

Coder Week 2: Write It, Trace It, Fix It

Mon, Nov 23 – Tue, Nov 24 · 2:30–4:00 PM · Room 105 (or live on Zoom; the link comes by email)

This week: Turn last week's plan into real Python, predict what code will print before you run it, and find bugs line by line. Tuesday is the Coder Gate.

Heads up

- Short week before Thanksgiving. Don't skip Tuesday; it's gate day.
- Tracing is the skill the exam tests most in this zone. Practice it with paper and pencil.

Read-aheads

Before Monday (15 minutes)

- Bring your pseudocode from last week.
- Read the Python basics notes below.
- Do 15 Coder study cards that show code.

Before Tuesday (20 minutes)

- Read the How to trace code notes below and trace the example by hand.
- Finish the Coder Study Path and try the Check yourself questions.

The missions

Monday, Nov 23 · Day 13

8A · Lab Flowchart to Code, Part 2

Tonight the pseudocode becomes real. Your crew writes last week's plan in Python, runs it, and uses a loop and a function to keep the code short. Then you trace it line by line to predict the output before you run it. Predicting is the skill the exam tests.

1. Translate the pseudocode into Python with at least one if/else, one loop, and one function.
2. Trace the program by hand for two inputs and write the predicted output.
3. Run the program, compare it with your prediction, and fix any difference.

Turn in: Working Python file plus the hand-trace sheet with predicted and actual output.

Tuesday, Nov 24 · Day 14

Coder Gate today. You must pass the Operator Gate first.

8B · Gate day Trace-the-Bug

Six short programs each hide one bug. Your crew traces variables step by step, finds the exact line that goes wrong, and explains why. Then every member takes the Coder Gate for badge four.

1. Trace all six buggy programs and record the variable values at each line.
2. Name the line with the bug and the fix for each program.
3. Take the 20-question Coder Gate solo; 17 of 20 (85 percent) passes.

Turn in: Bug report for all six programs and each member's Coder Gate result.

Notes

Python basics

- Assign a value: `score = 0`
- Decide: if `score >= 100`: `print("You win!")` else: `print("Try again")`
- Repeat a set number of times: `for i in [1, 2, 3, 4]`:
- Repeat while something is true: `while n < 10`:
- A function packages steps you can reuse: `def greet(name)`: then call it with `greet("Crew")`.
- Indentation matters in Python. The indented lines belong to the if, loop or function above them.

How to trace code

- Make a table with one column per variable.
- Go one line at a time and write the new value every time a variable changes.
- Example: `x = 3`, then `x = x * 2` (`x` is 6), then `x = x + 1` (`x` is 7). It prints 7.
- For a while loop, check the condition before every pass. `n = 1` doubles to 2, 4, 8, 16. At 16, `n < 10` is false, so the loop stops with `n = 16`.

Logic and common bugs

- AND is true only when both sides are true. OR is true when at least one side is true. NOT flips true and false.
- `"5" + "5"` joins two strings and gives `"55"`, not 10.
- Off-by-one: a loop runs one time too many or too few. Check where it starts and where it stops.
- Using `=` (assign) where you meant `==` (compare) is a classic bug.

Vocabulary

Function	A named block of reusable code
Loop	Code that repeats
Branch	An if/else choice between paths
Trace	Follow code line by line and track variable values
Bug	A mistake that makes code do the wrong thing
Concatenation	Joining strings together

Check yourself

1. `total = 0`, then `for i in [1, 2, 3, 4]`: `total = total + i`. What is total?
2. `age = 15`. if `age >= 16` print Drive, else print Wait. What prints?
3. Is `(true AND false)` true or false?

Answers: 1. 10 2. Wait 3. False

Bring

- A charged device
- Your student ID card
- Last week's pseudocode
- Pencil for the trace sheet