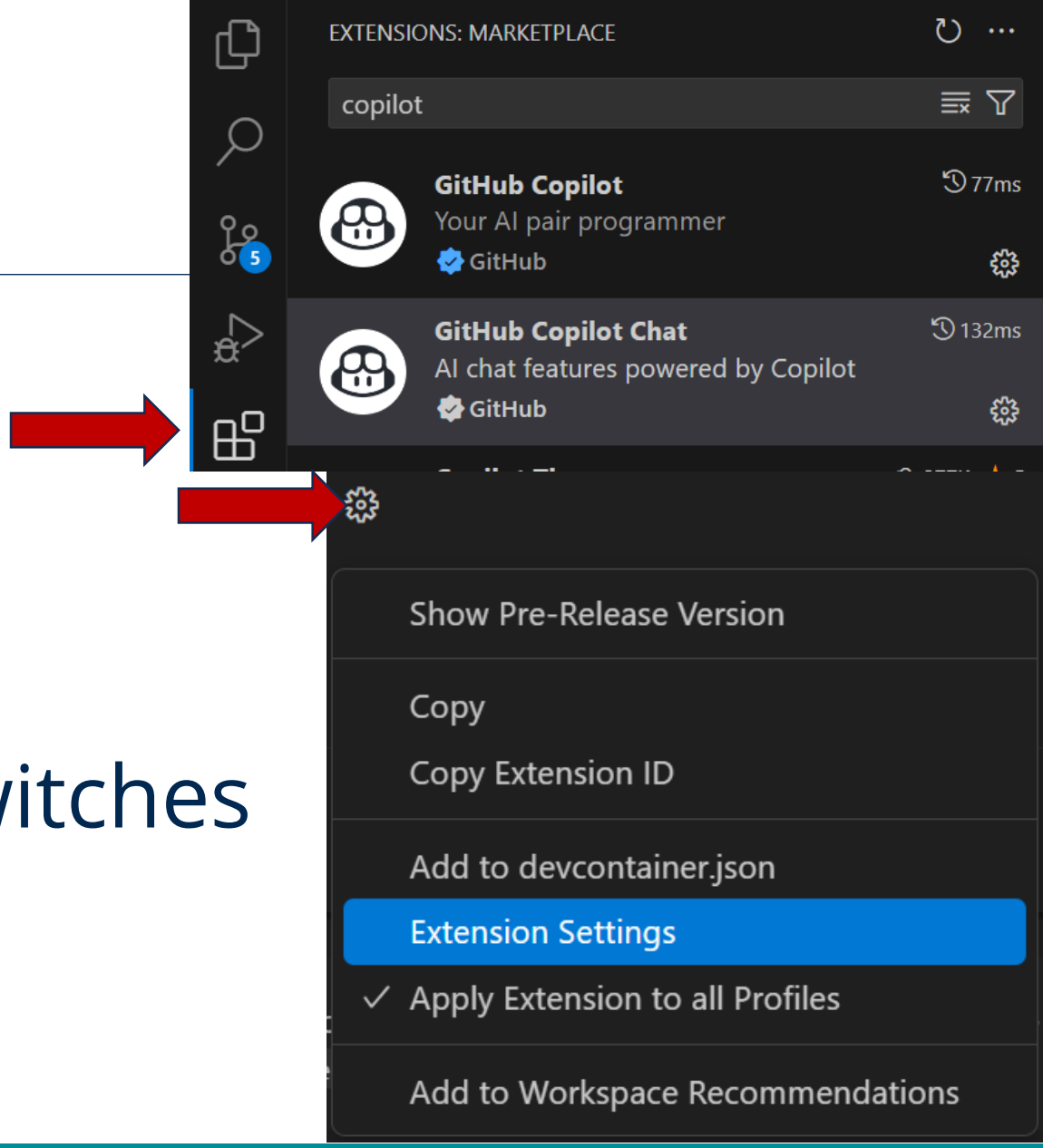
ELTE | FACULTY OF INFORMATICS

GitHub Copilot Best Practices

Configuration

- Click the extensions tab
- Locate cogwheel
- Click on settings

- Additional functionality switches



Prompt Engineering with Copilot

- You need to indirectly prompt Copilot!
- Lack of information will increase hallucinations

- **3S Principle**
  - **S**imple
  - **S**pecific
  - **S**hort

Simple

- **Decompose** the task to small, straightforward elements
- Prompt **iteratively** if needed
- **Complex** logic or formulas will be even **hard**er for Copilot to get

„Save a plot of the solution!”

„Load data!”
„Filter by…”
„Plot view of…”
„Save”

Specific

- **Include all** the necessary **details**, Copilot won’t read your mind
- Use **lists**, step-by-step definitions
- Provide **examples** if available

„Filter my data properly”

„Filter everything below … confidence by column …”

Short

- Use **short phrases**
- Do **not** try to be **polite** or **grammatically correct**
- Focus on **productivity**
- **Split it into two** requests if you cannot fit it in a simple sentence

„Filter everything below … confidence by column X”

↓

Get rid of X at <5

Information sources (by default)

- Currently open tabs (a bit unreliable)
- Code blocks (before and after)
  - Function/variable names are crucial
- Comments
  - Docstrings included

Autocomplete vs Inline chat

Autocomplete	Inline chat
Prompted indirectly by surrounding code	Prompted directly by instruction
Local (at cursor)	Local (sometimes inaccurate interval selection)
Straightforward / boilerplate code	More specific, specialized instructions
Modify by partial accept and continuation	Modify by iterative prompting
Automatic autocomplete model	Chat model with possible explanations and special commands

Where to chat?

Temp. Chat
CTRL + Shift +I

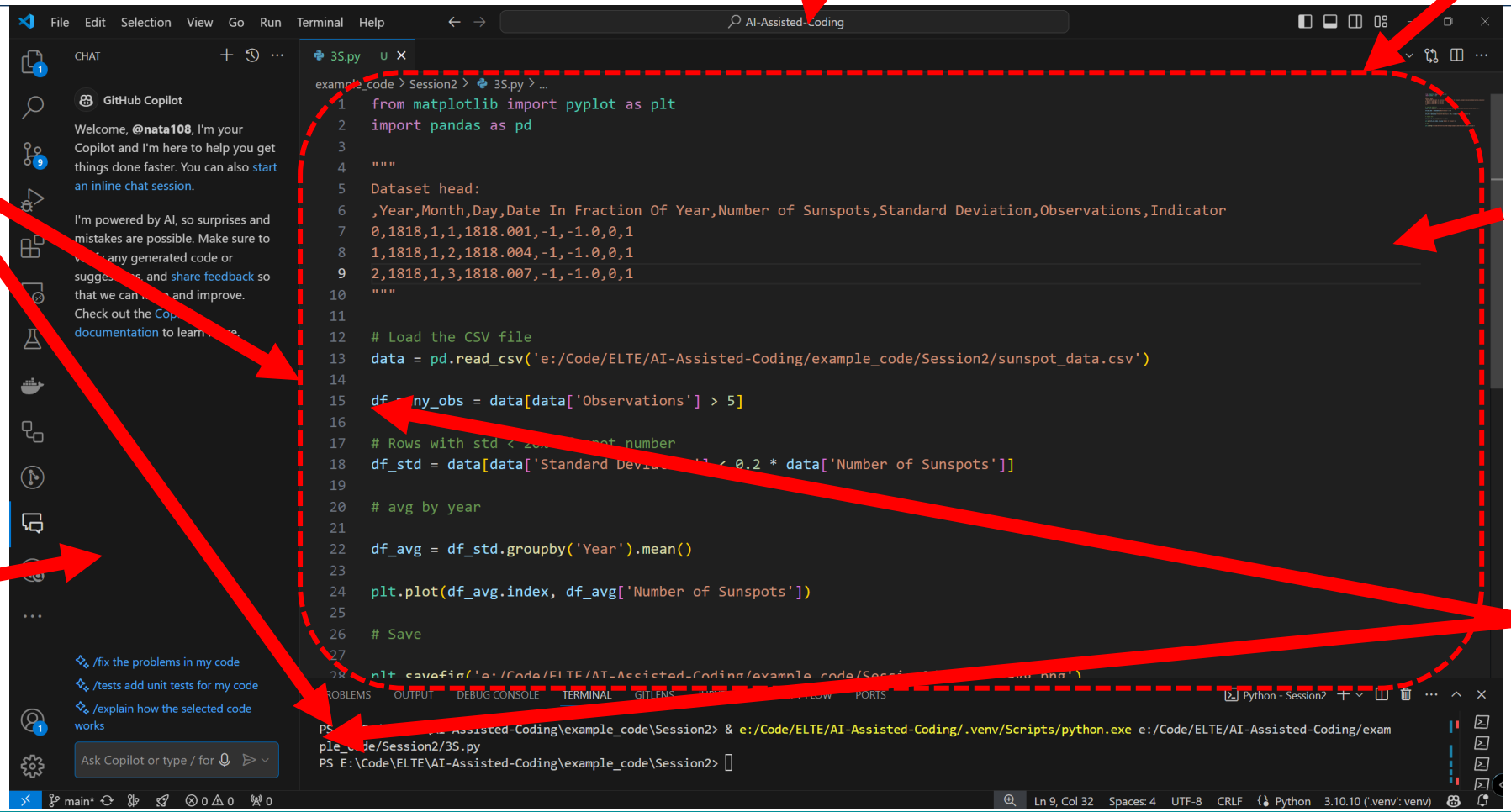
Autocomplete

Inline Chat
CTRL+I

Context Menu
RClick

Chat Panel

Sparkle ✨



Chat „participants”

- They introduce **specific knowledge** and **workflows** for generating the answer
- **@workspace**
  Gathers information from the currently open workspace, interprets the request in the specific exact context of it.
- Hybrid (keyword + semantic) processing method with **no completeness guarantee**

Chat „participants”

- They introduce **specific knowledge** and **workflows** for generating the answer
- **@vscode**
  For finding and handling VSCode’s functions, hidden settings, or extensions (provides support for extension development as well)

Chat „participants”

- They introduce **specific knowledge** and **workflows** for generating the answer
- **@terminal**
  Helps to execute/explain/create commands in the terminal. Takes the given terminal system (e.g.: Linux vs Windows PS) into account.

Chat sources

#selection - The current selection in the active editor This is used in the chat by default.

#editor - The visible source code in the active editor

#terminalLastCommand - The active terminal's last run command

#terminalSelection - The active terminal's selection

#file - Choose a file in the workspace

Chat commands

- „Canned” prompts
  - **Better** than writing your own
  - Also **quicker**
- Enabled in inline chat as well

Chat commands

@workspace		Try it!
/new /newNotebook	Create an intial workspace or Jupyter notebook	Initialize a workspace for data modeling!
/explain	Get the details of a given code	Ask for what PML is!

Chat commands

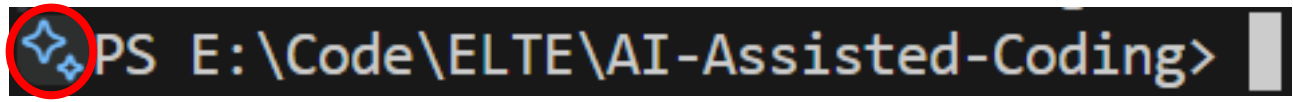
@workspace		Try it!
/doc	Create documentation for your code	Ask the model to create docstrings!
/tests	Generate unit tests for a functionality	Create tests for a data processor!

Chat commands

@vscode		Try it!
/search	Search for VSCode settings	Change copilot suggestion fonts!
/api	Learn more about VSCode's API	Find a way to add emojis to 50+ chars long lines.
@terminal		
/explain	Get the details of a given code	Ask for the details of the SLURM command!

Copilot in the terminal

- Look for the sparkles!



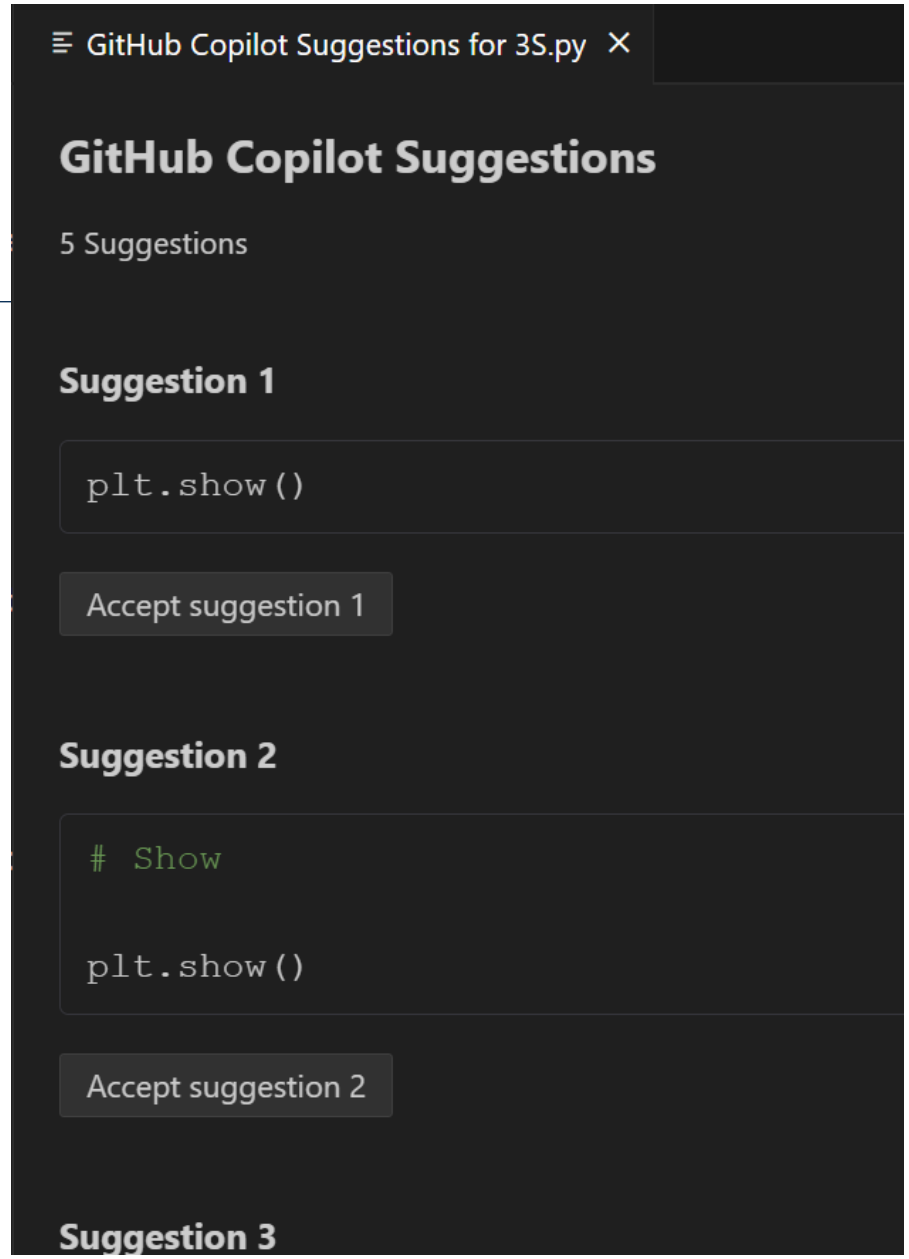
- Trigger inline chet, or select commands
  - Beware, single turn!
- Special commands are not (yet) enabled
  - For complex revisions use the standalone chat

Copilot for fixing bugs

- Refer to the **#terminalselection**
- Use **/fix**
- Currently the **chat panel** is the most stable
  - Hopefully this changes
- Look for the **sparkles** for fixing syntax errors

The alternatives panel

- Theoretically: Automatic alternatives should be generated
- In practice: You need to trigger it ***(CTRL+Enter)***
- Only advisable for long segments



Checking sources

- Check similar code sources
  - Navigate to the output tab
  - Choose Github Copilot Log
  - Check the links!

Showing **413** of **413** references in public code sources

Some of the original code references might not be available due to changes in the code

titansarus/Realtime-Project · clock/.gitignore copy

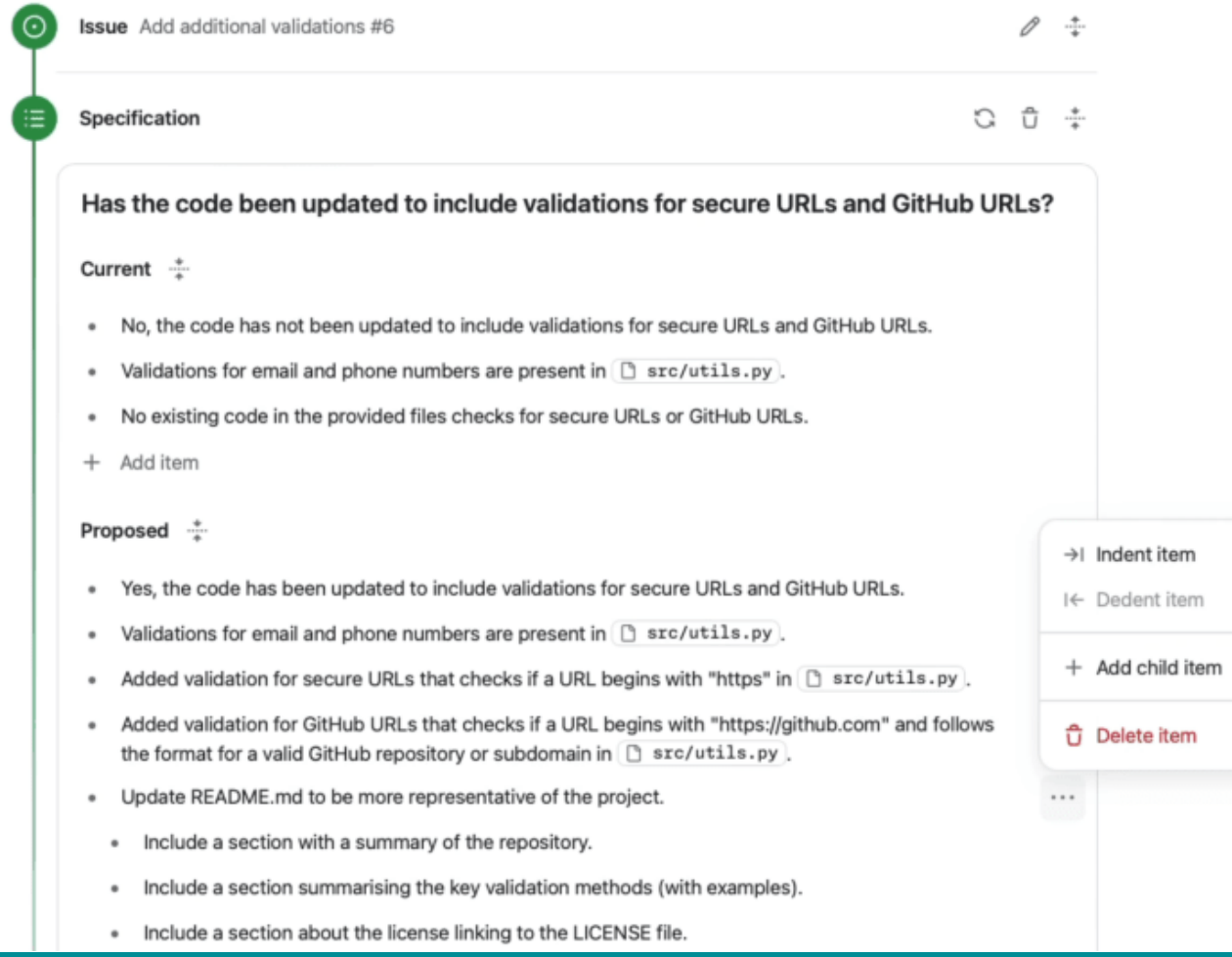
```
407
408 # Makeindex log files
409 *.lpz
410
411 # xwatermark package
412 *.xwm
413
414 # REVTeX puts footnotes in the bibliography by default, unless the nofoot
415 # option is specified. Footnotes are the stored in a file with suffix Not
416 # Uncomment the next line to have this generated file ignored.
417 #*Notes.bib
```

PROBLEMS 2 | OUTPUT | DEBUG CONSOLE | TERMINAL | GITLENS | JUPYTER | PROMPT FLOW | PORTS | GitHub Copilot Log

```
2024-06-03 09:09:48.638 [info] '\.gitignore' Similar code with 9 license types [AGPL-3.0, Apache-2.0, BSD-2-Clause, CC0-1.
0, GPL-3.0, LGPL-3.0, MIT, Unlicense, unknown] https://github.com/github-copilot/code_referencing?
cursor=b07a1435c647af32ae74017b157a617e&editor=vscode [Ln 307, Col 0] lpz # xwatermark package *.xwm # REVTeX puts
footnotes in the bibliography by default, unless...
```

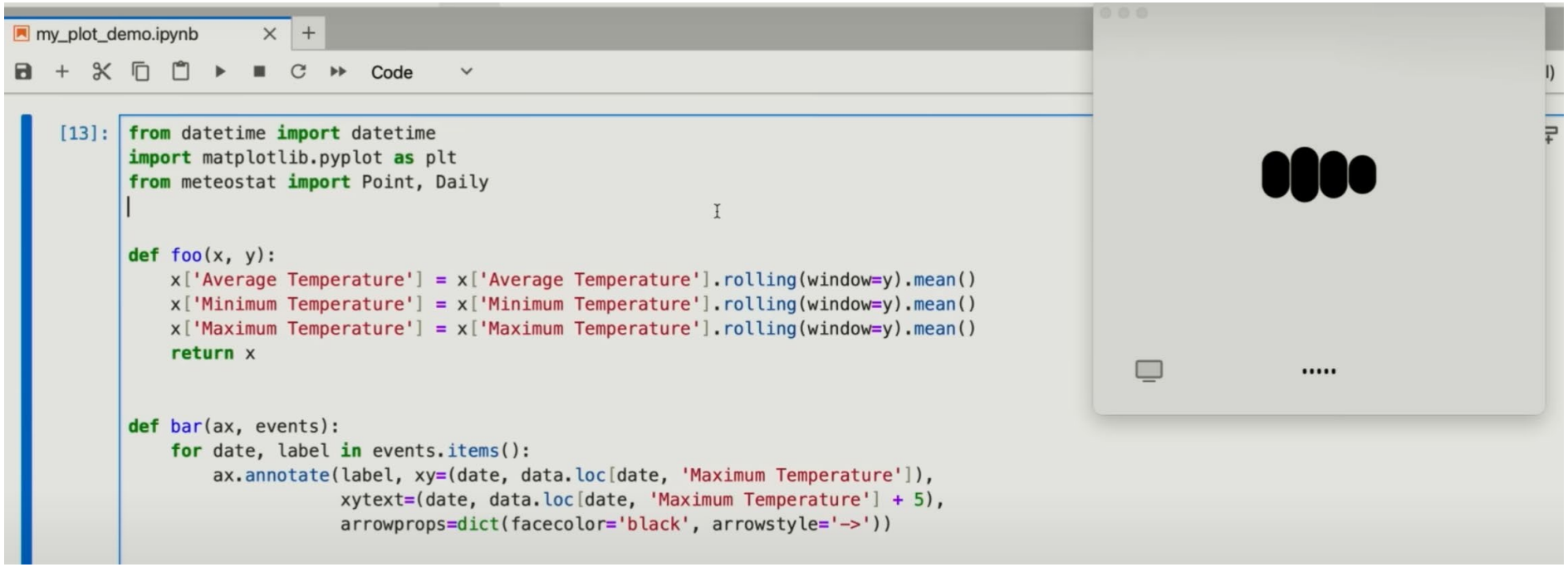
The future of copilot

- Copilot Workspace
- Issue-based agentic workflow
  - Planning
  - Human override possible



The future of copilot

- Real-time coding voice assistant
- With vision of your desktop



Example: Pendulum

- Create an animated **double pendulum simulation** in **matplotlib**
- Request an update to manually **set link lengths**!
- Ask for a local update to **display a „shadow”** [link positions at t-1]

Example: Data Visualization

- **Initialize** a new dataset visualization notebook that originates **from *3s_finished.py***
- **Violin plot**:
  - **Observations** as a function of **Month (sum)**
- **Bar plot**:
  - Create **an absolute floating time** variable (using fraction of the year)
  - Use **FFT** on all valid dates to find the most influential **components of the sunspot count**

Example: Latex-to-code

- Find your **favorite formula** in a paper!
- Go to **ChatGPT** (chat.openai.com)
- **Upload** a **printscreen** of your formula and ask for a **latex source** for it!
- Use **Copilot** to **generate a function** that implements the formula, **define** the required **inputs**!

Example: Markdown documentation

- Start writing a **pragraph** about your favorit **scientific topic**!
- Use **headers** to **outline** the documentation!
- Let Copilot suggest content
- **Type $$** and use Copilot to generate markdown **formulas** and **explain** the equations
- Use inline chat to **generate a table**