Large Language Models: A New Way to Teach Programming

Leo Porter
University of California San Diego
leporter@ucsd.edu

Daniel Zingaro
University of Toronto Mississauga
daniel.zingaro@utoronto.ca

UC San Diego
JACOBS SCHOOL OF ENGINEERING
Computer Science and Engineering

UNIVERSITY OF
TORONTO

Our Path with LLMs

- Computer science education researchers who have studied introductory programming courses (CS1) for over a decade

- We have NOT conducted research studies related to LLMs
  - We have read the literature that is coming out rapidly

- We have spent the last 6 months working with Copilot and writing a book incorporating LLMs into learning programming
  - Will be using this approach to teach in Fall 2023

Our Plan Today

1. **What we know from the growing body of research**
2. Discussion of where existing CS1 courses fall short
3. Demo of the workflow with LLMs
4. Our vision for how to teach CS1 incorporating Copilot
5. Q&A

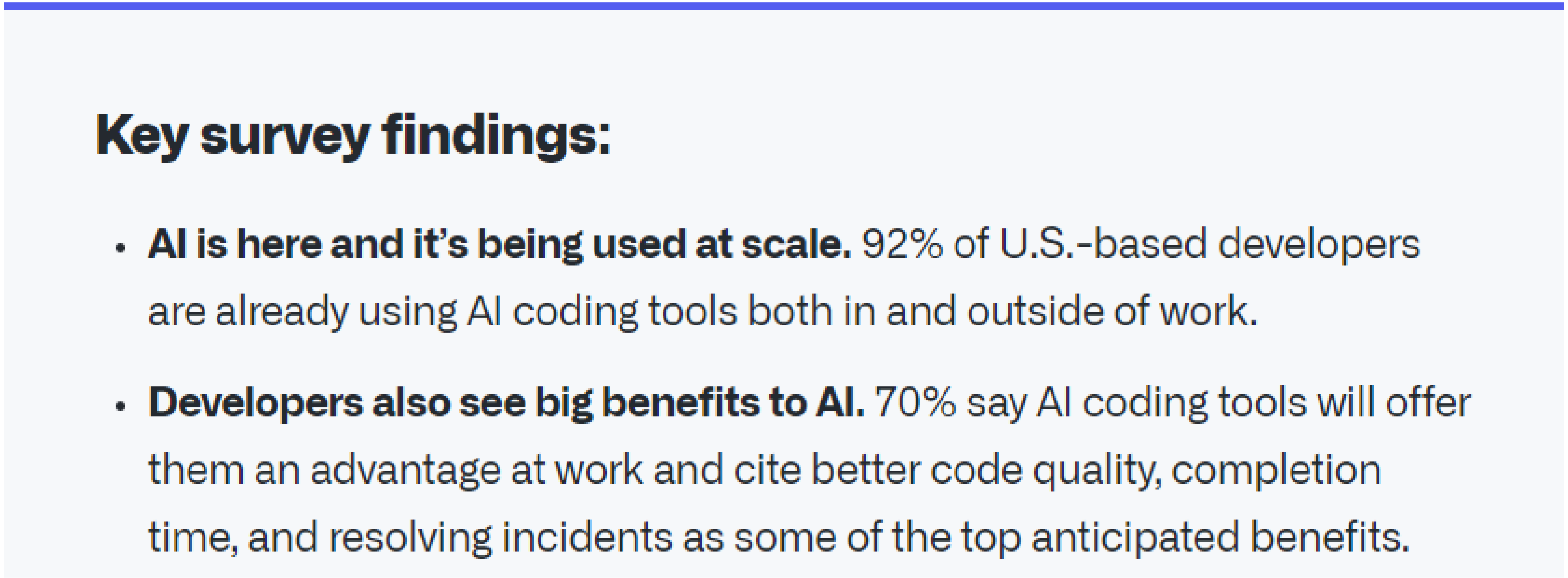
LLMs for programming in the news

- **Satya Nadella**, CEO of Microsoft:
  - "Just like the rise of compilers and interpreters, we believe AI-assisted coding will fundamentally change the nature of software development, giving developers a new tool to write better code easier and faster"
  - Developers are able to code 50% faster using Copilot

- **Jensen Huang**, Nvidia President and CEO:
  - "This computer doesn’t care how you program it, it will try to understand what you mean, because it has this incredible large language model capability. And so the programming barrier is incredibly low"

1. https://www.protocol.com/newsletters/protocol-enterprise/microsoft-ignite-defender-copilot-mobileye
2. https://www.cnbc.com/2023/05/30/everyone-is-a-programmer-with-generative-ai-nvidia-ceo-.html

LLMs can aid professional Software Engineers (1)



Key survey findings:

- **AI is here and it's being used at scale.** 92% of U.S.-based developers are already using AI coding tools both in and outside of work.
- **Developers also see big benefits to AI.** 70% say AI coding tools will offer them an advantage at work and cite better code quality, completion time, and resolving incidents as some of the top anticipated benefits.

Software developers are using these tools.
We need to help students use these tools responsibly.

https://github.blog/2023-06-13-survey-reveals-ais-impact-on-the-developer-experience/

LLMs can aid professional Software Engineers (2)

- Study on the Impact of Copilot on Developer Productivity [1]
  - Professional software developers (n=95) were asked to write an HTTP server in JavaScript
  - Controlled experiment: treatment group used Copilot, control group did not
  - Copilot group completed the task 56% faster
  - "Our results suggest that less experienced programmers benefit more from Copilot."

1. Peng et al. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. arXiv:2302.06590, 2023.

LLMs can aid experienced programmers

- Study on experiences of experienced students using LLMs
  - Within-subjects study (n=24); participants were students with prior programming experience
  - Participants completed tasks in Python (e.g. CSV editing, web scraping)
  - No difference between conditions on success rate or task completion time
  - But 19/24 preferred using Copilot
  - 12/24 participants found it hard to fix the code generated by Copilot

Vaithilingam et al. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. ACM CHI EA ’22. 2022.

How LLMs may impact CS1

- LLM performance on CS1 assignments [1]
  - Evaluation of Copilot on exercises (n=166) in public CS1 question bank
  - Authors used prompt engineering when Copilot gave incorrect code
  - Copilot solved 47.6% of problems on its first attempt and that went up to 79.5% after prompt engineering
- LLM performance on CS1 exams [2]
  - Asked Codex to solve all questions from two Python CS1 exams.
  - Codex solved almost half of the problems on its first attempt
  - Codex got 78.5% on Exam 1 and 78% on Exam 2 (rank 17 out of 71 students)

1. Denny et al. Conversing with Copilot: Exploring Prompt Engineering for Solving CS1 Problems Using Natural Language. ACM SIGCSE 2023.
2. Denny et al. Computing Education in the Era of Generative AI. arXiv:2306.02608. 2023.

Instructors vary in how to approach LLMs

- Instructor Opinions about Teaching LLMs
  - Researchers interviewed 20 programming instructors on how they plan to adapt to LLMs
  - Two categories of long-term strategies emerged:

Resist the use of AI coding tools:
- teaching Python fundamentals
- create AI proof assessments
- proctored exams

Embrace AI tools by integrating them
- give personalized help to students
- focus on code reading and critique
- have students collaborate with AI

Lau and Guo. From "Ban It Till We Understand It" to "Resistance is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. ACM ICER 2023. To appear.

LLMs in CS1 may be beneficial overall (still early though!)

- Novices (n=69), age 10-17, were asked to complete 45 Python tasks
- Each task consisted of a code authoring followed by a code modification part

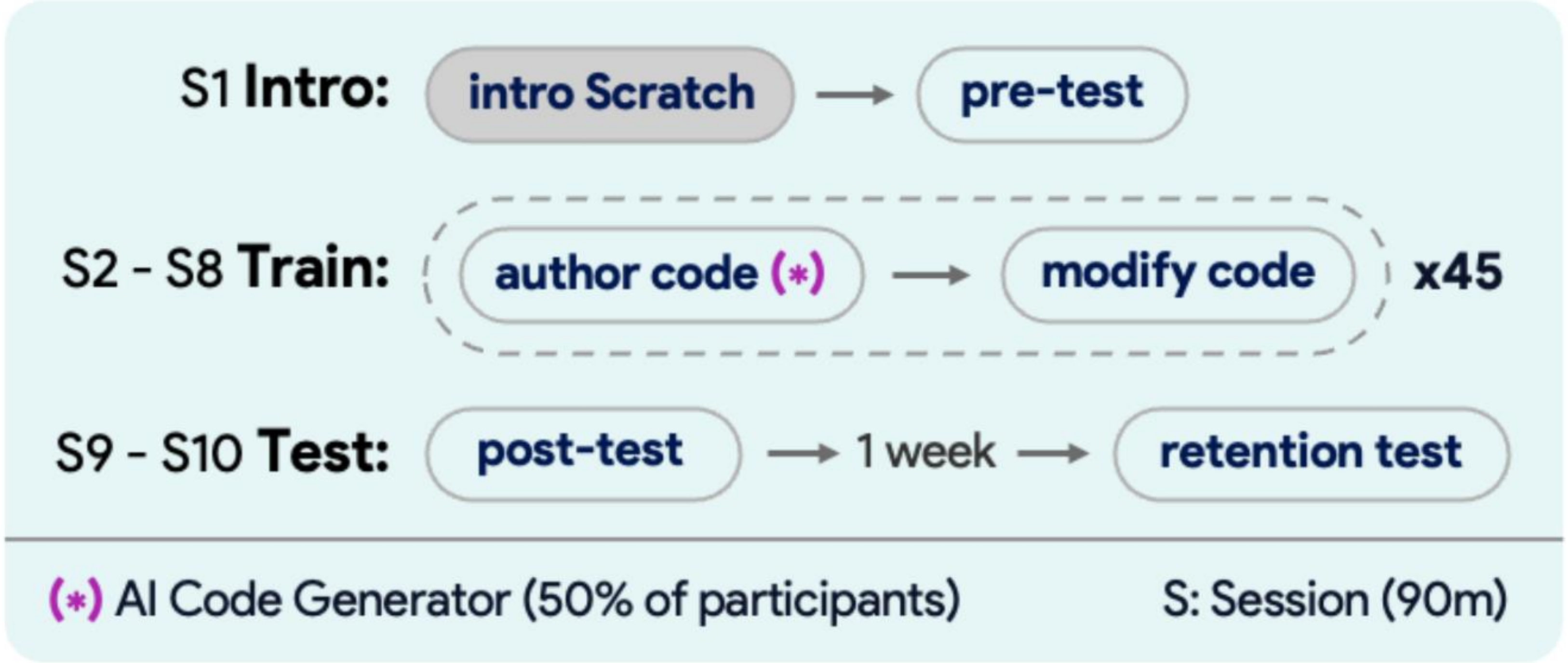


Figure 1. Summary of our controlled study over 10 sessions (from [1])

1. Kazemitabaar et al. Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming. ACM CHI 2023.

LLMs in CS1 may be beneficial overall (still early though!)

During Training

- **Code authoring**: Codex group had 1.8x higher correctness scores and 0.58x completion times
- **Code modification**: Codex group had higher correctness scores, but not statistically significant

- Students in the Codex group reported being more eager to learn programming and felt less stressed, discouraged, and irritated while completing the tasks than those without Codex.

1. Kazemitabaar et al. Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming. ACM CHI 2023.

LLMs in CS1 may be beneficial overall (still early though!)

Evaluation Phase

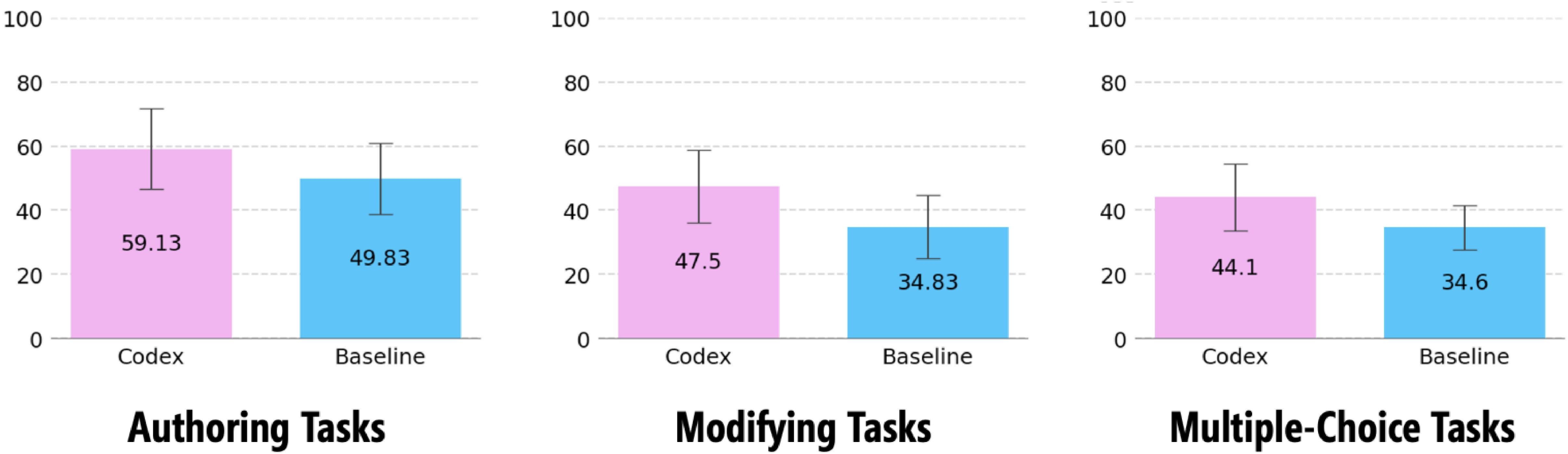


Figure 9. Correctness score of tasks on Retention Post Test (from [1])

1. Kazemitabaar et al. Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming. ACM CHI 2023.

Our Plan Today

1. What we know from the growing body of research
2. **Discussion of where existing CS1 courses fall short**
3. Demo of the workflow with LLMs
4. Our vision for how to teach CS1 incorporating Copilot
5. Q&A

State of Computer Science as Industry Preparation

- Decades long divide between what is taught in CS courses and the skills needed in industry [1]
  - Well-defined assignments vs. open-ended project requirements
  - Short time span on assignments vs. working with legacy code
  - Working individually vs. working on a team
  - Writing standalone programs vs. adding features to codebases
- Many faculty want to help prepare students for industry and are willing to improve their curriculum to do so [2]

1. Craig et al. Listening to early career software developers. CCSC-SW 2018.
2. Valstar et al. A Quantitative Study of Faculty Views on the Goals of an Undergraduate CS Program and Preparing Students for Industry. ACM ICER 2020.

State of CS1 Today

- Long known that students learn less than instructors expect [1]
  - For example: Students struggle writing a basic program (the Rainfall problem) that instructors expect they can complete [1]
- Although failure rates vary, failure rates in CS1 can be high (average failure rate of 28-33% globally) [2]
- Students experience an emotional toll learning how to program [3]

1. Soloway. Cognitive Strategies and Looping Constructs: An Empirical Study." Comm. of the ACM 1986.
2. Simon et al. Pass Rates in Introductory Programming and in other STEM Disciplines. ACM ITiCSE 2019.
3. Kinnunen and Simon. My program is ok – am I? Computing freshmen’s experiences of doing programming assignments. CSE 2012.

Our Plan Today

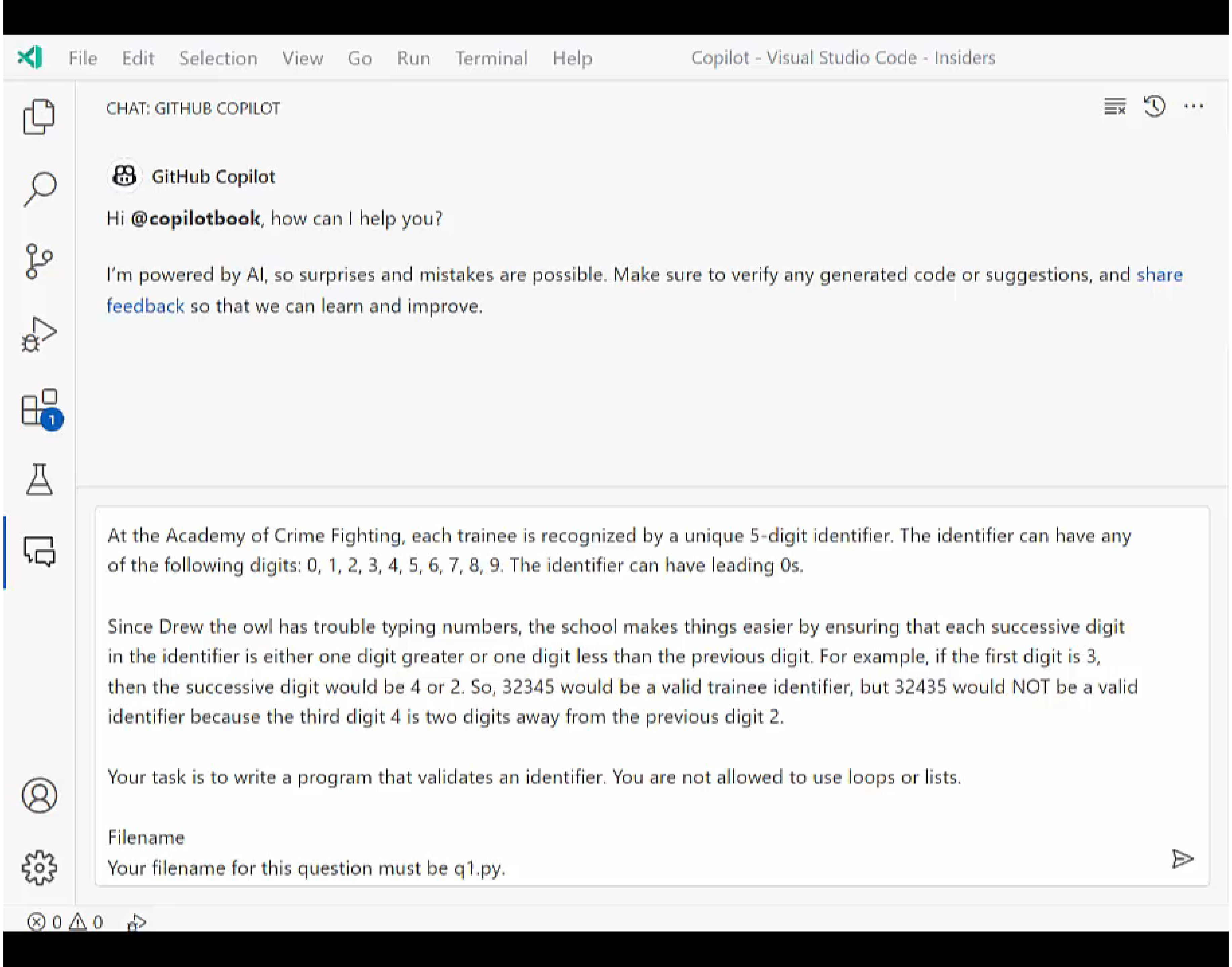
1. What we know from the growing body of research
2. Discussion of where existing CS1 courses fall short
3. **Demo of the workflow with LLMs**
4. Our vision for how to teach CS1 incorporating Copilot
5. Q&A

Solving Dan’s CS1 Assignment

At the Academy of Crime Fighting, **each trainee is recognized by a unique 5-digit identifier.** The identifier can have any of the following digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. The identifier can have leading 0s.

Since Drew the owl has trouble typing numbers, the school makes things easier by ensuring that **each successive digit in the identifier is either one digit greater or one digit less than the previous digit.** For example, if the first digit is 3, then the successive digit would be 4 or 2. So, 32345 would be a valid trainee identifier, but 32435 would NOT be a valid identifier because the third digit 4 is two digits away from the previous digit 2.

Your task is to write a program that validates an identifier. You are not allowed to use loops or lists.



GitHub Copilot

Hi **@copilotbook**, how can I help you?

I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and share feedback so that we can learn and improve.

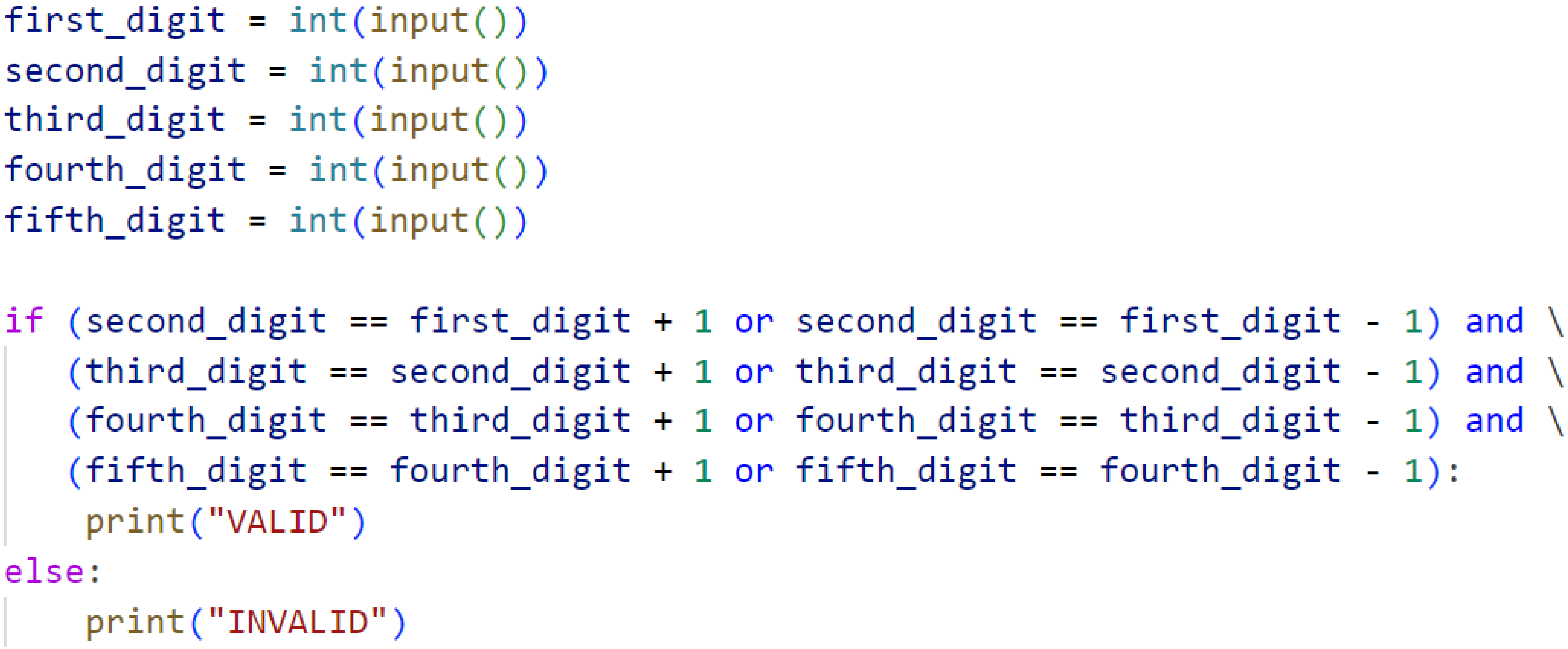
At the Academy of Crime Fighting, each trainee is recognized by a unique 5-digit identifier. The identifier can have any of the following digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. The identifier can have leading 0s.

Since Drew the owl has trouble typing numbers, the school makes things easier by ensuring that each successive digit in the identifier is either one digit greater or one digit less than the previous digit. For example, if the first digit is 3, then the successive digit would be 4 or 2. So, 32345 would be a valid trainee identifier, but 32435 would NOT be a valid identifier because the third digit 4 is two digits away from the previous digit 2.

Your task is to write a program that validates an identifier. You are not allowed to use loops or lists.

Filename
Your filename for this question must be q1.py.

Code from Copilot



```
first_digit = int(input())
second_digit = int(input())
third_digit = int(input())
fourth_digit = int(input())
fifth_digit = int(input())

if (second_digit == first_digit + 1 or second_digit == first_digit - 1) and \
   (third_digit == second_digit + 1 or third_digit == second_digit - 1) and \
   (fourth_digit == third_digit + 1 or fourth_digit == third_digit - 1) and \
   (fifth_digit == fourth_digit + 1 or fifth_digit == fourth_digit - 1):
    print("VALID")
else:
    print("INVALID")
```

When Copilot works well

- A natural language interface to asking computers to do what we want.
- Hasn’t the last 70 years of computer progress leveraged advancements in the ease of communicating our goals with computers?
- For CS1, this could means communicating with the LLM and not needing to write code or learn syntax.
- But Copilot isn’t always right…

Solving Another of Dan’s CS1 Assignments

Problem is to add commas (if missing) to a list in a sentence (where the sentence structure is quite rigid). Solution can use string `replace`, but no conditionals or loops.

- Example 1:

```
Input: Cats like sleeping eating and purring.

Exp. Output: Cats like sleeping, eating, and purring.
```

- Example 2:

```
Input: Alana eats apples oranges, kiwis and bananas.

Exp. Output: Alana eats apples, oranges, kiwis, and bananas.
```

Solving Another of Dan’s CS1 Assignments

Dan’s writing a new book, but he needs a bit of help with his grammar. Correct the given sentences by adding commas in the correct locations throughout the sentence.

Input

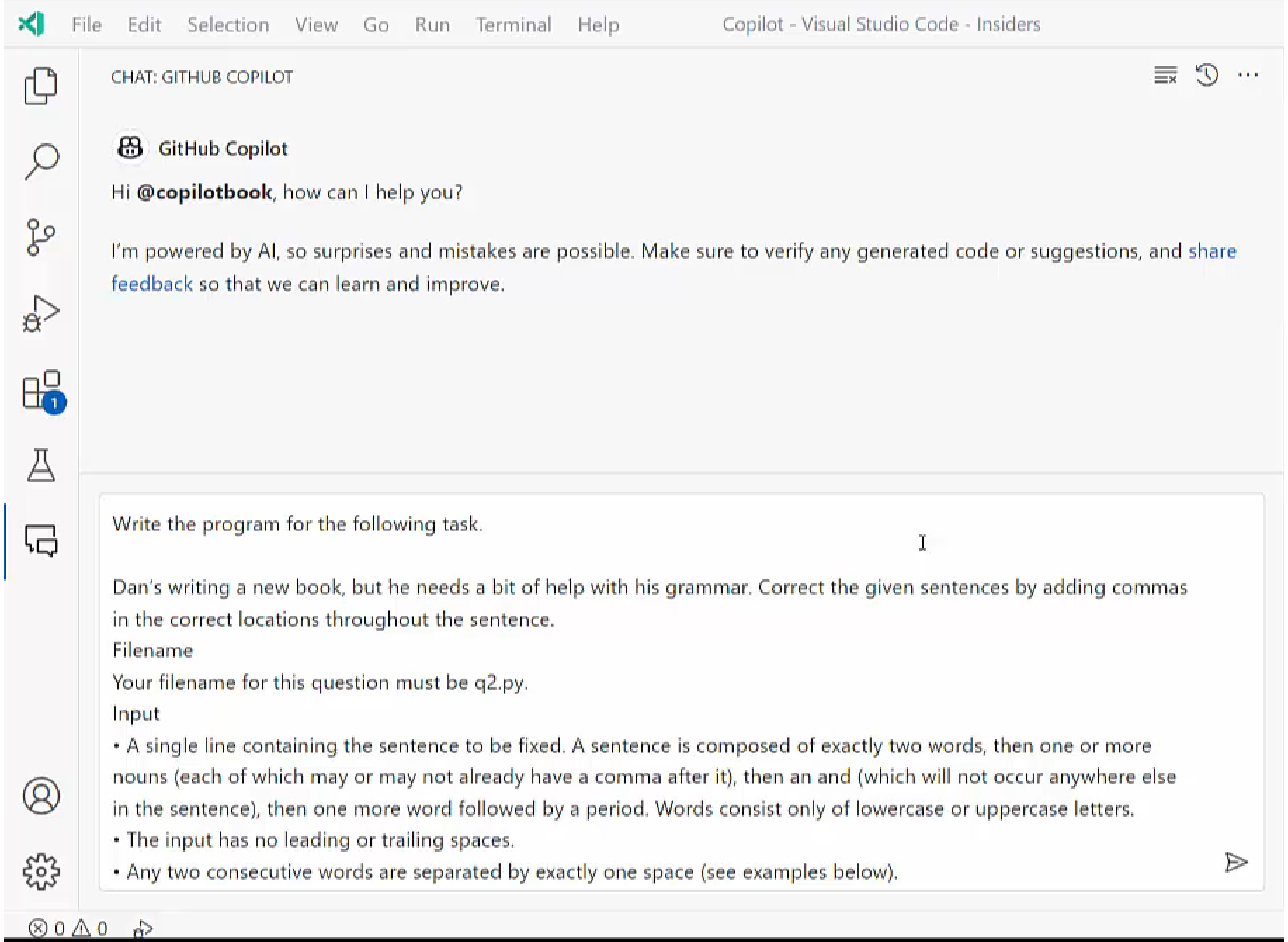
- A single line containing the sentence to be fixed. A sentence is composed of exactly two words, then one or more nouns (each of which may or may not already have a comma after it), then an and (which will not occur anywhere else in the sentence), then one more word followed by a period. Words consist only of lowercase or uppercase letters.
- The input has no leading or trailing spaces.
- Any two consecutive words are separated by exactly one space (see examples below).
- For example, the sentence: *Dinner is chocolate, goldfish and mustard*. is a valid input, and so is *TAs like code exams tests essays and grading*.

Output

- Output the sentence with one comma after each noun (nouns in this problem are the words indicated above in the input section). Do not add or modify anything else about the given sentence.

Requirements

- You MUST use the string method replace to solve this problem (i.e. no lists, split, or anything else after week 2).
- Using if-statements or loops of any kind is not allowed.



Hi **@copilotbook**, how can I help you?

I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and share feedback so that we can learn and improve.

Write the program for the following task.

Dan's writing a new book, but he needs a bit of help with his grammar. Correct the given sentences by adding commas in the correct locations throughout the sentence.

Filename

Your filename for this question must be q2.py.

Input

- A single line containing the sentence to be fixed. A sentence is composed of exactly two words, then one or more nouns (each of which may or may not already have a comma after it), then an and (which will not occur anywhere else in the sentence), then one more word followed by a period. Words consist only of lowercase or uppercase letters.
- The input has no leading or trailing spaces.
- Any two consecutive words are separated by exactly one space (see examples below).

Code from Copilot

Input:
Alana eats apples oranges, kiwis and bananas.

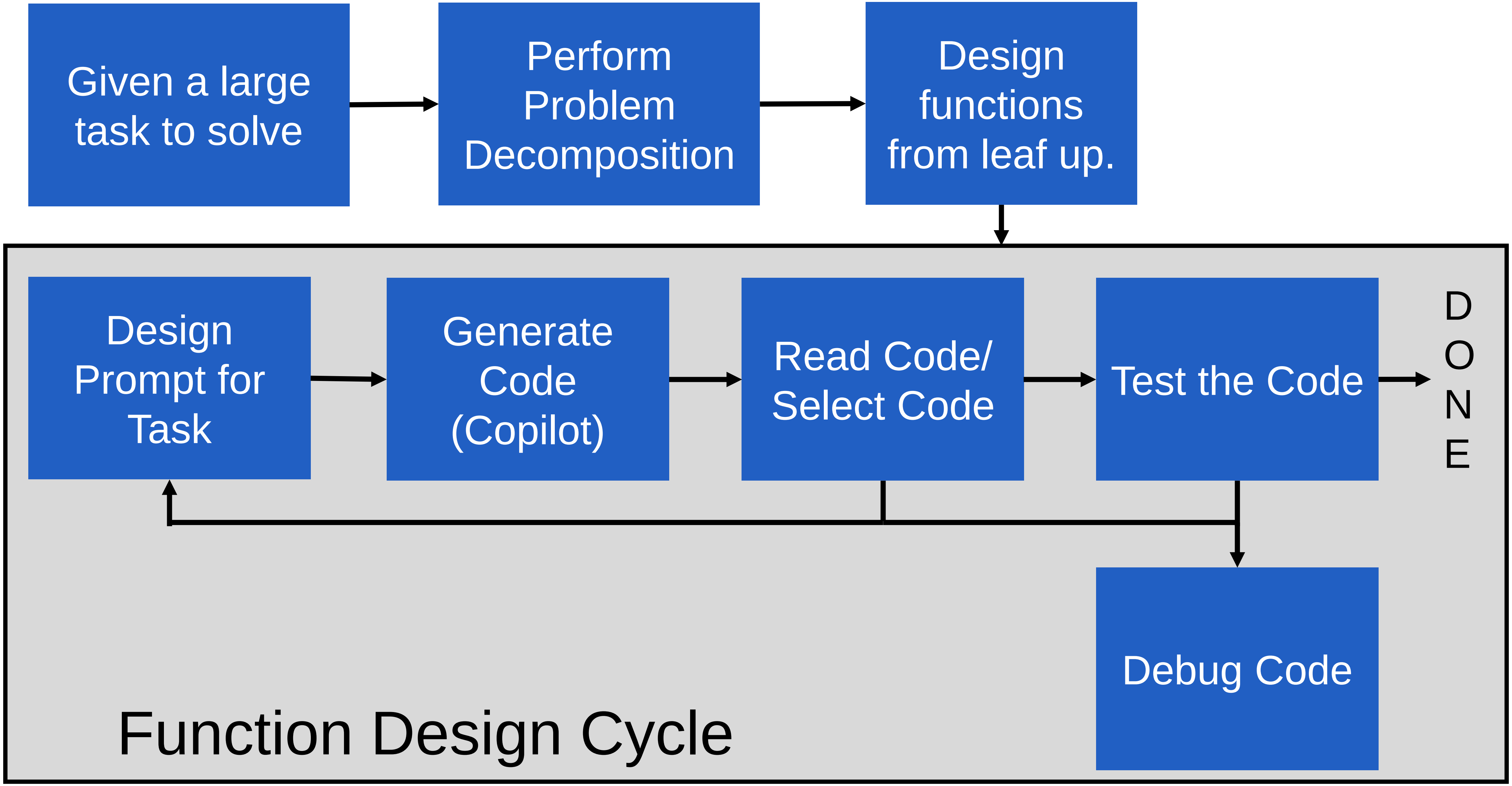
Output:
Alana, eats, apples, oranges,, kiwis,, and, bananas.

```
# Read the input sentence
sentence = input().strip()

# Replace ' and ' with ', '
sentence = sentence.replace(' and ', ', and ')

# Replace the last space with a comma and a space
sentence = sentence.rsplit(' ', 1)[0] + ', ' + sentence.rsplit(' ', 1)[1]

# Replace all spaces before the last word with a comma and a space
sentence = sentence.replace(' ', ', ', sentence.count(' ') - 1)

# Print the corrected sentence
print(sentence)
```

Our Plan Today

1. What we know from the growing body of research
2. Discussion of where existing CS1 courses fall short
3. Demo of the workflow with LLMs
4. **Our vision for how to teach CS1 incorporating Copilot**
5. Q&A

The workflow when writing software with Copilot


Given a large task to solve

Perform Problem Decomposition

Design functions from leaf up.

Design Prompt for Task

Generate Code (Copilot)

Read Code/ Select Code

Test the Code

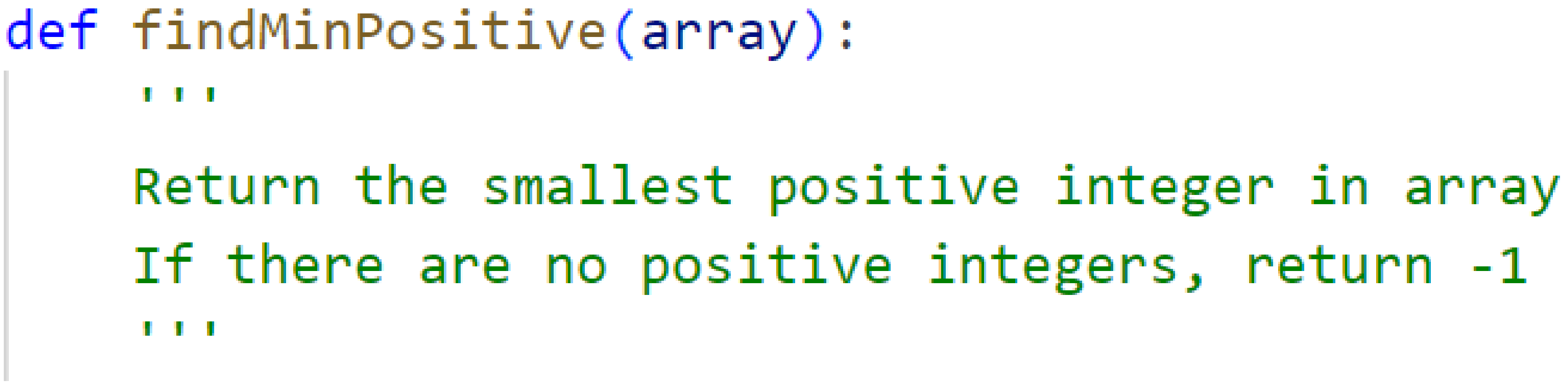
DONE

Debug Code

Function Design Cycle

Skill: Function Design and Prompting

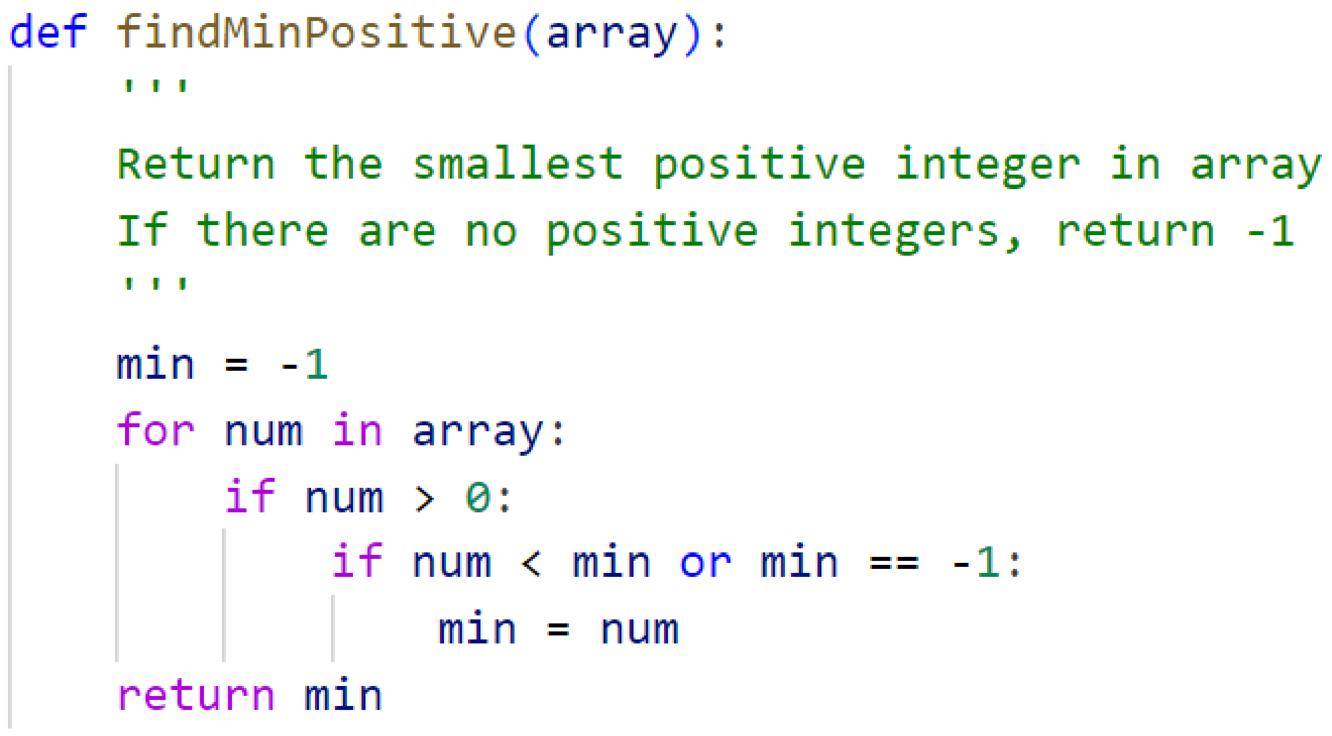
- Create the function signature
- Describe the behavior of the function
    - In sufficient detail for Copilot to try to solve it



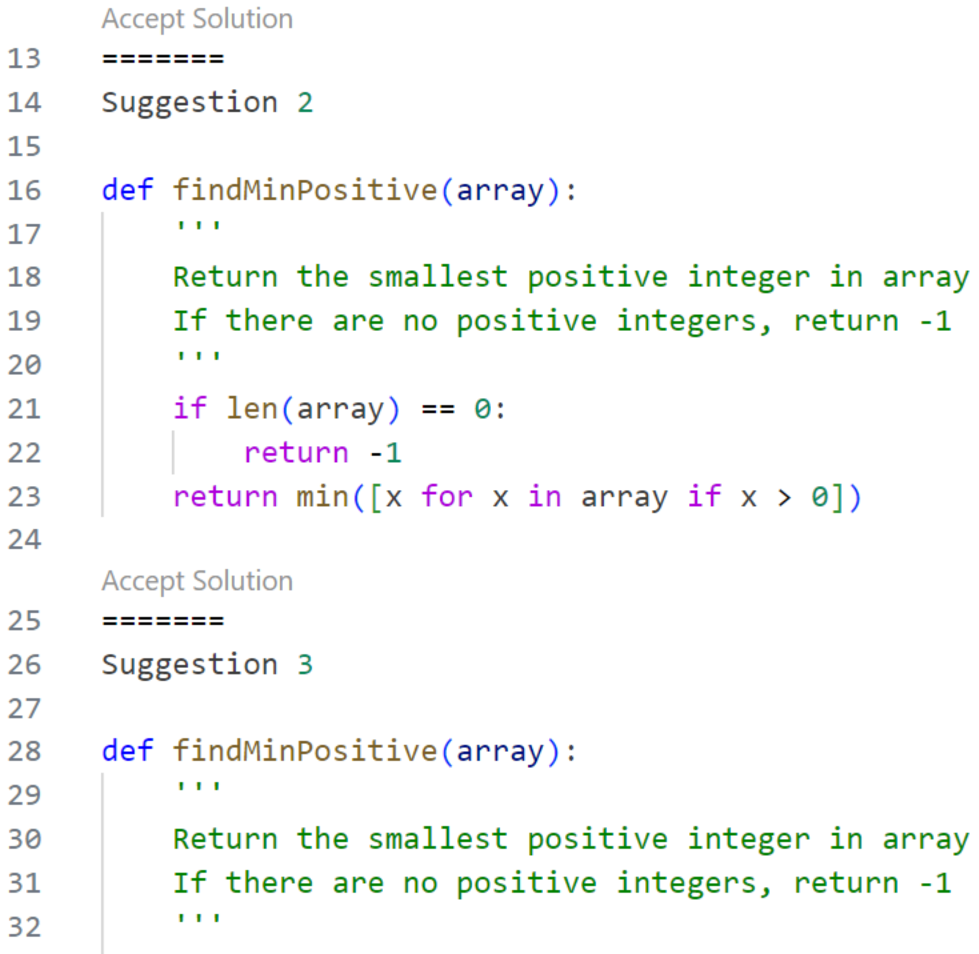
```python
def findMinPositive(array):
    '''
    Return the smallest positive integer in array
    If there are no positive integers, return -1
    '''
```

Skill: Code Reading and Selection

- Understand the code well enough to recognize if it’s trying to solve the task.
- Be able to read multiple suggestions of code and select one that seems likely to help solve the problem



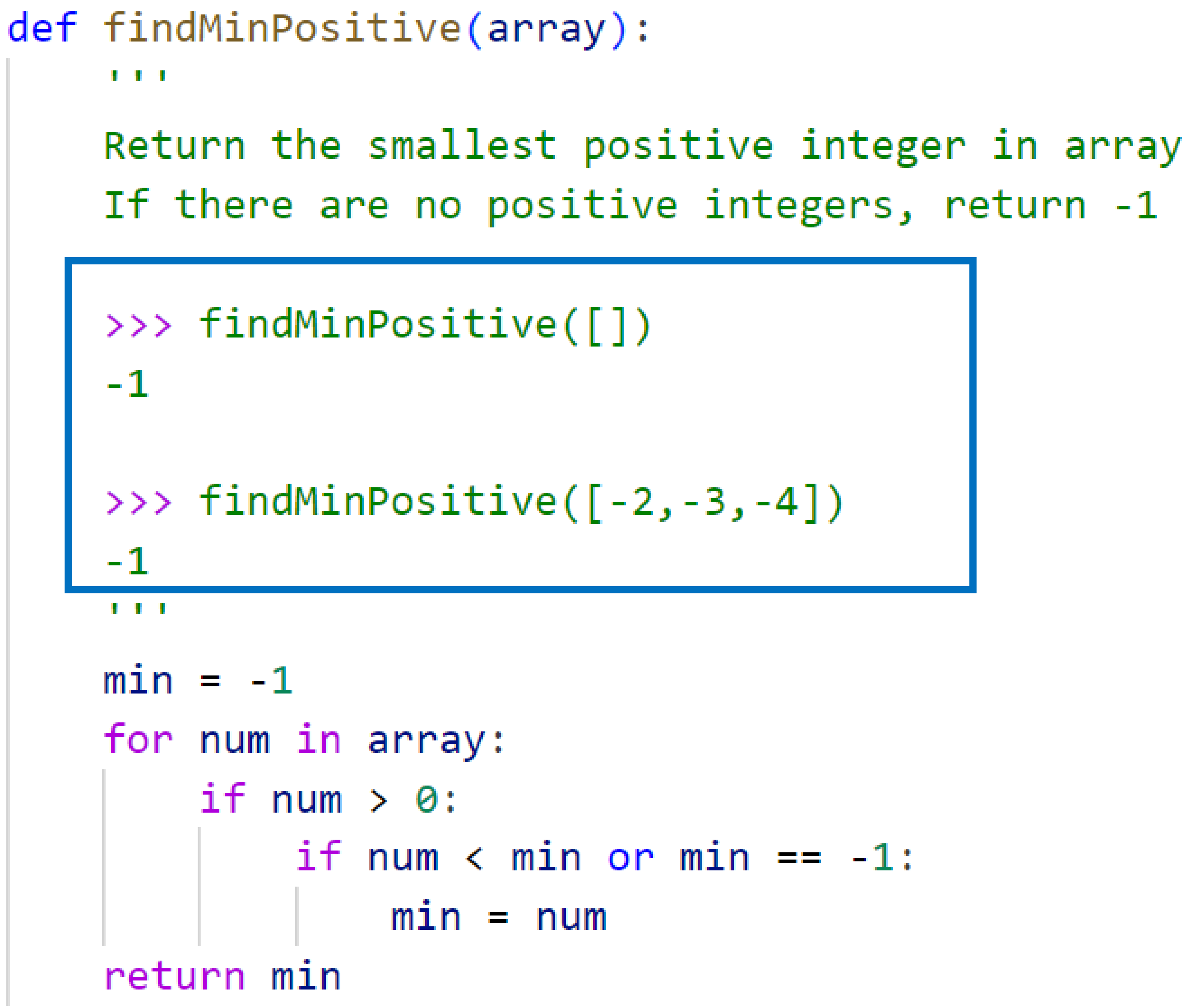
```
def findMinPositive(array):
    '''
    Return the smallest positive integer in array
    If there are no positive integers, return -1
    '''
    min = -1
    for num in array:
        if num > 0:
            if num < min or min == -1:
                min = num
    return min
```



```
Accept Solution
=======
Suggestion 2

def findMinPositive(array):
    '''
    Return the smallest positive integer in array
    If there are no positive integers, return -1
    '''
    if len(array) == 0:
        return -1
    return min([x for x in array if x > 0])

Accept Solution
=======
Suggestion 3

def findMinPositive(array):
    '''
    Return the smallest positive integer in array
    If there are no positive integers, return -1
    '''
```

Skill: Code Testing

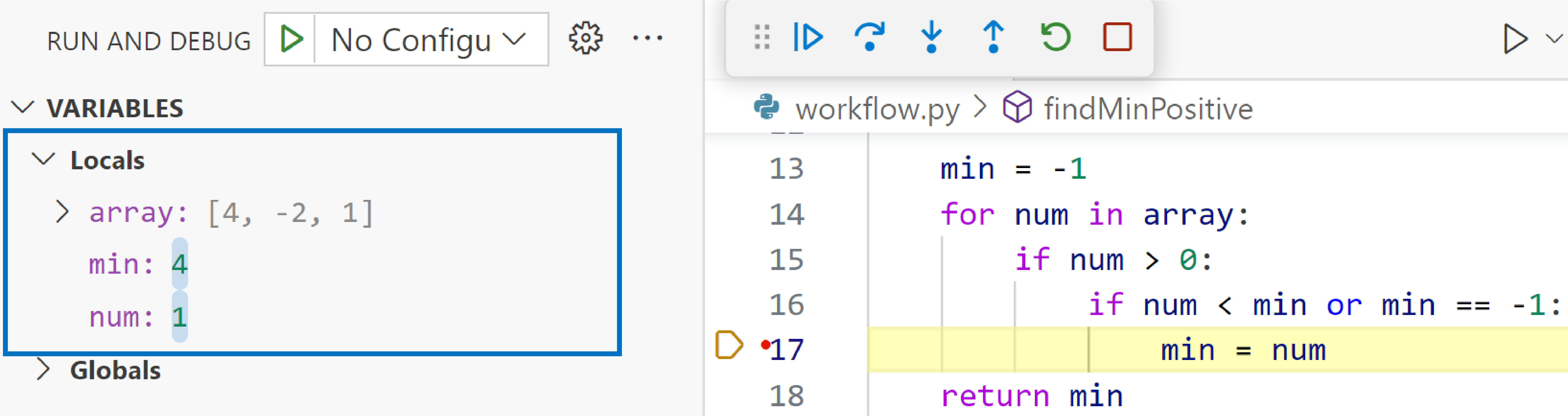
- Open- and Closed-box testing
- Understand common case vs. edge-cases
    - understand code well enough to identify possible edge cases
- Use doctest to test the code



```
def findMinPositive(array):
    '''
    Return the smallest positive integer in array
    If there are no positive integers, return -1

    >>> findMinPositive([])
    -1

    >>> findMinPositive([-2,-3,-4])
    -1
    '''
    min = -1
    for num in array:
        if num > 0:
            if num < min or min == -1:
                min = num
    return min
```

Skill: Debugging

- Use VSCode Debugger to gain insight into the behavior of the program
  - Know how to add breakpoints, follow variable state during execution



Locals

array: [4, -2, 1]

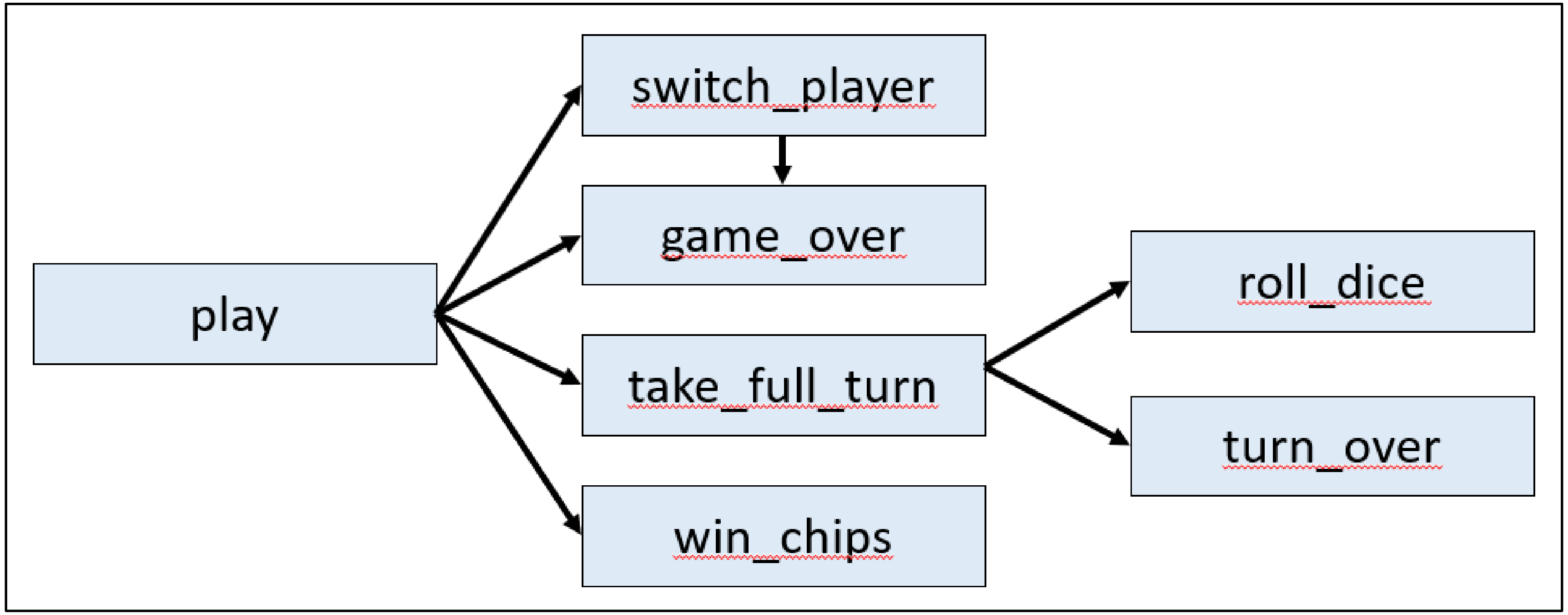
min: 4

num: 1

```python
    min = -1
    for num in array:
        if num > 0:
            if num < min or min == -1:
                min = num
    return min
```

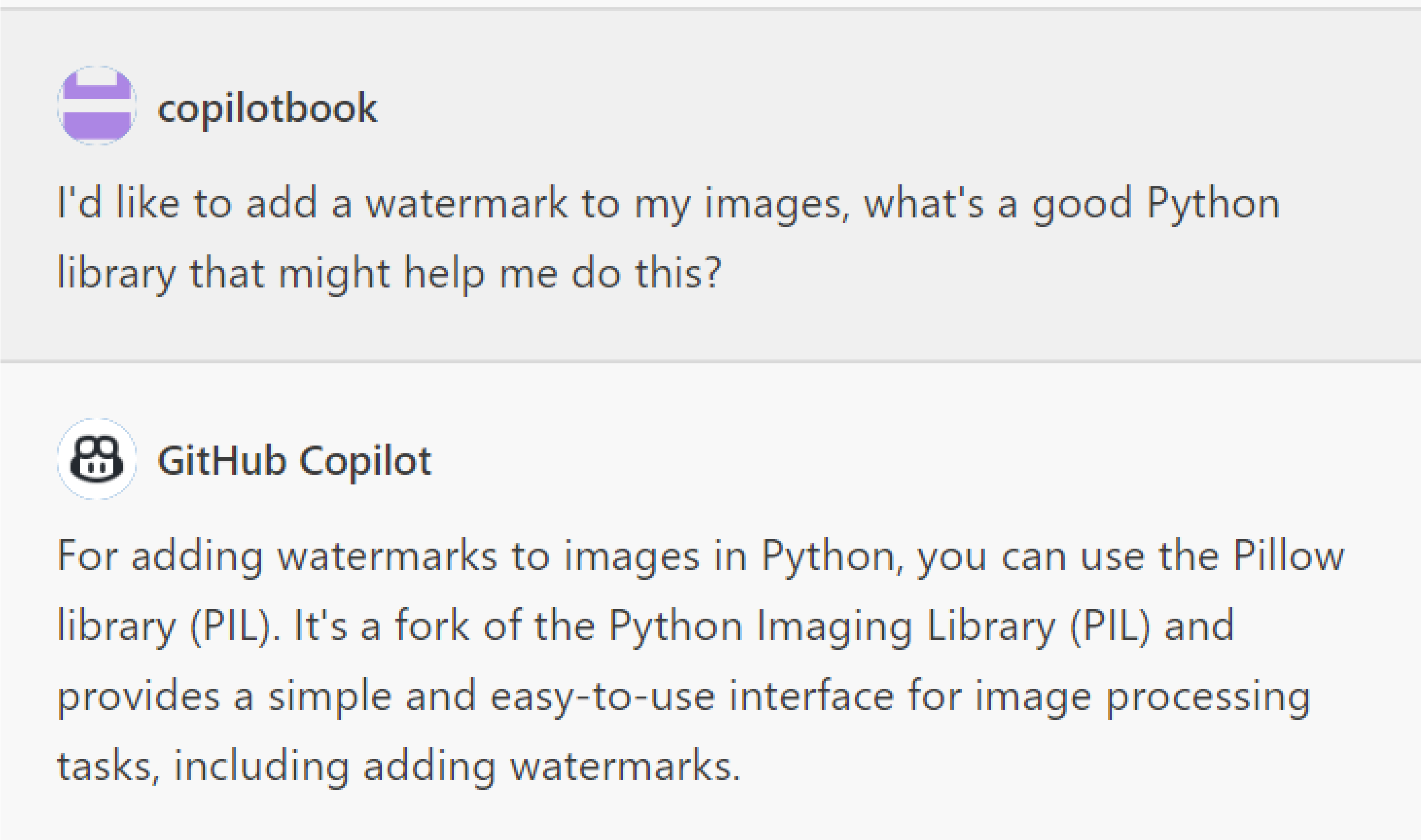
Skill: Problem Decomposition

- Necessary to solve any medium/large task and we want students solving real tasks
  - Need to be able to break this into functions
- Often helpful for debugging
  - Breaking a function that is incorrect into smaller pieces helps Copilot



Skill: Identifying Useful Modules

- Knowing which module to use can help guide Copilot to better solutions.
- Conversations with Copilot Chat can make it easy to identify which module to use



copilotbook

I'd like to add a watermark to my images, what's a good Python library that might help me do this?

GitHub Copilot

For adding watermarks to images in Python, you can use the Pillow library (PIL). It's a fork of the Python Imaging Library (PIL) and provides a simple and easy-to-use interface for image processing tasks, including adding watermarks.

Assessing students in this new CS1

- Write software (with Copilot) to solve real tasks:
  - Engage in the full design process
  - Creative and open-ended tasks [1]
- Create alternative representations for the code (e.g., a flow chart) similar to math education [2]
- Computer-based Testing [3]
  - Allows for practice and frequent feedback
  - Let students have access to Copilot for some questions and not others

Figure 1 (from [1])

1. Porter and Simon. Fostering Creativity in CS1 by Hosting a Computer Science Art Show. ACM Inroads 2013.
2. Jacques. Teaching CS-101 at the Dawn of ChatGPT. ACM Inroads 2023.
3. Smith et al. Investigating the Effects of Testing Frequency on Programming Performance and Students' Behavior. ACM SIGCSE 2023.

Why we’re excited about Copilot CS1

- We look forward to teaching students how to:
  - Read code and select correct code from among multiple choices
  - Test code
  - Decompose large problems into smaller tasks
  - Learn about modules that can help solve problems in new domains

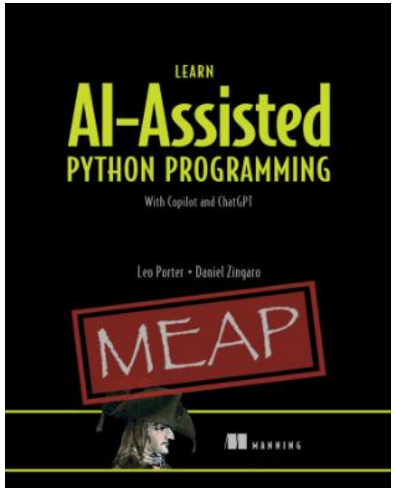
- Students will likely be able to actually produce software after CS1

Challenges

- Need to change not just CS1, this impacts many courses in our curricula
- Tools are free for now for students
  - Could be equity issues if these tools are no longer free
- Ethical challenges of LLMs (e.g., copyright, bias, etc.)

Our Timeline for Fall 2023

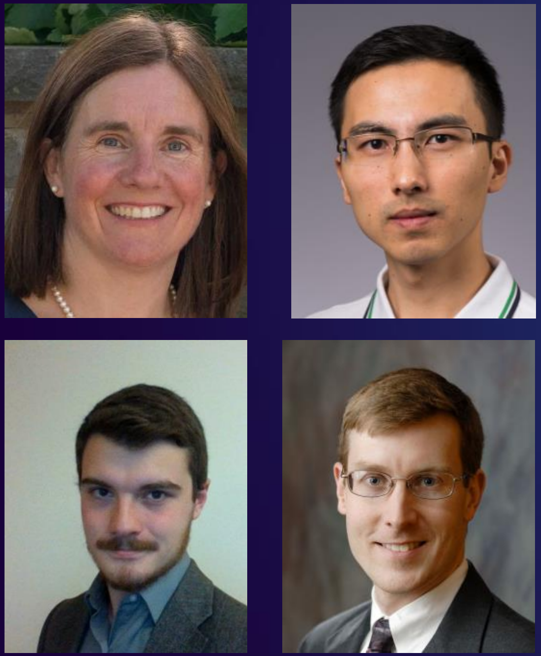
- Will be teaching a class following this approach at UC San Diego
  - Happy to collaborate on course material design
- Book available in MEAP: show book cover
  - Happy to share instructor codes to get access to evaluate.
- Aim to release book by end of August / early September to correspond with the fall term



For an instructor copy, please e-mail: **leporter@ucsd.edu**

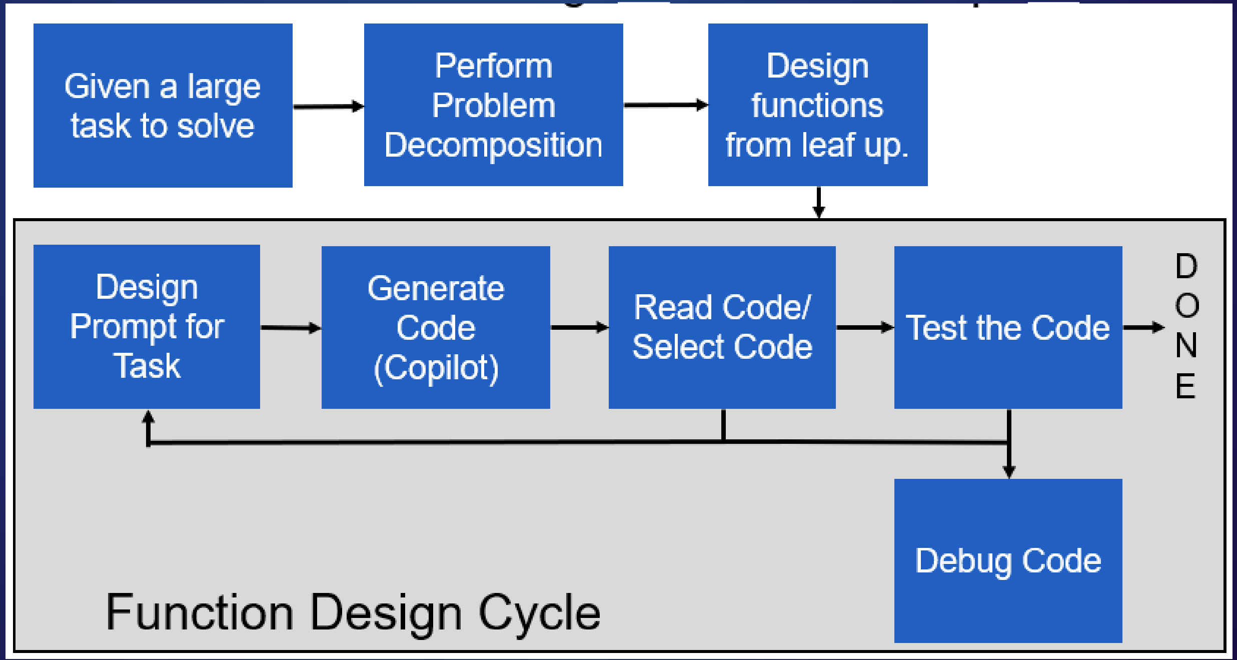
bit.ly/CS1-Copilot

Thank you and questions!

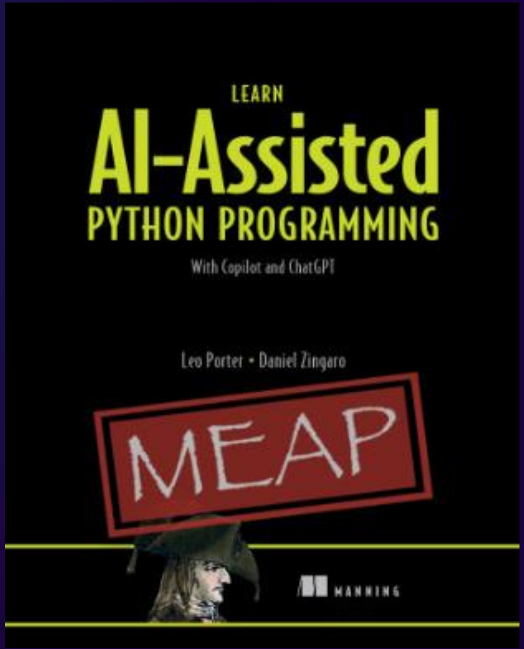


Resist the use of AI coding tools:
- teaching Python fundamentals
- create AI proof assessments
- proctored exams

Embrace AI tools by integrating them
- give personalized help to students
- focus on code reading and critique
- have students collaborate with AI



MANNING



LEARN
AI-Assisted
PYTHON PROGRAMMING
With Copilot and ChatGPT
Leo Porter • Daniel Zingaro
MEAP
MANNING

E-mail for instructor copy: leporter@ucsd.edu

bit.ly/CS1-Copilot

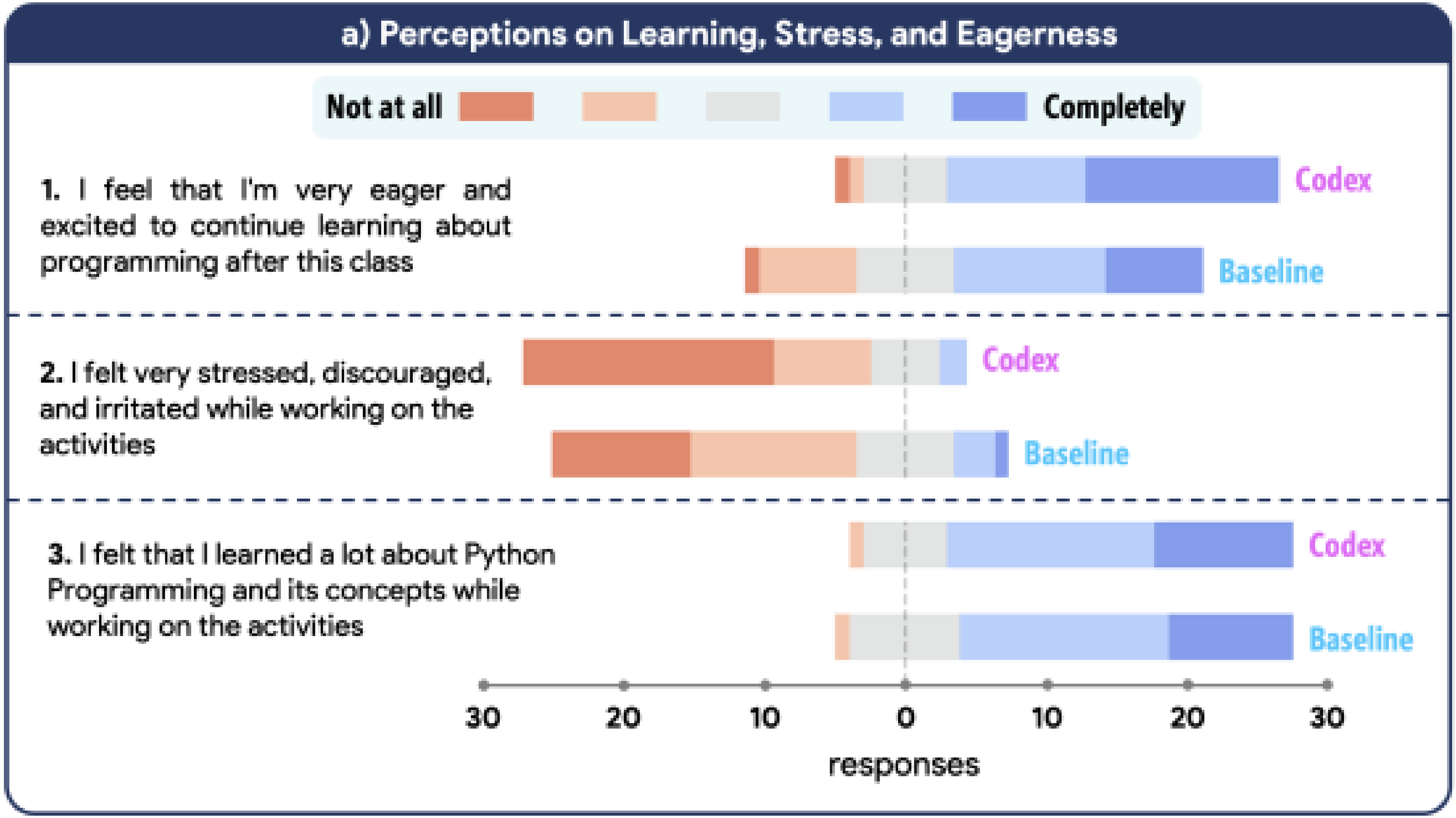
Copilot CS1 - Learning Goals Part 1

- **Level 1: Knowledge**
  - Define nondeterminism, Large Language Model (LLM), prompt, prompt engineering, code correctness, problem decomposition, and top-down design.
- **Level 2: Comprehension**
  - Illustrate the workflow that is used when programming with an AI assistant.
  - Describe the purpose of common Python programming features, including variables, conditionals, loops, functions, lists, dictionaries, and modules.
- **Level 3: Application**
  - Apply prompt engineering to influence code generated by an AI assistant.

Copilot CS1 - Learning Goals Part 2

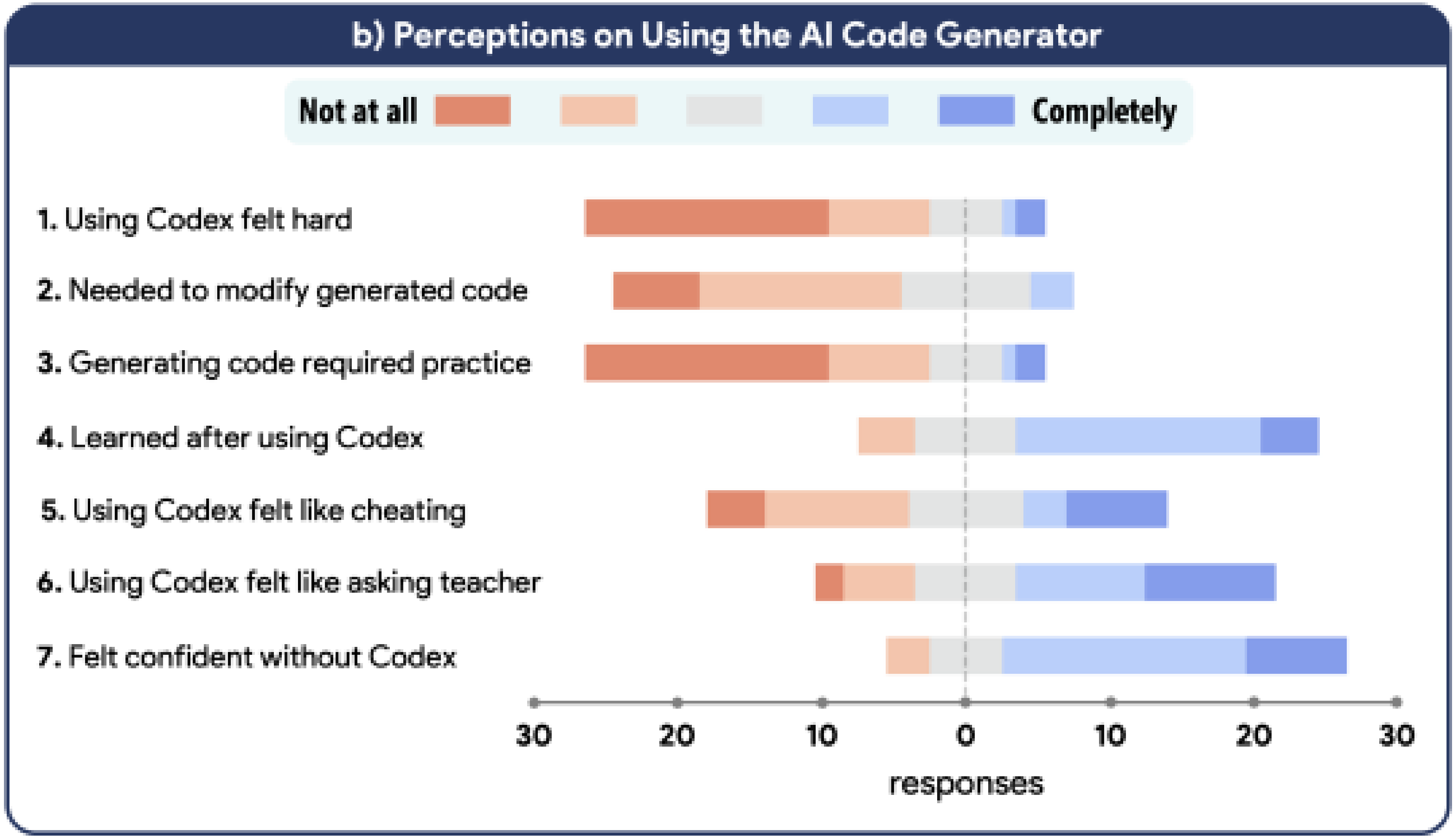
- **Level 4: Analysis**
  - Analyze and trace a Python program to determine its behavior.
  - Divide a programming problem into subproblems as part of top-down design.
  - Debug a Python program to locate bugs.
- **Level 5: Synthesis**
  - Design open- and closed-box tests to determine whether code is correct.
  - Identify and fix bugs in Python code.
  - Perform modifications to Python code to have the code perform a different task.
  - Write complete and correct Python programs using top-down design, prompting, testing, and debugging.
- **Level 6: Evaluation**
  - Judge whether a program is correct using evidence from testing and debugging.

Student perceptions of Codex [from 1]



1. Kazemitabaar et al. Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming. ACM CHI 2023.

Student perceptions of Codex [from 1]



1. Kazemitabaar et al. Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming. ACM CHI 2023.