WELCOME TO CS1!

- Jim Fix, lecturer and lab instructor

Also: Several 121 alums as TAs

REED COLLEGE CSCI 121 SPRING 2026

COURSE OVERVIEW

LECTURE 00

JIM FIX, REED COLLEGE CSCI 121

COURSE OVERVIEW

- There is a course webpage at http://jimfix.github.io/csci121
  - It has the syllabus and a schedule of topics covered.
  - There I'll post lecture materials, assignments, and supplemental readings.

ASSIGNMENT SUBMISSION THRU GRADESCOPE

- There is a **Gradescope** "course" for submitting completed assignments.
  - ➡ You should have received an invitation to join it.
  - ➡ You hand in homework and project work there.

- Today (by 5pm) I will post a **Project 0** for you to work on tonight.
  - ➡ It will help you set up your computer.
  - ➡ It will give you practice submitting assignments.
  - ➡ Complete it before our first lab meeting tomorrow.

- Tomorrow a **Homework 1** will be started in lab, due by the next lab.
  - ➡ You'll write some basic interactive Python programs.

- I post homework descriptions at http://jimfix.github.io/csci121/assign.html

CS1 IS AN INTRODUCTION TO SEVERAL THINGS

- Course topics:
  - An introduction to programming. We will use the **Python** language.
  - An introduction to the discipline of computer science.
  - An introduction to object-oriented programming.
  - An introduction to data structures and algorithms.

CS1 IS AN INTRODUCTION TO SEVERAL THINGS

- Course topics:
  - An **introduction** to programming. We will use the **Python** language.
  - An **introduction** to the discipline of computer science.
  - An **introduction** to object-oriented programming.
  - An **introduction** to data structures and algorithms.

CS1 IS AN INTRODUCTION TO SEVERAL THINGS

- Course topics:
  - An **introduction** to programming. We will use the **Python** language.
  - An **introduction** to the discipline of computer science.
  - An **introduction** to object-oriented programming.
  - An **introduction** to data structures and algorithms.

No prerequisites.
No prior programming experience expected.

WHY PYTHON?

- It's a good first language.
  - ➡ It's easy to learn, loose, feature-rich.
  - ➡ Has features from several language families.

- It has a large programmer base.
  - ➡ Used by the open source community for scripting, glue.
  - ➡ Used by several scientific communities:
    - ✦ bioinformatics, computational chemistry, SAGE math.
  - ➡ Programming tools and docs are freely available.

- It's great fun.

A PYTHON PROGRAM

```
name = input("Enter your name: ")
print("Hello, " + name + ".")

course = int(input("What's the course's #? "))
print("Ahh, yes, CSCI " + str(course) + "!")
```

```
square = 0
count = 0
while square + 2 * count + 1 <= course:
    square += 2 * count + 1
    count += 1

if course % square == 0:
    print("Did you know " + str(course))
    print("equals "+ str(count) + " squared?")
```

ANOTHER PYTHON PROGRAM

```
import time

def newton(guess, target):
    time.sleep(0.5)
    next = guess - (guess * guess - target) / (2 * guess)
    while abs(next - guess) > 0.001:
        print(guess)
        guess = next
        next = guess - (guess * guess - target) / (2 * guess)

course = 121
name = input("Enter your name: ")
print("Okay, " + name + " let me think...")
approx = newton(course/2.0, course)
print("Did you know " + str(course))
print("is roughly "+ str(approx) + " squared?")
```

WEEKLY PROGRAMMING TOPICS

- Here are the first several weeks of programming topics:
  - **WEEK 1:** scripting; program input and output; calculating things
  - **WEEK 2:** defining functions and procedures; checking conditions
  - **WEEK 3:** loops
  - **WEEK 4:** lists and dictionaries
  - **WEEK 5:** recursion
  - ...

This schedule can be found at http://jimfix.github.io/csci121/sched.html

A PYTHON PROGRAM

WEEK 1: SCRIPTING

```
name = input("Enter your name: ")
print("Hello, " + name + ".")

course = int(input("What's the course's #? "))
print("Ahh, yes, CSCI " + str(course) + "!")

square = 0
count = 0
while square + 2 * count + 1 <= course:
    square += 2 * count + 1
    count += 1

if course % square == 0:
    print("Did you know " + str(course))
    print("equals "+ str(count) + " squared?")
```

WEEK 1: INPUT

WEEK 1: OUTPUT

WEEK 1: CALCULATING

A PYTHON PROGRAM

```
name = input("Enter your name: ")
print("Hello, " + name + ".")

course = int(input("What's the course's #? "))
print("Ahh, yes, CSCI " + str(course) + "!")
```

```
square = 0
count = 0
while square + 2 * count + 1 <= course:
    square += 2 * count + 1
    count += 1

if course % square == 0:
    print("Did you know " + str(course))
    print("equals "+ str(count) + " squared?")
```

WEEK 2: LOOPS

WEEK 2: CHECKING CONDITIONS

ANOTHER PYTHON PROGRAM

```
import time

def newton(guess, target):
    time.sleep(0.5)
    next = guess - (guess * guess - target) / (2 * guess)
    while abs(next - guess) > 0.001:
        print(guess)
        guess = next
        next = guess - (guess * guess - target) / (2 * guess)

course = 121
name = input("Enter your name: ")
print("Okay, " + name + " let me think...")
approx = newton(course/2.0, course)
print("Did you know " + str(course))
print("is roughly "+ str(approx) + " squared?")
```

WEEK 3: FUNCTIONS

WEEK 2: LOOPS

WEEKLY PROGRAMMING TOPICS

- Here are the first several weeks of programming topics:
  - **WEEK 1:** scripting; program input and output; calculating things
  - **WEEK 2:** checking conditions; looping
  - **WEEK 3:** defining functions and procedures
  - **WEEK 4:** lists
  - **WEEK 5:** dictionaries
  - …

The schedule can be found at http://jimfix.github.io/csci121/weeks.html

- We move somewhat quickly, but it has worked well to do so!
- Though we use **Python**, these concepts are universal.
  - You are learning the structure of algorithms; ***algorithmic problem solving***.

WEEKLY PROGRAMMING TOPICS

- The remaining weeks provide a transition to more advanced programming.
  - **WEEK 6:** recursion
  - **WEEK 7**: object-oriented programming
  - **WEEK 8**: function objects
  - ***SPRING BREAK***
  - **WEEK 9**: linked lists
  - **WEEK 10**: algorithmic efficiency; sorting and searching
  - **WEEK 11**: binary search trees
  - **WEEK 12**: file I/O and exceptions
  - **WEEK 13**: wrap-up and review

ASSIGNMENTS

- There will be ***weekly lab homework***.
  - Assigned in Tuesday lab, due the following Tuesday before lab.

- There will be four ***programming projects***.
  - We'll give you 2-4 weeks to complete each.

FOUR PROGRAMMING PROJECTS

- **Project 1: roll100**
  - ➡ program a strategy for a two-player dice game of chance

- **Project 2: ciphers**
  - ➡ crack some codes

- **Project 3: hawks and doves**
  - ➡ simulate a population of birds

- **Project 4: adventure / arcade**
  - ➡ build an 80s-style game, either text-based or graphical

These will be due on occasional Thursdays.

ASSIGNMENTS

- There will be ***weekly lab homework***.
  - ➡ Assigned in Tuesday lab, due the following Tuesday before lab.

- There will be four ***programming projects***.
  - ➡ We'll give you 2-4 weeks to complete each.

- There will be several ***in-class quizzes***
  - ➡ Starting in a few weeks, then (roughly) every two weeks.
  - ➡ One or two short programming puzzles each.
  - ➡ Write code on paper, no use of a computer.

- There will be two ***in-class exams*** and a ***comprehensive final***
  - ➡ Also written on paper.

MEETING TIMES

- **LECTURE**: Mondays and Wednesdays, 80 minutes
  - ➡ 1:10-2:30pm in Elliot 314

- **LAB MEETING**: Tuesdays, 80 minutes
  - ➡ 9-10:30am in LIB **204**
  - ➡ 1:40-3:00pm in LIB 389

- **EVENING TUTORING**: Sunday through Thursdays, 7-9pm in LIB 340

- **MY OFFICE HOURS (subject to change; see web page)**:
  - ➡ 11:40am-12:40pm Monday and Wednesday in LIB 314 (my office)
  - ➡ most Tuesday and Wednesday mornings 11-noon, drop by my office!
  - ➡ most Monday and Wednesday late afternoons, drop by my office!
  - ➡ other times by appointment; Thursday and Friday over Zoom

RESOURCES

- I will post all my slides and code examples. You can probably work mostly from these.

- I'll suggest supplemental readings from **three textbooks**:

  - **Principled Programming** by Adam Groce, Reed College.
    - Closest to what I'll be teaching. Home-grown.

  - **Think Python! How To Think Like a Computer Scientist** by Allen Downey, Green Tea Press
    - Follows the topics of the course somewhat closely.

  - **COMPOSING PROGRAMS** by John DeNero, UC Berkeley
    - This is a **Python** rewrite of MIT's famous
      Structure & Interpretation of Computer Programs ("SICP"; uses **Scheme**)
    - Interesting supplement. Only use Chapters 1 and 2.

  - All are freely available on-line.

CS1 IS AN INTRODUCTION TO SEVERAL THINGS

- Course topics:
  - An introduction to programming. We will use the **Python** language.
  - **An introduction to the discipline of computer science.**
  - An introduction to object-oriented programming.
  - An introduction to data structures and algorithms.

- **Q:** What is computer science as an academic discipline?

Q: WHAT IS COMPUTER SCIENCE?

- **A:** It's programming.

Q: WHAT IS COMPUTER SCIENCE?

- **A:** It's programming.

- **A:** It's about programming.

Q: WHAT IS COMPUTER SCIENCE?

- **A:** It's programming.
- **A:** It's about programming.
- **A:** It's about "about programming."
- Etc.
-

Q: WHAT IS COMPUTER SCIENCE?

- **A:** It's programming.

- **A:** It's about programming.

- **A:** It's about "about programming."

- Etc.

- You will learn to be reflective about programming, and also to be reflective about the tools that run programs.
- If you continue studying CS, you will learn to make tools that help people write programs. And make tools that help tools that run programs. Etc.