Copilot and ChatGPT for Java and C# Developers

Essential prompting skills for generative AI-enhanced code

https://github.com/andrewstellman/genai-training

O'REILLY
Head First
C#
A Learner's Guide to Real-World Programming with C# and .NET
Andrew Stellman & Jennifer Greene
Fifth Edition
A Brain-Friendly Guide
Beautiful Teams

Andrew Stellman

I've been training people on software development and other technical topics for over 20 years. *Head First C#*, one of the six books I've written for O'Reilly, is going into its 5th edition. I'm also a full-time software developer and team lead, and I'm passionate about all things code.

Go to https://bit.ly/sens-ai-course
for the slides in this deck.

Take-Home Exercise

This is the take-home exercise from the end of the course. You can do it in any language, using whatever AI tool works best for you.

⚠️ Generating code is uncertain! ⚠️

- **This exercise may take longer than the remaining course time. It's a take-home project for you to practice AI skills.**
- **Always review AI-generated code carefully.** Copilot and other AI tools may sometimes produce code that doesn't compile or work as expected.
- **You're in charge!** Apply the best practices and critical thinking skills we've discussed throughout the course.
- **I'm still here to help.** If you encounter difficulties or want to see the process in action, a great way to get in touch with me is by raising an issue on the GitHub page for the course: https://bit.ly/coding-ai-course
- **Every AI experience is different.** But don't be discouraged if your AI doesn't work the way you want it to. Each interaction with AI can produce unique results! Remember—iterate, iterate, iterate.

Create a new project in your favorite IDE

- **Before you begin:** Make sure you have your language's development environment set up. You can use any IDE—Visual Studio Code, Visual Studio, IntelliJ, PyCharm, or whatever you prefer. If your IDE has an AI extension (like GitHub Copilot), make sure it's installed.
- **For Java:** Create a new Java project. For example, you're using VS Code, choose "Java: Create Java Project..." from the Command Palette. If you're using Maven, the maven-archetype-quickstart archetype works well. Name your main class MathQuiz.
- **For C#:** Create a new Console App project. If you're using VS Code, choose ".NET: New Project..." from the Command Palette. If you're using Visual Studio, choose "Console App" from the new project templates. Add an MSTest or NUnit test project to your solution.
- **For Python:** Create a new folder for your project and add a file called math_quiz.py. Optionally, create a virtual environment with python -m venv venv. For unit tests, you'll use pytest—install it with pip install pytest.

Replace the code with a comment (Java/C#)

Java or C# version

You can use this comment as a prompt in an AI chat, or paste it directly into your code file if your IDE has an AI extension like GitHub Copilot – just remove all boilerplate from the Main method (Java) or top-level statements (C#) and add this comment:

```
// Create a console application that quizzes the user with
// random math problems. Generate two random numbers from
// 1 to 9 and pick either addition or multiplication.
// Prompt for an answer, congratulate if correct, retry
// if incorrect. Exit if user enters a non-numeric value.
// Put the game in a class that is testable, with public
// methods to generate operators, numbers, and questions.
```

Replace the code with a comment (Python)

Python version

You can use this comment as a prompt in an AI chat, or paste it directly into your code file if your IDE has an AI extension like GitHub Copilot:

```
# Create a console application that quizzes the user with
# random math problems. Generate two random numbers from
# 1 to 9 and pick either addition or multiplication.
# Prompt for an answer, congratulate if correct, retry
# if incorrect. Exit if user enters a non-numeric value.
# Put the game in a class that is testable, with public
# methods to generate operators, numbers, and questions.
```

Carefully review the AI’s output

- If you used an AI chat, review the generated code carefully before copying it into your IDE. Look for a suggestion that has public methods that can be tested, including a method that generates a question.
- If you're using an AI extension in your IDE (like GitHub Copilot), review the suggested code carefully before accepting it. Look for a suggestion that has public methods that can be tested, including a method that generates a question.
- If you don't like your options, modify the prompt. Iterate until you have a suggestion you like.
- Accept or copy the suggestion into your project. Run the app and make sure it works. If it does, you can delete the prompt comment at the top of the file.

Generate unit tests

Add this comment to your test file if you're using an AI extension in your IDE, or use it as a prompt in your AI chat:

Java / C#

```
// Create unit tests for the math quiz game. Test
// random number generation, operation selection,
// and answer checking.
```

Python

```
# Create unit tests for the math quiz game. Test
# random number generation, operation selection,
# and answer checking.
```

Generate unit tests (continued)

- If you're using an AI extension, let it generate suggestions. If you're using an AI chat, copy the generated tests into your test file.
- Find a suggestion that looks like it has good tests. Accept the suggestion.
- Make sure the code builds with no problems. If there are problems, either fix them by hand or delete the code and ask the AI to regenerate the unit tests.

Run the unit tests and fix them if needed

- Run your unit tests:
  - **Java:** Use your IDE's test runner, or run mvn test from the command line
  - **C#:** Use the Testing view in VS Code, or run dotnet test from the command line
  - **Python:** Run pytest from the command line
- If no tests are displayed, or if there's an error, make sure your code builds without problems.
- If a test doesn't pass, ask the AI to help you fix it. Give it this prompt:

```
Why didn't this test pass?
```

- Read the response and fix the test so it passes.

Ask the AI to test edge cases

You've been using AI to generate code, but there are other ways to use it. Let's add more tests using comments and prompts.

- Go to the bottom of the test class (above the closing bracket) and add this comment:
  - **Java / C#:**

  ```
  // Add a unit test to check an edge case
  ```

  - **Python**

  ```
  # Add a unit test to check an edge case
  ```

- If you're using an AI extension, it should suggest code to complete the test. Accept it and run the test to make sure it passes.
- If you’re using a chat, copy the tests and paste them into your IDE.

Ask the AI to test edge cases (continued)

Now ask the AI to generate additional edge case tests. Use this prompt in your AI chat or IDE:

```
Add unit tests for additional edge cases
```

Review the suggested tests. Add them to your test file, run them, and make sure they pass. Fix any errors.

Refine and document the code

- Go back to your main code file and ask the AI to add documentation. Use one of these prompts:

Java:	`Add XMLDoc comments to all methods`
C#:	`Add JavaDoc comments to all methods`
Python:	`Add docstrings to all functions and classes`

- Review the generated documentation and make any necessary adjustments.
- Use this prompt to get a full explanation of your refined code:

```
/explain this entire app
```

Use Copilot Chat to modify your code

AI can suggest changes to your code. Here are a few things to try:

- Split the code that generates the question into two methods or functions, one to generate a question and one to check the answer. Add unit tests for both.
- Modify the app so it keeps track of how many questions in a row the player gets right. Remember the longest question streak. Implement this with public methods and generate unit tests for those methods.
- Implement robust error handling and input validation. Create a separate method to validate user input, ensuring it handles various edge cases (e.g., non-numeric input, extremely large numbers, negative numbers). Generate unit tests for this validation method, including tests for different types of invalid input.

linkedin.com/in/andrewstellman/ github.com/andrewstellman @andrewstellman.bsky.social

O'REILLY
Head First
C#
Fifth Edition
A Learner's Guide to Real-World Programming with C# and .NET
Andrew Stellman & Jennifer Greene
A Brain-Friendly Guide
PMP
4th Edition
Beautiful Teams

Thank you!

I hope you enjoyed my course! If you want to learn more, follow me on social media, and check out *Head First C#* and my other books on O'Reilly Learning.

Thanks so much for attending!