

Copilot/ChatGPT for productivity FAQ

Table of Contents

Copilot/ChatGPT for productivity FAQ	1
How to install	1
Why copilot completion so slow?	4
How can I get next suggestion	5
Sometimes the copilot suggests the wrong function, not the one I was expecting	5
Some auto fill cases	8
How to write unit test:	9
How to explain code:	10
How to translate code	11
How to review code:	14
How to edit Jupiter notebook? (@Data Scientist)	14
Copilot short cut	17
How to install Copilot Chat (in Copilot X)	17

How to install

[Copilot organization | Microsoft Open Source Management](#)

1. accept the guidelines:

Review guidelines

Guidelines for GitHub Copilot at Microsoft

You are responsible for the code you write with Copilot. All teams using Copilot are reminded to follow existing team practices for source code - you should ensure that the suggestions you accept abide by the same policies that apply to code you write without Copilot, for example, [Security Development Lifecycle \(SDL\)](#) requirements for your organization.

Source code and data collection notice. Code Snippets Data and User Engagement Data are collected by GitHub to be used by GitHub and Microsoft as described on the feature page. Code Snippets Data includes source code that you are editing, related files, and other files open in the same IDE or editor, URLs of repositories and file paths. User Engagement Data includes users edit actions like completions accepted and dismissed, and error and general usage data to identify metrics like latency and features engagement, and may include pseudonymized personal data.

Your use of Copilot must be consistent with any agreements and tenting requirements that apply to your project. If you are (i) working on a tented project, (ii) working in a source repo that also contains third-party confidential source code, or (iii) working on code that is subject to export controls or is otherwise considered sensitive, please consult with your frontline CELA before using Copilot.

Flag potential problems. Email copilot-safety@github.com if you identify privacy, security, offensive content, or other issues in Copilot suggestions, and do not use suggestions that present issues. If you are not sure if a Copilot suggestion presents a problem, consult your CELA contact.

Personal use may be commingled with Microsoft data. If you use Copilot in your personal capacity (i.e., outside of your use as an employee or vendor of Microsoft), please be aware that your Copilot telemetry may be combined with Microsoft data within Microsoft's account. This issue may be fixed in an upcoming release.

Some teams onboarding after Copilot for Business Launch. Following the November announcement of Copilot for Business, some Microsoft businesses have decided to wait for this launch to onboard to Copilot. If you are unable to onboard to the system at this time, this could be why - check with your management or CELA contact if you have questions.

☒ I understand the guidelines

Continue

You may receive surveys & comms regarding Copilot to share your feedback and learnings. We'll email you these guidelines, too.

Copilot for your Enterprise Managed User account

Install the extensions

C

Ol

Gd

Gd

Pu

Int

Vi

Onb

Cc

Cc

Cc

M

2. Enable Github Copilot

3. Download extension

GitHub Copilot at Microsoft

You have GitHub Copilot access.

Review guidelines

Show guidelines

GitHub Copilot enabled for your y1qhou_microsoft Enterprise Managed User account

Install the extensions, sign in to your y1qhou_microsoft account, and enjoy your Copilot

Visual Studio Code Extension

Visual Studio Extension

Optional: would you also like to use Copilot with your own GitHub account for open source use?

Yes, show me how

4. Use your alias_microsoft to login GitHub. Do not need password

Sign in to GitHub

Username or email address

Password [Forgot password?](#)

Sign in with your identity provider

Device Activation

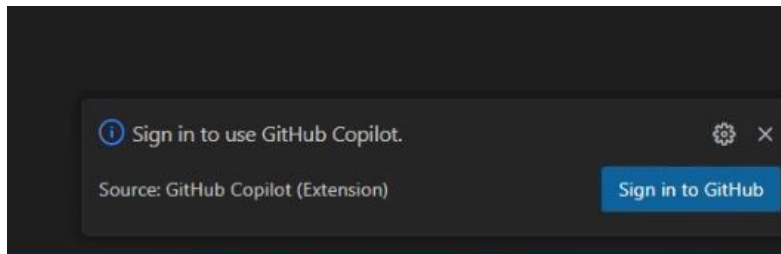


Single sign-on to **Microsoft**
EMU

Authenticate your account by logging into
Microsoft EMU's single sign-on provider.

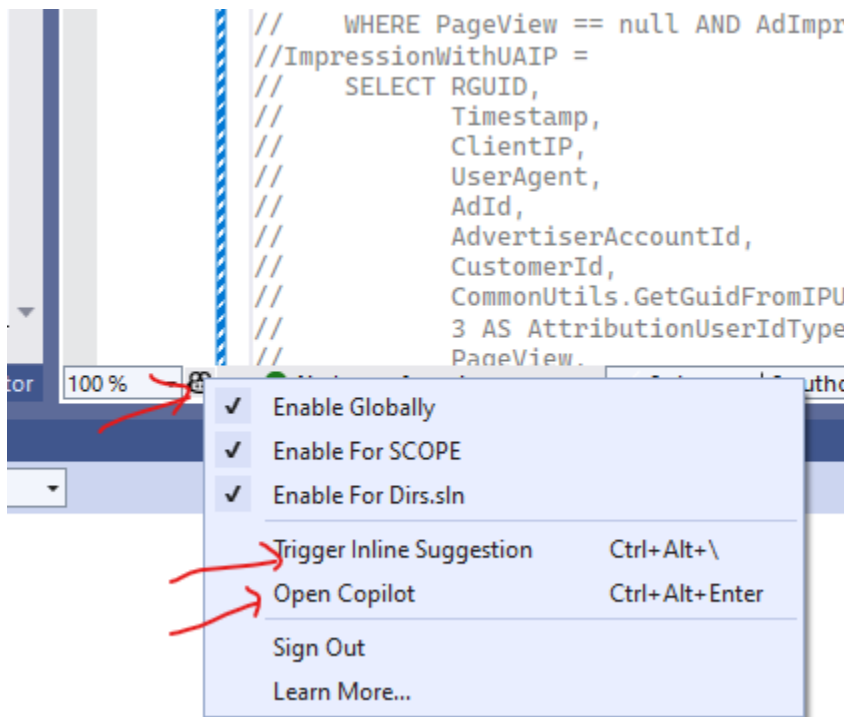
Continue

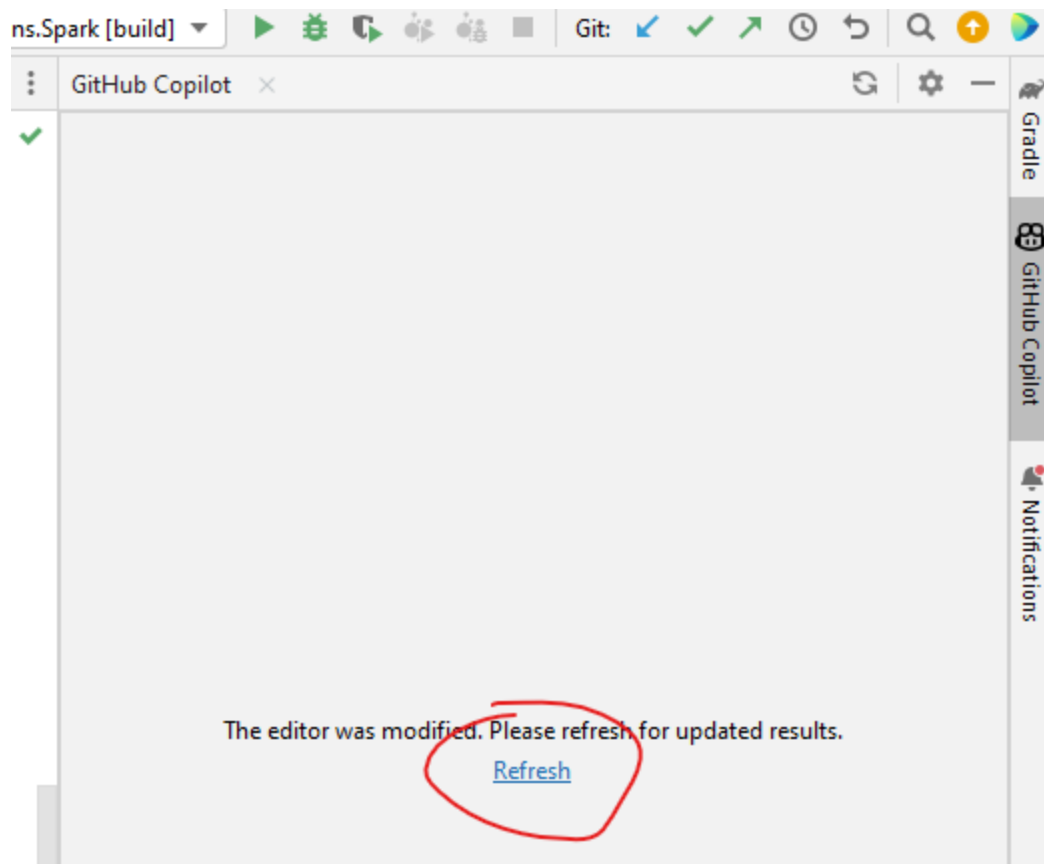
5. Login and active your extension



Why copilot completion so slow?

You can manually trigger. Click left bottom in VS, or short cut Ctrl+Alt+\\, Ctrl+Alt+Enter





Open Github Copilot tab in JetBrains IDEA, and click “Refresh”

How can I get next suggestion

[Configuring GitHub Copilot in Visual Studio - GitHub Docs](#) **Alt + .** next ; **Alt + ,** previous

[Configuring GitHub Copilot in Visual Studio Code - GitHub Docs](#) **Alt +]** next; **Alt + [** , previous

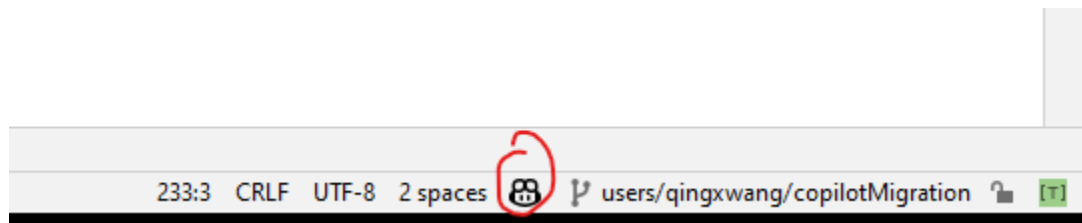
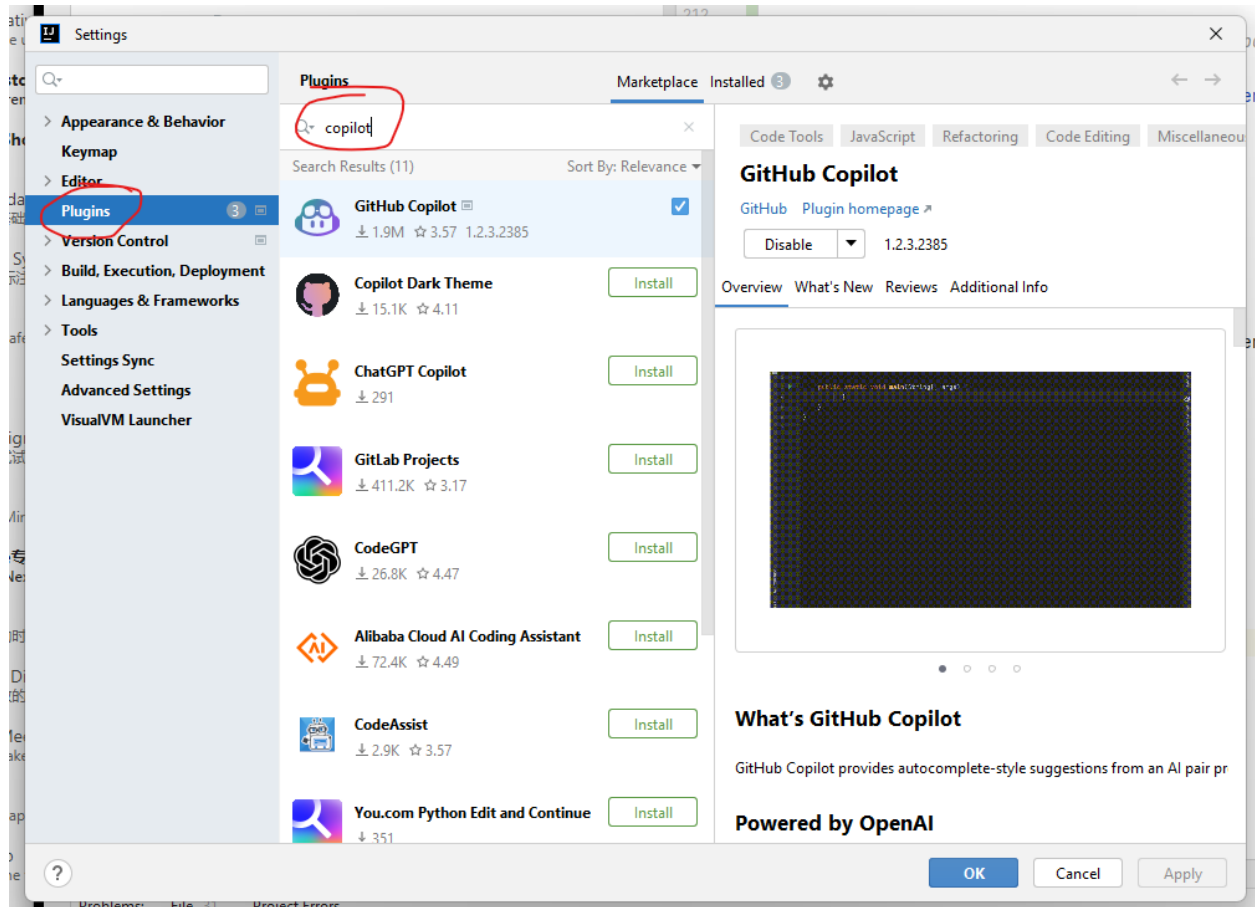
[Configuring GitHub Copilot in a JetBrains IDE - GitHub Docs](#) **Alt +]** next, **Alt + [** , next

Sometimes the copilot suggests the wrong function, not the one I was expecting

Paste the right function/class's interface to the comment

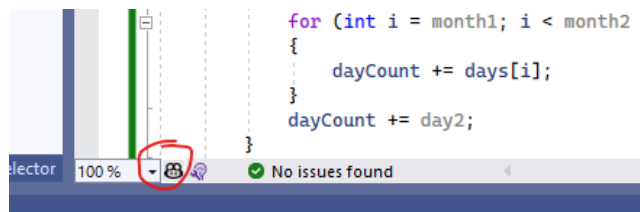
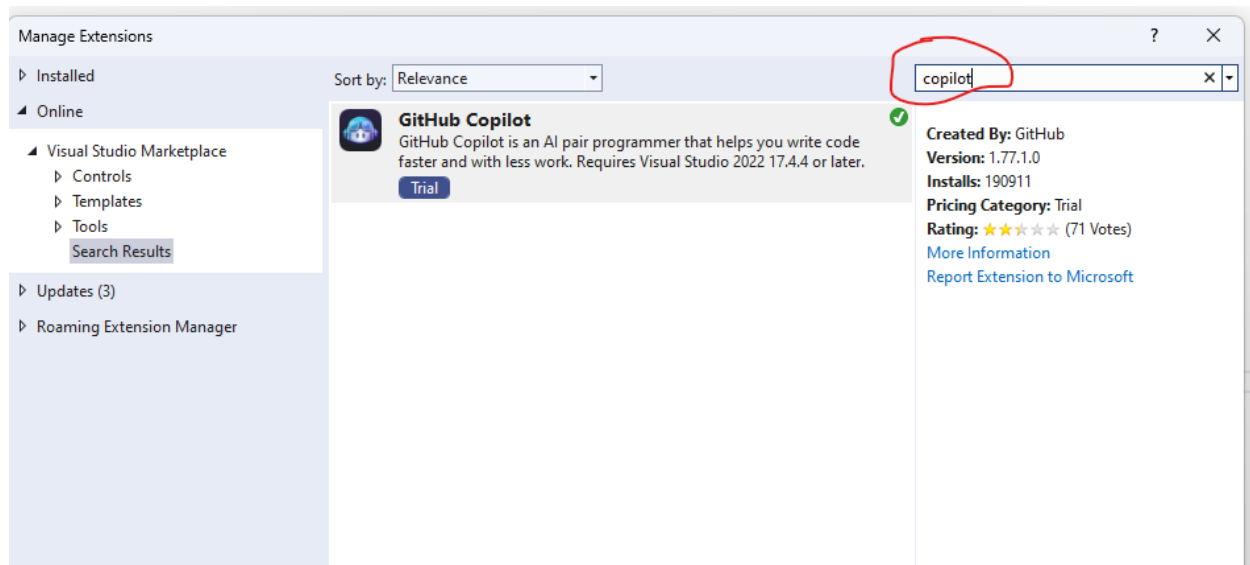
Thanks to Jie Li, Taiyuan, Andrew Chen, Yiwen, Zhen Xuan, Zehua, Zheng Li, Jason Guo's feedback!

JetBrains IDEA

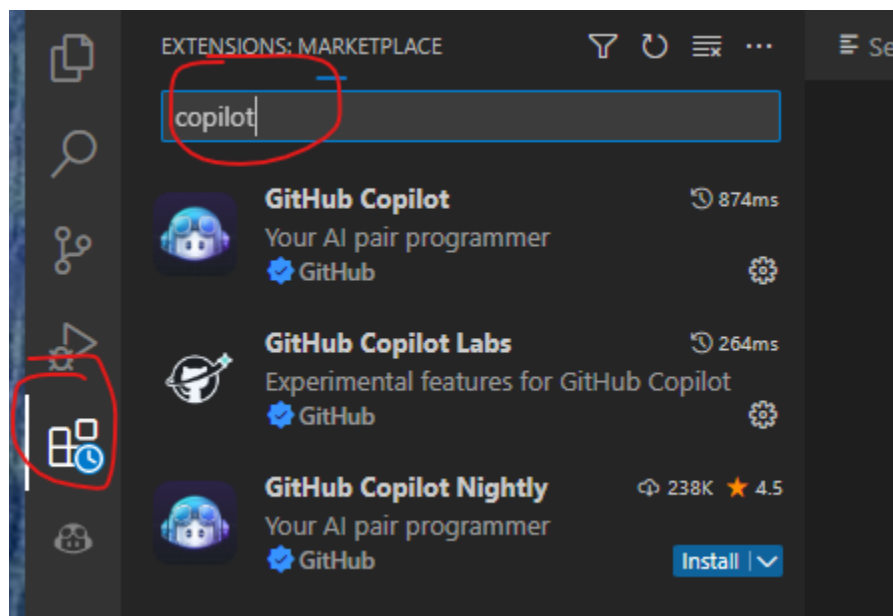


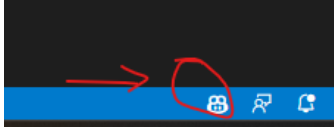
Visual Studio 2022

Extensions -> Manage Extension



Visual Studio Code:





Some auto fill cases

You can directly experience what the copilot is capable of doing:

```
public void SimplePerformanceTest()
{
    long startTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();
    string[] lines = File.ReadAllLines(pathname + filename);
    foreach (var line in lines)
    {
        ProcessLine(line);
    }
    long endTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();
    long elapsedTime = endTime - startTime;
    long lenght = lines.Length;
    Console.WriteLine($"Total time: {0} ms, total lines: {1}, average time: {2} ms", elapsedTime, lenght, elapsedTime / lenght));
}
```

```
def convert_str_to_datetime(date: str):
    return datetime.datetime.strptime(date, "%Y-%m-%d")
```

```
"""
    Get price from a crypto currency
    """
def get_price():
    url = 'https://api.coinmarketcap.com/v1/ticker/bitcoin/'
    response = requests.get(url)
    response_json = response.json()
    return float(response_json[0]['price_usd'])
```

```
"""
    Send an email using gmail
    """
def send_email(subject, msg):
    try:
        server = smtplib.SMTP('smtp.gmail.com:587')
        server.ehlo()
        server.starttls()
        server.login('
```



```

import os
import zip
import zipfile
import sys

def unzip(filename):
    try:
        z = zipfile.ZipFile(filename, 'r')
        for name in z.namelist():
            (dirname, filename) = os.path.split(name)
            # Create the directory
            if dirname and not os.path.exists(dirname):
                os.makedirs(dirname)
            # Write the file
            if filename:
                data = z.read(name)
                file = open(name, 'wb')
                file.write(data)
                file.close()
            z.close()
    except:
        print 'Error: %s' % sys.exc_info()[1]
        return False
    return True

```

Write unit test:

Add the class/function as context, and give instructions in comment to write unit test:

```

//write test case for the method toAdjustmentRecord: some fields are null
test( testName = "test toAdjustmentRecord method: some fields are null" ) {
    val csvString = "1,20348126,1.0,2.0,USD,Restate,"
    val record = csvString.toAdjustmentRecord( delimiter = ',' )
    assert(record.transactionId == "1")
    assert(record.goalId == 20348126)
    assert(record.goalId64 == 20348126L)
    assert(record.adjustmentValue.get == 1.0)
    assert(record.adjustmentCredit.get == 2.0)
    assert(record.adjustmentValueCurrency == "USD")
    assert(record.adjustmentType == "Restate")
    assert(record.adjustmentTime == null)
}

```

```
//write test case for the method toAdjustmentRecord: some fields are null
test("test toAdjustmentRecord method: some fields are null") {
    val csvString = "1,20348126,1.0,2.0,USD,Restate,"
    val record = csvString.toAdjustmentRecord(',')
    assert(record.transactionId == "1")
    assert(record.goalId == 20348126)
    assert(record.goalId64 == 20348126L)
    assert(record.adjustmentValue.get == 1.0)
    assert(record.adjustmentCredit.get == 2.0)
    assert(record.adjustmentValueCurrency == "USD")
    assert(record.adjustmentType == "Restate")
    assert(record.adjustmentTime == null)
}
```

Explain code:

Give construction "explain the code", it also works in GPT and Github Copilot. I found GPT-3.51, GPT4 works better. The explain of Quick Sort:

The code you provided is an implementation of the Quicksort algorithm in Python.

The Quicksort algorithm is a divide-and-conquer algorithm that works by selecting a pivot element from the array and partitioning the other elements into two sub-arrays, according to whether they are less than or greater than the pivot. The sub-arrays are then sorted recursively.

In this implementation, the pivot is chosen as the last element in the array. The `partition()` function is used to find the partition position, which is the index of the pivot element after partitioning. The `quickSort()` function is then called recursively on the left and right sub-arrays of the pivot until the entire array is sorted.

The `partition()` function works by iterating through the array and comparing each element with the pivot. If an element is smaller than the pivot, it is swapped with the element at the pointer `i`, which keeps track of the elements smaller than the pivot. At the end of the iteration, the pivot is swapped with the element at `i+1`, which is the partition position.

The code also includes a test case for the `quickSort()` function, which sorts an array of integers and prints the sorted array.

Overall, this implementation of Quicksort is a simple and efficient way to sort an array of integers in ascending order.

[Translate code](#)

The simplest way is to ask Copilot/GPT to translate it to Scala, C++, and so on:

To translate the Quicksort implementation in Python to Scala, you can follow the same logic and structure of the original code. Here is an implementation of Quicksort in Scala:

```
object Quicksort {  
  // Function to find the partition position  
  def partition(array: Array[Int], low: Int,  
high: Int): Int = {  
    // choose the rightmost element as pivot  
    val pivot = array(high)  
  
    // pointer for greater element  
    var i = low - 1  
  
    // traverse through all elements  
    // compare each element with pivot  
    for (j <- low until high) {  
      if (array(j) <= pivot) {  
        // If element smaller than pivot is found  
        // swap it with the greater element  
pointed by i  
        i += 1  
  
        // Swapping element at i with element at  
j  
        val temp = array(i)  
        array(i) = array(j)  
        array(j) = temp  
      }  
    }  
  
    // Swap the pivot element with the greater  
element specified by i  
    val temp = array(i + 1)  
    array(i + 1) = array(high)
```

To translate the Quicksort implementation in Python to C++, you can follow the same logic and structure of the original code. Here is an implementation of Quicksort in C++:

```
#include <iostream>
#include <vector>

using namespace std;

// Function to find the partition position
int partition(vector<int>& array, int low, int high) {
    // choose the rightmost element as pivot
    int pivot = array[high];

    // pointer for greater element
    int i = low - 1;

    // traverse through all elements
    // compare each element with pivot
    for (int j = low; j < high; j++) {
        if (array[j] <= pivot) {
            // If element smaller than pivot is found
            // swap it with the greater element pointed by i
            i++;
        }
    }
}
```

The Visual Studio Code plugin also have the ability to translate code.

Translate SCOPE Script prompt (using GPT)

Scope is a language using SQL syntax with C# types, variables and expressions support.

I want you to help me translate Scope code block to scala code.

Here're some rules:

Translate code block into a scala function 'scope' with 'Unit' as return type.

For '#IF' and '#ELSE', use scala's if/else.

For '#DECLARE', declare a scala variable using 'var'.

For '#SET', update the scala variable.

For '#WARNING', use 'logger.warn'.

For '#RESOURCE', there's a 'download(path:String)' method for you to download it, just invoke this method.

For '#REFERENCE', just ignore it.

All variables are passed in using parameters.

All 'DateTime' should be changed to 'ZonedDateTime' and when parsing string to 'ZonedDateTime', add an additional 'Z' to the end.

For DateTime APIs like 'Add*', change to 'plus*'.

For 'String.format(variable, ...)', change to 'variable.format(...)'.

For C#-style date time formatting placeholders like '{{0:yyyyMMdd}}', change it to java's format placeholder.

Variables should be changed to camel case.

Just translate the code in the \001SCOPE:\001 block. Do not add code. Translate it as-is.

Here's the scope code:

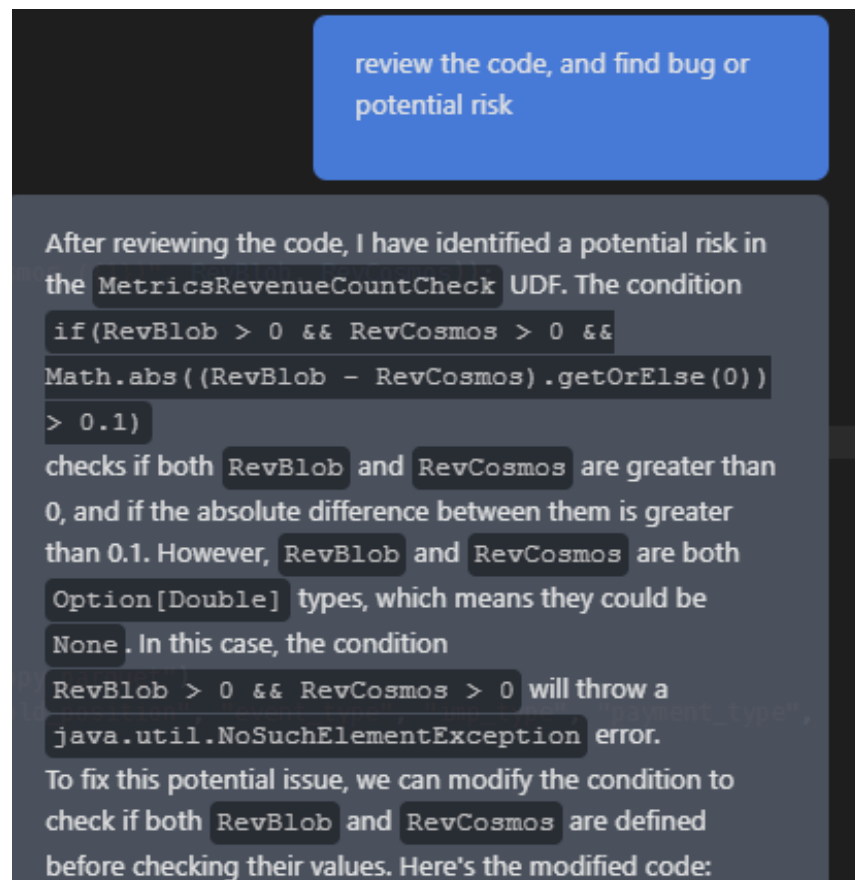
```
\001SCOPE:\001
```

```
{content}
```

```
\001Scala:\001
```

Review code

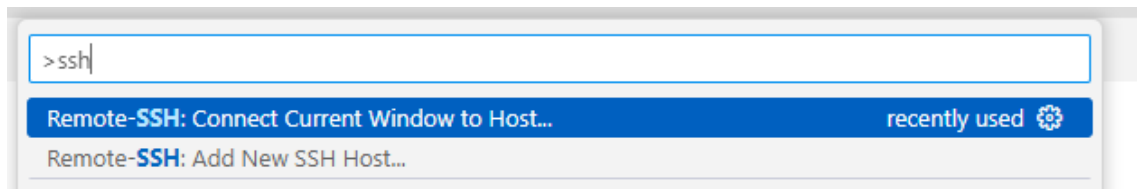
It can find the potential risk:



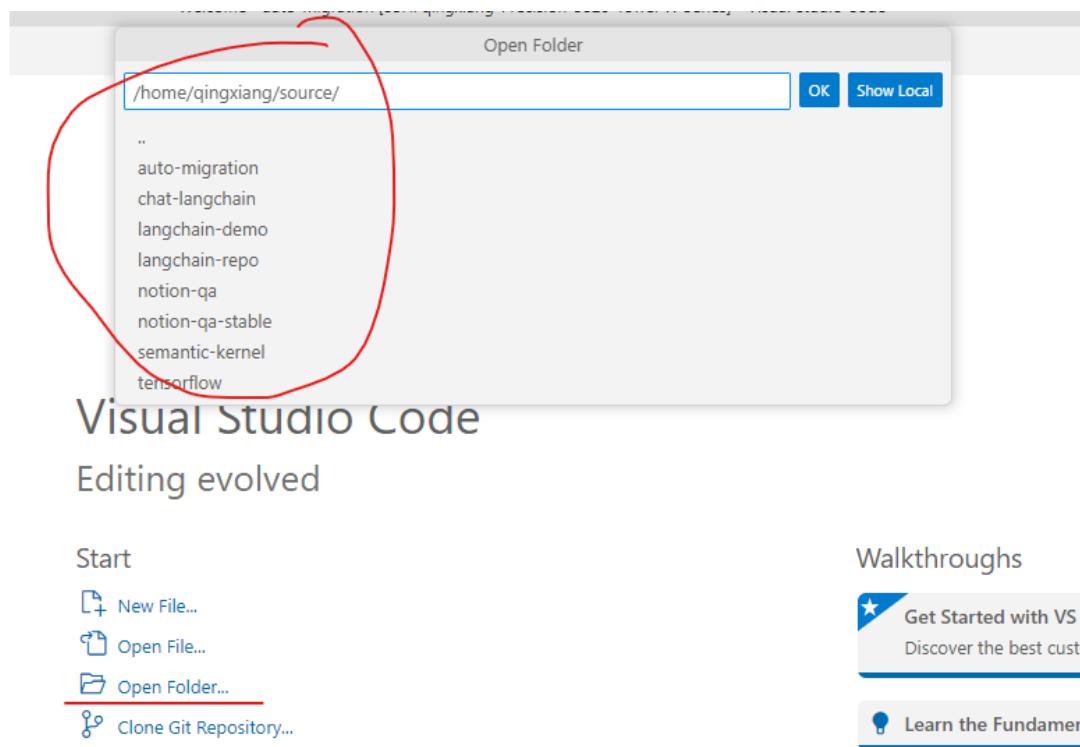
How to edit Jupiter notebook? (@Data Scientist)

Method One: Install SSH Extension

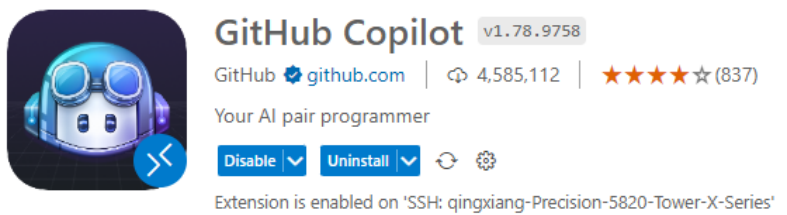
1. Install Visual Studio Code SSH Extension: [Developing on Remote Machines using SSH and Visual Studio Code](#)
2. Press F1, input "SSH:Open" select Remote-SSH: Add new host, add a new ssh host, username@host, input password if prompted. Strongly suggestion use id_rsa.pub ([Setting up SSH Trust Between Two Servers \[ClearOS Documentation\]](#))
3. Restart Visual Studio, Press F1, input SSH, choose "connect current window to Host"



4. Open the remote folder



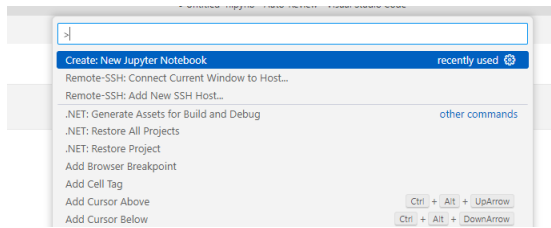
5. You need also enable it again for remote SSH host



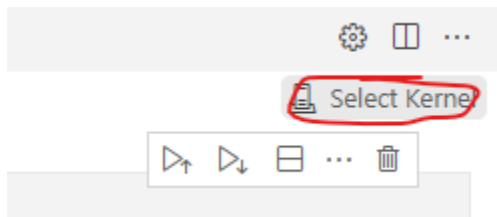
Method Two: Connect remote kernel, just use kernel

Working with Jupyter Notebooks in Visual Studio Code

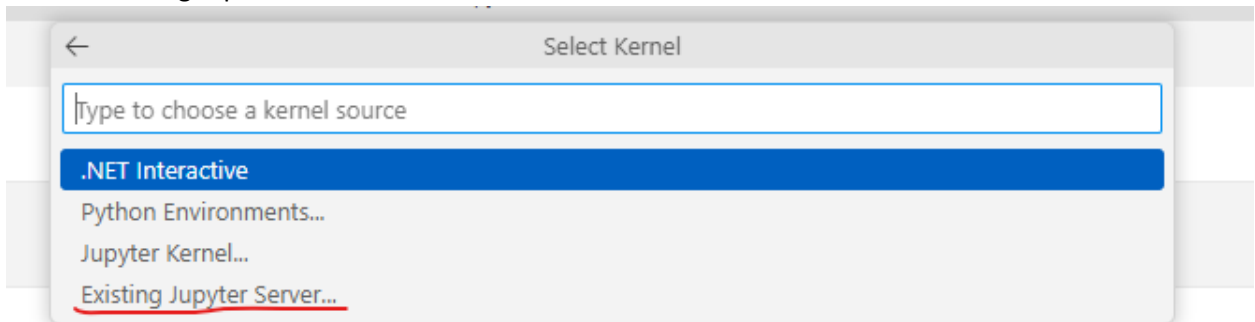
1. F1, then type “Create: New Jupyter Notebook”



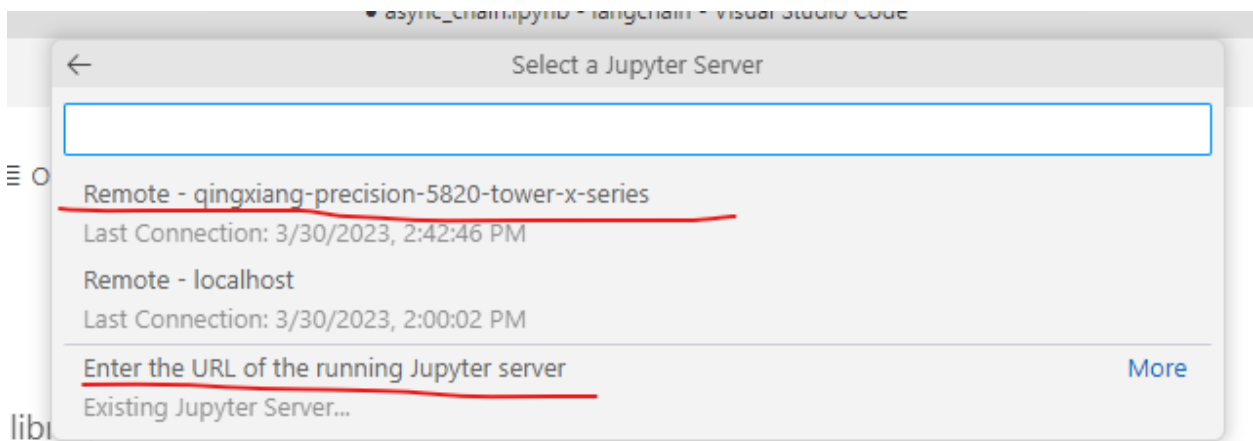
2. Create left top, “Select Kernel”



3. Choose Existing Jupyter Kernel:



4. First time, choose “Enter the URL of the running Jupyter server”
After then, choose the kernel



Copilot short cut

Visual Studio Code

Action	Shortcut	Command name
Accept an inline suggestion	Tab	editor.action.inlineSuggest.commit
Dismiss an inline suggestion	Esc	editor.action.inlineSuggest.hide
Show next inline suggestion	Alt+]	editor.action.inlineSuggest.showNext
Show previous inline suggestion	Alt+[editor.action.inlineSuggest.showPrevious
Trigger inline suggestion	Alt+\	editor.action.inlineSuggest.trigger
Open GitHub Copilot (additional suggestions in separate pane)	Ctrl+Enter	github.copilot.generate
Toggle Git Hub Copilot on/off	<i>No default shortcut</i>	github.copilot.toggleCopilot

JetBrains IDEA

Action	Shortcut
Accept an inline suggestion	Tab
Dismiss an inline suggestion	Esc
Show next inline suggestion	Alt+]
Show previous inline suggestion	Alt+[
Trigger inline suggestion	Alt+\
Open GitHub Copilot (additional suggestions in separate pane)	Alt+Enter

How to install Copilot Chat (Copilot X) tech preview?

Application link:

[GitHub · Where software is built](#)

Installation after you get approved:

- **You must use [VS Code Insiders](#).**
- Open the Extensions view (Ctrl+Shift+X), search for **GitHub Copilot Nightly** and install the extension.
- When prompted, authenticate with your GitHub ID.
- To open the in-editor Chat, optionally select a block of code and press Cmd+I on macOS or Ctrl+I on Windows/Linux. Ask Copilot to write a Quick Sort function.
- A "Chat" icon will appear in the Activity Bar, click on it to open the Chat view. Go ahead, ask Copilot to "write a program to calculate the airspeed velocity of an unladen swallow".

Reference: [Visual Studio Code and GitHub Copilot AI](#)

How to use my own library/framework instead of the public library?

TO BE DONE