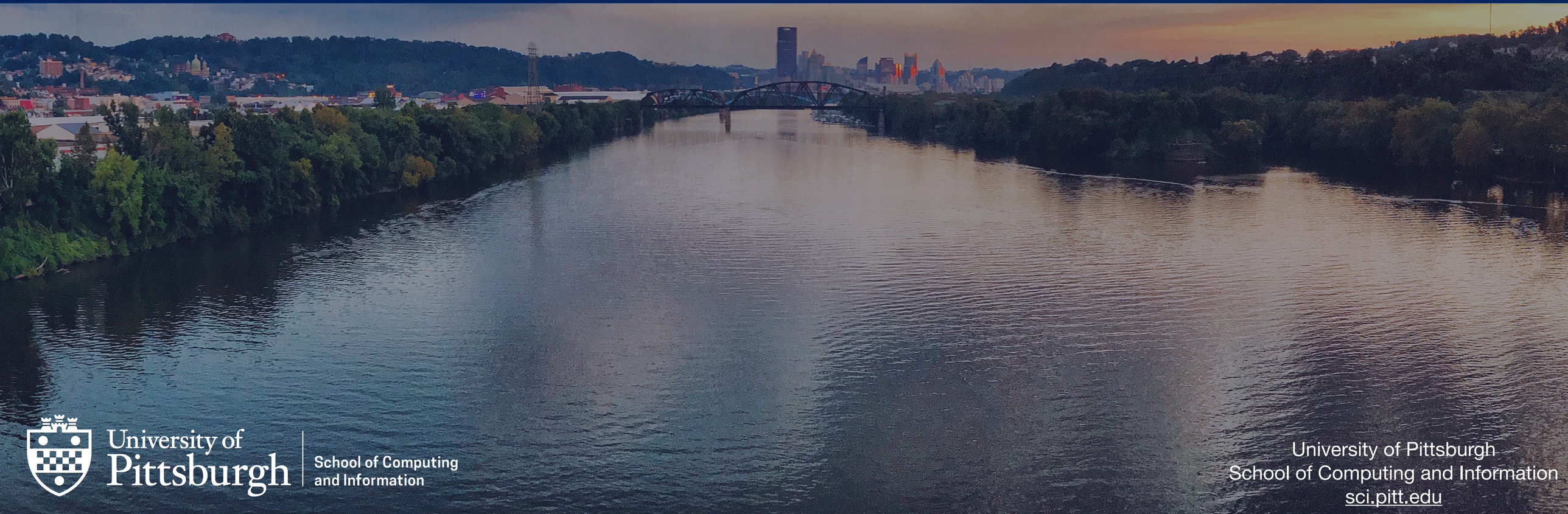
Fall 2025

L01 Introduction

CS 1530 Software Engineering

Nadine von Frankenberg



University of Pittsburgh
School of Computing and Information
sci.pitt.edu

Copyright

- These slides are intended for use by students in CS 1530 at the University of Pittsburgh only and no one else. They are offered free of charge and must not be sold or shared in any manner. Distribution to individuals other than registered students is strictly prohibited, as is their publication on the internet.
  - All materials presented in this course are protected by copyright and have been duplicated solely for the educational purposes of the university in accordance with the granted license. Selling, modifying, reproducing, or sharing any portion of this material with others is prohibited. If you receive these materials in electronic format, you are permitted to print them solely for personal study and research purposes.
  - Please be aware that failure to adhere to these guidelines could result in legal action for copyright infringement and/or disciplinary measures imposed by the university. Your compliance is greatly appreciated.
- Material from these notes is obtained from various sources, including, but not limited to, the following:
  - Bruegge, & Dutoit. Object–oriented software engineering. using UML, patterns, and Java. Pearson, 2009.
  - Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns. Pearson, 1994.
  - Sommerville, Ian. "Software Engineering" Pearson. 2011.
  - http://scrum.org/

Today's roadmap

- Organizational & syllabus
  - Prerequisites
  - Learning goals & course organization
  - Grading
  - Tools
  - Communication & Feedback
- Q&A
- Intro to Software Engineering

Organizational matters

- Please review the syllabus & course policies on Canvas!
- If you need specific **accommodations**, please **email me & the DRS ASAP!**
- Make sure you fulfill the prerequisites for this course

Prerequisites & assumptions for this course

University of Pittsburgh

- You have some programming experience
  - Previous experience with Java OR Python, C, C++, Visual Basic, …
  - Lecture code examples & coding homework assignments will be in Java
  - Team project (*more later*) can be in any language
- You have access to a computer and use it during the lectures

Material to refresh your knowledge (Java)

University of Pittsburgh

- Interactive online tutorials
  - https://www.learnjavaonline.org
  - https://www.w3schools.com/java/
- Java tutorials covering the basics: https://docs.oracle.com/javase/tutorial/

Today's roadmap

- Organizational & syllabus
  - Prerequisites
  - Learning goals & course organization
  - Grading
  - Tools
  - Communication & Feedback
- Q&A
- Intro to Software Engineering

Course learning goals

- You know the most important software engineering terms and workflows
- You understand the basic concepts and phases of a software project
- You can apply suitable concepts and methods for concrete problems in complex projects
- You are familiar with and can apply problem-solving techniques
- You are able to analyze and evaluate given problems

Teaching philosophy

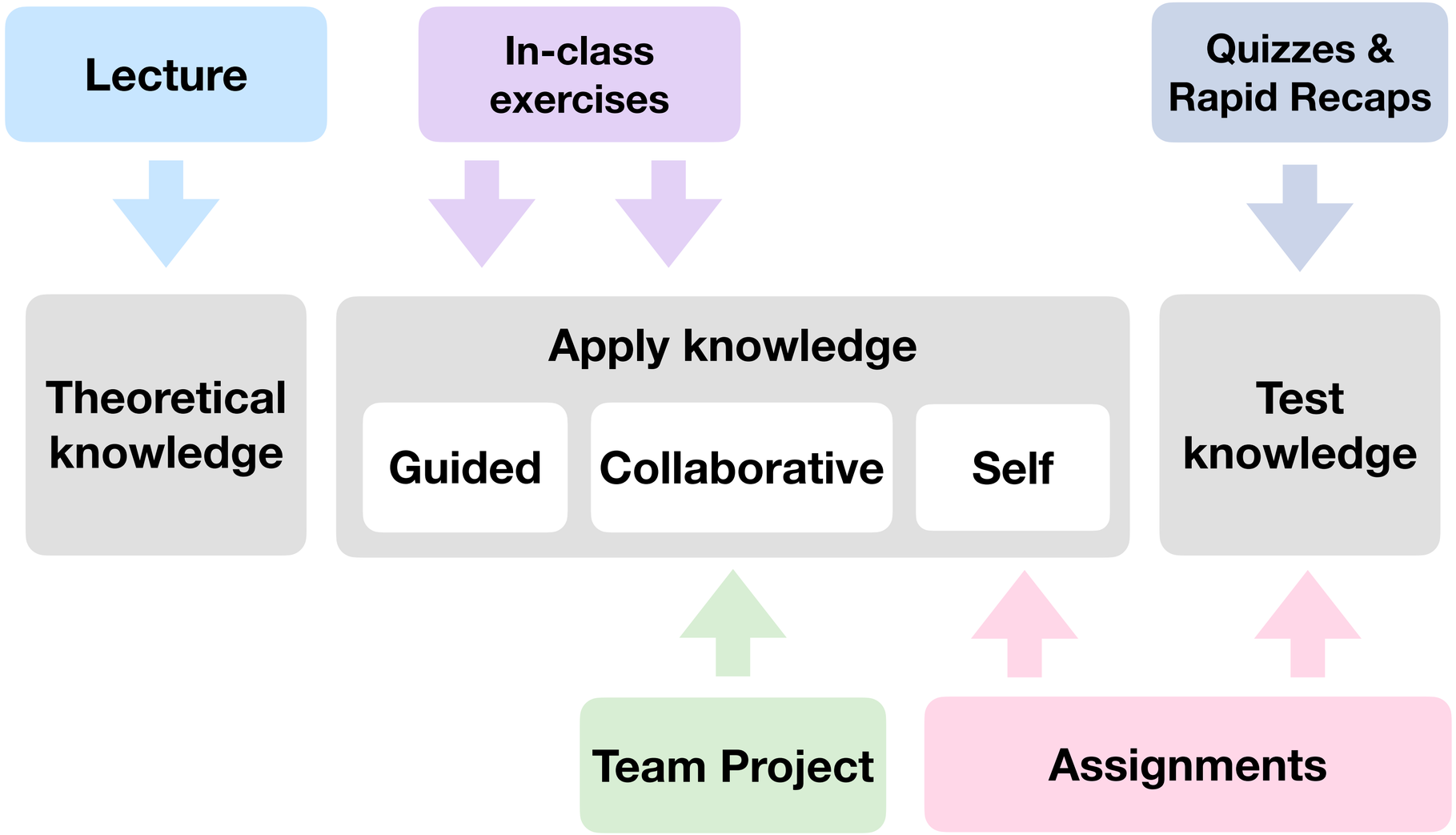
Tell me and I forget,

teach me and I remember,

involve me and I learn.

— Chinese proverb | Confucius

Teaching & learning framework



This course's roadmap

- Intro to Problem-Solving
- Model-based Engineering (UML)
- Software Development Lifecycle Models (Waterfall, Spiral, Agile, …)
- Requirements Engineering
- System Design and Architecture
- Object-Oriented Analysis and Design
- Design Patterns
- Software Testing and Quality Assurance
- Project Management in Software Engineering
- Software Configuration Management & Deployment

Order is subject to change

- User Interface Design and Usability
- Emerging Trends in Software Engineering (AI in SE, Blockchain, etc.)
- Verification and Validation

If time permits

Course organization — Lectures

#	Date	Subject
1	08/26/2025	Introduction
	08/28/2025	
2	09/02/2025	Model-based Engineering
	09/04/2025	
3	09/09/2025	Requirements Analysis
	09/11/2025	
	…	…

Today's roadmap

- Organizational & syllabus
  - Prerequisites
  - Learning goals & course organization
  - Grading
  - Tools
  - Communication & Feedback
- Q&A
- Intro to Software Engineering

Grading — Scale

%	LG	Breakdown	
>= 90%	A	100% — 97%	A+
		96% — 94%	A
		93% — 90%	A-
>= 80%	B	89% — 87%	B+
		86% — 84%	B
		83% — 80%	B-
>= 70%	C	79% — 77%	C+
		76% — 74%	C
		73% — 70%	C-
>= 60%	D	69% — 67%	D+
		66% — 64%	D
		63% — 60%	D-
< 60%	F	59% — 0%	F

Grading — Composition

- Rapid recaps (in-class) [10%]
- In-class exercises [20%]
- Individual Homework Assignments [30%]
  - You will complete several assignments on your own
  - More information soon!
- Team Project [40%]
  - In small teams, you will complete several assignments (more soon)
  - Each team must present their project (end of term)
- No exams :-)

Type	Percentage
In-class Rapid Recaps	**10%**
In-class Exercises	**20%**
Individual Homework Assignments	**30%**
Team Project	**40%**

Rapid Recaps (RRs) [10%]

- Short quiz questions about the previous week's content
- Every **Tuesday** at the beginning of class, **11:00 am on the dot** ⏰
- Open book, up to 5 questions, time will vary from ~ 3 - 10 min
- There are no repetitions! No exceptions!
- Your worst two RR results will be dropped

In-class Exercises [20%]

- You will be given some time in-class to work on a problem in **pairs**
- Follow-alongs or short exercises to be completed in class (latest by 11:59 pm)
  - E.g., programming exercises, modeling exercises, quiz questions, …
- After some time, a pair present their results
  - You receive feedback from peers & me
    - Allows you to adapt your solution before submitting it!
  - You see other solutions and understand other approaches
  - Practice communication skills

Individual Homework Assignments [30%] — Preview

University of Pittsburgh

- Individual submissions
- Assignments allow you to apply & deepen your knowledge
- *More & schedule soon*

Team Project [40%] — Preview

- In small teams, you will work together on designing a software system (and implementing parts of it) throughout the term
- You will come up with your own idea / find someone with an idea
- Each team member must contribute somewhat equally (will be assessed!)
- Each team must present their system
- *More & schedule soon*

Today's roadmap

- Organizational & syllabus
  - Prerequisites
  - Learning goals & course organization
  - Grading
  - Tools
  - Communication & Feedback
- Q&A
- Intro to Software Engineering

Main course tools (1)

Canvas

- Access lecture material
- Take rapid recaps

TopHat

- Follow-along with the slides
- Complete in-class exercises

- Access & submit coding exercises (in-class, assignments)

Main course tools (2)

Discord

- Communicate with instructor, TA, & fellow students
- Ask general questions about the organization, lectures, and exercises
- Receive announcements
- You can find the invite link on Canvas

Today's roadmap

- Organizational & syllabus
  - Prerequisites
  - Learning goals & course organization
  - Grading
  - Tools
  - Communication & Feedback
- Q&A
- Intro to Software Engineering

"Life happens" - Late submissions & Extensions

- Reach out **asap** in case of medical/family/… emergencies
  - I will need some **proof**, e.g., doctor's note
    - (E.g., your roommate's friend's COVID test is NOT a valid form of proof)
  - **Alternative**:
    Reach out to **Lynnsey Doane** (led114@pitt.edu, Director of Student Success)
- Non-emergency late submissions/extensions
  - Please reach out if unsure how to handle a situation before the deadline (no later than 3 days past the due date)!
  - Late assignments/in-class exercises:
    see late policy in the syllabus
  - RR: in total, you can miss two
    (your best 10 RR scores will count)

!! Expectations — Questions (in-class & Discord)

University of Pittsburgh

- **Ask questions!**
  - If asking a question "publicly" makes you a little uncomfortable, I still encourage you to do so; it will be good practice & benefit you in the long run (e.g., being more comfortable to ask questions in a job later on)
  - **THERE ARE NO STUPID QUESTIONS!**
  - Remember, you are here to **learn**!
    If you have a question, there is a high chance of someone else having the same question
  - If my/your TA's answer does not satisfy you, also please let do me/them know!
    What is obvious to us might not be obvious to you, but if we don't know, we can't change anything
- **If there are no questions, I will assume that everyone understood everything!**

Expectations — Feedback

- If you have any feedback, **please talk to me!**
- I can only make changes if I know about what's working for you and what isn't
  - Knowing what's working for you is important for me!
  - Don't wait for the surveys, please reach out anytime!
- There will be frequent questions/feedback TopHat discussions - do use them!
- There will be two 'official' surveys throughout the course
  - **Midterm evaluation**: 09/08 - 10/20/2025
  - **!! Final course evaluation: 11/17 - 12/07/2025**

Based on this feedback, I will make changes for the remainder of the semester

Most important one!

University of Pittsburgh

Previous Knowledge

⏳ 5-10 min

📊 Survey

Please complete this survey by the end of today!

Today's roadmap

- Organizational & syllabus
  - Prerequisites
  - Learning goals & course organization
  - Grading
  - Tools
  - Communication & Feedback
- Q&A
- Intro to Software Engineering

University of Pittsburgh

Questions?
Concerns?
Feedback?

Useful websites every programmer should know & use

- Google 
- Stack Overflow 

- GitHub 
- Other useful websites:
  - Geeks for Geeks
  - SourceForge
  - Codecademy
  - CodeBeautify

University of Pittsburgh

What are your expectations for this class?

Today's roadmap

- Organizational & syllabus
  - Prerequisites
  - Learning goals & course organization
  - Grading
  - Tools
  - Communication & Feedback
- Q&A
- Mini-Intro to Software Engineering

What is software?

- Software is **complex**
  - Typically composed of many parts
- Software can be **complicated**
  - No sufficient documentation = not easy to understand
  - Often requires in-depth knowledge to make a change
- Building & maintaining software involves many **stakeholders**
  - E.g., developer, customer, user, …

What is software engineering?

Code?

Coding?

Software Engineering is not rocket science nor textbook knowledge

Software Engineering is the use of **common sense** and **discipline** to **solve real-world problems**

What is software engineering?

University of Pittsburgh

- Software engineering is much more than 'just writing code'
  - Programming is a subset of software engineering!
- **Software engineering is the process of applying scientific principles to the design & creation of software**
  - Systematic approach to designing, building, testing, and maintaining software
  - Focuses on problem-solving and decision-making throughout the project lifecycle
  - Collaboration and teamwork across various roles & stakeholders
  - Emphasizes quality assurance, scalability, and sustainability
- Involves both technical and non-technical skills

Communication & project management

Key concepts in software engineering

- Software engineering builds on **problem-solving** abilities
  1. Understand the problem
  2. Propose a solution and a project plan
  3. Engineer a system using best practices and good design principles

The proposed solution is often called and used as a 'prototype'

- Software engineering is about **dealing with complexity**
  - Creating abstractions and models
  - Notations for abstractions
- Software engineering is concerned with **dealing with change**
  - Requirements elicitation, analysis, design, implementation, validation of the system, delivery and maintenance

Why you should learn about software engineering

University of Pittsburgh

- Building software without discipline is crazy!
- Software is critical to society
  - Economy
  - Enterprises
  - Key part of many complex systems
  - Essential for designing products
  - Embedded in systems you are using daily
  - …
- Realizing large, complex projects is hard
- Software crisis (*more next lecture*)
- It’s fun 🤓

University of Pittsburgh

RR00 - Demo

⏳ 2 min

⚡ Rapid Recap

Practice time!
(Today's results will count as a bonus 💪)

University of Pittsburgh

How to build a complex system?

Where to start?

I01 — Group Exercise

⏳ 10 min

💬 In-class

University of Pittsburgh

Problem Statement: *Imagine you're in a new city for the first time, and you're very hungry. You want to find a good place to eat but don't know the area.*
How would you go about solving this problem?

Your Task:

- Form small groups of 2 — 3 students
- Identify steps you would naturally need to solve this problem
- You now want to share your approach with many people in various cities!
- **Design & build this system:** *How many possible solutions are there? Which solution is correct? Where to start?*
- Share your results!

I1 - Defining a system

2 points

I01 — Group Exercise

⏳ 10 min

💬 In-class

University of Pittsburgh

Problem Statement: *Imagine you're in a new city for the first time, and you're very hungry. You want to find a good place to eat but don't know the area. How would you go about solving this problem?*

Questions you might consider:

- What do you need to build this system?
- What assumptions are you making?
- How will the user interact with the system?
- How will you collect information about available restaurants?
- Are other stakeholders involved? Who are they?
- How will you measure the success of the system?
- What happens after you have built this system?
- …

Via **Gradescope**.
One submission per group, please!
You can add your group members after submitting the exercise.

I1 - Defining a system

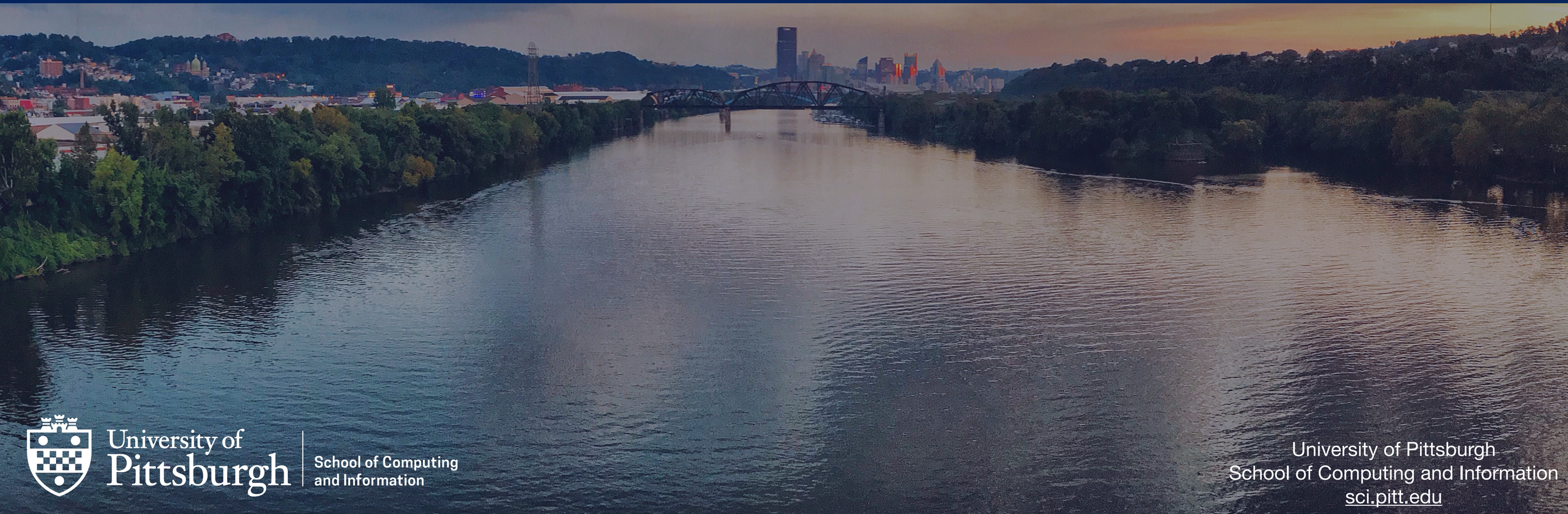
2 points

Fall 2025

L01 Introduction

CS 1530 Software Engineering

Nadine von Frankenberg



University of Pittsburgh
School of Computing and Information
sci.pitt.edu