THE PROCESS OF WRITING PROGRAMS I

REMINDERS

- HW03 due tonight
- HW03 Presentations this week
- Office Hours are there for you!

TODAY'S PRIMARY QUESTION

> **WHAT DO I DO WHEN I RUN INTO A PROBLEM I DON'T IMMEDIATELY KNOW HOW TO SOLVE?**

Note: Today's topic is based on the first parts of the framework presented in *How to Design Programs* (a textbook by Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, and Shriram Krishnamurthi) and the PEDAC process designed by the LaunchSchool.
If you want more information or practice I suggest you check them out.

WRITING A PROGRAM SYSTEMATICALLY

Overall, solving a programming problem take ~9 steps -->

1. Understand the Problem
2. Write your Signature(s) and Purpose
3. Identify Examples/Test Cases
4. Solve your Examples by Hand
5. Identify Your Data Types
6. Write your Outline
7. Fill in the Gaps
8. Code!
9. Test

WRITING A PROGRAM SYSTEMATICALLY

Overall, solving a programming problem take ~9 steps -->

Today we're only going to worry about three of them

1. **Understand the Problem**
2. **Write your Signature and Purpose**
3. **Identify Examples/Test Cases**
4. Solve your Examples by Hand
5. Identify Your Data Types
6. Write your Outline
7. Fill in the Gaps
8. Code!
9. Test

STEP 1: UNDERSTAND THE PROBLEM

Questions to ask yourself:

- What is the overall goal of this program?
- What variables do we need to track? How are they represented/what data types make sense for them?
- What are our expected inputs and outputs?
- What other information (if any) do we need to do this task?

LET’S TRY IT!

You are building a digital version of a domino-style word game called "Word Train." To make a valid train, each boxcar (word) must connect to the next one by matching its last letter to the first letter of the next boxcar.

A train is represented by a list of strings, where each string is a word (boxcar).

Your task is to write a Python program that checks if a given train is valid.

STEP 1: UNDERSTAND THE PROBLEM

Questions to ask yourself:

- What is the overall goal of this program?
- What variables do we need to track? How are they represented/ what data types make sense for them?
- What are our expected inputs and outputs?
- What other information (if any) do we need to do this task?

Your answers **[C1]**:

STEP 1: UNDERSTAND THE PROBLEM

Questions to ask yourself:

- What is the overall goal of this program?
- What variables do we need to track? How are they represented/ what data types make sense for them?
- What are our expected inputs and outputs?
- What other information (if any) do we need to do this task?

Your answers:

- To check a given list of strings to see if each string ends in the same letter the next starts with
- The train (list[str]), last letter in prev word (str), first letter in current word (str)
- in --> list[str], out --> boolean
- None

STEP 2: WRITE YOUR SIGNATURE(S) AND PURPOSE

Using the information you wrote for Step 1, write both a function *signature* and *docstring* for the function(s) you will write
(in this case, you just need one function) **[L2]**

1. Write a comment or a function signature (or both!) to help yourself understand the moving in and outputs
2. If you identified that you'll need additional information (more functions), you can write those signatures too!

STEP 2: WRITE YOUR SIGNATURE(S) AND PURPOSE

1. Write a comment or a function signature (or both!) to help yourself understand the moving in and outputs
2. If you identified that you'll need additional information (more functions), you can write those signatures too!

```
def is_valid_train(train : list[str]) -> bool:
    """
    Description: checks if all strings in list train follow the rule -->
        the last letter of index i must be the first of index i+1
    Input: a list of strings that is to be checked
    Output: a boolean, true if valid, false otherwise
    """
```

STEP 3: IDENTIFY EXAMPLES/TEST CASES

Based on your understanding and signature, write down some test cases for the proposed function.

Questions to help you:

1. What are some examples that you can use to check your program?
2. Find some obvious edge cases that you can use to check yourself

STEP 3: IDENTIFY EXAMPLES/TEST CASES

1. What are some examples that you can use to check your program?
2. Find some obvious edge cases that you can use to check yourself as well

Add some additional examples to the table below to create some test cases **[C2 (but leave some space)]**

INPUT	EXPECTED OUTPUT
["hi", "is" "so"]	True

STEP 3.5: WRITE YOUR TEST CASES AS UNIT TESTS

You can prepare for later (when you will write you code) by writing your test cases as unit tests *right now*.

This gets you ahead of the game so that once you've written your code you can run it on your unit tests and see if it's correct!

Rewrite one of the test cases you wrote but as a unit test.

Example:

```
def test_two_letters_list():
    input = ["hi", "is" "so"]
    output = is_valid_train(input)
    assert output  # <- why can I write it without ==?
```

IF TIME: LET'S TRY IT ALL TOGETHER [C3]

The question:

You are working the gate at a sold-out concert. People are streaming in, and their ticket numbers are being logged into a list (as strings). Because of a glitch in the online system, you suspect a counterfeit ticket was scanned twice. You need to find the very first ticket number that appears a second time.

Write a function that returns the first element that shows up more than once. If every ticket is unique, return `None`.

Step 1: Understand the problem

- What is the overall goal of this program?
- What variables do we need to track?
  How are they represented/what data types make sense for them?
- What are our expected inputs and outputs?
- What other information (if any) do we need to do this task?

Step 2: Write Your Signature(s) and Purpose

Step 3: Identify Examples/Test Cases