



TRAINING

# Hands-on AI-Assisted Development

## IT track Day 1

---

Build and fix with Copilot, add features to the handout app, and write an instructions file



## Do not redistribute these materials

---

**Redistribution of these materials is prohibited.**

These materials are copyrighted by Givery, Inc. Use them only for your own learning and work reference. Do not share them outside your company, on social media or with third parties.

During the course, do not take screenshots or recordings that show these materials.

# Today's goals

---

## Today you learn to **check AI output before you accept it**

### **Copilot responds in VS Code**

In VS Code on the loaner PC, ask in English and get an answer in English. Everyone reaches this point.

### **Choose between Ask, Agent and Plan**

Pick one of three: just ask, let it edit, or get a plan first. Read the diff before you click Keep.

### **Add features to an existing app**

Have Copilot read the handout app, then add filtering and totals. If it breaks, undo and ask again.

Finish by adding an instructions file. Next session, you test the features you add today.

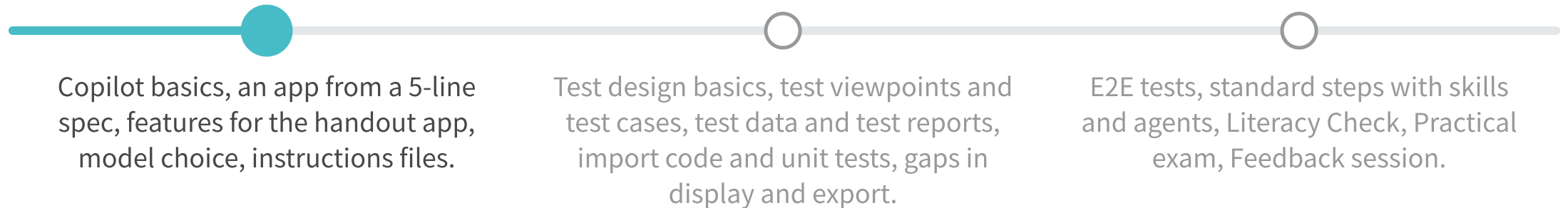
## Where today fits in the 3 days

# Build first, then **test**, then automate and get certified

### Day 1 Build and fix

### Day 2 Build tests with AI

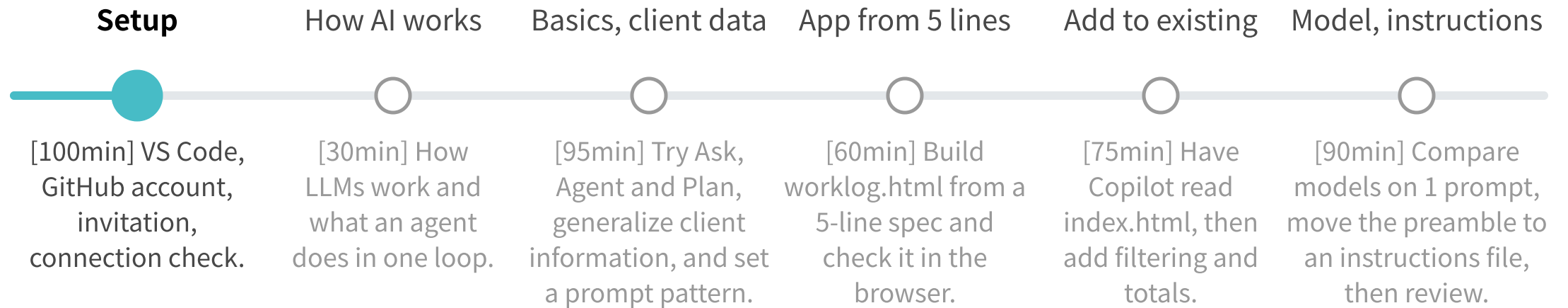
### Day 3 Automation and certification



Each day is 8 hours, about 2 weeks apart. The days are called Session 1, Session 2 and so on.

# Today's schedule

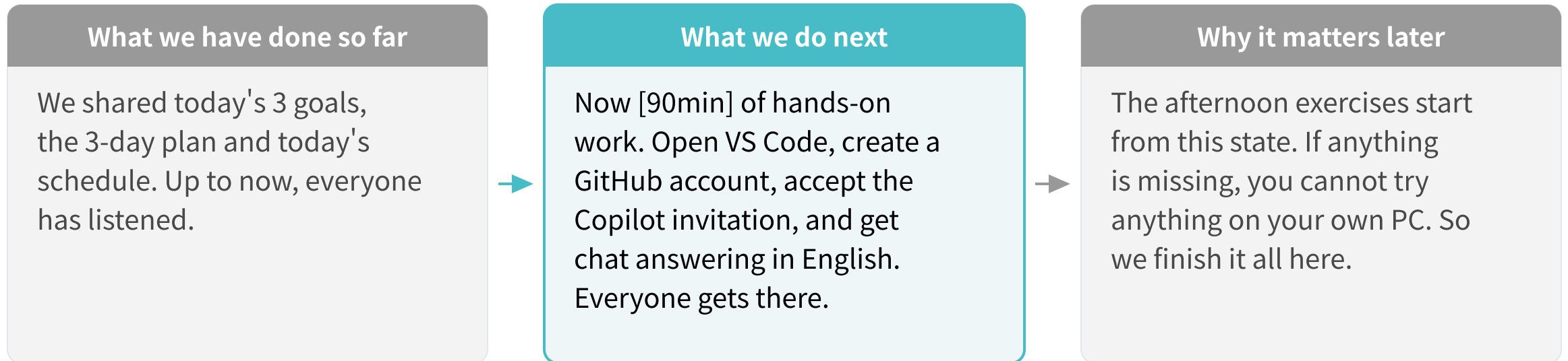
This afternoon, new and existing apps use **the same steps**



Lunch break comes after How AI works. There is a [10min] break between blocks.

## From talk to hands-on work

### Trouble here blocks the afternoon lab, so we move on together





# 01

## Environment setup

---

Get VS Code and GitHub Copilot working on this PC

## Today's tools

---

### Write code in VS Code; run it **only in the browser**

#### Write

VS Code. Version 1.116 and later include Copilot Chat, so you do not install any extensions.

#### Run

Browser. Open the HTML file directly to check it. No server is used.

#### Load data

To load data, drag and drop the file onto the window. Pages opened from file:// cannot load files automatically with fetch.

#### Not used

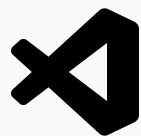
Python, Node.js, the command line. No admin rights needed.

Even if you usually use Visual Studio or IntelliJ, today everyone uses VS Code. The ideas carry over to your IDE. Source: [https://code.visualstudio.com/updates/v1\\_116](https://code.visualstudio.com/updates/v1_116)

## The 3 products you use today

---

**Write and run in separate places; use Copilot in VS Code**



### VSCode

Where you write code.  
Version 1.116 and later  
include Copilot Chat.



### GitHub Copilot

Provides suggestions, chat  
and agents. Runs inside  
VS Code.



### Microsoft Edge

Runs your app. Open HTML  
with a double-click.

Source: Visual Studio Code 1.116 release notes [https://code.visualstudio.com/updates/v1\\_116](https://code.visualstudio.com/updates/v1_116)

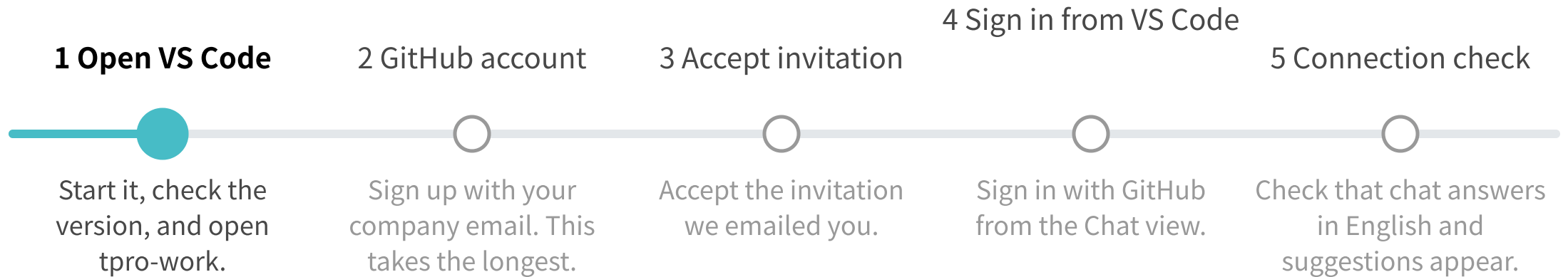
## Contents of the handout ZIP

Extract the handout ZIP to your desktop as **tpro-work**

| Contents                       | What it is for                                                                                     | When you use it |
|--------------------------------|----------------------------------------------------------------------------------------------------|-----------------|
| <b>worklog-app/ index.html</b> | An existing app that records 4 columns: date, category, description and hours. Used on all 3 days. | All day         |
| <b>worklog-app/ README.txt</b> | A list of what the app can and cannot do yet.                                                      | Afternoon       |
| <b>spec5.txt</b>               | The 5-line spec you paste into the AI this afternoon. Copy it from here; do not type it.           | Afternoon       |
| <b>worklog-complete.html</b>   | The finished version of this afternoon's app. Use it if your generated app does not work.          | Only if needed  |
| <b>lib folder (3 files)</b>    | xlsx.full.min.js, papaparse.min.js, chart.umd.js. About 1.1 MB in total.                           | Day 2 and later |

## 5 setup steps

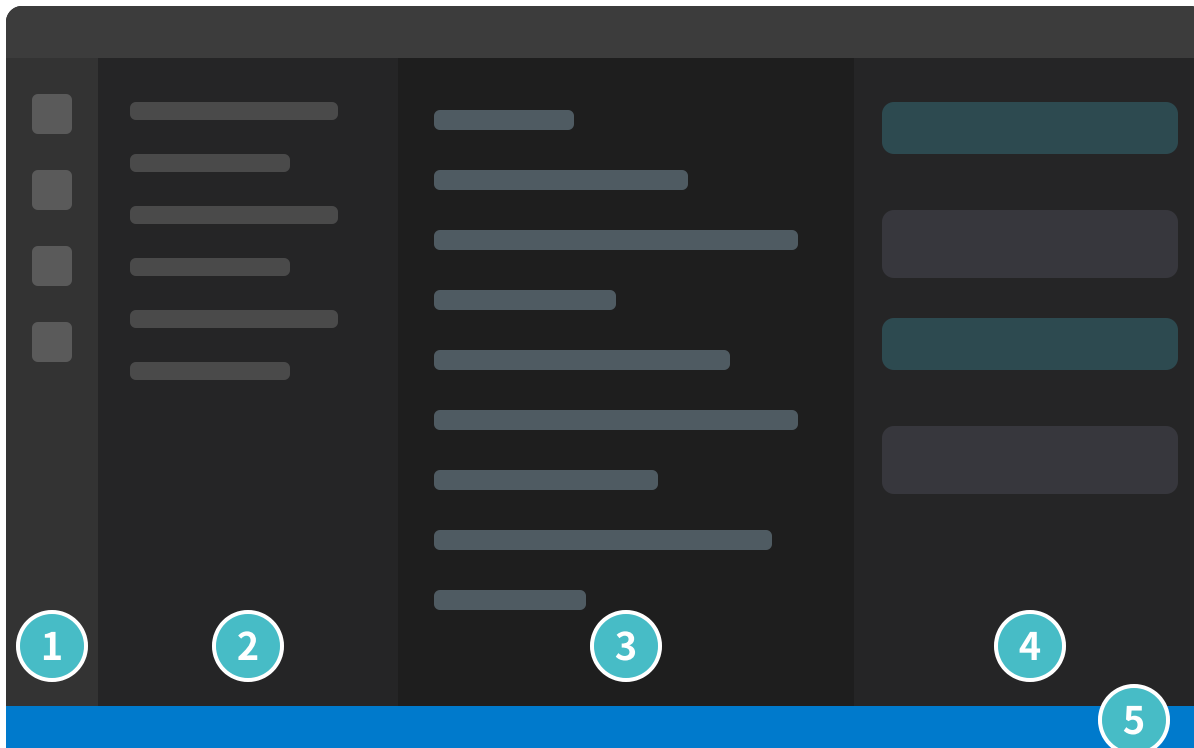
We finish each of the 5 steps together before moving on



If you get lost, raise your hand. We finish each step together before moving on.

# Parts of the VS Code window

## You only need these 5 parts of the window today



### 1 Activity Bar

The column of icons on the far left. Today you only use Explorer at the top and the Chat icon.

### 2 Explorer

Contents of the open folder. The files in tpro-work are listed here.

### 3 Editor

Center. Where you write code. Inline suggestions appear here in light gray text.

### 4 Chat view

Right side. You use it most today. You also sign in from here.

### 5 Status Bar

Bottom row. Shows the language of the open file. Check here if no suggestions appear.

### Keep all your work in one place: **tpro-work**

#### Scenario

On your first day with a client's PC, you first decide on one place to work. Today, keep every file you create in tpro-work on the desktop. If files are scattered, you cannot find your work in the afternoon exercises.

#### Steps

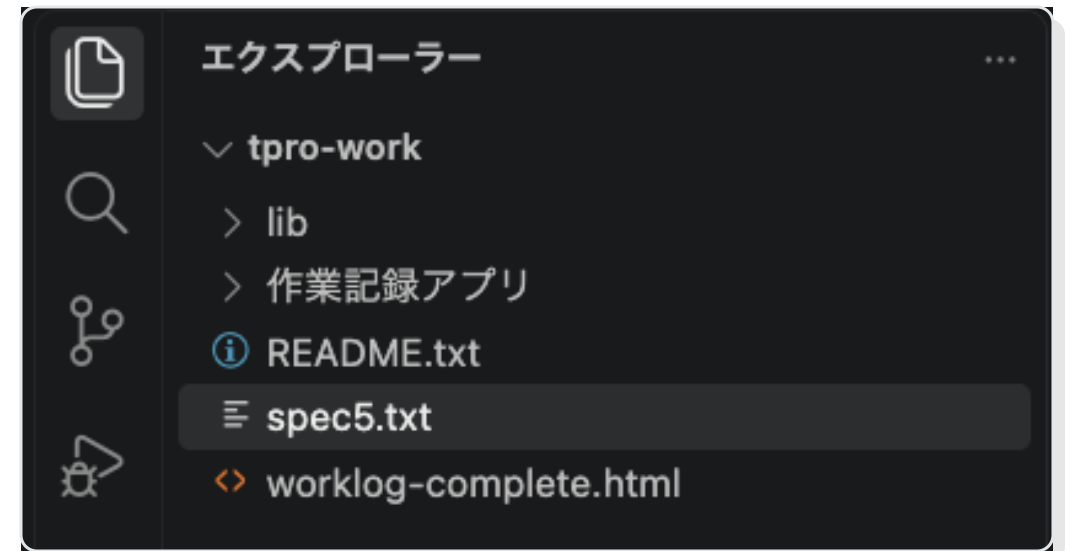
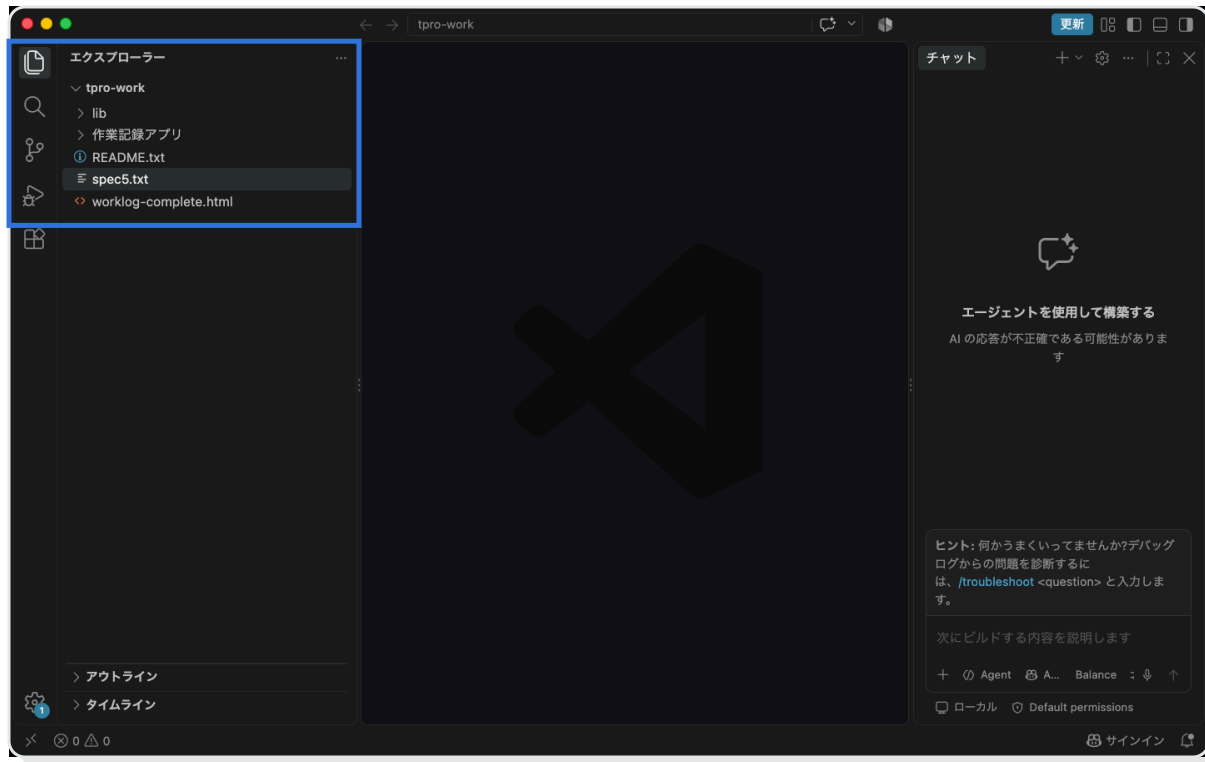
- 1 Start Visual Studio Code from the Start menu. If you cannot find it, raise your hand.
- 2 Open Help > About and check that the version is 1.116 or later.
- 3 Extract the handout ZIP to the desktop and check the folder name is tpro-work.
- 4 Choose tpro-work from File > Open Folder. Check that worklog-app and lib appear in Explorer on the left.

#### Result

No deliverable. You are done when Explorer in VS Code shows the tpro-work files.

## The screen after opening the folder

The Explorer on the left lists **the contents of tpro-work**



If tpro-work is at the top and the worklog-app and lib folders sit under it, the folder is open correctly.

Captured on a Mac from the Japanese edition. The Explorer shows the same layout on Windows.

## Completion criteria for Step 1

### Move on when you see all 3 on your own screen

#### Done when

- tpro-work appears in the title bar or at the top of Explorer.
- Explorer shows 2 folders: worklog-app and lib.
- The About dialog shows version 1.116 or later.

#### If you're stuck

- Your version is older than 1.116. The Chat view does not appear, so install GitHub Copilot Chat from Extensions. No admin rights needed.
- You cannot write to the desktop. Create tpro-work directly in your Documents folder, and read "desktop" as Documents on later pages.
- You opened a single file, not the folder. The AI cannot read the files around it. Use File > Open Folder again.
- The extracted folder is nested twice. If tpro-work contains another tpro-work, move its contents up one level.

## Completion criteria for Step 1 (continued)

---

### If time allows

- From the View menu, open Appearance and look for where to switch the display language. Keep the display language in English for today.
- Open README.txt in worklog-app and read the list of what the app cannot do yet. You add these features this afternoon.

### Create the account with your **company email address**

#### Scenario

Copilot seats go to individual GitHub accounts. The invitation goes to your company email, so you need an account with that address to get a seat. This takes the longest this morning.

#### Steps

- 1 Open github.com in the browser. If you are signed in with a personal account, sign out first.
- 2 Click Sign up and enter the email address your company gave you. The invitation does not go to personal addresses.
- 3 A password of 15 or more characters passes the rules without trouble. Follow the requirements shown on screen.
- 4 Usernames must be unique and alphanumeric. Quickly pick your last name plus a number.
- 5 Solve the puzzle on screen, then enter the verification code sent to your email.

#### Result

No deliverable. You are done when your username shows at the top right after sign-in.

## Completion criteria for Step 2

### If your username is on screen, you are ready to get a seat

#### Done when

- Clicking the icon at the top right of github.com shows your username.
- Settings > Emails lists the email address your company gave you.

#### If you're stuck

- The verification code does not arrive. Check your spam folder, then your company mail filter. If it does not arrive in 5 minutes, the instructor notes it and we move on.
- The username is taken and shows red text. Add 2 digits and it passes. Do not spend time here; decide quickly.
- The puzzle is slow. The network is just busy. Wait and do not close it.
- Still signed in with your personal account. Sign out and create the account again.

#### If time allows

- In Settings, open Copilot in the left menu and check that nothing is assigned yet. This changes in the next step.

## Accept the invitation and sign in

### Use **the same account** to accept and to sign in

#### Scenario

Getting a seat takes 2 steps: accept the invitation, then sign in to VS Code. If you mix accounts, Copilot does not work even after accepting. This is the most common problem.

#### Steps

- 1 Open the invitation email we sent in advance, with GitHub Copilot in the subject. If you cannot find it, tell the instructor.
- 2 First, sign in to the account you just created in the browser. Otherwise, the seat goes to the wrong account.
- 3 Open it from the button in the email and accept the invitation.

## Accept the invitation and sign in (continued)

### Steps

- 1 Go back to VS Code and open Chat from the chat icon in the Activity Bar on the left, or from the View menu.
- 2 Click Sign in in the Chat view to open the browser. Confirm out loud that it is the account you used to accept, then authorize.
- 3 In the "Open Visual Studio Code?" dialog, click Open. You are done when you can type in the VS Code chat box.

### Result

No deliverable. You are done when you can type in the Chat view input box.

## Completion criteria for Steps 3 and 4

### If you can type in the input box, Copilot is ready

#### Done when

- GitHub Settings > Copilot shows your seat. Menu names may differ on the day.
- The Sign in button is gone from the VS Code Chat view, and you can type in the input box.
- The Accounts icon at the bottom left of VS Code shows your company email account.

#### If you're stuck

- The accept link says there is no invitation. This is almost always the wrong account. Sign out and sign in again.
- Your company email rewrites the link and it does not open. Copy the URL and paste it into the browser address bar.
- Stuck at the browser dialog asking to return to VS Code. It is OK to click Open.
- The Chat view does not appear. Open it from the View menu. If it still does not appear, check the version. If VS Code is older than 1.116, install the extension.
- Sign-in fails because of your company proxy. This cannot be fixed on the spot. The instructor records your PC and the problem. Today, follow along on the instructor's screen.

## Completion criteria for Steps 3 and 4 (continued)

---

### If time allows

- In Settings > Copilot, find the setting that blocks suggestions matching public code. We cover it this afternoon.

### The afternoon exercises assume **all 3 checks pass**

#### Scenario

If one fails, you cannot try anything on your PC this afternoon. You can judge each check yourself, without the instructor. When all 3 pass, go to the task on the next page.

#### Steps

- 1 In Explorer, open index.html in worklog-app.
- 2 In the Chat view, type "What does the app in this file do? Answer in English." and send it.
- 3 Create and save a new test.html in tpro-work. Type html in the editor and wait a few seconds.
- 4 Click the model name below the chat input box and just check that a list opens. We switch models this afternoon.

#### Result

You get one test.html in tpro-work on the desktop. It can be empty.

# Completion criteria for the connection check

## Ignore accuracy. **Just check that a reply comes back**

### Done when

- The chat answers in English. Whether it matches the actual code does not matter here.
- When you type `html` in `test.html`, a suggestion appears in light gray text.
- When you click the model name, a list of several names opens.

### If you're stuck

- The reply is not in English. Add "Answer in English." at the end to fix it. This is a live example of how prompts work, so we share it with everyone.
- No suggestions appear. Usually the file is named just `test`. Save it again with `.html` added, and check the Status Bar.
- No reply comes back. Often the network is just busy, so wait 30 seconds and send again. If it still does not come, tell the instructor.
- The reply has nothing to do with `index.html`. Check that the file was open when you sent it. If it is closed, its content is not sent.

## Completion criteria for the connection check (continued)

---

### If time allows

- When all 3 pass, go to tasks A to D on the next page. No need to wait.
- Send the same question again and see that the reply differs from the first one. The next chapter explains why.

## Check it yourself. The answer appears on your screen

| Task     | Steps                                                                                        | What you learn                                                                                                                                             |
|----------|----------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>A</b> | Send the same question 3 times in a row and compare the replies                              | Each reply differs. An LLM gives different answers to the same input by design; the afternoon lecture explains why. If all 3 match, try a longer question. |
| <b>B</b> | Compare the result of "Make it nice" with "Add a total hours row at the bottom of the table" | Vague requests miss the point. Naming exactly what you want works. This afternoon's prompt pattern splits this into 4 elements.                            |

## If you finish early (continued)

Try this afternoon's topics **yourself, ahead of time**

| Task | Steps                                                                                                                              | What you learn                                                                                 |
|------|------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|
| C    | In Notepad, create sample-list.txt with 3 made-up names and save it. With it open in VS Code, ask "What files are in this folder?" | The open file's content appears in the answer. You see that just opening a file sends it.      |
| D    | Have it write code for the Worklog app total. Open it in the browser and check Console.                                            | If it uses a method that does not exist, it stops only there. Reading alone will not catch it. |

Delete the file you made in task C once checked. Never use real personal information.

## From setup to how it works

# Time to stop typing. First learn what it cannot do

### What we have done so far

Everyone has VS Code chat answering in English and suggestions appearing. You have a working setup.



### What we do next

The next [80min] is for listening. We cover what an LLM looks at to answer and what it means for AI to act on its own.



### Why it matters later

This afternoon you apply this on the same screen. Without knowing how it works, you cannot verify its output.



# 02

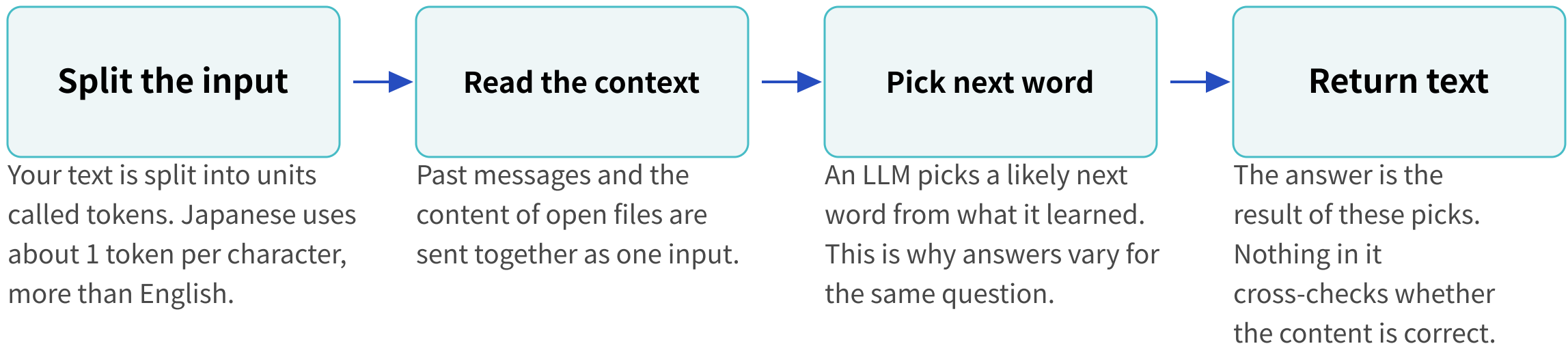
## **The world of AI-driven development**

---

How LLMs work, agents, and the design ideas behind Copilot

# How an LLM produces an answer

## LLMs split text and pick the next word **by probability**



Different answers each time are not a bug. The LLM picks each next word by probability.

# What is sent and what is not

## Information outside the context **is not used**

### Not sent

Company rules, client specs, run results. In Ask, unsent information is not used, and you get a plausible guess. Agent may find and read files itself.

### Sent

Open files, the selection, the conversation so far, and files you attach. Only what is here is used to decide.

# Inline suggestions, chat and agents

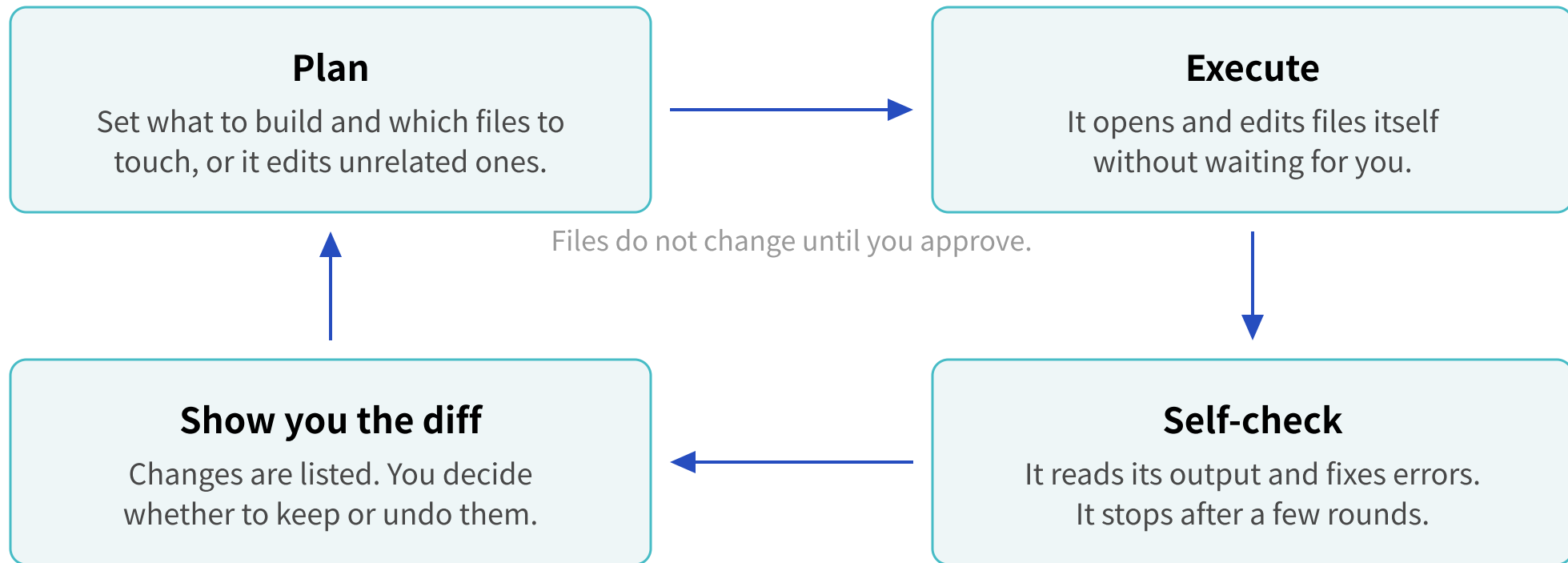
All 3 are the same AI, but **the size of the task** differs

| Feature            | What it handles                       | What you do                               | When to use it                                         |
|--------------------|---------------------------------------|-------------------------------------------|--------------------------------------------------------|
| <b>Suggestions</b> | A few lines after the cursor.         | Accept or reject at once.                 | When you know how to start and want to keep typing.    |
| <b>Chat</b>        | Questions on a selection or one file. | Choose what to send and verify the reply. | When asking how to build, or what existing code means. |
| <b>Agent</b>       | Edits across files, with retries.     | Set the scope first, then read the diff.  | Changes across files, or building one from scratch.    |

In chat, you open files and send them. An agent opens and reads files itself. We find the switch on screen this afternoon.

## One agent loop

An agent plans, acts and **corrects itself** on its own



Step in at 2 points: when it picks the files and when the diff appears. Otherwise, let it run.

## Decisions you keep

---

Faster writing makes **reading and cutting** matter more

**AI proposes the options**  
**You decide what to adopt**

AI cannot see whether it runs in the client's environment. Testing and deciding to accept it stay with you, today and after.

## Morning takeaways

---

### Check **what you now have** and what you can now explain

#### **A working setup**

VS Code chat answers in English, and gray suggestions appear in the editor. Everything later today depends on this.

#### **Choosing among the 3**

You can explain suggestions, chat and agents by the size of task they take, not by feature name.

#### **Responsibility for approval**

Lines you approve from an agent's diff count as lines you wrote. This stays with you.

If your connection check is unfinished, we help you during the afternoon exercises. Do not stay stuck.

## From morning to afternoon

# Knowing it by reading is not the same as doing it

### What we have done so far

This morning ended with 3 points: a working setup, choosing among the 3, and approval responsibility. After lunch you practice all 3. The morning's hands-on work was only the setup check.



### What we do next

In this chapter you try all 3 Copilot entry points. Have it read the handout Worklog app, run one slash command, and learn where to switch Ask, Agent and Plan.



### Why it matters later

Later this afternoon you write a spec and generate a web app. It uses only the operations in this chapter. Learn the mode switch and file approval here, and you will not get stuck later.

# 03

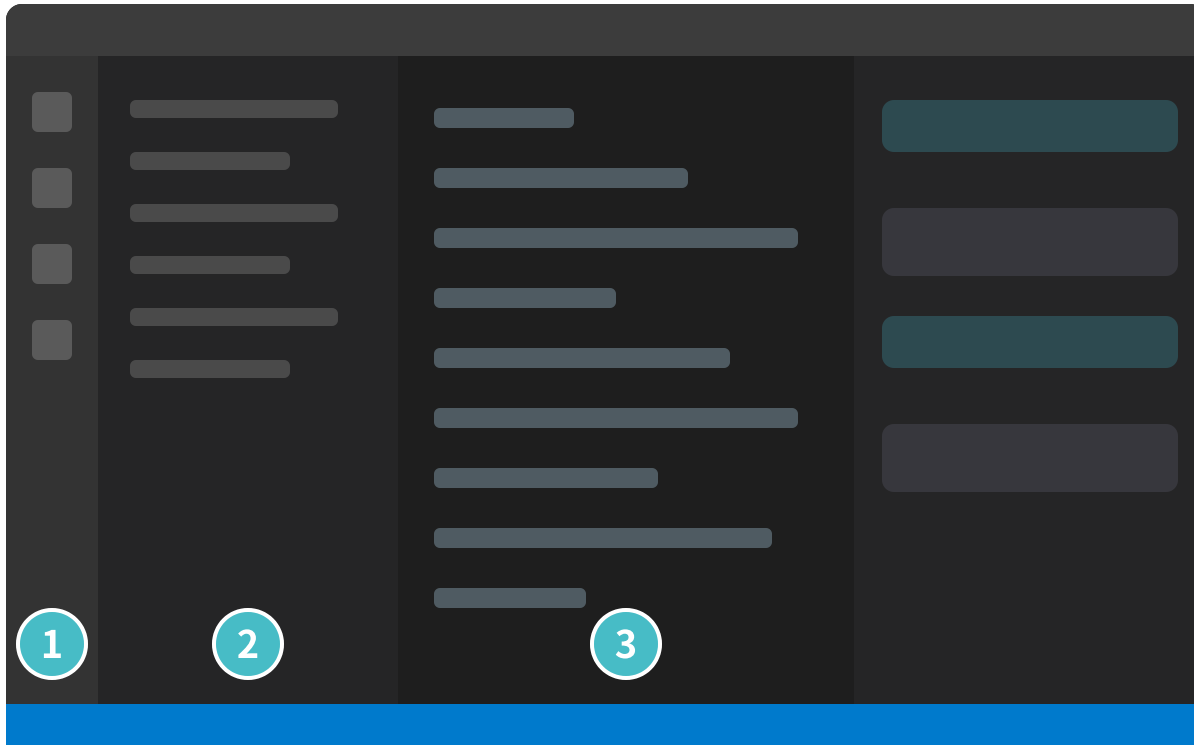
## Using Copilot

---

Get hands-on with chat, slash commands, Ask, Agent and Plan

## 3 places where Copilot appears

### Copilot appears in 3 of the 5 areas named this morning



#### 1 Activity Bar

The column on the far left. Today you click only 2 icons: Explorer and Chat.

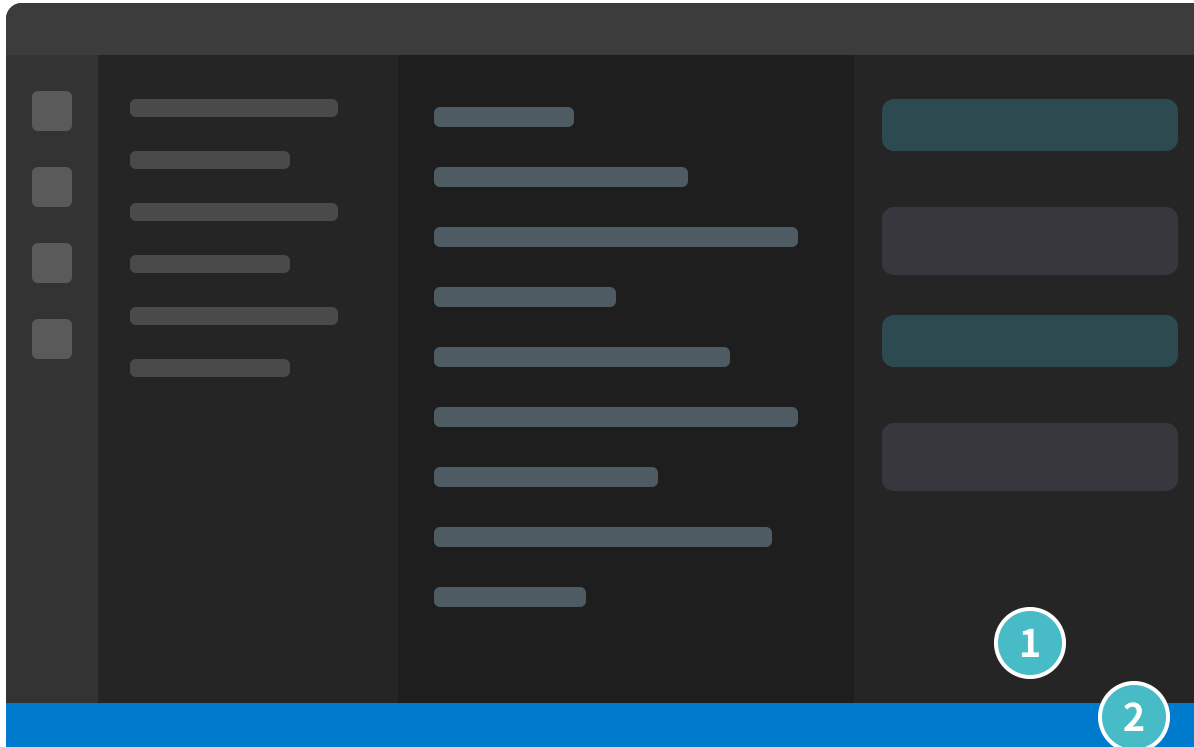
#### 2 Explorer

The contents of tpro-work, opened this morning. If worklog-app and spec5.txt are listed, the right folder is open.

#### 3 Editor

Copilot place 1. Gray text that continues the line you are typing is an inline suggestion.

## 3 places where Copilot appears (continued)



### 1 Chat view

Place 2, today's main tool. Ask your questions in English. Switch Ask, Agent and Plan below the input box.

### 2 Status Bar

Place 3. The Copilot icon appears here. If it has a slash or warning mark, check your sign-in or the extension.

### Plain English works. Everyone sends the same line first

Type the 2 lines below into the Chat view input box as they are and send. Enter sends; Shift+Enter adds a new line. Learn this first, or long pasted text may get sent halfway.

```
What files are in the folder that is open now?  
Answer in English.
```

```
If the reply is not in English, send just this line next.  
Answer in English.
```

Do not worry about accuracy here. If the reply is in English, you pass.

## A slash command saves retyping the same prompt every time

Type `/` in the input box to see a list of commands. No need to memorize the names. Here are 4 common ones, exactly as you type them.

```
/explain  Explain what the selected code does
/fix      Find the cause of an error and how to fix it
/tests    Suggest tests for the selection
/clear    Clear the chat history and reset the context
```

How to send it. Select the code first, then send this line.

```
/explain Explain what this function does in 3 lines for a first-time reader.
```

The command list changes between versions. If one is missing, write the same request in plain English for the same result.

# Ask, Agent and Plan

Before asking, choose **whether it may change files**

| Mode  | What you get                      | Files                    | Use it to                                      |
|-------|-----------------------------------|--------------------------|------------------------------------------------|
| Ask   | Text and code snippets            | Unchanged                | Ask about code, have it explain, compare ideas |
| Agent | New and edited files, plus a diff | Not final until you Keep | Build, fix, add                                |
| Plan  | Only a plan and steps             | Unchanged                | Check the plan before a big change             |

Keep accepts an Agent diff; Undo reverts it. Keep means you claim the code as yours.

# Approve a diff without reading, and **it becomes your code**

1

### The diff appears

Planned changes are listed as added lines and removed lines. The file has not changed yet.

2

### Read the diff

See which file changes and how. If the file is unexpected, stop and fix the prompt.

3

### Approve to apply

Click the approve button to write the changes to the file. The button label varies by version, so check it on screen on the day.

4

### Revert

Undo the last change with Ctrl+Z, or use Undo in the chat. We do not use Git today, so these are your only 2 ways back.

## Exercise 1: Tour of the entry points

Have it read the handout app, then **try all 3 entry points**

### Scenario

At a client, you usually start by reading the app you inherit. Fixing code you do not understand causes trouble. Have Copilot read the handout Worklog app so you can explain it in your own words, and learn where to switch to Agent.

### Steps

- 1 In tpro-work, open worklog-app, then open index.html in the editor.
- 2 Select the entire render function inside the script tag.
- 3 In the chat input box, type `"/explain Explain what this function does in 3 lines for a first-time reader."` and press Enter.

## Exercise 1: Tour of the entry points (continued)

---

### Steps

- 1 Read the explanation. Create notes.txt in the tpro-work folder and write one line on a point that differs from your understanding or seems missing.
- 2 Open the Ask / Agent / Plan dropdown below the chat input, choose Agent, see it change to Agent, then switch back to Ask. Do not send anything.

### Result

notes.txt is in the tpro-work folder on the desktop with at least 1 line. No other files are created. index.html is not changed.

# Completion criteria for Exercise 1

If all 3 pass, **you know every step needed later today**

## Done when

- The /explain reply is in English and about the render function. It says, in effect, that the function rebuilds and re-sorts the table.
- notes.txt in the tpro-work folder has at least 1 line on a point that differed from your understanding or seemed missing.
- The dropdown below the chat input changed to Agent, and you switched it back to Ask.

## If you're stuck

- /explain is not in the command list. The list varies by version. Keep the selection and type "Explain the selected part in 3 lines." for the same result.
- You got an explanation of the whole file. The selection was lost. Select again, send /clear once, then send the prompt again.
- You cannot find the dropdown. Look above and below the input box; its position varies by version. If you still cannot find it, raise your hand.
- The reply is not in English. Next, send "Answer in English." The context carries over.
- You changed index.html by mistake. Undo it with Ctrl+Z. You use this file in a later exercise.

## Completion criteria for Exercise 1 (continued)

---

### If time allows

- Send /explain again for the same render function and confirm the wording differs from the first time. Check whether the meaning stays the same.
- Send /tests with the same selection. Just check whether the test ideas include "no records entered". Do not implement them.
- In index.html's script, type // returns the total hours, press Enter, and watch the gray inline suggestion. Press Esc to discard; leave the file unchanged.

# From operations to writing prompts

## Set the rules first to keep client names out of practice

### What we have done so far

In the last chapter you tried chat, slash commands, Ask, Agent and Plan. You also saw on screen that a diff is not written to the file until you approve it. That covers all the operations.



### What we do next

Next: writing and rules. First decide how to generalize client information, then build the prompt pattern. In reverse order, client names get into practice prompts.



### Why it matters later

In the next block, paste this pattern as is to build a web app from a spec. Use the same pattern to ask for help when it breaks. You take home the 1 file you make in this chapter.

# 04

## **Prompt design and governance**

---

Decide how to generalize client information, and take home 1 prompt pattern you can use as is

# Generalizing client information

## Generalize client information, then **check it yourself**

| Rule                                             | Examples                                                                                                                             |
|--------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <b>Generalize names to industry and phase</b>    | "Company A's order management system" becomes "a core system in distribution". Test: can a search for that word identify the client? |
| <b>No real data, real files or screenshots</b>   | Do not paste production CSVs. Make your own dummy with the same column meanings and give that instead.                               |
| <b>Write goals and issues, not exact figures</b> | "30,000 records in 30 minutes a day" becomes "high daily volume, and processing time is an issue"                                    |
| <b>Check yourself that it is safe to share</b>   | Read it once yourself before you click Send. Do not assume someone will stop you later.                                              |

Today's exercises all use dummy material, so this rule does not come up here. The risk starts tomorrow, back at your own desk.

## When information goes to the AI

It goes **when you open a file and when you ask**

| Your action                       | Sent to the AI?                                                |
|-----------------------------------|----------------------------------------------------------------|
| File names are listed in Explorer | No. The tree on the left only lists names.                     |
| Open a file in the editor         | Yes. It is added to the chat even if you do not specify it.    |
| Select text in a file             | Yes. Selected text counts more than a file that is only open.  |
| Ask "What is this project?"       | Yes. Related files are picked up from the index automatically. |

Source <https://code.visualstudio.com/docs/copilot/reference/workspace-context> (checked Sep 9, 2026)

## When exclusion settings do not work

### Even if excluded, **opening a file bypasses it**

#### **Even in gitignore, a file is read once opened**

The official docs say gitignore is bypassed for an ignored file if it is open or text in it is selected. Listing a file in an exclusion does not make it safe.

#### **Search matches are sent even if the file is not opened**

Even if the AI does not open the file, content that matches a search goes into the chat. Do not assume that what is not on screen is not sent.

#### **The 3 exclusion settings cover different scopes**

gitignore removes files from the index and search. files.exclude also hides them in Explorer. search.exclude affects search only.

Source <https://code.visualstudio.com/docs/copilot/reference/workspace-context> (checked Sep 9, 2026)

### Not opening confidential files in VS Code is the surest way

- 1 Do not open folders with confidential files in VS Code at all**  
Create a separate work folder and open only that. This is the surest way and needs no settings.
- 2 If you cannot, move confidential files to a folder outside the workspace**  
Files on the same PC are not indexed if they are not in the workspace.
- 3 Hide them in Explorer too with files.exclude**  
What you cannot see, you will not open by mistake. Unlike gitignore, this hides the files from view.
- 4 If your organization has a Copilot admin, ask them to set exclusions**  
They can exclude content per repository or per organization, so you do not rely on individual habits.

Back at your desk, first check once what gets indexed when you open your current project.

## The 4 parts of a prompt

Of the 4, **constraints and output format** matter most

### Role

Who answers. For example, "as a JavaScript code reviewer". Polishing this has limited effect.

### Context

The target and where it runs. For example, "a single HTML file opened directly in the browser". This 1 line describes today's setup.

### Constraints

What not to do: "Do not add external libraries." Without this, you get code that does not run in today's setup.

### Output format

How to return it: "Only the changed lines, as a diff." Without this, you get the whole file and cannot see what changed.

No need to memorize the order. Before sending, just ask: did I write the constraints and output format?

### With a pattern, **you know how to start a prompt**

Here are the 4 parts from the previous page as a full prompt you can send. Save these 5 lines as prompt.txt in the tpro-work folder on the desktop. Today you add prompts to this file.

```
You are a JavaScript code reviewer.  
The target is a single HTML file that I open directly in the browser.  
The problem: the hours total field shows the numbers joined together instead of added.  
Constraint: do not add external libraries.  
Output: 3 possible causes first, then only the fixed lines.
```

```
Replace 5 parts: role / target and environment / problem / constraint / output format  
Only the problem line changes each time. The other 4 lines stay mostly the same.
```

You can delete lines you do not need. Adding 1 constraint helps more than polishing the role.

## Exercise 2: Rewrite the pattern

Write it once for your work, **generalized**. Do not send it yet

### Scenario

The biggest risk at your desk is pasting client or table names as is when you are in a hurry. While you are calm, practice writing a generalized version once. You do not send or submit it.

### Steps

- 1 Open prompt.txt in the tpro-work folder on the desktop. You saved it on the previous page. If you did not, create it now and copy the 5 lines on screen.
- 2 Rewrite the 5 lines for the work you are doing now. Generalize client, product, project and partner names to industry and phase terms.
- 3 When done, read it once and check that no word could identify the client in a search
- 4 If any remain, replace them with generalized words and save. Do not send it.

### Result

prompt.txt in the tpro-work folder on the desktop has the 5 lines in your own words. No sending or submitting. Just keep it.

## Completion criteria for Exercise 2

### Constraints and output format filled in, **no proper nouns left**

#### Done when

- Of the 5 lines, the constraint line and the output format line are not empty
- No proper nouns remain, such as company, product, system or person names. You checked that no word identifies the client in a search.
- The problem line states visible facts, not causes: "the total field shows numbers joined together", not "reduce is broken".

#### If you're stuck

- You cannot think of anything to write. Recall 1 question you asked someone in the past week and write that. Do not spend time looking for a topic.
- Generalizing seems to make the question meaningless. A dummy with the same structure is enough. Keep the column meanings and change only the names.
- You cannot think of a constraint. Write today's constraints as is: do not add external libraries, and keep it in a single HTML file.
- You cannot find prompt.txt. If you did not create it on the previous page, create it now. It goes in the tpro-work folder on the desktop.

## Completion criteria for Exercise 2 (continued)

---

### If time allows

- Write the same request again without the constraint line and compare. See which one you would rather send.
- Write the problem line 3 ways: stating a cause, stating only facts, and facts plus steps to reproduce. Confirm the third is strongest.
- Send what you wrote only after you are back at your desk. Before sending, read it again to check it is safe to share outside the company.

# Break

---

## Next, we build an app from 5 lines of English

### **Keep open**

The tpro-work folder and prompt.txt with your prompt pattern

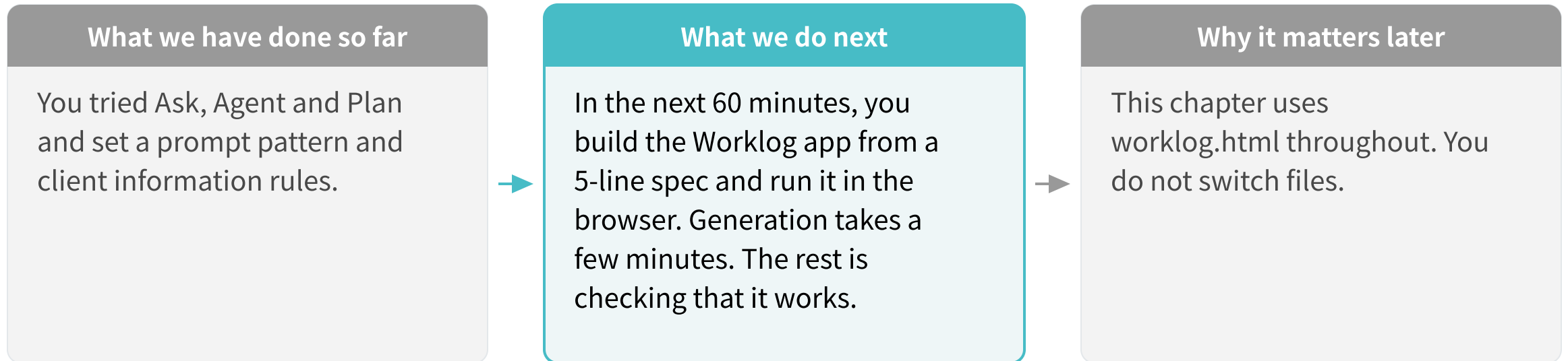
### **After the break**

We start by looking at the actual 5-line spec

Stop your work here

## From practice to development

From here on, **hands-on time** is longer than listening





# 05

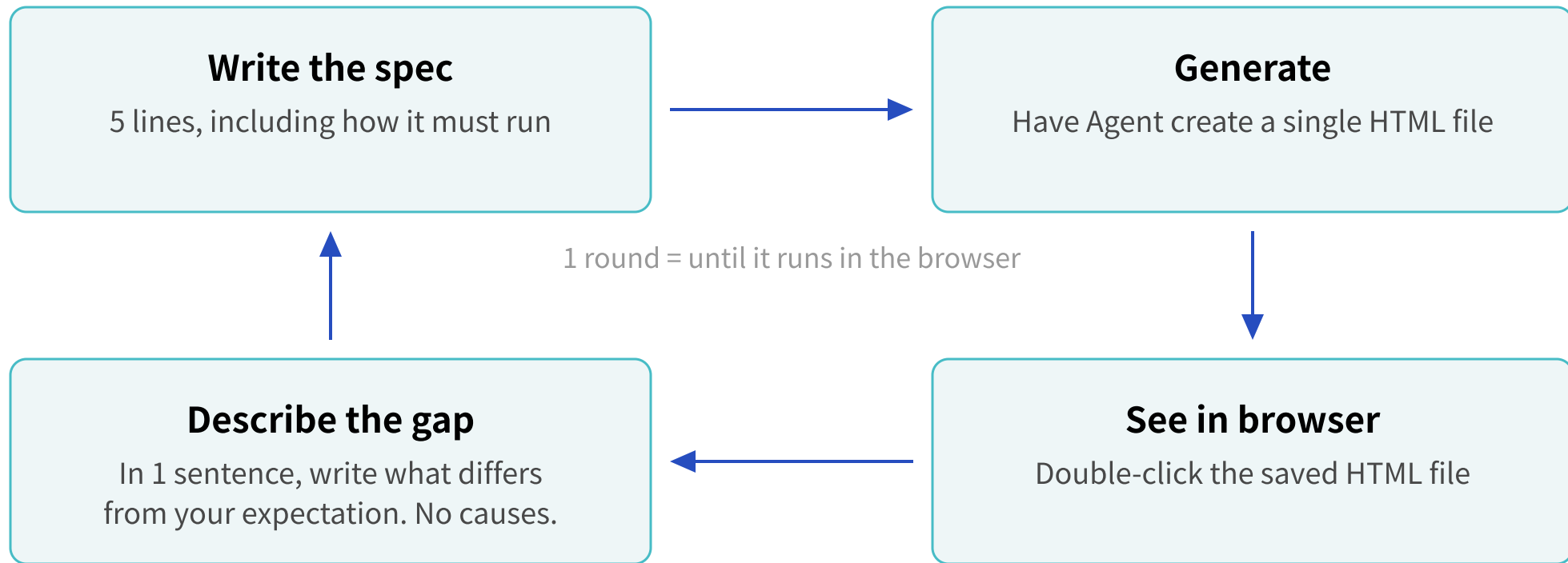
## **A working web app from a 5-line spec**

---

Paste a 5-line spec and get the Worklog app running in the browser

## One round of generating and checking

A round ends in the browser. **Open it before asking more**



## 4 steps to run it

---

# Write in VS Code, run in the browser: **no server needed**

1

### Write in VS Code

Edit the HTML in the editor. Generated changes also appear in the editor.

2

### Save

Press Ctrl+S. When the dot next to the file name on the tab disappears, the file is saved.

3

### Open in File Explorer

Open the tpro-work folder on the desktop in File Explorer. The sure way is to right-click the file in VS Code and select Reveal in File Explorer.

4

### Double-click

Double-click worklog.html to open it in your default browser. The address bar shows a path starting with file:///

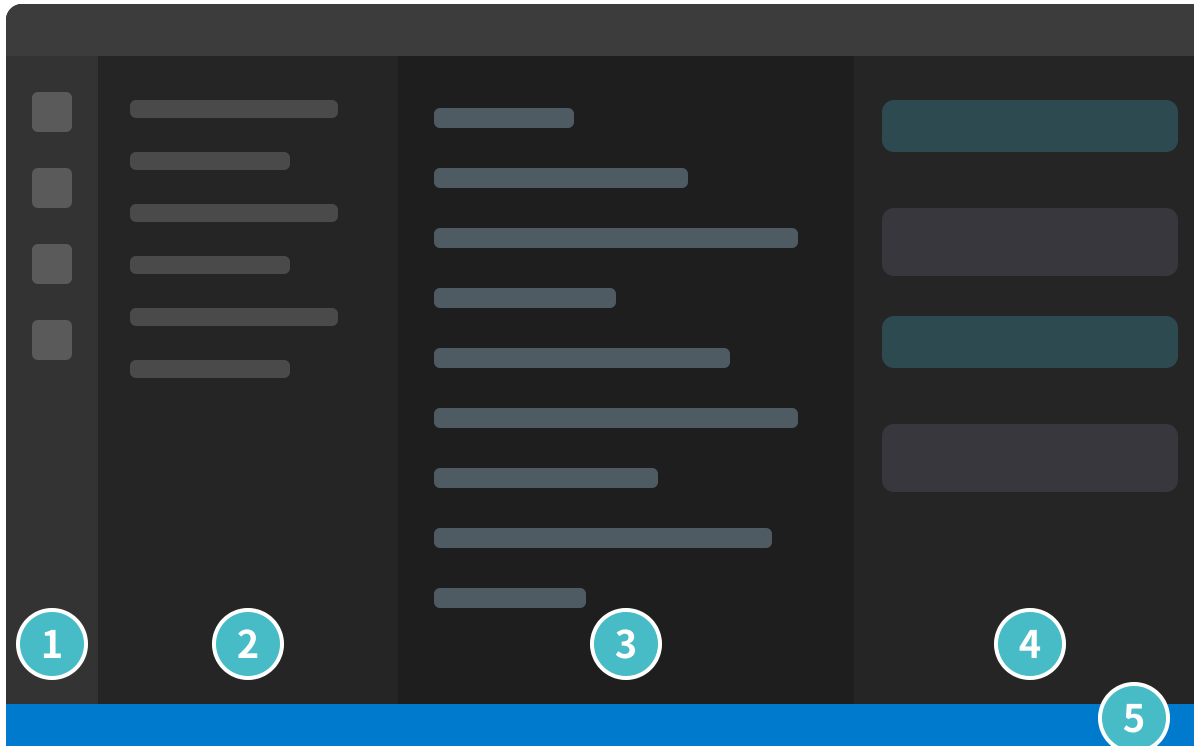
Because it is opened with file:///, JavaScript cannot load external files. On Day 2, you pass CSV and Excel files by dragging and dropping them onto the page.

### Line 5 decides how it runs in the browser

Copy tpro-work/spec5.txt as is and paste it into the chat. Do not edit it.

1. Put one form for entering Date, Owner, Phase, Task and Time (min)
2. The "Add" button adds the input as one row to the list table below
3. Below the list table, show the total time in minutes
4. If any field is empty, do not add the row, and show "Some fields are empty" below the form
5. Put HTML, CSS and JavaScript in one HTML file that runs just by opening it in a browser.  
Name the file worklog.html

### Agent's changes **reach the file only after you approve**



**1 Activity Bar**

The chat icon. Open the Chat view from here.

**2 Explorer**

Contents of tpro-work. When generation works, worklog.html appears here.

**3 Editor**

Changes show as a diff. Added and removed lines are shown in different colors.

**4 Chat view**

Ask / Agent / Plan below the input box. Select Agent today. The Keep button appears above the input box or above the diff.

**5 Status Bar**

The open file's type. If it shows HTML, you saved it correctly.

# 4 checks for completion

Looks differ by person. **It is done when these 4 work**

| Where      | Input                                                           | What should appear                                   |
|------------|-----------------------------------------------------------------|------------------------------------------------------|
| Form       | Fill in Date, Owner, Phase and Task, and enter 30 in Time (min) | You can click the Add button                         |
| List table | Add 30, 15 and 45 minutes in a row                              | The table has 3 rows                                 |
| Total      | With the 3 entries above                                        | It shows 90                                          |
| Message    | Leave one field empty and click Add                             | No row is added, and "Some fields are empty" appears |

## When you add an entry, the list and total change together

### 作業記録

日付・担当・工程・作業内容・作業時間（分）を記録します

|              |    |     |            |         |    |
|--------------|----|-----|------------|---------|----|
| 日付           | 担当 | 工程  | 作業内容       | 作業時間（分） |    |
| 10/10/2026 📅 | 鈴木 | テスト | 例: 一覧画面の修正 |         | 追加 |

| 日付         | 担当 | 工程  | 作業内容      | 作業時間（分） |
|------------|----|-----|-----------|---------|
| 2026-10-10 | 佐藤 | 設計  | 画面遷移図の作成  | 30      |
| 2026-10-10 | 佐藤 | 実装  | 入力フォームの作成 | 15      |
| 2026-10-10 | 鈴木 | テスト | 入力チェックの確認 | 45      |

合計: 90 分

The handout worklog-complete.html after adding three entries of 30, 15 and 45 minutes.

If any field is blank, **no row is added and a notice appears**

### 作業記録

日付・担当・工程・作業内容・作業時間（分）を記録します

|                                                                                              |    |     |            |         |    |
|----------------------------------------------------------------------------------------------|----|-----|------------|---------|----|
| 日付                                                                                           | 担当 | 工程  | 作業内容       | 作業時間（分） | 追加 |
| 10/10/2026  | 鈴木 | テスト | 例: 一覧画面の修正 | 30      |    |

未入力の項目があります

| 日付         | 担当 | 工程  | 作業内容      | 作業時間（分） |
|------------|----|-----|-----------|---------|
| 2026-10-10 | 佐藤 | 設計  | 画面遷移図の作成  | 30      |
| 2026-10-10 | 佐藤 | 実装  | 入力フォームの作成 | 15      |
| 2026-10-10 | 鈴木 | テスト | 入力チェックの確認 | 45      |

合計: 90 分

Only Task was left blank before pressing Add. The list stays at three rows.

## Exercise 3: Worklog app

### Paste the 5 lines and **get it running in the browser yourself**

#### Scenario

Showing a client's verbal requirements as a working app the same day speeds things up. First, see on your own PC how long it takes to go from a spec to a working app.

#### Steps

- 1 Open the tpro-work folder in VS Code and select Agent below the Chat view input box
- 2 Open tpro-work/spec5.txt and copy the 5 lines with Ctrl+A, Ctrl+C
- 3 Paste them into the chat input box and send
- 4 Read the proposed diff from the top, then approve it
- 5 Open tpro-work on the desktop in File Explorer and double-click worklog.html
- 6 Enter 30, 15 and 45 minutes, and check the total field and the list table

#### Result

worklog.html is created in the tpro-work folder on the desktop. If the file name differs, send "Rename the file to worklog.html" in the chat to fix it.

## Completion criteria for the Worklog app

### Check all 3 yourself before you move on to the next block

#### Done when

- Double-clicking worklog.html opens your default browser, with an address starting with file:///
- Adding 30, 15 and 45 minutes makes 3 rows in the list, and the total shows 90
- With one field empty, clicking Add adds no row and shows "Some fields are empty"

#### If you're stuck

- The screen is blank. Press F12 and check Console. If external files fail to load, send "Fix it to run as a single HTML file without external libraries"
- Double-clicking opens VS Code, not the browser. Right-click the file, choose Open with, and select a browser.
- The total shows joined numbers such as 3015. This is the same problem we cover later today. For now, send 1 sentence: "Add the totals as numbers."
- The empty check fails only for time. Try each of the 3 input fields and name the field that does not work.

## Completion criteria for the Worklog app (continued)

---

### If time allows

- Send just 1 more sentence, "Also show the total in hours", and read only the changed lines.
- Send "Add a delete button to each row", then check that the total of the 3 entries is still right after adding.
- Check that reloading the page clears your entries. The spec does not mention saving, so this is correct.

# Break

---

## In the second half, you edit **the handout app**

### **Keep open**

The tpro-work folder. You can close worklog.html.

### **What you open next**

index.html in worklog-app. You do not need to open it yet.

Stop your work here

## From your app to the handout app

### Now you edit **code you did not write**

#### What we have done so far

You built worklog.html yourself, so you knew roughly what was in it.



#### What we do next

At work, you often change code that someone else wrote. The handout index.html has no filtering or totals, on purpose.



#### Why it matters later

Have Copilot read it first, then add these 2 features. Next session, you test this index.html.

# 06

## **Add features to an existing app**

---

Have Copilot read the code first, then add filtering and totals

## The Worklog app in the handout

It lists and adds records, but has **no filtering or totals**

| Item                 | Contents                                                                                                 | Today                          |
|----------------------|----------------------------------------------------------------------------------------------------------|--------------------------------|
| <b>Files</b>         | Just one file, index.html. The page, styles and logic are all in it.                                     | You add code to this file.     |
| <b>Input fields</b>  | 4 fields: Date, Category, Description and Hours. Category is Design, Implementation, Review or Testing.  | Used for filtering and totals. |
| <b>Data format</b>   | An array named records, with 4 fields: date, kind, memo, hours. Nothing is saved, so a reload clears it. | Keep field names               |
| <b>Not there yet</b> | Filter by category, total hours, and a check on the hours input                                          | You add the first 2 now.       |

A record you add disappears when you reload the browser. Keep in mind that this app does not save data.

## The app as it runs now

It can list and add records, but has **no filtering or totals**

**作業記録アプリ**  
日付・作業種別・内容・工数を記録して一覧に表示します

日付: 2026/08/28 作業種別: 設計 内容: 例: 画面遷移図の作成 工数(時間): 例: 1.5 追加

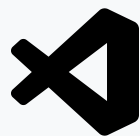
| 日付         | 作業種別 | 内容       | 工数(時間) |
|------------|------|----------|--------|
| 2026-08-24 | 設計   | 画面遷移図の作成 | 3      |
| 2026-08-24 | 実装   | 一覧画面の実装  | 4.5    |
| 2026-08-25 | レビュー | 設計書のレビュー | 1.5    |
| 2026-08-25 | 試験   | 単体試験の実施  | 2      |
| 2026-08-26 | 実装   | 検索処理の実装  | 6      |
| 2026-08-26 | 設計   | データ項目の整理 | 2.5    |
| 2026-08-27 | 試験   | 結合試験の準備  | 3.5    |
| 2026-08-27 | レビュー | コードレビュー  | 1      |

The handout Worklog app open in the browser. It contains 8 records.

## Where you write and where you run

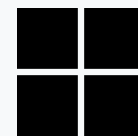
---

Write in VS Code, **run it in the browser**



### VSCode

Edit index.html and save with Ctrl+S. The page does not run here.



### Edge

Open the saved index.html to run it. After a fix, reload with Ctrl+Shift+R.

### The answer changes **if Copilot reads the file first**

#### Asking without the file

In Ask mode, if you keep index.html closed and send only "Add filtering by category", the reply may be a rewrite that ignores the existing render. Pasted in, it can show the list twice or break the Add button.

#### Asking after Copilot reads it

If you attach index.html, have it explained, then ask the same thing, the change reuses the existing render and records. The reply even says which function to add it below.

### Check the explanation **yourself, in the code**

1

#### **Open the whole folder**

In VS Code, open all of tpro-work with File > Open Folder. If you open a single file, you cannot reference the other files.

2

#### **See how it works now**

Open index.html in the browser. Check that the list shows and that you can add a record. See the working state before anything breaks.

3

#### **Attach the file in chat**

In the Copilot Chat input box, type # and select index.html. The attached file name appears above the input box.

4

#### **Ask only for an explanation**

Send the prompt on the next page as is. Do not let Copilot write code yet.

5

#### **Check the answer yourself**

Search index.html for the function names in the reply. If you cannot find them, do not trust the explanation.

## If you know **3 names**, you can read the reply

| Name              | Role                                 | In today's exercises                                            |
|-------------------|--------------------------------------|-----------------------------------------------------------------|
| <b>records</b>    | Array that holds the entered records | Rows are added to it or filtered                                |
| <b>render</b>     | Redraws the table from records       | Filtering and totals both call it at the end to update the page |
| <b>addRecord</b>  | Runs when you click Add              | Checks the input, then adds it to records                       |
| <b>createCell</b> | Helper that makes one table cell     | Do not touch it. Knowing the name is enough                     |

If the reply mentions render or records, it builds on the existing code. If not, Copilot may be rewriting without reading.

## A prompt to have Copilot read the code

### The last line stops Copilot from writing code too early

Paste this prompt as is. Attach index.html before you send it. Change the 3 points and it works for other apps too.

```
#index.html
```

```
Explain the role of this app and the flow until the list appears on the page.
```

```
Always include these 3 points:
```

- Where the work log data is stored
- The name of the function that draws the list
- The name of the function called when the Add button is clicked

```
Answer in English. Do not write any code yet.
```

The 3 points let you cross-check the reply yourself. Search for the function names to confirm them.

## Exercise 4: Filter by category

### Write a 3-line spec first, and the reply says where to paste

#### Scenario

Your first change at a client site is usually a small addition to a page someone else wrote. There is no spec, only a working page and a verbal request. From there, you add one feature without breaking anything, using the handout Worklog app.

#### Steps

- 1 Write a 3-line spec in chat: a category selector above the list, selecting a category shows only its rows, and selecting "All" shows every row again.
- 2 Attach index.html with # and send the 3-line spec. Add one sentence at the end: "Use the existing render as is."
- 3 In VS Code, find the place the reply names before you paste. If unclear, ask "Which function should I put it below?"

Change one thing at a time. If you ask for 2 at once, you cannot tell which one broke it.

## Exercise 4: Filter by category (continued)

---

### Steps

- 1 Save with Ctrl+S, then press Ctrl+Shift+R in the browser to reload. Change the selection and check that the list changes.
- 2 If it does not work, press F12 and paste the error message into the chat as is. Do not put it in your own words.

### Result

index.html in the worklog-app folder in tpro-work gets a category selector, and the list changes with the selection.

# Check separately that the new feature works and that the existing features are not broken

### Done when

- A category selector appears above the list. Selecting Design leaves only Design rows.
- Switching back to "All" restores the row count before filtering.

### If you're stuck

- The selector appears, but the list does not change. The event listener is missing. Just say "The list does not update when I change the selection" and have Copilot fix it.
- The page went blank. Press F12 and paste the first red line as is. Do not guess the cause.
- You saved, but the page did not change. It is the browser cache. Reload with Ctrl+Shift+R.
- The Add button stopped working. The existing render was replaced. Undo the paste, add "use the existing render" to the prompt, and ask again.
- You do not know where to paste. Ask "Which function should I put it below?" and the reply gives the exact line.

## Completion criteria for filtering (continued)

---

### Done when

- You can add a record from the form while filtered. Existing features still work.
- The new code in index.html calls the existing render. render has not been replaced by a different function.

## Completion criteria for filtering (continued)

---

### If time allows

- Show the count of shown rows above the list, like "Showing 12 rows".
- Add filtering by date range as well as category. Add one at a time.
- Check whether a record added while filtered shows in the list. If it does not, decide the spec yourself.

## Exercise 5: Total hours by category

### Pick one total from AI and **cross-check it by hand**

#### Scenario

Totals take only a few lines of code, so it is easy to trust the table as is. If one digit is off, the page looks the same. Make it a habit to add up one category yourself and compare.

#### Steps

- 1 Decide what to show: a table of total hours by category below the list. When a filter is on, total only the filtered rows.
- 2 Attach index.html the same way and send the spec. Add one sentence at the end: "Do not break the existing filter."
- 3 Pick one category, add up the hours in the list yourself, and compare with the total on the page. Always do this cross-check by hand.

## Exercise 5: Total hours by category (continued)

---

### Steps

- 1 Switch the category and check that the totals change too. If not, describe the symptom as is and have Copilot fix it.
- 2 Create model-memo.txt in tpro-work and write one line on the cross-check: did it match, and if not, what caused the gap.

### Result

index.html in the worklog-app folder in tpro-work gets a totals table, and model-memo.txt gets one line with the cross-check result.

### Check that **the numbers are right**, not just that they appear

#### Done when

- A table of total hours by category appears below the list.
- For one category, your hand-added sum matches the page total. You checked it yourself.
- Changing the filter changes the totals too.
- model-memo.txt has a new line with the cross-check result. If it was off, it says why.

#### If you're stuck

- The totals cover all records and ignore the filter. Just say "The totals do not change when I switch the category."
- Totals join like 1.52.5. They are added as strings. Paste the symptom as is to fix it.
- The totals table breaks into the list. The spec did not say where to show it. Add "Show it as a separate table below the list."
- The filter is gone. You left out the sentence from Step 2. Undo the paste, then ask again.

## Completion criteria for totals (continued)

---

### If time allows

- Sort by total, largest first.
- Add the record count to each total row to show how many records make up the hours.
- Have Copilot break one total on purpose and see if your cross-check catches it. Always copy the file before breaking it.

## The app with filter and totals added

When filtered to Design, **the totals cover only the 2 listed rows**

| 作業種別で絞り込み   |      |          |        |
|-------------|------|----------|--------|
| 設計          |      |          |        |
| 日付          | 作業種別 | 内容       | 工数(時間) |
| 2026-08-24  | 設計   | 画面遷移図の作成 | 3      |
| 2026-08-26  | 設計   | データ項目の整理 | 2.5    |
| 作業種別ごとの工数合計 |      |          |        |
| 作業種別        |      | 工数合計(時間) |        |
| 設計          |      | 5.5      |        |
| 実装          |      | 0        |        |
| レビュー        |      | 0        |        |
| 試験          |      | 0        |        |

A sample of the app after Exercises 4 and 5. Adding 3 and 2.5 by hand matches the total of 5.5.

## How to recover when it breaks

Keep **a working version**, and you can always go back to it

### Keep a working copy

When a feature works, copy index.html and save it under a new name. Add a suffix like \_ok1 to the file name.

### One change at a time

If you ask for 2 changes at once, you cannot tell which broke it. Decide on one change before you ask.

### Undo and ask again

If it gets out of hand, go back to the saved copy. Rewrite the prompt and ask again. The same prompt gets stuck in the same place.

This course does not use Git. Copies take its place. If you can use Git at work, it is more reliable.

# Break

---

## Last: choosing a model and **instructions files**

### **What you have so far**

index.html with filtering and totals. Check that it runs in the browser before the break.

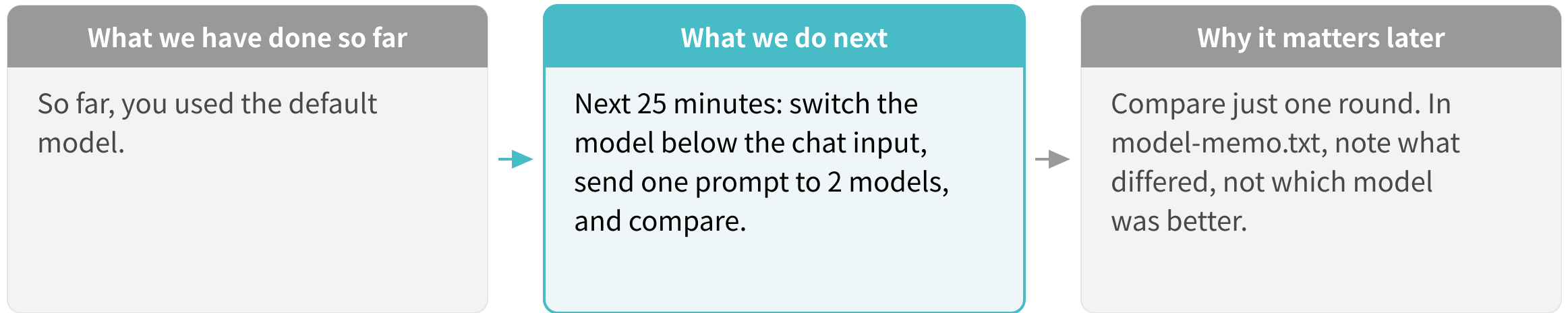
### **55 minutes to go**

Compare models on one prompt, then move the preamble to an instructions file

Stop your work here

## From adding features to models

The same prompt gets different answers **from a different model**





# 07

## Choosing models

---

Send one prompt to 2 models and note each difference in one line

## Before you choose

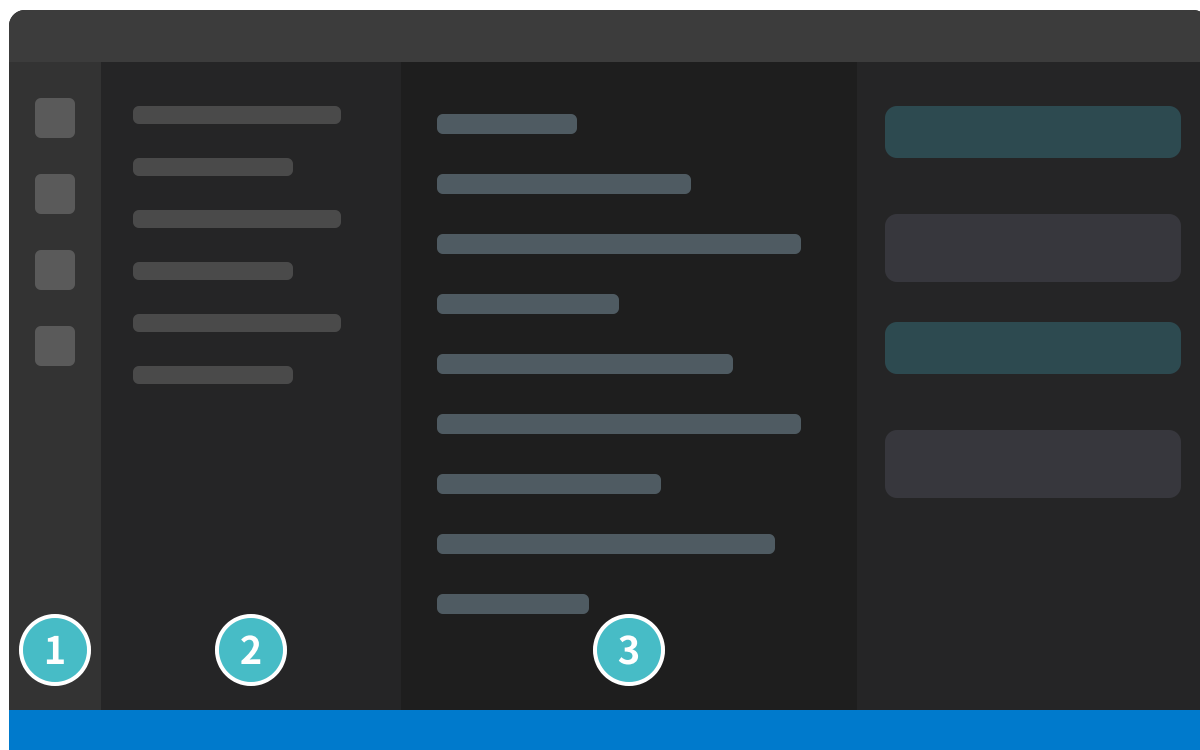
---

Choose a model **for the task at hand**, not by ranking

### **Not about memorizing rankings But choosing for the task at hand**

Copilot Chat lists models from several providers. The list changes every few months, so rankings will not help at your next site. Remember 2 things: where to switch, and what switching does not change.

## The model name **in the chat input box** is where you switch



### 1 Activity Bar

Click the Copilot icon to open the Chat view.  
If you cannot find it, press Ctrl+Alt+I.

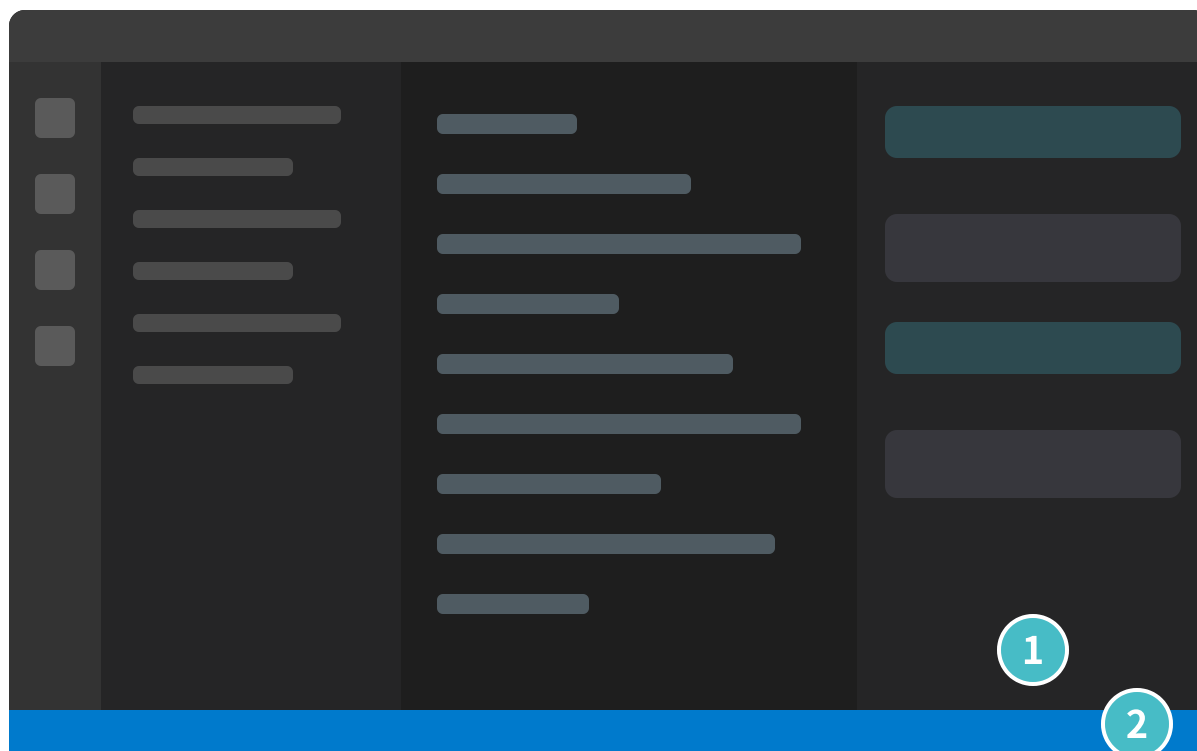
### 2 Explorer

Click worklog.html in tpro-work to open it.  
Copilot reads it in the next exercise.

### 3 Editor

This is worklog.html, now open. Picking files  
for the chat from the open files is faster.

## Where to switch models (continued)



### 1 Chat view

The input box shows the current model name. Click it to open the list of models. This is where you switch.

### 2 Status Bar

The Copilot icon appears on the right. A slash through it means you are signed out.

## The 3 model families

Models from 3 providers and Auto are listed **in the same place**

| Family            | Provider        | Common uses                                                  |
|-------------------|-----------------|--------------------------------------------------------------|
| <b>Claude</b>     | Anthropic       | Explaining existing code, handling long files at once        |
| <b>GPT, Codex</b> | OpenAI          | Adding small functions, rewriting into a fixed format        |
| <b>Gemini</b>     | Google          | Requests with screens or documents, research-style questions |
| <b>Auto</b>       | Copilot chooses | When you cannot decide                                       |

The models change. See the official docs for the latest list:

<https://docs.github.com/en/copilot/reference/ai-models/supported-models>

## Two ways to pick a model

---

If unsure, **keep the default** and switch only when it misses

### Use the default

The default is what you get if you change nothing. Usually it is fine.  
No need to stop working to choose a model again.

### Choose it yourself

Only when the same request misses twice in a row, switch to a model from another provider and resend it. Keep the prompt unchanged.

If you change the prompt and the model together, you cannot tell which helped. Change one at a time.

## Exercise 6: Compare models

### Send **the same prompt twice**; see what a model switch changes

#### Scenario

At client sites, the organization's settings may limit the models you can use. Even where you can choose, if you have never seen what switching changes, you switch models whenever something fails. Send the same prompt twice and see what changes and what does not.

#### Steps

- 1 Open worklog.html in tpro-work in VS Code. If you do not have the file you made, use index.html in the worklog-app folder.
- 2 Open model-memo.txt, which you made in the cross-check, and paste the prompt as one line to keep it. Use this prompt: "Explain what this file does for someone seeing it for the first time. Answer in English."
- 3 Type a half-width # in the chat input box to attach the file, and send without touching the model name. Read only the first 5 lines of the answer.

## Exercise 6: Compare models (continued)

---

### Steps

- 1 Click the model name in the input box, switch to a model from another provider, paste the saved prompt and send it again. Do not change a single character.
- 2 Add one line to model-memo.txt on what differed. Do not write which was better.

### Result

model-memo.txt in the tpro-work folder on the desktop. It holds one line with the prompt and one line on the differences.

### "No visible difference" is also a valid result

#### Done when

- model-memo.txt keeps the prompt you sent as one unchanged line. You can later check that you sent the same prompt both times.
- The line on differences says what differed, not which one was better.

#### If you're stuck

- Typing # shows no list. It is the full-width #. Switch to half-width and retype.
- The file is not attached. Check for a chip with the file name above the input box. If there is none, the file is not attached.
- The model list does not open. You are signed out. Sign in to GitHub again from the account icon at the bottom left.
- The list shows only one model family. Your organization's settings limit it. Send once with the default and ask a neighbor to show you the difference on their screen.
- You changed the prompt the 2nd time. Paste the saved line again in a new chat.

## Completion criteria for model comparison (continued)

---

### Done when

- You sent the 1st time with the default and the 2nd time with a model from another provider. The list shows the provider changed.
- If you saw no difference, you are done when you wrote "No difference found for this prompt".

## Completion criteria for model comparison (continued)

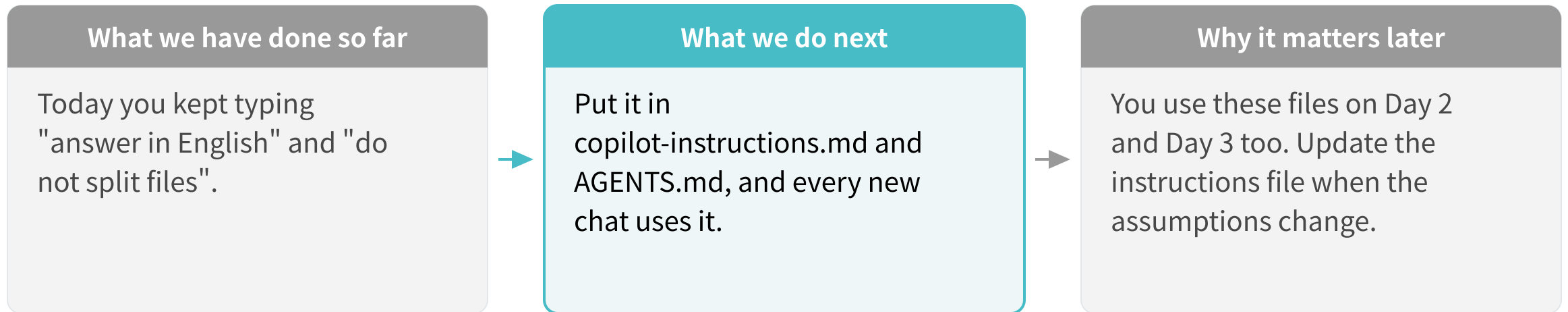
---

### If time allows

- Send the same prompt to a third model family and add one line comparing all 3.
- Send the same prompt twice to the same model and see if its answers vary.
- Add "Keep it under 200 characters" to the prompt, resend, and check the length.

## From models to instructions files

Move the preamble you type every time **into one file**



# 08

## Sharing instructions with the team

---

Move the preamble to copilot-instructions.md and AGENTS.md

# Move the preamble into **one file** and stop retyping it

### Type it at the start of every chat

A new thread does not carry over the previous chat, so you retype the language, banned libraries and comment style. Each person types something different, so code for the same screen comes out in different shapes. Code from people who skip the preamble gets more review comments.

### Write it in a file in a fixed place

Put it in `tpro-work/.github/copilot-instructions.md` and Copilot Chat reads it automatically in every chat. Participants do not need to paste it. To change the style, edit this one file and it applies to everyone's requests.

## In the instructions file, also write **what is not allowed**

The full copilot-instructions.md for the Worklog app. It has 4 headings: project premise, language and style, review points, and how to present changes. You may copy it as is in the exercise.

### # Project premise

- Write HTML, CSS and JavaScript in a single index.html. Do not split the file.
- Open index.html directly in the browser to run it. There is no build step.
- Do not use fetch. It does not work over file://, so receive files by drag and drop.

### # Language and style

- Write in JavaScript (ES2020). Do not use TypeScript.
- Do not add external libraries. Use only the ones that are bundled.
- Write variable and function names in English, and write comments in English too.
- Get DOM elements only with document.getElementById and querySelector.

Continued from the previous page.

- Do not delete the existing render function. Redraw the screen by calling render.

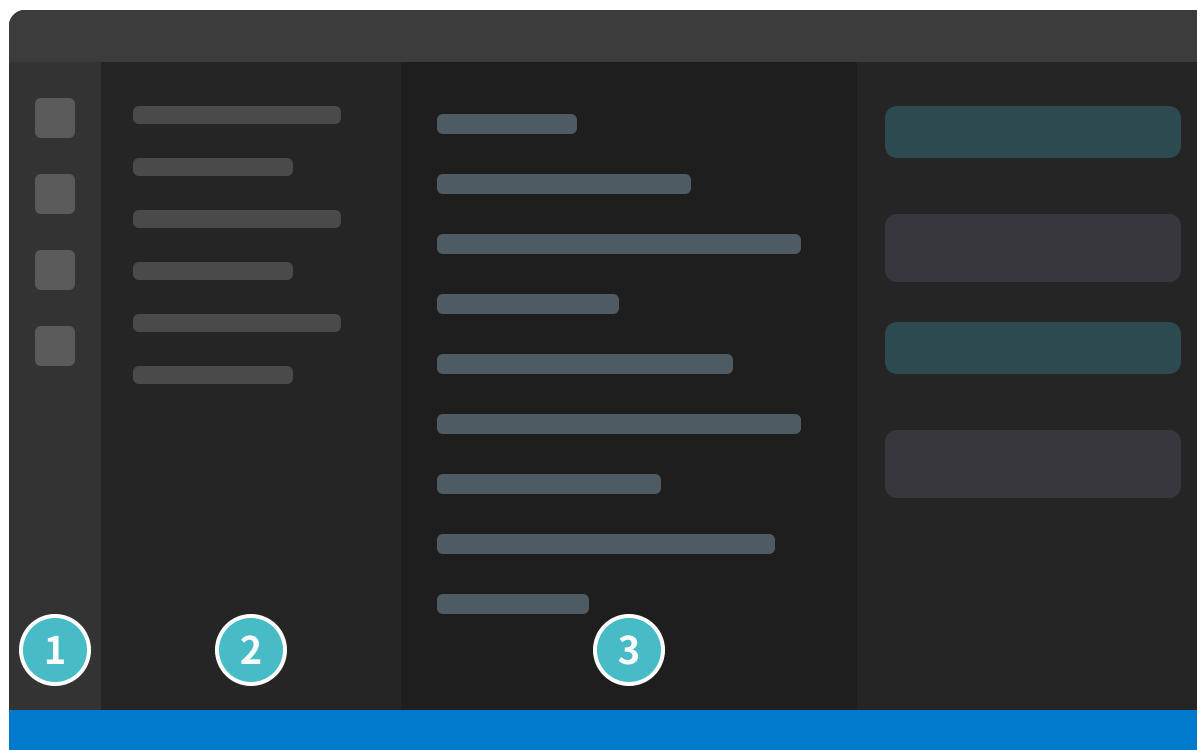
### # Review points

- Does the code add to records only after the input check passes?
- Is the text shown on screen kept in one place instead of scattered through the code?
- Does the code show a message in the on-screen message area instead of swallowing exceptions?

### # How to present changes

- Start with one line that gives the changed file names and function names.
- Show each affected function in full, without ellipses.

## Create .github right inside tpro-work, not one level up



### 1 Activity Bar

Click the top file icon to open Explorer. If it is closed, open it again from here.

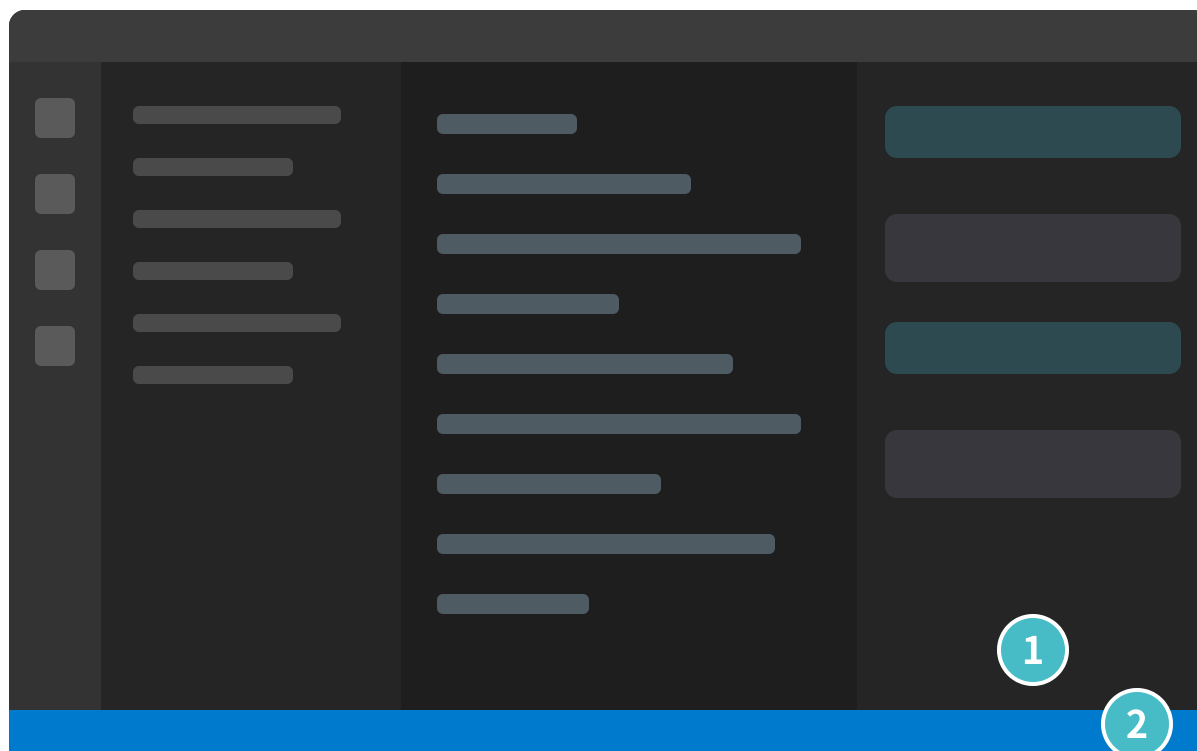
### 2 Explorer

Click the tpro-work row once to select it, then create .github with the New Folder icon. Without the leading dot, it is not read.

### 3 Editor

Select .github, create copilot-instructions.md with New File, paste the full text from the last page, and press Ctrl+S.

## Placing the instructions file (continued)



### 1 Chat view

After you send a request, check that `copilot-instructions.md` is in the list of references above the answer.

### 2 Status Bar

No dot next to the tab name means the file is saved. A file with the dot is not read.

## Screen after placing the file

Once it is **at this level**, it is ready to be read



### 1 .github folder

Directly in tpro-work. One level up, it is not read

### 2 copilot-instructions.md

Plural "instructions". One wrong letter and it is not read

### 3 Contents

4 headings, 17 lines. Copy this in the exercise

### 4 Chip above the input box

Shown because it is open. That does not mean it is used as instructions

## Two instructions files

### Different tools read them, so place and detail differ

| Item     | copilot-instructions.md                                       | AGENTS.md                                                          |
|----------|---------------------------------------------------------------|--------------------------------------------------------------------|
| Location | In tpro-work/.github/                                         | Directly in tpro-work                                              |
| Read by  | GitHub Copilot. Chat reads it every time                      | A common format read by several AI coding tools, including Copilot |
| Contents | Coding style: language, naming, what not to do, review points | Project premise: structure, how to run, what may be changed        |
| Length   | About 20 lines, most important rules first                    | About 30 lines. Facts only, no judgments                           |

Do not copy the same text into both. If you edit one, the other goes stale and results vary.

### Write **only facts** in AGENTS.md, not judgments

The AGENTS.md for the same Worklog app. It covers the structure, how to run it, and what may be changed. Put it directly in tpro-work.

```
# Worklog app
```

```
A one-screen app. Enter the date, category, description and hours, and see them in a list.
```

```
## Structure
```

- index.html ... screen, styles and logic in one file
- .github/copilot-instructions.md ... coding instructions

```
## How to run it
```

- Open index.html in the browser. Do not start a server.

## The full AGENTS.md (continued)

---

Continued from the previous page.

- After a change, save with Ctrl+S and press Ctrl+Shift+R in the browser to reload.

### ## What may be changed

- Do not rename the fields stored in records (date, kind, memo, hours).
- Do not delete and rebuild the existing render. Add to it instead.

### ## Constraints

- Do not write code that talks to external services.
- Do not put real company, person or project names in the data.

## Exercise 7: Set up the instructions file

### Compare one request **before and after** the instructions file

#### Scenario

At client sites, several people work on the same screen's code. If some type a careful preamble and others skip it, the same request returns code in different shapes, and review comments vary by person. See for yourself how much one instructions file closes that gap.

#### Steps

- 1 Open the tpro-work folder on the desktop in VS Code and send only "Add a function that validates the hours input" to Copilot Chat. Do not reword it.
- 2 Paste the generated code into before.txt in the tpro-work folder and save it.
- 3 In Explorer, select tpro-work, create .github with New Folder, and create copilot-instructions.md inside it.

## Exercise 7: Set up the instructions file (continued)

### Steps

- 1 Paste all of the "Full instructions file" page, add just 2 lines for your own workplace, and save with Ctrl+S.
- 2 Start a new chat thread and send exactly the same text as in Step 1. Paste the generated code into after.txt and save it.
- 3 Open before.txt and after.txt side by side (drag a tab to the right edge to split). Compare variable names, comment language and how the DOM is accessed.

### Result

Three files in the tpro-work folder on the desktop: before.txt, after.txt and .github/copilot-instructions.md.

## Completion criteria for the instructions file

### Instructions file: "read" and "followed" are **separate checks**

#### Done when

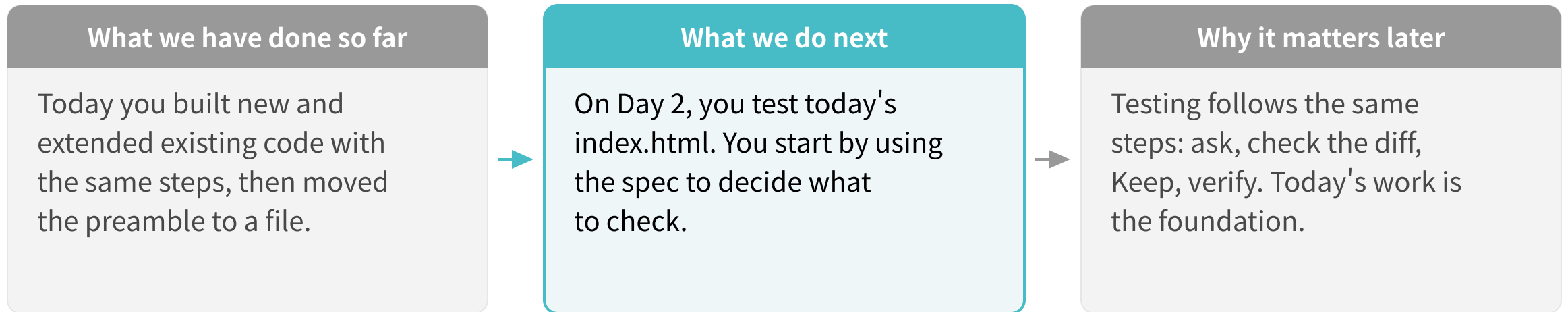
- copilot-instructions.md appears in the list of references above the answer. This confirms that it was "read".
- You can say how many of the instructions were applied and how many were not.

#### If you're stuck

- You cannot create the .github folder. File Explorer may reject names that start with a dot. Click New Folder in the VS Code Explorer and type .github.
- The name is not in the references list. The file is not saved, or .github is one level above tpro-work. Check the dot on the tab and the folder level in Explorer.
- No difference before and after. If you added details to the Step 1 request, they overlap with the instructions and hide the difference. Restore the wording and send again.
- You sent the 2nd request in the same thread. The earlier chat affects the answer, so you cannot see the difference. Open a new thread and resend.
- You named it copilot-instruction.md. "instructions" is plural. One wrong letter and the file is not read.

## From exercises to review

# Count today's cycles and choose **what to bring to Day 2**





# 09

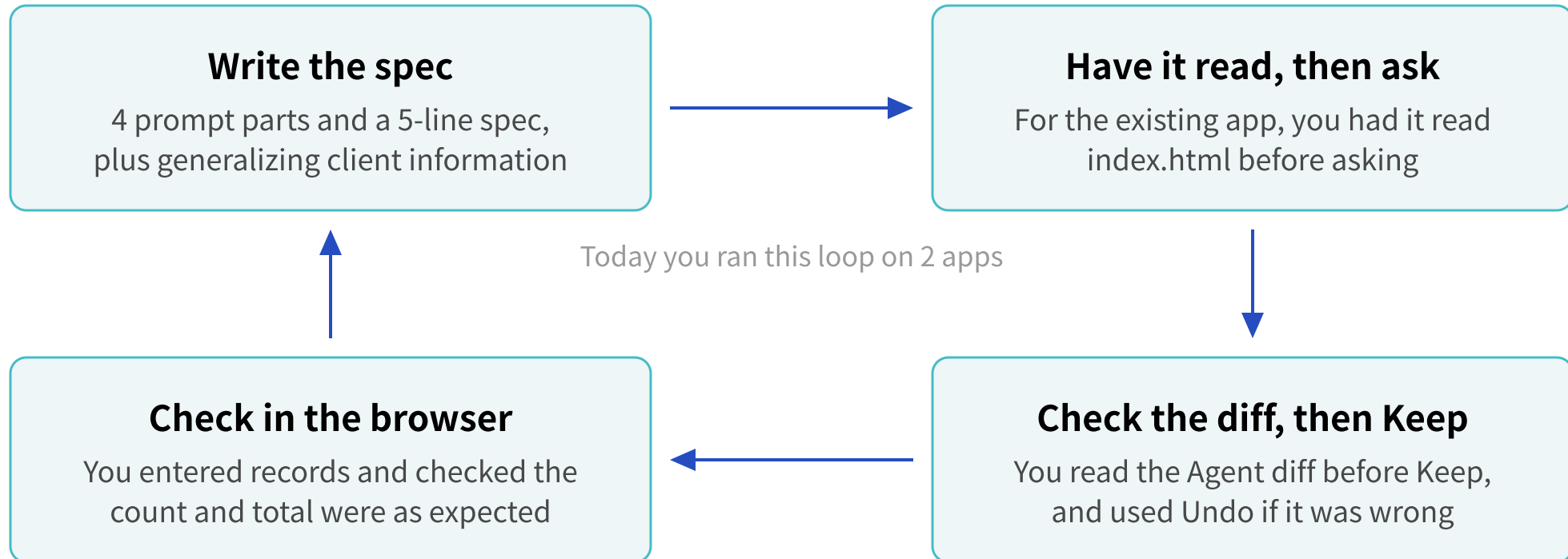
## **Review and Day 2 preview**

---

Go over today's loop and see where Day 2 begins

## What today's loop covered

Everything you did today **fits into these 4 steps**



## Today's takeaway

---

**Checking speed**, not generation speed, decides results

**Generation took only minutes**

**The key is a 10-second check every time**

You found every issue today by entering 3 records and checking the total. No special tools needed.

## Day 2 overview

---

# On Day 2, you **build tests from the spec** for today's features

### **Test design basics**

Trace each expected result to a requirement ID in the specification. Build cases with a decision table.

### **Test viewpoints, test cases and test data**

For today's index.html, list test viewpoints, make a test case table and test data, and write a test report.

### **Implementing import and unit tests**

Implement the import in the specification and judge the AI-written tests. Also fix display and export bugs.

If you work in testing, this day is the closest to your daily work.

## What to keep for next session

---

### **Keep** your tpro-work folder and stay signed in

#### **tpro-work folder**

Keep worklog.html, the updated index.html, both instructions files and model-memo.txt. Day 2 starts with them.

#### **Copilot sign-in**

The next session is about 2 weeks away. Do not sign out. Day 2 has no setup from scratch.

#### **One try at your desk**

Try today's pattern once in your own work. Pick a task where you can generalize client information.



Givery, Inc.

Teito Shibuya Bldg. 8F, 15-13 Nampeidai-cho, Shibuya-ku, Tokyo 150-0036, Japan