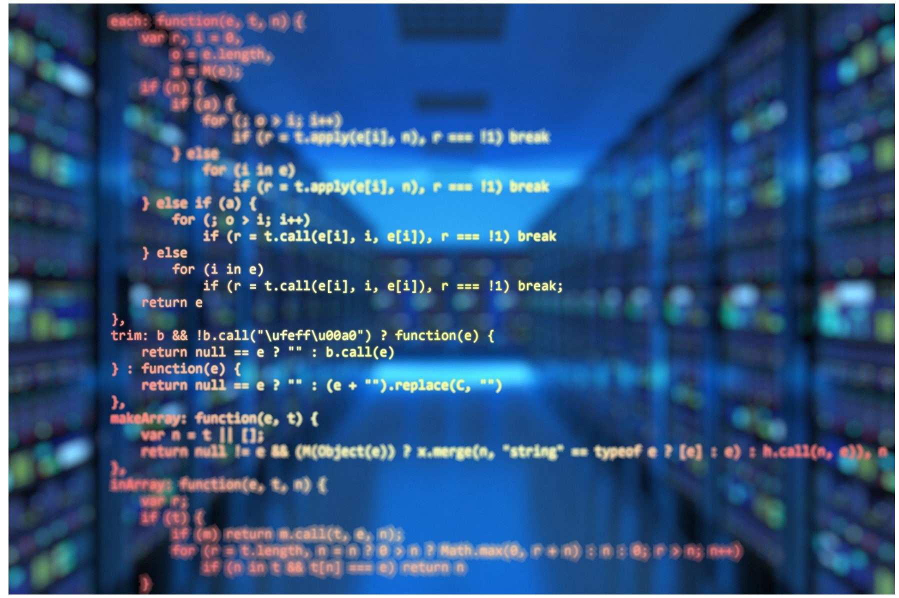


CSE 331

Software Design & Implementation

Kevin Zatloukal

About Your Instructors

Kevin Zatloukal

- Programmer for 28 years
  - started at Microsoft in 1996
- My first computer
  - had 64 kilobytes of memory

    had to scrimp and save every bit
  - drew on the screen by writing directly to video memory

    no libraries of any kind

About Your Instructors

Kevin Zatloukal

- Programmer for 28 years
  – started at Microsoft in 1996

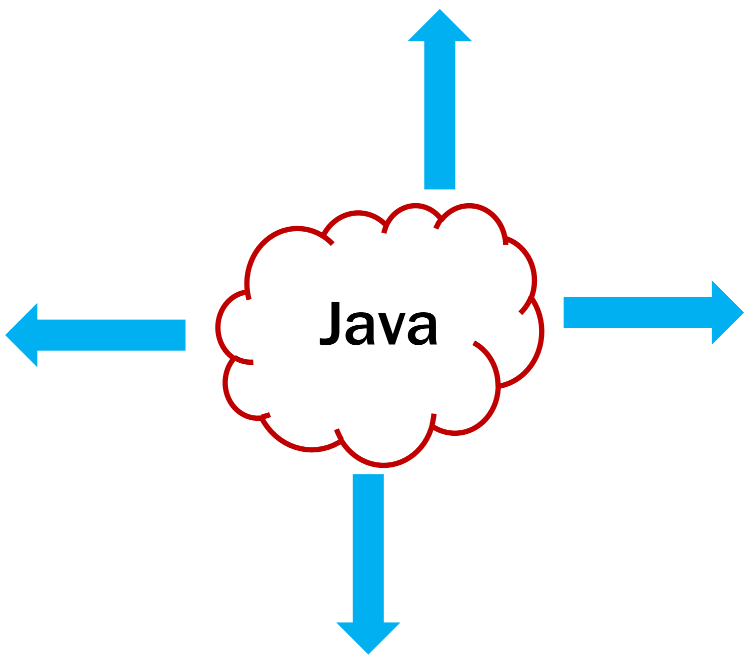
- Built a wide range of systems and applications

Systems
- compilers
- operating systems
- distributed systems
- networking systems
- database systems
- graphics
- ...

Applications
- desktop apps
- web apps
- phone apps
- IDE
- games
- ...

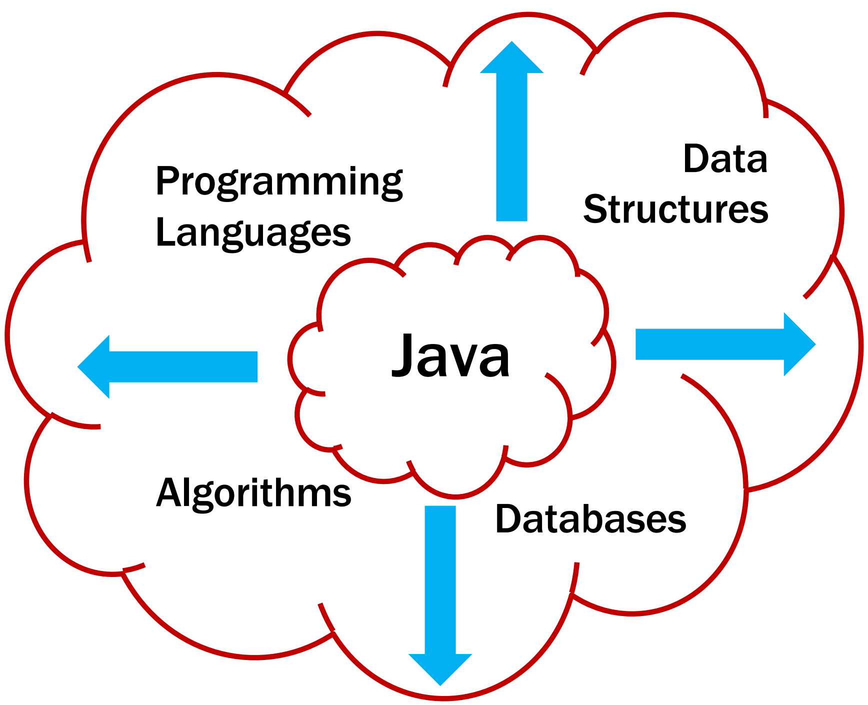
Learning Computer Science

- You already know Java
  - some basic data structures and algorithms
- Working on expanding your knowledge

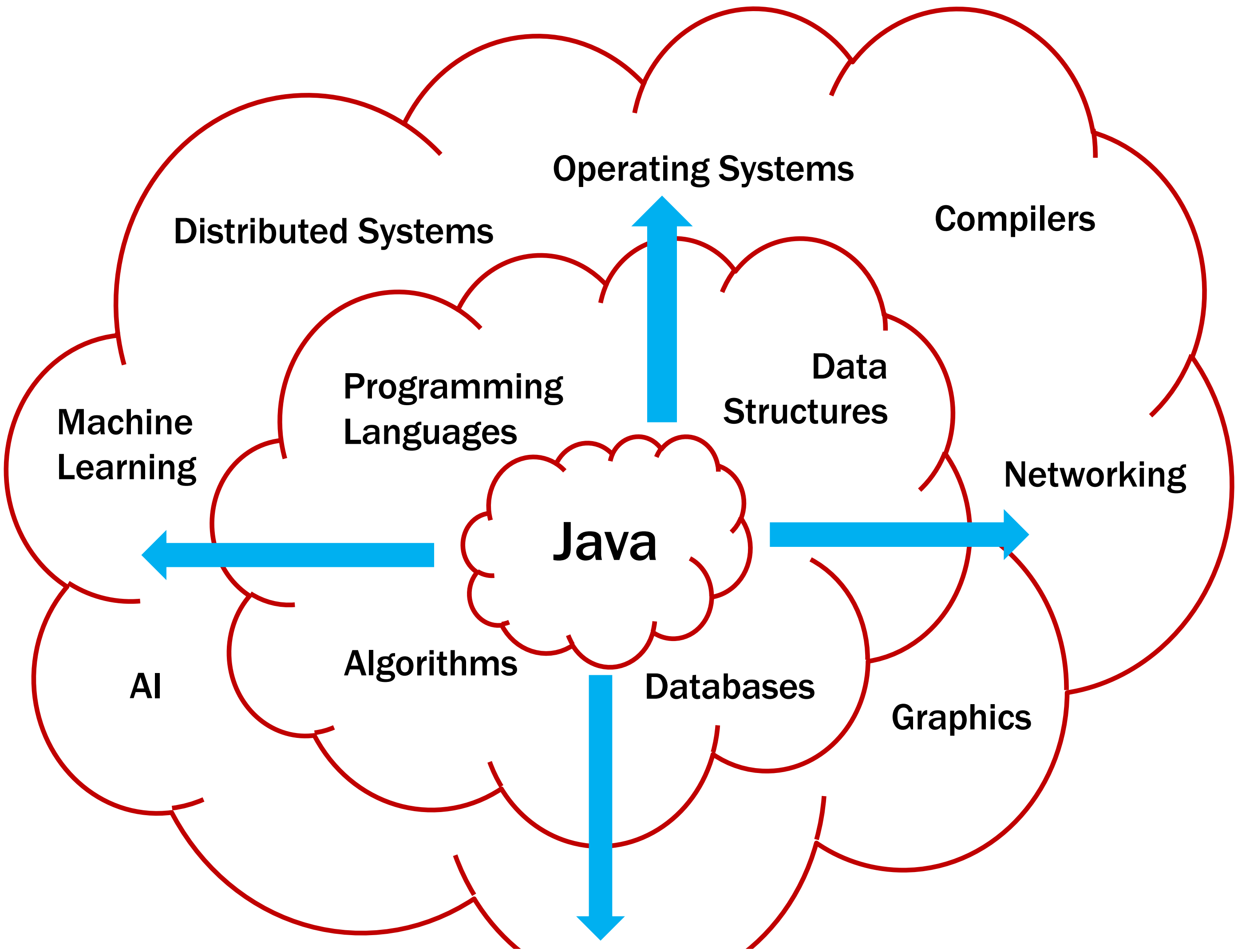


Learning Computer Science

- You already know Java
  - some basic data structures and algorithms
- Working on expanding your knowledge



Learning Computer Science



Learning Computer Science

1. First time solving this kind of problem

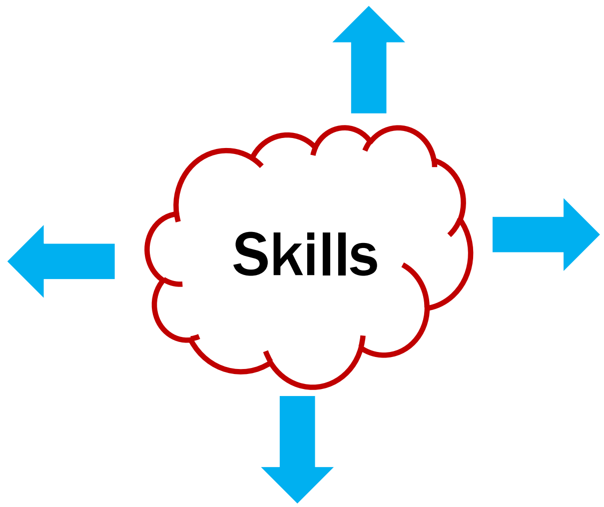
2. Given lots of help

   will often tell you if it's right

3. Expected to make mistakes

   90% is an "A"!

All of these are
<u>different</u> in industry



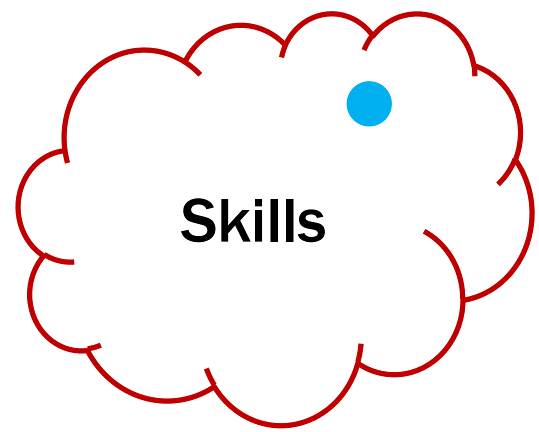
Practicing Computer Science

1. Not the first time solving this kind of problem 😀

normal to hire someone with prior experience

learn new skills in class or in spare time

E.g., write Huffman encode/decode in some new setting



Practicing Computer Science

1. **Not the first time solving this kind of problem** 😀

   normal to hire someone with prior experience

   learn new skills in class or in spare time

2. **No one to tell you if your code is right**

   That's your job!

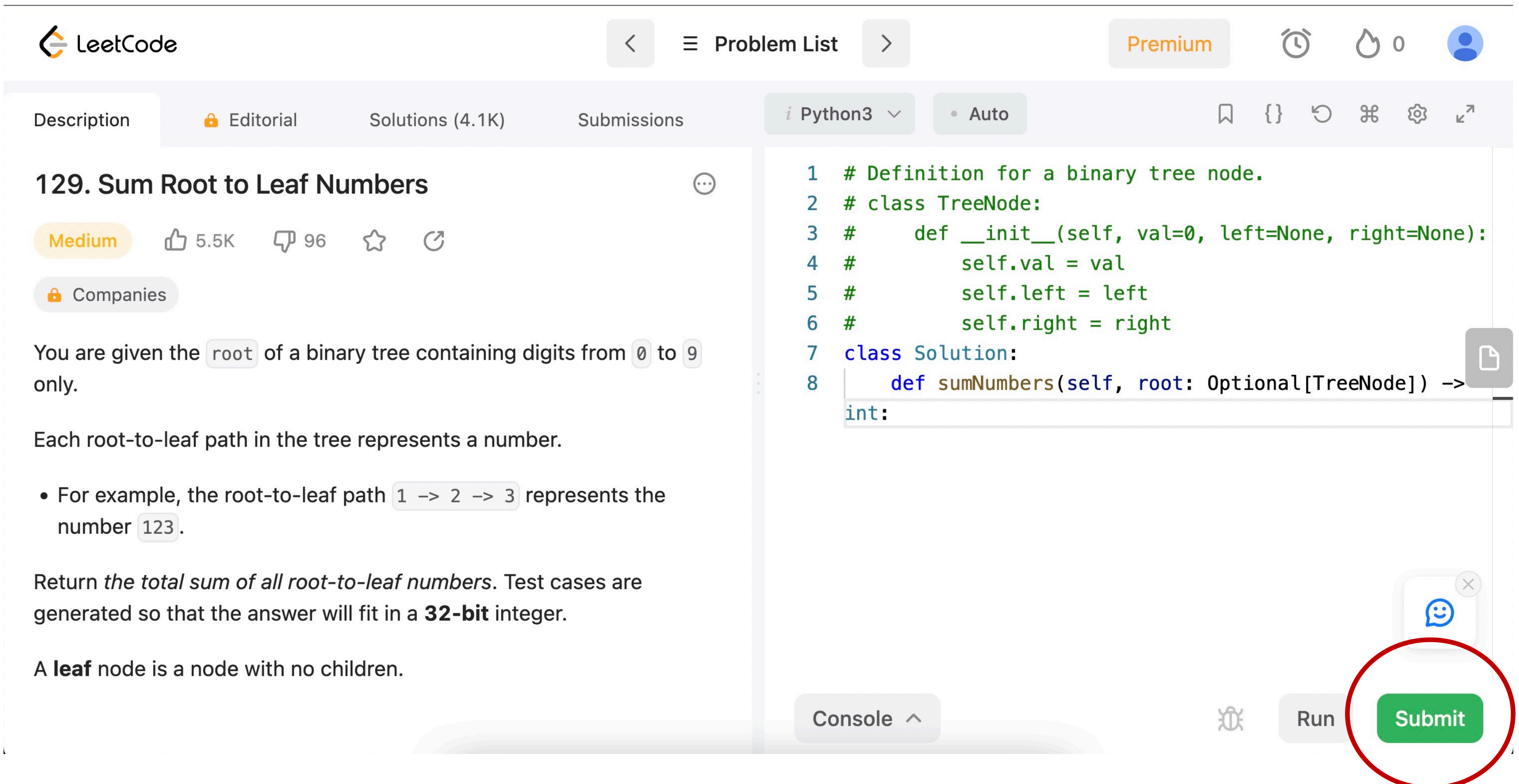
   (senior engineers will *double check* your work, but they expect it to be right)

   you will almost never be given tests

Skills

Least "Real World" Setting Possible

Would give you a button to click to see if it's right…



Someone else already solved this problem.
They only need you for new problems.

Practicing Computer Science

1. **Not the <u>first time</u> solving this kind of problem** 😀

   normal to hire someone with prior experience

   learn new skills in class or in spare time

2. **<u>No one</u> to tell you if your code is right** ☹️

   That's your job!

   (senior engineers will *double check* your work, but they expect it to be right)

   you will almost never be given tests

Skills

Practicing Computer Science

1. Not the first time solving this kind of problem 😀

normal to hire someone with prior experience

learn new skills in class or in spare time

2. No one to tell you if your code is right ☹️

That's your job!

(senior engineers will *double check* your work, but they expect it to be right)

you will almost never be given tests

3. Mistakes are not acceptable

90% is not an "A"

10% of 1m users is 100k users calling customer service

1% of 1m users is 10k users calling customer service

Cyberpunk 2077

just before launch

Cyberpunk 2077

CrowdStrike in 2024

Lost 50% ($45b) in a few months

Users hate buggy software

- Almost all software works properly almost always

- People are not used to buggy software
  - when it happens, they lose their minds

- Hard to appreciate until you see it yourself…

Practicing Computer Science

1. Not the <u>first time</u> solving this kind of problem 😀

normal to hire someone with prior experience

learn new skills in class or in spare time

2. <u>No one</u> to tell you if your code is right ☹️

That's your job!

(senior engineers will *double check* your work, but they expect it to be right)

you will almost never be given tests

3. Mistakes are not acceptable 😬

90% is <u>not</u> an "A"

10% of 1m users is 100k users calling customer service

1% of 1m users is 10k users calling customer service

What This Class is About

- Learning what engineers do to make sure their code is correct <u>before</u> sending it to users

- Learn a toolkit for being 100% sure it is right
  - any "computer scientist" must know this

- Learn when to use the toolkit
  - not every problem requires it

Properties of High-Quality Code

- Professionals are expected to write high-quality code
- Correctness is the most important part of quality
  - users **hate** products that do not work properly
- Also includes the following
  - easy to understand
  - easy to change
  - modular

will also discuss these

Other Properties of High-Quality Code

- Note that list did not include **efficiency**

- We will focus on correctness
  - a prerequisite for efficiency
    speed doesn't matter if the code is not correct
  - other classes give plenty of attention to efficiency

- General rules about programmers (50+ years of evidence):
  - overestimate the importance of efficiency
  - underestimate the difficulty of correctness

Will Focus on Timeless Skills

- Focus our time on skills that will be useful 10+ years on…

- Not specific languages or libraries
  - specific knowledge is only impressive to junior programmers
    - you will know 3-5 languages by the time you graduate!
    - you will not be impressed by someone who knows 1 more
  - AI knows how to to write a loop in every language and how to call every well-known function in every library
    - do not expect to be paid for this knowledge

We Will Ask You to Write Code Differently

- Our goal is **not** to teach you to write code that looks exactly like what you will see in industry
  - nor is it to use the libraries most common in industry

    the most popular languages and libraries change all the time

- Our goal is to teach you to **think** through your code and to **understand** how all the parts work

- That is best served by writing slowly and carefully

- We will force that by
  1. changing programming languages to something *unfamiliar*
  2. having *unusual* coding conventions at times

Homework

- CSE 331 is a **hard** class
  - because coding & debugging are hard!
- Most of the work is done **outside of class**
  - university policy is 2 hours per hour of class time
  - plan for 8 hours per week, but…
- Wide variation in time required
  - some students will average 10-15 hours

    but this is not expected!

    be sure to get help if you are averaging over 15 hours

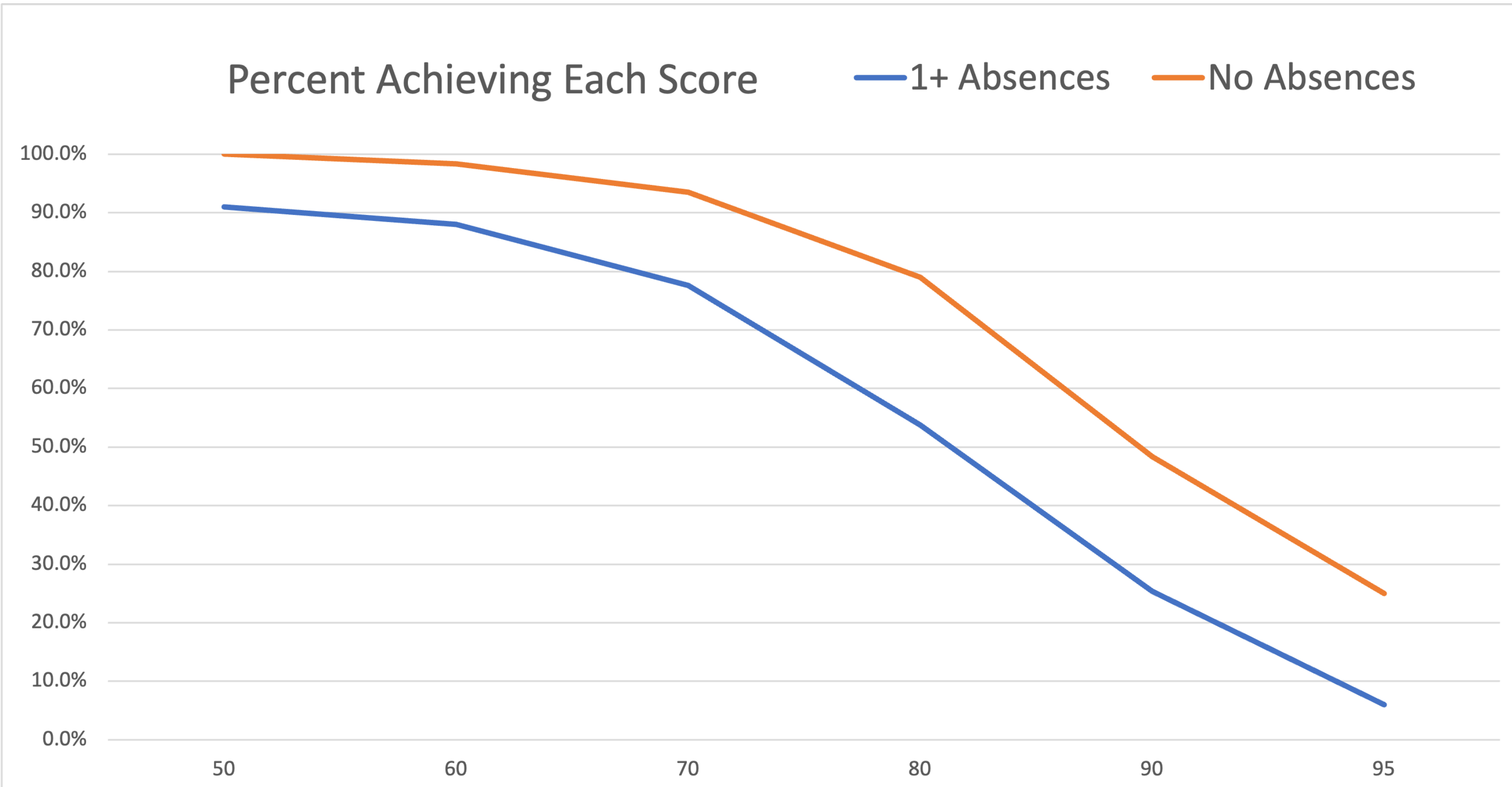
Homework Assignments

- Nine assignments split into these groups:

HW1 HW2 HW3	learn to write more complex apps practice debugging
HW4 HW5 HW6	learn how to be <u>100% sure</u> the code is correct (most of the work done on *paper*)
HW7 HW8 HW9	learn to use the tools productively (when to use then and when not to)

Quiz Sections

- Get an ungraded attempt at solving HW-like problems
  - extremely beneficial to student success...



Quiz Sections

- Get an ungraded attempt at solving HW-like problems
  - extremely beneficial to student success
- Plan to attend all quarter
- If you are unable to attend, can do the work online
  - submit solutions to all worksheet problems by 5pm
- Participation is not required, but non-participation is interpreted as confidence that you do not need extra help
  - specifically, that you do not need to attend **office hours**
  - OH time is the most limited resource in the course, so it will be **restricted** to those who attended that week's section

Late Days

- Students are allowed 4 late days total
  - anything more than 10 minutes late is 1 day
  - no more than 1 late day per assignment
  - 1 late day covers both the written and coding parts

    some assignments have separate due dates for the written and coding parts

    for those assignments, a late day allows **both parts** to be submitted 24 hours late

- To go over these limits, you must talk to the instructors
  - no reason to ask for more until you are out of late days

- Plan to complete assignments on time
  - schedule is set up to be done on the due date
  - save late days for emergencies

Exams

- No midterm exam

- Final exam at an unusual time and place
  - on Tuesday, December 10^{th} at 12:30
  - in BAG room 131 or 154

Collaboration

- Students are expected to write their solutions individually and unassisted
  - these are not group work
  - AI is not allowed

- Fine to talk about the assignment with other students, but you must either:
  1. Keep the discussion "high level" or
  2. Keep no written or electronic records of your discussion

- There is <u>no acceptable reason</u> to share your solution with any other student

Advice

- Start HW4-9 early
  - wide variation in the time required
  - never know how long **debugging** will take!
- Use the message board whenever possible
  - will get an answer promptly (during working hours)
- Do not skip class to work on homework

More Advice

- Start homework assignments early!
- Make sure you understand how lectures apply
  - seeing no connection to lecture is a giant red flag
- Focus on understanding, not points
  - understanding, not points, will not get you a job
  - losing points is the best reminder to review later