# Writing Your First Function With GitHub Copilot

**To best understand the future of software development, try programming this factorial calculator with GitHub Copilot!**

In software development, there is a constant demand for the quick delivery of efficient code. With the recent creation of GitHub Copilot, a cutting-edge AI-driven coding assistant, developers can now code better innovations more easily and quickly than ever before. To show-off just how easy Copilot can make programming, this project will walk through crafting a factorial calculator. [Factorials](https://simple.wikipedia.org/wiki/Factorial) serve as a great example of a simple problem that has many applications in statistics and algorithm design, such as counting the many arrangements of a group of objects. Even if it has been decades since your last statistics class, this functionality can still be easily built by leaving the specifics of these mathematical properties to Copilot.

### Setting Up GitHub Copilot

Before diving into this new way of coding, it’s crucial to be properly set up. For an in-depth guide on setting up Visual Studio Code, we recommend reviewing our [GitHub Copilot setup article](https://www.codecademy.com/courses/intro-to-github-copilot/articles/setting-up-git-hub-copilot). For those more familiar with the process, here is a high-level overview of what is needed to follow along with this project:

1. **GitHub Account:** If you don’t have one, sign up [here](https://github.com/).
2. **GitHub Copilot Subscription:** Copilot requires a subscription to use. This can be purchased with the option of a free trial [here](https://github.com/features/copilot).
3. **Visual Studio Code (VSCode):** While Copilot can work in a variety of code editors, it integrates best with VSCode. Download it [here](https://code.visualstudio.com/" \t "_blank).
4. **GitHub Copilot Extension:** Once you’re in VSCode, head over to the left extensions sidebar and search for “GitHub Copilot”. Install this official extension and log in to your GitHub account when prompted.

### Working with GitHub Copilot

With the setup complete, the easiest way to get started with Copilot is just to start typing away. While you code as usual, Copilot will make suggestions to speed up your development and proactively prevent errors. More precisely, this loop looks like this:

1. **Start Typing:** As you begin typing in VSCode, Copilot will automatically provide suggestions.
2. **Accept Suggestions:** Press Tab to accept the auto-completion if Copilot correctly understood your intention.
3. **Ask Copilot:** If you’re unsure about how to implement the next feature, just write a comment describing the problem in a comment, such as **// How do I...**. Copilot will then attempt to give you the answer as another code suggestion in your text editor.
4. **Continue Coding:** Keep the momentum going, coding the functions easiest to implement for you, and letting Copilot take the lead when feeling stuck.

### Having Copilot Write the Factorial Function

Now that we know the workflow of working with GitHub Copilot, let’s kick things off with the backbone of our calculator: the factorial function.

Factorials are easy to calculate. They are the product of all positive integers less than or equal to the given number. For example, the factorial of five, written as **5!**, is **5 × 4 × 3 × 2 × 1 = 120**.

To get started, create a new Python file called **factorial\_calculator.py**. Then, have GitHub Copilot help write the function to calculate the factorial:

1. Start by typing **def factorial(n):** in your Python file.
2. GitHub Copilot should automatically suggest code similar to the following:

def factorial(n):   
    if n == 0:   
        return 1   
    else:   
        return n \* factorial(n-1)

1. Press Tab to accept the suggestion.

Voila! In no time at all, you’ve got your function!

Do keep in mind that while the above example is the most common way to calculate a factorial, it is also possible for Copilot to recommend a different solution entirely. For example, instead of recursively multiplying each integer, a loop could be used to calculate the factorial iteratively. This may offer better performance in certain cases and highlights why general technical knowledge is still important even when leaving the coding to Copilot.