# Hands-On Activity: Debugging Using Copilot

00:00 Start visual description. Logo: Microsoft. End visual description.  
00:01 Start visual description. Text: Hands-On Activity: Debugging Using Copilot. Slide: Learning Objectives. End visual description.  
00:04 Dev: Today, we're going to tackle a hands-on activity where you'll get to apply GitHub Copilot to debug your code. This will give you practical experience and reinforce what we've learned so far.  
00:07 Dev: In this activity, you'll be debugging code using GitHub Copilot's inline chat, quick chat, and the chat pane. We'll provide you with a sample file, but if you're feeling more ambitious, you can also use a code repository that you've worked with before. This hands-on approach will help you get comfortable with Copilot's debugging features and enhance your ability to maintain code quality.  
00:18 Dev: First, download the sample files from the course Resources section or use your own code. For instance, if you have a project in JavaScript that you want to debug, feel free to use it for this activity. We'll be using a simple Python script with a few bugs. For this activity, we want you to use GitHub Copilot's inline chat, quick chat, and chat pane for a debugging task. If you're using the sample file we provided, we want you to identify and fix three bugs using each of Copilot's chat functionalities. Make sure you understand each of the fixes, utilizing Copilot to help explain, if necessary.  
00:43 Dev: If you're using a more complex repository that you own, just work on debugging it and employ the different Copilot chat capabilities.  
00:53 Dev: For our first task, we'll be working with a Python Flask application. The code has several issues that need to be addressed. Your job is to use GitHub Copilot to identify and fix these issues. You'll use the quick chat feature for the first bug, inline chat for the second, and the chat pane for the third. Make sure you understand each of the fixes, utilizing Copilot to help explain if necessary.  
01:17 Start visual description. Slide: Task 1: Steps. End visual description.  
01:18 Dev: Open the Python file app.py in your code editor. Use GitHub Copilot to analyze the code and find potential bugs or areas of improvement. Identify and fix at least three issues using different Copilot features. If you're using a more complex repository that you own, just work on debugging it and employing the different Copilot chat capabilities. You should pause here and take a look at all of the steps for Task 1. Try to implement the fixes using GitHub Copilot. I'll go through the solution now.  
01:49 Start visual description: Slide: Task 1: Solution. End visual description.   
01:51 Dev: For the first issue, open the GitHub Copilot quick chat and type "enhance error handling in the data API function." Press Enter.  
02:17 Start visual description. Sample.py code screen. Dev presses the Chat icon on the screen's lefthand dashboard. He types into the prompt bar at the bottom of the screen. A window for GitHub Copilot opens and generates lines of code. End visual description.  
02:24 Dev: Copilot will handle error handling to manage cases where the data might be null and void. This ensures that your API endpoint is robust and can handle unexpected inputs gracefully.  
02:34 Start visual description. Dev copies and pastes a section of code from Copilot into his code window. End visual description.   
02:40 Dev: For the second issue, use the inline chat and type "improve error handling in sum of squares."  
02:42 Start visual description. Dev hovers his mouse over a line of code, and an "Ask Copilot" text line appears. He types into the text line. He presses Accept underneath the text line.   
02:48 Dev: Copilot will add a check to ensure that the input is an iterable list of numbers and return a JSON response for the error message if it's not.  
02:56 Start visual description. Dev types his prompt into the inline chat at the top of the screen. He presses Enter, and Copilot generates code. End visual description.  
03:05 Start visual description. Dev presses the Apply in Editor button at the top of the window. End visual description.  
03:08 Dev: For the third issue, type "modify the about route to return the HTML template."  
03:11 Start visual description. Slide: Task 2: Sample Code With Bug (Java). The Sample.py code screen. Dev zooms in on the lines of code. He highlights lines 1 through 9, opens the inline chat window, and types "Fix the bug in the findMax function when no argument is passed" into it. End visual description.  
03:11 Dev: Copilot will modify the about function to render an HTML template instead of returning a plain string.  
03:29 Start visual description. A highlighted segment is inserted into the code, and Dev presses the Accept button under the text line. End visual description.  
03:38 Dev: Next, let's assume we want to copy any code from a different language and have it debugged by Copilot.  
03:45 Start visual description. Slide: Task 3: Sample Code With Bug (Java). The Sample.py code screen. Dev opens the chat and enters the prompt. A highlighted segment is inserted into the code. End visual description.  
03:49 Dev: Here's a simple JavaScript code snippet with an error. Type "fix the bug in the findMax function when no argument is passed" into the inline chat.  
03:55 Start visual description. Dev presses the Accept button under the text line. A window appears with the solution typed out. End visual description.   
04:06 Dev: Observe how Copilot adds a check to ensure the numbers argument is provided and an array is present before attempting to find the maximum value.  
04:16 Start visual description. Copilot types out its suggestions in the window. Dev scrolls down the suggestions and then applies the top suggestion to his code. End visual description.  
04:25 Dev: Finally, let's look at an example with Java code that has an issue handling null values. Open GitHub Copilot inline chat. Type "fix the null pointer exception." Press Enter. Copilot will suggest adding a null check before accessing the string length. To further improve the code, you can ask Copilot to suggest ways to handle null values more elegantly.  
04:44 Dev: Copilot will provide a solution using Java's optional class to handle the null cases more elegantly, demonstrating a more modern approach to null handling in Java.  
04:51 Dev: Remember, while Copilot provides these solutions, it's critical to understand and verify the proposed changes before implementing them in your code.  
05:00 Dev: In this hands-on activity, we've explored the capabilities of GitHub Copilot in assisting with debugging tasks across multiple programming languages.  
05:11 Dev: Throughout these tasks, we saw how Copilot can assist in identifying potential issues, suggesting fixes, and improving overall code quality. We also learned the importance of critically reviewing and understanding the AI-generated code before implementation.  
05:12 Start visual description. Logo: Microsoft. End visual description.