Pair Programming Today:
Learning Dictionaries & Learning From Each Other

CS 1 – In-Class Handout

November 12, 2025

Why Are We Pair Programming?

Today you are not just "getting through" dictionary exercises. You are practicing a real industry skill: **working on code with another human without driving each other crazy.**

Pair programming helps you:

- **Catch bugs early.** One person types; the other watches for logic mistakes, typos, and missing cases.
- **Explain your thinking.** Saying your ideas out loud makes you understand them better (and shows you what you do not understand yet).
- **Learn faster.** You pick up tricks, habits, and mental models from each other.
- **Build professional habits.** Most serious software and engineering work is done in teams, not alone in a cave.

Where Does Industry Use Pair Programming?

Pair-style work shows up in many professional settings:

Software and Computer Science Roles

- **Pair programming on tricky features.** Two developers sit together to design a data structure or debug a nasty bug.
- **Code reviews.** One person walks through their code while another asks questions and looks for edge cases.
- **Onboarding new hires.** A junior developer "shadows" a senior developer, working as a pair to learn the codebase.

Manufacturing & Engineering Technology Roles

- **Programming PLCs, robots, and test equipment.** One engineer writes the program; the other verifies safety logic and test conditions.
- **Configuring production systems.** Two people check that sensor names, tag names, and database fields match what is on the shop floor.
- **Data collection and dashboards.** One person builds the script that reads data into dictionaries or tables; the other checks units, labels, and ranges.

In both CS and manufacturing engineering technology, **it is normal and expected** that you will:

- talk through designs out loud,
- challenge each other's assumptions respectfully, and
- use another person's brain to help keep humans and systems safe.

Roles in Pair Programming

Today, you will mainly use two layers of roles:

Driver and Navigator

Driver: Has hands on the keyboard.

- Types the code.
- Focuses on syntax, indentation, and getting the current idea into the editor.
- Talks out loud about what they are typing.

Navigator: Eyes on the big picture.

- Watches for bugs, missing cases, and clarity.
- Reads the problem statement and checks that the code actually solves the right problem.
- Suggests test cases and edge cases to try.

Mentor and Learner (and How They Switch)

Sometimes one of you knows more Python or more about dictionaries right now. That is normal. For this pair:

Mentor:

- Explains ideas, not just answers.
- Asks questions like: "What do you think this line will do?" instead of grabbing the keyboard.
- Slows down enough that the learner can follow the reasoning.

Learner:
- Asks questions *before* getting completely lost.
- Tries to predict the output or behavior before running the code.
- Summarizes what they just learned in their own words.

Important: These roles are not permanent labels. They should *swap* during the exercise. A strong teammate can be a mentor on one problem and a learner on the next.

What Should You Be Doing *Today*?

You are working on Python **dictionaries**. Here are concrete habits to practice during this lab:

Before You Type

- As a pair, read the problem **out loud**.
- Identify what your dictionaries represent:
  - What are the **keys**? (e.g., student names, part IDs, sensor names)
  - What are the **values**? (e.g., grades, quantities, temperature readings)
- Decide who is Driver and who is Navigator for this problem.

While You Are Coding

- **Driver:**
  - Talk as you type: "Now I am creating an empty dictionary called `grades`".
  - Ask the Navigator to double-check key names and spelling.
- **Navigator:**
  - Watch for common dictionary mistakes:
    - Using the wrong key name.
    - Confusing `[]` access with `get()`.
    - Forgetting to handle missing keys.
  - Keep connecting the code to a real-world picture: "This dictionary is like a little look-up table on the factory floor."
- **Both:**
  - Suggest test cases and run them *often*.
  - If something breaks, talk through *why*, not just "fix it fast".

After Each Problem

- Switch Driver and Navigator roles.
- Do a 30-second debrief:
  - What did we each learn about dictionaries?
  - What communication habit helped us the most?

Connecting to Your Future Career

If You Are a Computer Science Major

Ask yourself:

- Could I explain this dictionary code to a future teammate during a code review?
- Would another developer be able to read my variable names and understand what they mean?
- How would I feel doing this same exercise with a senior engineer watching? What would I keep the same? What would I improve?

If You Are in Manufacturing Engineering Technology

Think about how dictionaries show up in your world:

- Mapping sensor IDs to readings.
- Mapping station names to status flags.
- Mapping part numbers to quantities or locations.

Ask yourself:

- Could this code safely handle missing data?
- If a production line was down, would my pair programming habits help us find the problem faster?

How to Get Better at Pair Programming

Here are a few skills you can deliberately practice today:

Communication Skills

- Use phrases like:
  - "Can you walk me through your thinking?"
  - "I am confused about this line; can we pause and talk about it?"

  - "I have an idea; can I share it before we type?"
- Be kind and specific with feedback:
  - Instead of: "That is wrong."
  - Try: "I think that key might not exist yet; what happens if we try it with an unknown student?"

Technical Habits

- Always know what your dictionary is supposed to represent.
- Keep keys and values consistent and well-named.
- Add small print statements or tests as you go to confirm what is inside your dictionaries.

Reflection

At the end of class, each of you should be able to answer:

- What did I learn about dictionaries?
- What did I learn about working with another person on code?
- What is one habit I want to keep using next time we pair program?

Goal for Today: By the time you walk out, you should understand dictionaries better, *and* be slightly more comfortable solving problems with another human. Both skills matter in the real world.