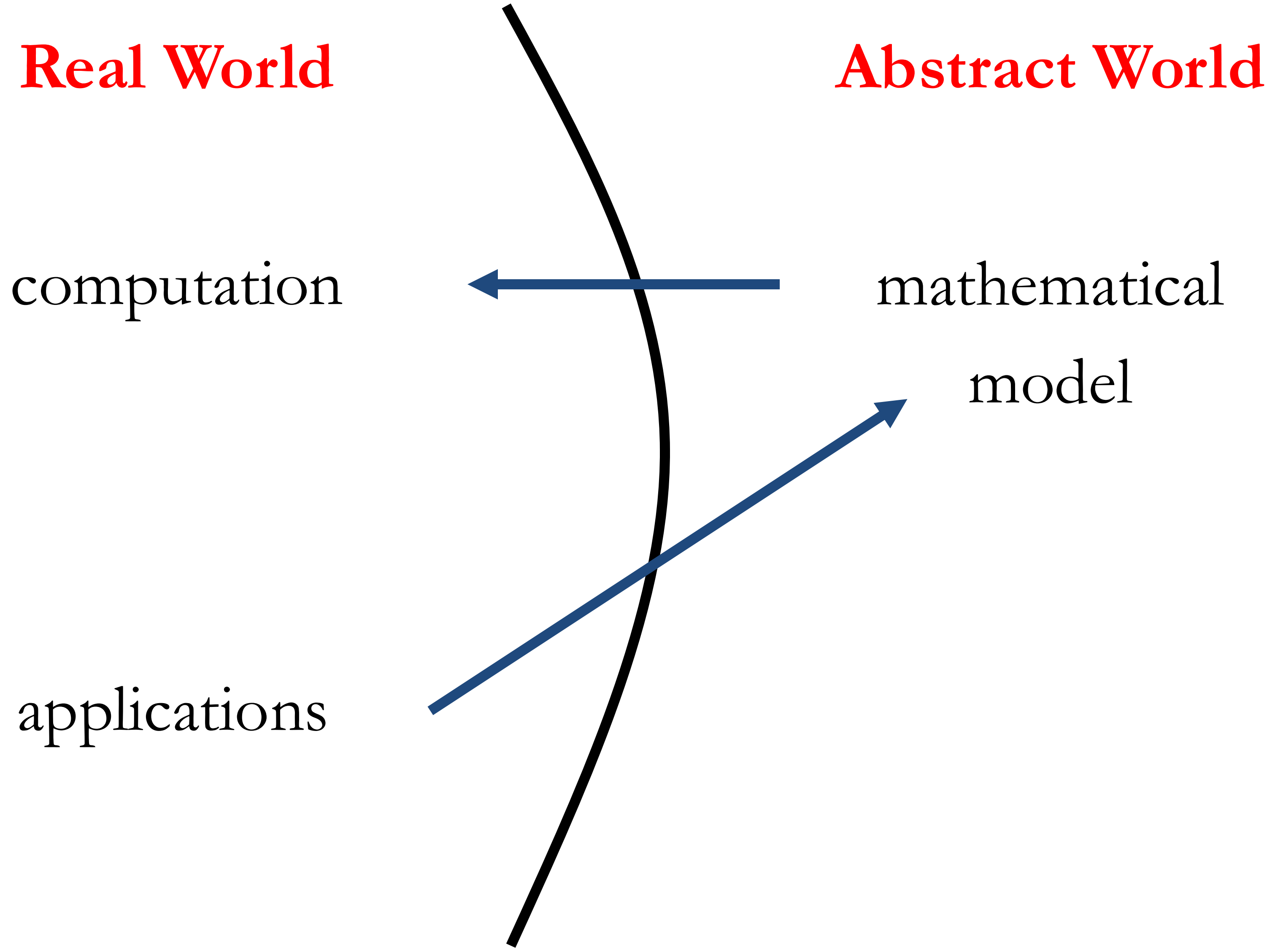


Alan Turing

Princeton grad

class of 1938

the role of theoretical computer science



What is this course about?

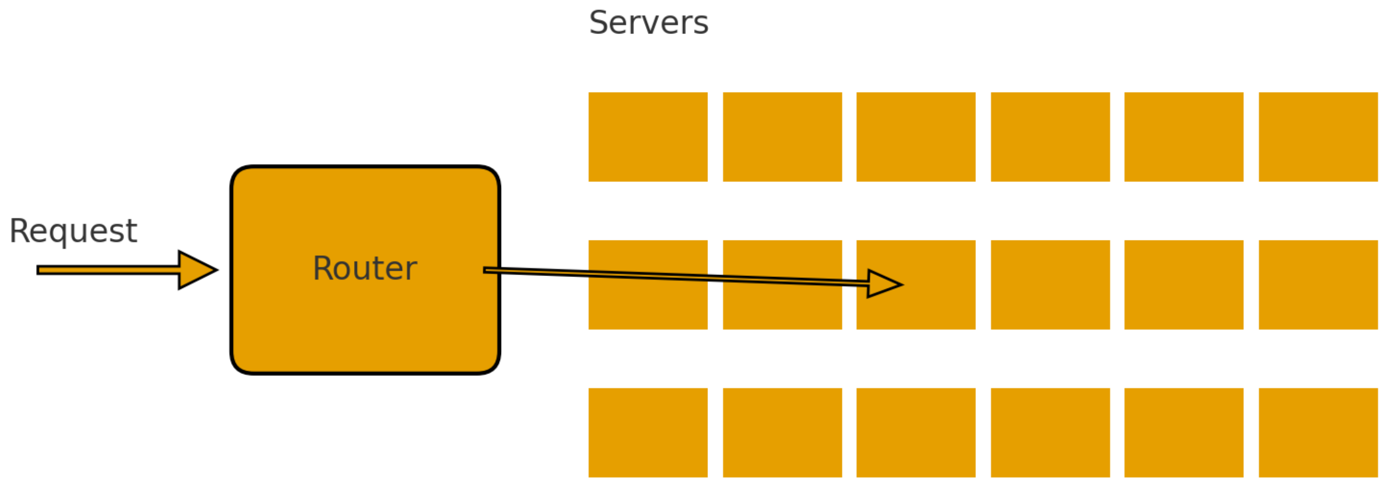
powerful, broadly impactful theoretical ideas

Let me give an example

Let’ say you run a large-scale AI bot service like ChatGPT.

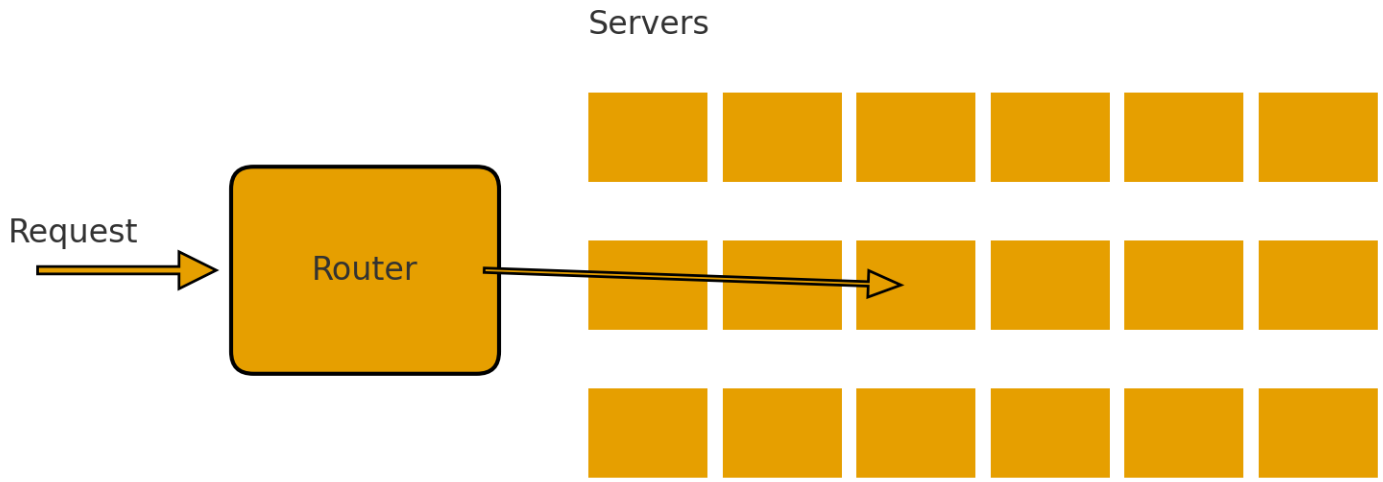
Many servers…to address incoming prompts/requests.

A new request arrives. Which server should this be “routed to” for service?



Dynamic Resource Allocation

A new request arrives. Which server should this be "routed to" for service?



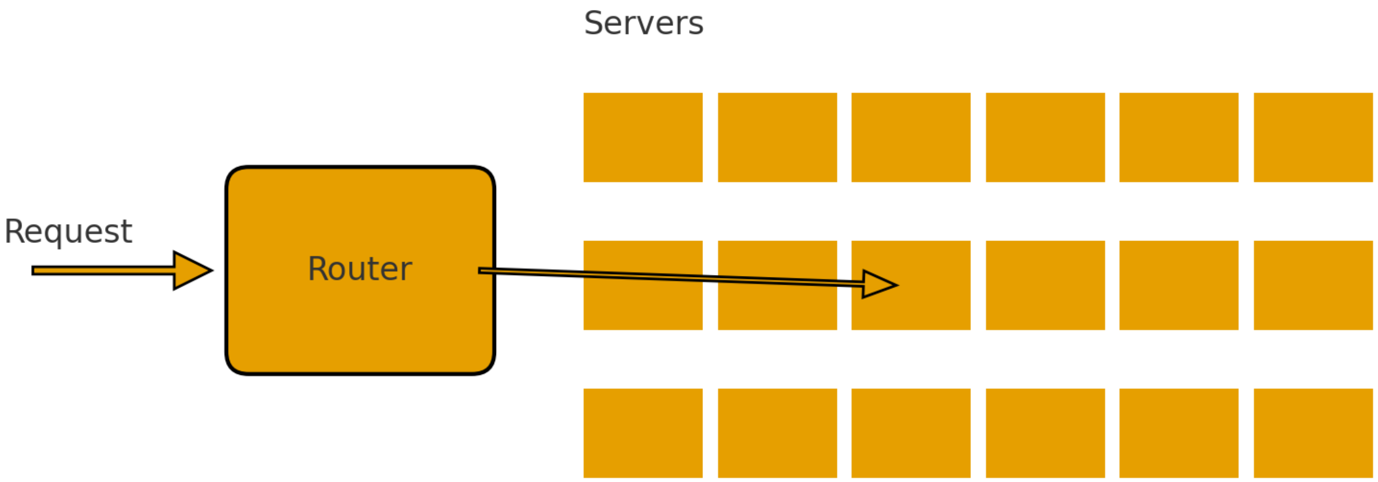
Easy. The least loaded one.

Minimizes "latency".

But…load at servers is *dynamic*. Must query each server for its current load for every request…. horrible latency.

Dynamic Resource Allocation

A new request arrives. Which server should this be "routed to" for service?



~~Easy. The least loaded one.~~

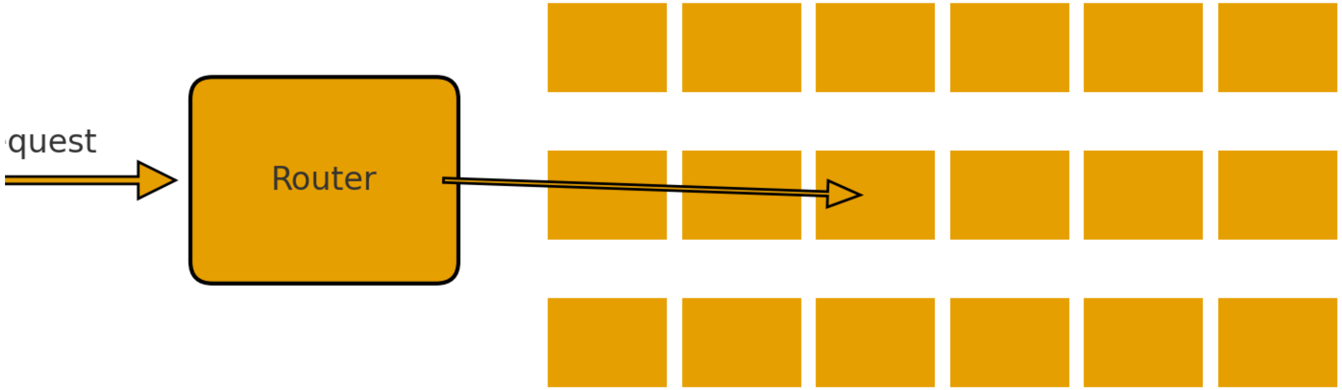
~~Minimizes "latency".~~

Idea Route it to a *random* server.

\+ Super lightweight. No queries to servers!

How good is it?

Dynamic Resource Allocation: Random Choice!



A (simple) theoretical model

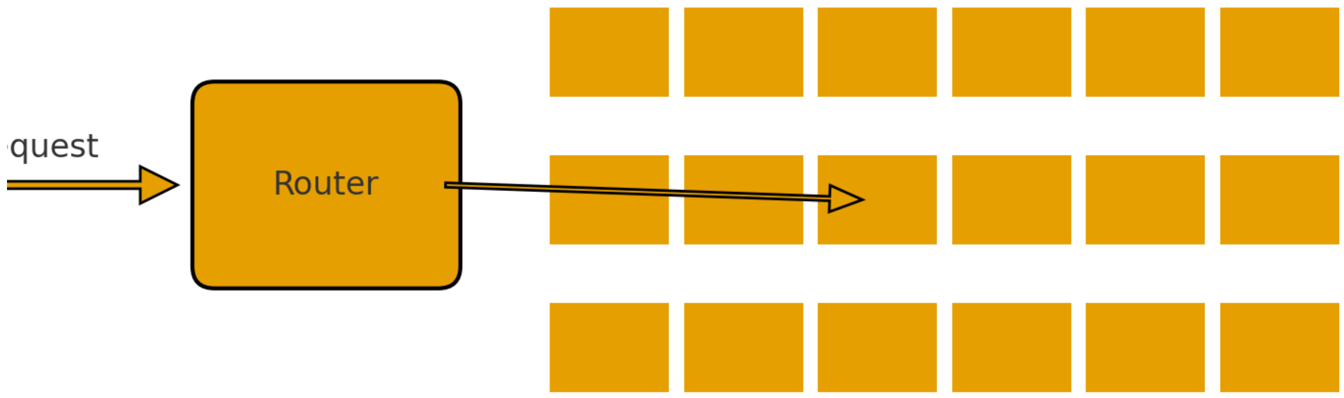
n servers. n requests.

What's the maximum "load" on any server? Proxy for max latency.

Idea Route it to a *random* server.

+ Super lightweight. No queries to servers!

Dynamic Resource Allocation: Random Choice!



A (simple) theoretical model

n servers. n requests.

What's the maximum "load" on any server? Proxy for max latency.

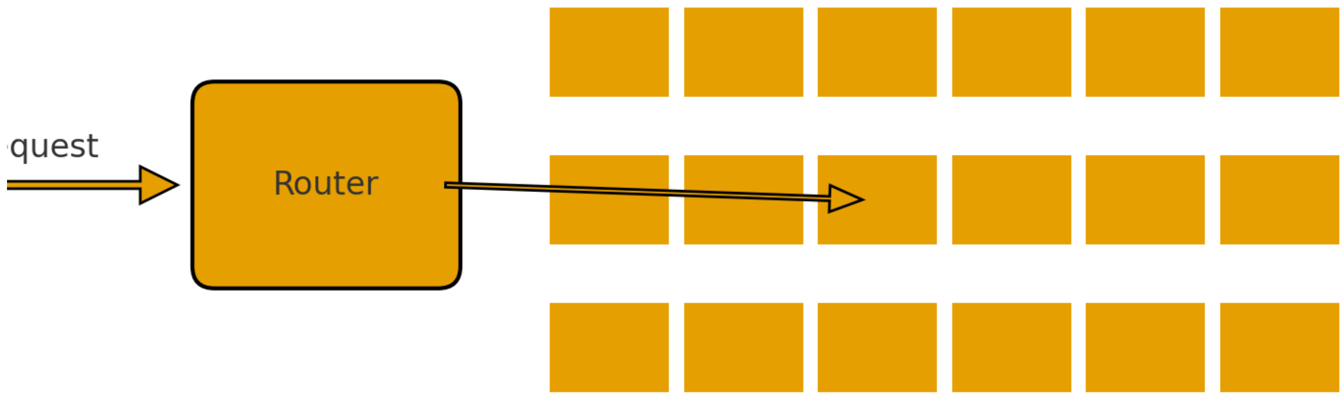
Label the servers 1,2,…,n.

the chance that server j receives at least k requests? $\leq \binom{n}{k} \cdot \frac{1}{n^k} \sim \frac{1}{k!}$.

For $k \sim (\log n)/\log\log n$, this quantity is $\ll 1/n^2$.

By a union bound with prob $1 - 1/n$, no server has load $\gtrsim \frac{\log n}{\log \log n}$.

Dynamic Resource Allocation: Random Choice!



A (simple) theoretical model

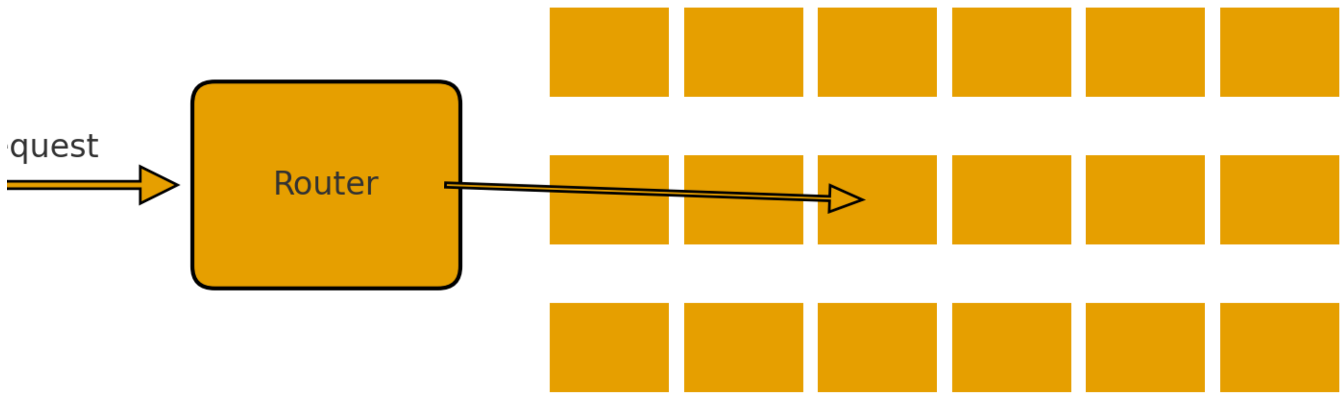
n servers. n requests.

What's the maximum "load" on any server? Proxy for max latency.

With prob $1 - 1/n$, no server has load $\gtrsim \frac{\log n}{\log \log n}$.

Cloudflare handles ~45 million http requests a second and designed for 100s of millions of requests per second. So, a max load of ~6-8. Not bad!

Dynamic Resource Allocation: Can we improve the simple idea?



A (simple) theoretical model

n servers. n requests.

Idea 2 Query **two** *random* servers and route to the one with smaller load.

+ Still quite lightweight. Just two queries!

How much can it help?

The Power of Two Choices

BALANCED ALLOCATIONS*

YOSSI AZAR†, ANDREI Z. BRODER‡, ANNA R. KARLIN§, AND ELI UPFAL¶

Abstract. Suppose that we sequentially place n balls into n boxes by putting each ball into a randomly chosen box. It is well known that when we are done, the fullest box has with high probability $(1 + o(1)) \ln n / \ln \ln n$ balls in it. Suppose instead that for each ball we choose two boxes at random and place the ball into the one which is less full at the time of placement. We show that with high probability, the fullest box contains only $\ln \ln n / \ln 2 + O(1)$ balls—exponentially less than before. Furthermore, we show that a similar gap exists in the infinite process, where at each step one ball, chosen uniformly at random, is deleted, and one ball is added in the manner above. We discuss consequences of this and related theorems for dynamic resource allocation, hashing, and on-line load balancing.

Symposium on Theory of Computing, STOC, 1994.

Idea 2 — Query **two** *random* servers and route to the one with smaller load.

+ — Still quite lightweight. Just two queries!

Load reduces to just $\log \log n$! – Exponentially smaller. For n=100 mill, <3.

The Power of Two Choices

BALANCED ALLOCATIONS[*]

YOSSI AZAR[†], ANDREI Z. BRODER[‡], ANNA R. KARLIN[§], AND ELI UPFAL[¶]

Abstract. Suppose that we sequentially place n balls into n boxes by putting each ball into a randomly chosen box. It is well known that when we are done, the fullest box has with high probability $(1 + o(1)) \ln n / \ln \ln n$ balls in it. Suppose instead that for each ball we choose two boxes at random and place the ball into the one which is less full at the time of placement. We show that with high probability, the fullest box contains only $\ln \ln n / \ln 2 + O(1)$ balls—exponentially less than before. Furthermore, we show that a similar gap exists in the infinite process, where at each step one ball, chosen uniformly at random, is deleted, and one ball is added in the manner above. We discuss consequences of this and related theorems for dynamic resource allocation, hashing, and on-line load balancing.

***Symposium on Theory of Computing, STOC*, 1994.**

30 year test of time award in 2024.

Load reduces to just $\log \log n$! – Exponentially smaller. For n=100 mill, <3.

Incredibly impactful: L4/L7 routing in HTTP, RPC clients, shared caches,…

Incidentally, not much gain if you increase the choices to 3,4,…

The Power of Two Choices

Simple model. Simple, powerful (even if counterintuitive) theoretical idea.

This course is about such theoretical ideas.

PS: we don’t expect you to follow the calculation in the previous slide (we will build up to it in a lecture).

COS 330 Topics Overview

Module 1: Classic Algorithms

(fundamental algorithmic techniques, or, COS 226++, rigorous analyses of resources, limits of algorithms)

Two camps:

1. Trying to come up with efficient algorithms (algorithm designers)
2. Trying to show no efficient algorithm exists (complexity theorists).

Multiplying two integers

factoring integers

detecting communities in social networks

DNA sequence alignment

Computing Nash equilibria of a game

COS 330 Topics Overview

Module 1: Classic Algorithms

(fundamental algorithmic techniques, or, COS 226++, rigorous analyses of resources, limits of algorithms)

Module 2: Power of randomization.

(using randomness to improve algorithms, data structures, and more, *randomness is a superpower!*)

Module 3: Optimization

(continuous optimization to solve problems, e.g., linear programming)

Module 4: New Computational Models

(beyond traditional models: e.g., dynamic inputs, data streams, distributed computation…)

Module 5: Information and Coding Theory

(reliable communication through unreliable channels, applications)

Module 6: Wildcard topics!

COS 330 Topics Overview

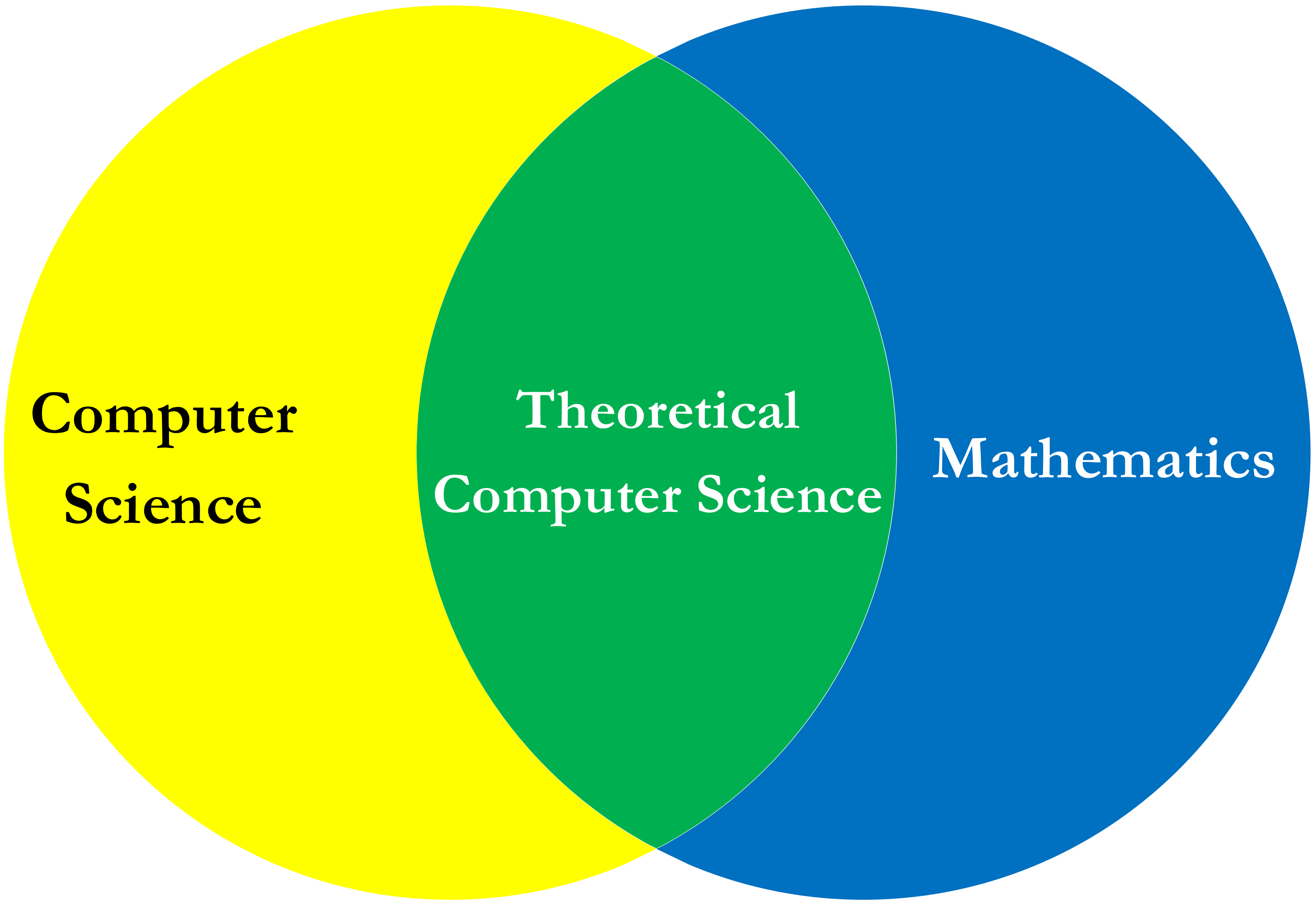
New ambitious course, somewhat experimental in topics/teaching strategy.

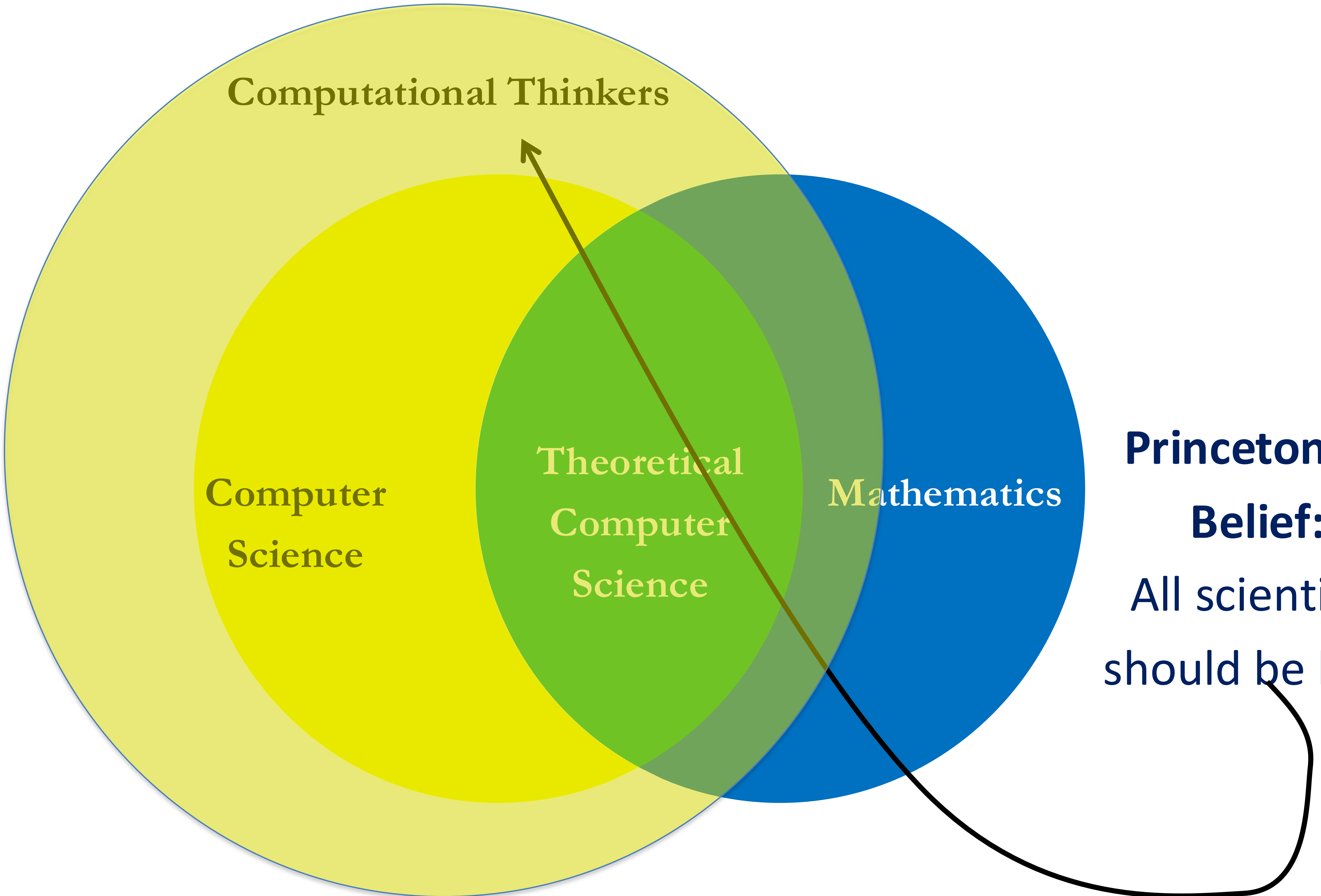
We hope it will be fun learning for all of you.

Let's talk about math

These topics require some math.

Part of the goal of the course is to train you to use math to aid rigorous, logical, abstract thinking.


Computer
Science
Theoretical
Computer Science
Mathematics

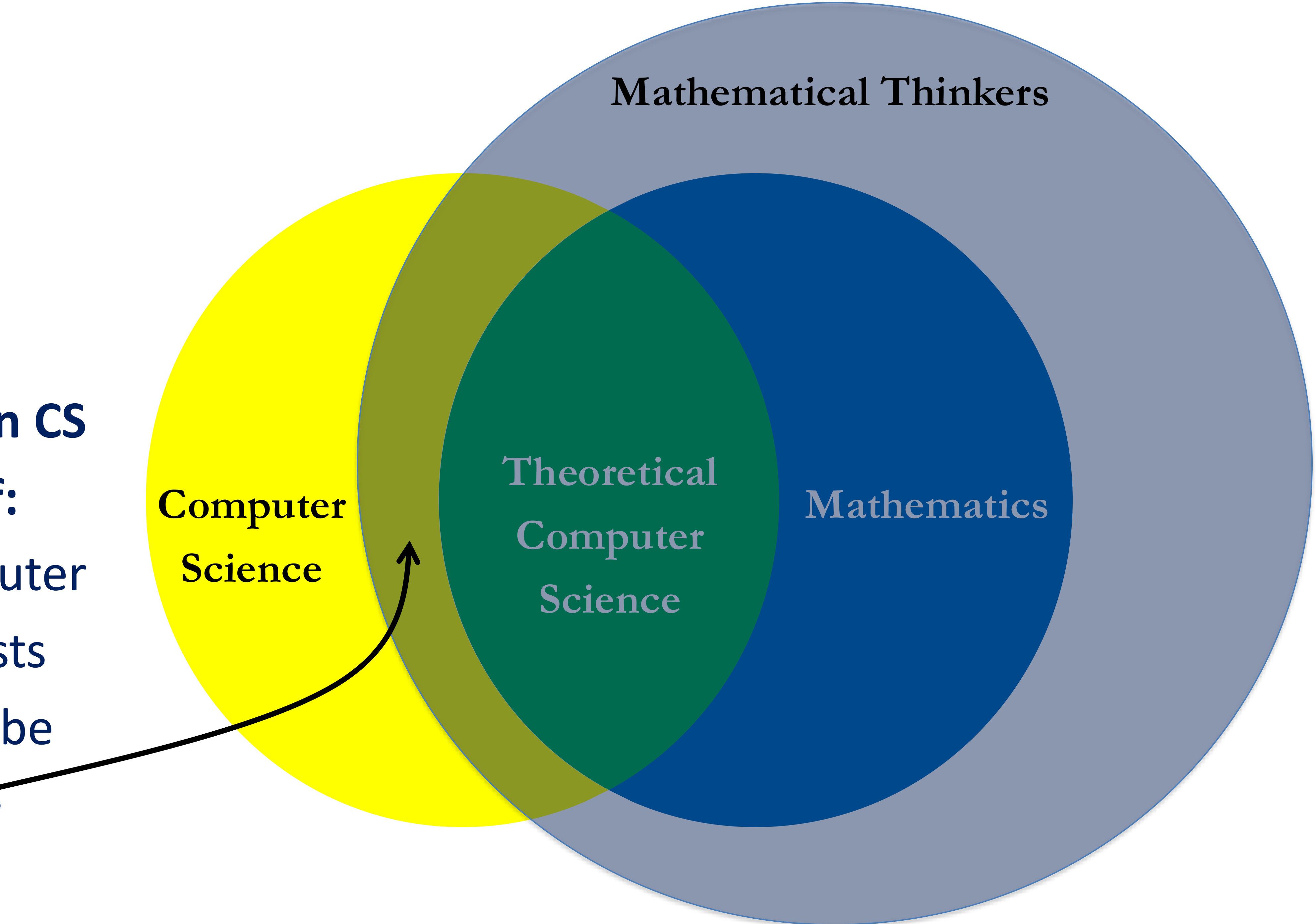


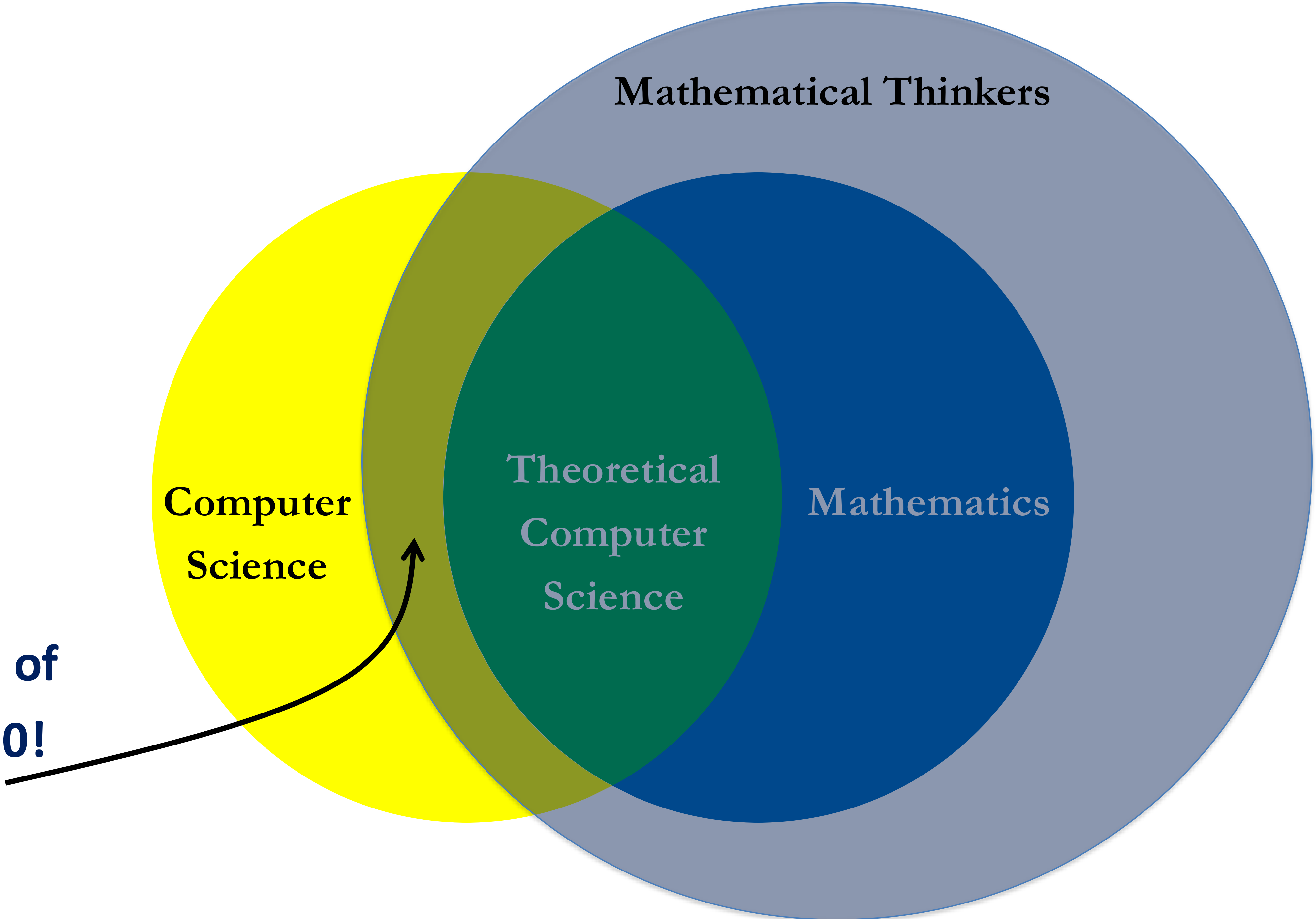
Princeton CS Belief:

All scientists should be here

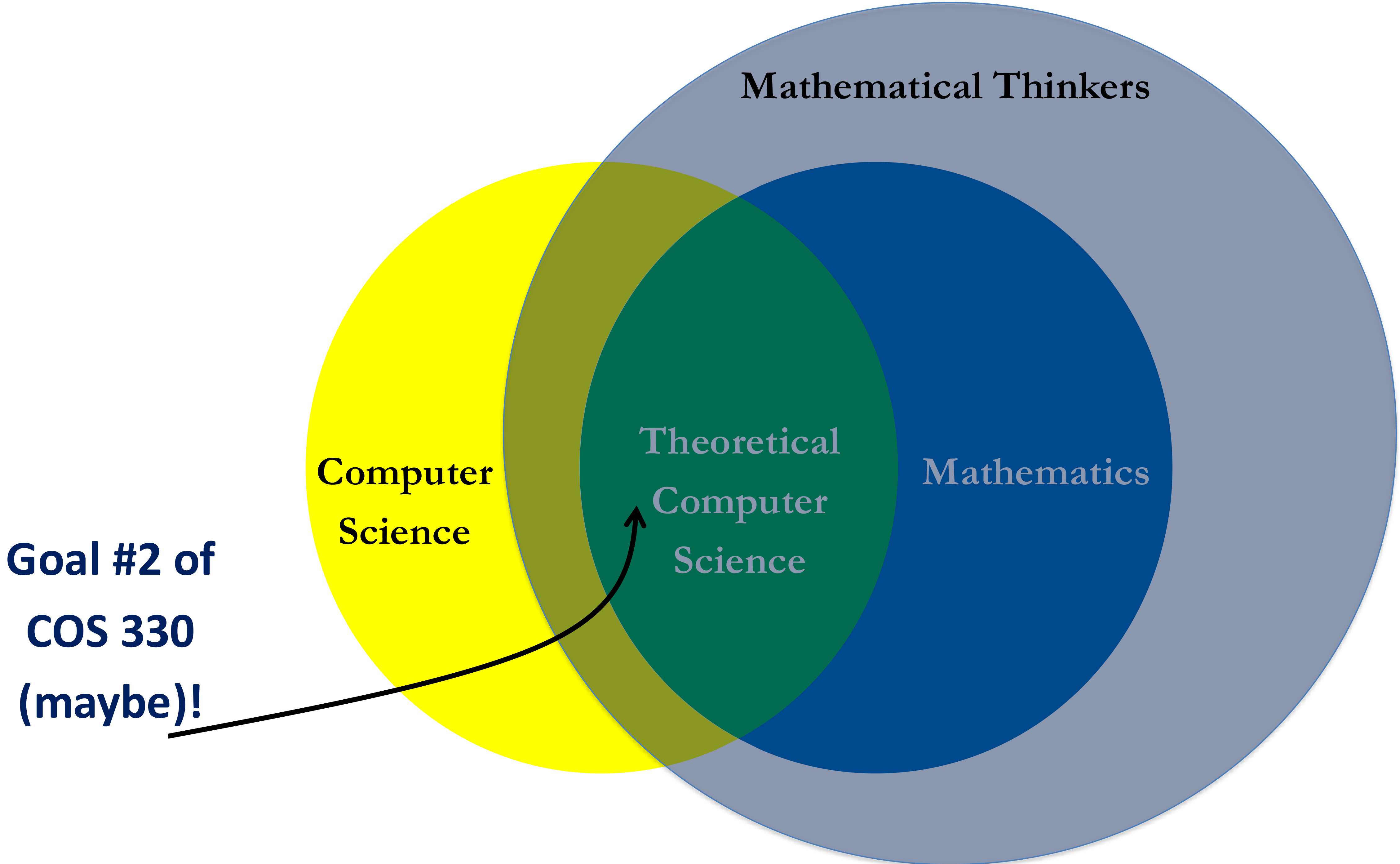
Princeton CS Belief:

All computer scientists should be here





Goal #1 of COS 330!


Mathematical Thinkers
Computer Science
Theoretical Computer Science
Mathematics
Goal #2 of COS 330 (maybe)!

Let’s talk about math

You don’t have to love math/TCS. But if you do, we hope you will like this course.

Not everyone has to be passionate about math of CS, but ideally, all computer scientists are happy to think mathematically.

The course material, precepts, exams are designed to help you with this. Use these resources!

Ryan O'Donnell's metaphor for the
mathematical / theoretical
part of computer science.

Math is like… **cilantro**

There are **5** kinds of people
when it comes to cilantro.

1. Those who don't know what cilantro is

1. Those who don't know what cilantro is

1. Those who don’t know what cilantro is

Coriandrum sativum

Coriander (leaves)

香菜

धनिया

고수

گشنیز

الكزبرة

ngò

φύλλα κόλιανδρου

கொத்தமல்லி

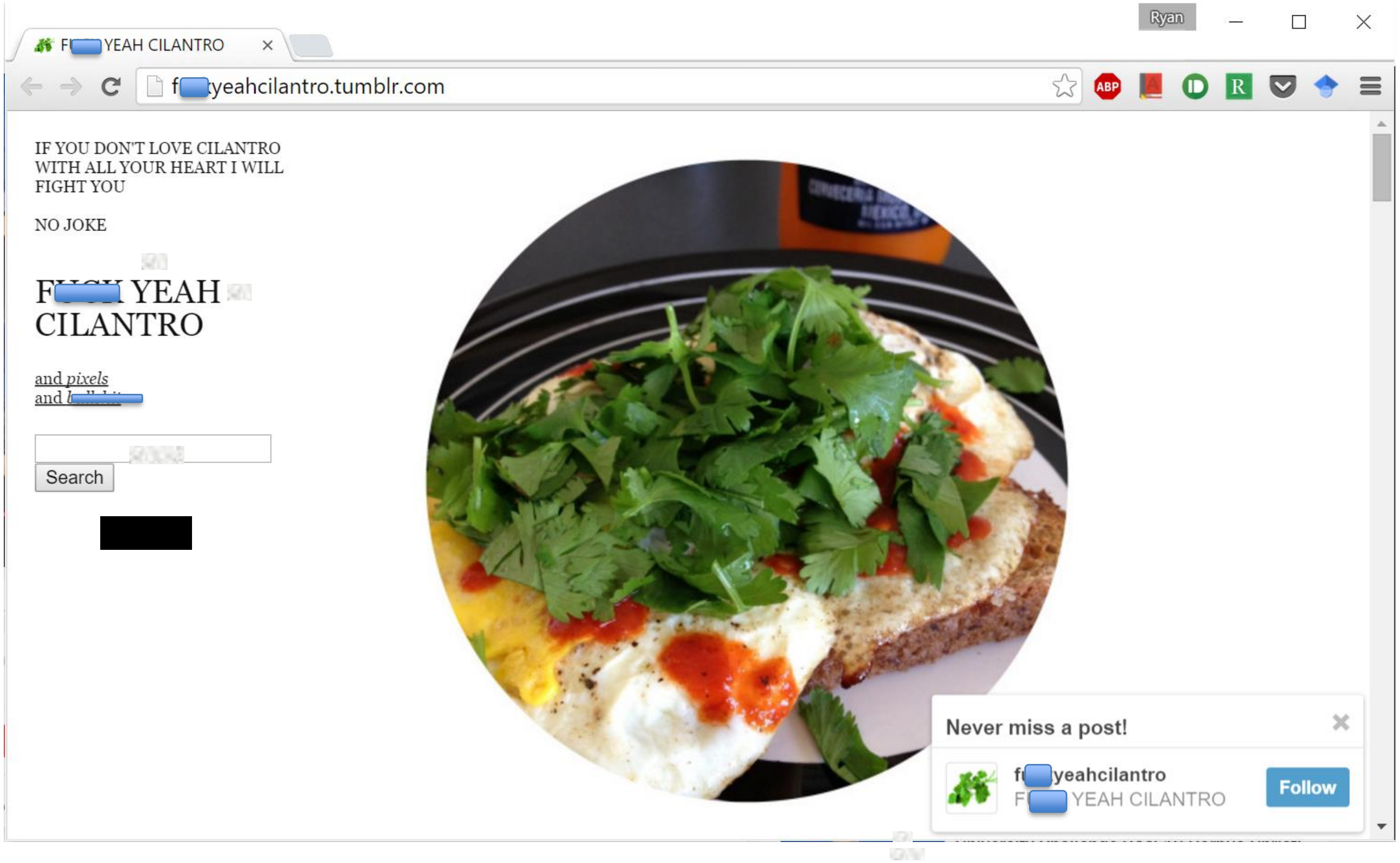
ধনে

kişniş

кинза

כוסברה

2. People who LOVE cilantro



2. People who LOVE cilantro

You don’t have to LOVE math/TCS.

But if you do, we hope you will like this course.

3. People who think cilantro is fine

Our dream is that everyone is in this category
at the end of COS 330.

Not everyone has to be passionate about
the math of CS, but ideally all
computer scientists are happy to think
mathematically.

3. People who don’t like cilantro

We want you to try it.

We’ll try to show you some of the tastiest math dishes.

Hope you can eat it if necessary.

4. People with a genetic condition that makes cilantro taste like soap

I'm not 100% sure this really exists.

I sympathize, but you still have to eat a little.

Course Logistics

Lectures

Section	Time	Location	Instructors
L01	MW 10:40am-12:00pm	Arch Bldg N101	Pravesh/Pedro/Matt

Slides to be posted shortly after lecture.

Precepts

Section	Time	Location	Instructor
P01	Thursday 1:20pm - 2:10pm	Friend 109	Ilya
P04	Friday 1:20pm - 2:10pm	Friend 110	Daniel
P05	Friday 1:20pm - 2:10pm	Friend 007	TBD

Precept Format:

- **Learn:** Extension of lecture material
- **Practice:** Exam-style problems to solve with guidance
- **Challenge:** (Optional) Harder problems for extra practice

Problem Sets and Exams

- **Problem Sets:** 8 PSets + 1 warmup (i.e., PSet 0)
- **Release Dates:** Typically Mondays
- **Exams:** Midterm in-class (Oct 6, Monday) and Final in-person (Dec 16, Tuesday)

Schedule:

PSet 0 is a warmup ▪ PSets released on Mondays ▪ M = Midterm, F = Final

Coaching

Problem Sets Are Not graded – purpose is exam preparation

- You won't submit your PSet solutions, and we won't do any traditional grading
- **Fully open**: AI, collaboration, etc. allowed

New course element: one-on-one coaching

Coaching Model: Weekly 30-minute 1-on-1 with TA/UCA

- **Coaching Credits:** Earn up to 8 credits for engaged participation (worth 40% of final grade)
- **Coaching dates:** Typically Tuesdays and Wednesdays (to be scheduled), starting next week

We highly recommend you attempt the PSet exactly how you would in any other class for a grade, and then use coaching to get feedback

Suggested Weekly Format

Before the Coaching Session:

- Attempt the PSet exactly how you would in any other class
- When stuck, use {grind it out, work with friends, office hours, ask AI}
- **When "done," write up your solution as if submitting for a grade**

During the Coaching Session:

- Take your written PSet to your Coach
- Ask them to live-grade the problem you're least certain about
- Ask for honest but kind feedback on your thought process

End of Session (last 5 minutes):

- Brief self-assessment: what's going well, what to improve?
- Make a plan: what will you do differently next week?

You'll spend a few minutes in the first meeting overviewing your background with the TA/UCA

Grading Breakdown

Let x = number of coaching credits (0 to 8)

- **Midterm**: (40-2x) %
- **Final**: (60-3x)%
- **Pset Engagement:** 5x% **(does not depend on correctness of your solutions)**

- **Baseline credit**: 40% midterm, 60% final
- **Credit with max allotment to Psets+Coaching**: 24% mid terms + 36% final + 40% Psets

Grades in this class are not curved, they do not depend on those of your classmates!

Resources

Course Website: http://www.cs.princeton.edu/courses/archive/fall25/cos330 (up soon)

- **Syllabus:** Course policies, grading, schedule
- **Lecture Slides:** Posted after each lecture
- **PSet Archive:** PSets
- **Precept Archive:** Precept notes with solutions

Getting help:

- **Ed Discussion:** Primary forum for questions (no private questions)
- **Office Hours:** For additional help with PSets and concepts
- **1-on-1 Coaching:** Weekly personalized feedback sessions

Disclaimer

This is a new course! So, expect some bumps along the way.

We will be trying some new (hopefully exciting) things this semester, and we want your feedback!

Please read the policy materials we will post on the course website. We know it's boring, but we are trying a lot of new things so you should know what to expect