



Simplifying Data Analysis With GitHub Codespaces, Jupyter Notebooks & Open AI

Nitya Narasimhan, PhD
Senior Cloud Advocate, AI
Microsoft



#PyDataNYC | Nov 2023

<https://aka.ms/pydata-workshop-2023>



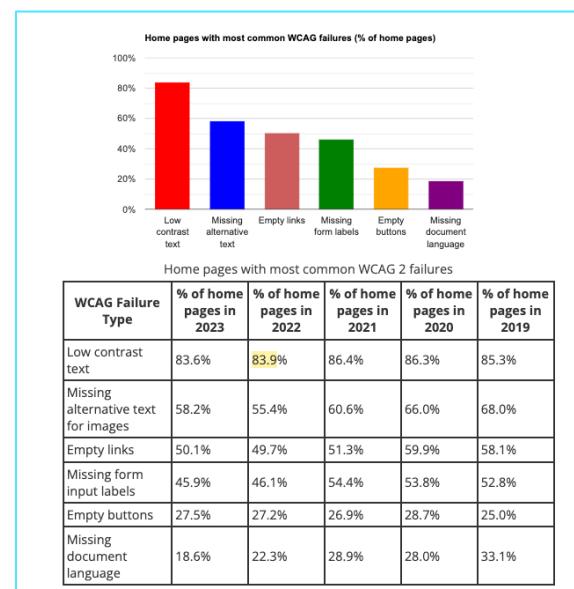
Motivation

“I’m an app developer not a Python expert. How can I learn enough to use it to tackle specific goals in my apps?”

Examples:

- The [USA Computing Olympiad](#) allows Python but lacks parity for resources vs. C/C++, Java. *Can I create self-driven learning resources for Python competitors like my 15yo?.*
- [Accessibility Testing](#) generates rich data (1K+ elements per page, 100 rules, 3-5 checks per rule = ~250-500K data points. *Can I give developers insight at authoring time, so they can fix issues earlier in development cycle?*
- [Information APIs](#) can generate pages of JSON data with dense information. *Can I visualize information in ways that help users get actionable insights?*

The screenshot shows the USACO Guide's Problems (Beta) section. At the top, there are search and filter fields for "Search" and "Algolia". Below that, there are dropdown menus for "Easy", "Modules", "Bronze", "Solved", and "Starred". A search bar is also present. The main area displays several problem cards, each with a title, a "View Solution" link, and a brief description of its difficulty level and topics covered. The problems shown are "Where Am I?", "Sleepy Cow Herding", "Shell Game", and "The Bucket List".



```
uploads.s3.amazonaws.com/uploads/organization/profile_image/512/64ce0b82/73bd-4ca0-8359-2c21513a0
383 "profile_image_90": "https://res.cloudinary.com/practicaldev/image/fetch/c--kA8Nj0Kg-
384 ", "type_of": "article",
385 "id": 1644652,
386 "title": "Want to buildintelligentApps with AKS & Serverless on Azure? Join us to #Learn
387 #DevOps with @mashuofficial & @joshduffney on the upcoming #LearnToBuild focused on stu-
388 "readable_publish_date": "Oct 24",
389 "slug": "want-to-buildintelligentapps-with-aks-serverless-on-azure-join-us-to-buildintelligent-
390 "url": "https://dev.to/azure/want-to-buildintelligentapps-with-aks-serverless-on-azure-join-us-to-build-
391 "content": "Want to buildintelligentapps with AKS & Serverless on Azure? Join us to #Learn
392 "public_reactions_count": 4,
393 "collection_id": 25071,
394 "published_at": "2023-10-24T13:15:00Z",
395 "positive_reactions_count": 4,
396 "cover_image": "https://res.cloudinary.com/practicaldev/image/fetch/s-u2j4XJ/-c_-Imag-
397 e_1644652.jpg",
398 "social_image": "https://res.cloudinary.com/practicaldev/image/fetch/s-bckB2oX/-c_-Imag-
399 e_1644652.jpg",
400 "uploads.s3.amazonaws.com/uploads/article/kmrk3u8phnbgplispp.json"
401 uploads.s3.amazonaws.com/uploads/article/kmrk3u8phnbgplispp.json"
402 "created_at": "2023-10-24T13:03:42Z",
403 "edited_at": null,
404 "updated_at": null,
405 "published_at": "2023-10-24T13:15:00Z",
406 "tag_list": [
407 "aws",
408 "beginners",
409 "webedv"
410 ],
411 "tags": "learnlive, azure, beginners, webdev",
412 "user": {
413 "name": "Sherry List 🎉",
414 "username": "sherrylst",
415 "twitter_username": "sherrylst",
416 "github_username": null,
417 "url": null,
418 "website_url": null,
419 "bio": null
420 },
421 "organization": {
422 "name": "Microsoft Azure",
423 "url": null,
424 "profile_image": "https://res.cloudinary.com/practicaldev/image/fetch/s-WMdo21B-/c-
425 uploads.s3.amazonaws.com/uploads/organization/kmrk3u8phnbgplispp.json",
426 "profile_image_90": "https://res.cloudinary.com/practicaldev/image/fetch/c--kA8Nj0Kg-
427 ", "type_of": "article",
428 "id": 1637344,
429 "title": "#BuildIntelligentApps Just #AskTheExpert !",
430 "description": "Join experts from @ZContainerApps @AzureFunctions, Azure AppService and
431 "readable_publish_date": "Oct 19",
432 "slug": "buildintelligentapps-just-asktheexpert-13d",
433 "url": "https://res.cloudinary.com/practicaldev/image/fetch/c--kA8Nj0Kg-
434 "content": "Want to buildintelligentapps just #AskTheExpert ?",
435 "public_reactions_count": 134,
436 "collection_id": 1637344,
437 "positive_reactions_count": 134,
438 "cover_image": "https://res.cloudinary.com/practicaldev/image/fetch/s-u2j4XJ/-c_-Imag-
439 e_1637344.jpg",
440 "social_image": "https://res.cloudinary.com/practicaldev/image/fetch/s-bckB2oX/-c_-Imag-
441 e_1637344.jpg",
442 "url": "https://dev.to/azure/o-buildintelligentapps-just-asktheexpert-13d",
443 "uploads.s3.amazonaws.com/uploads/organization/kmrk3u8phnbgplispp.json"
444 }
```

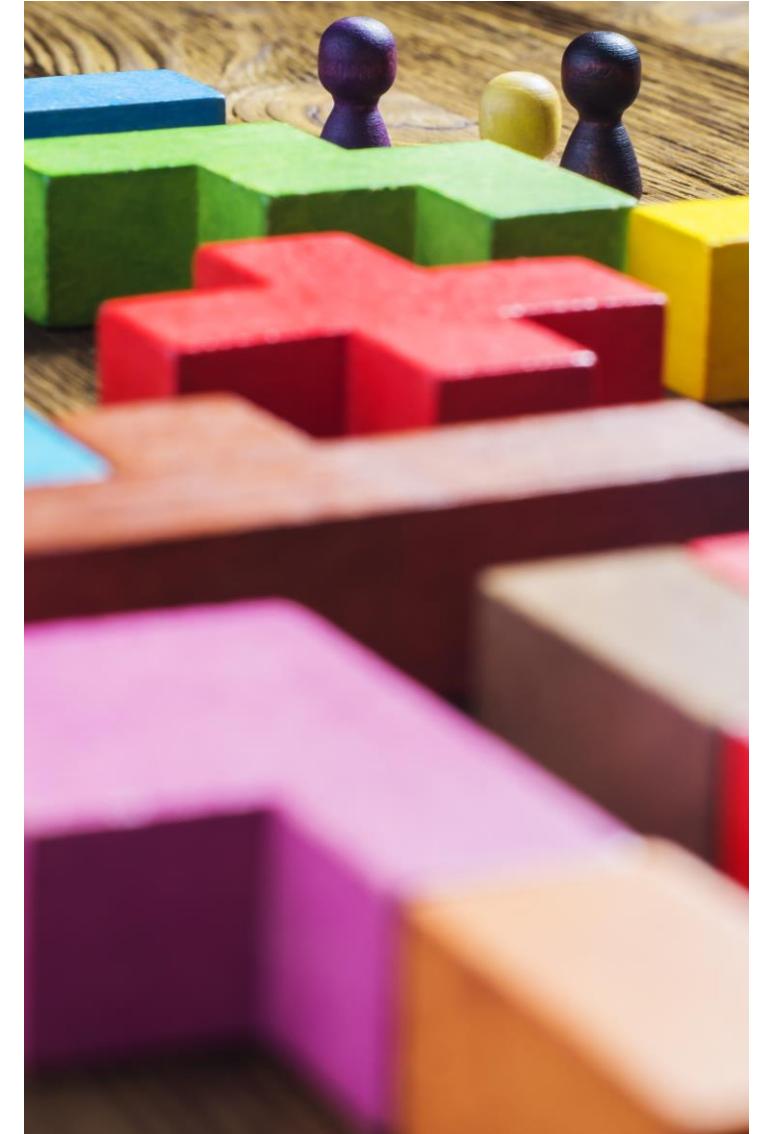
Objectives

Focused Learning – I want to prioritize things that help me get closer to my goal.

"Don't boil the ocean. Tell me how I can optimize my time,"

Transferable Learning – I want to make it easy for others to reproduce or extend my work.

"What tools or environments should I use that make this possible?"



Challenges

Developer Environment – setup can be hard to navigate for beginners.

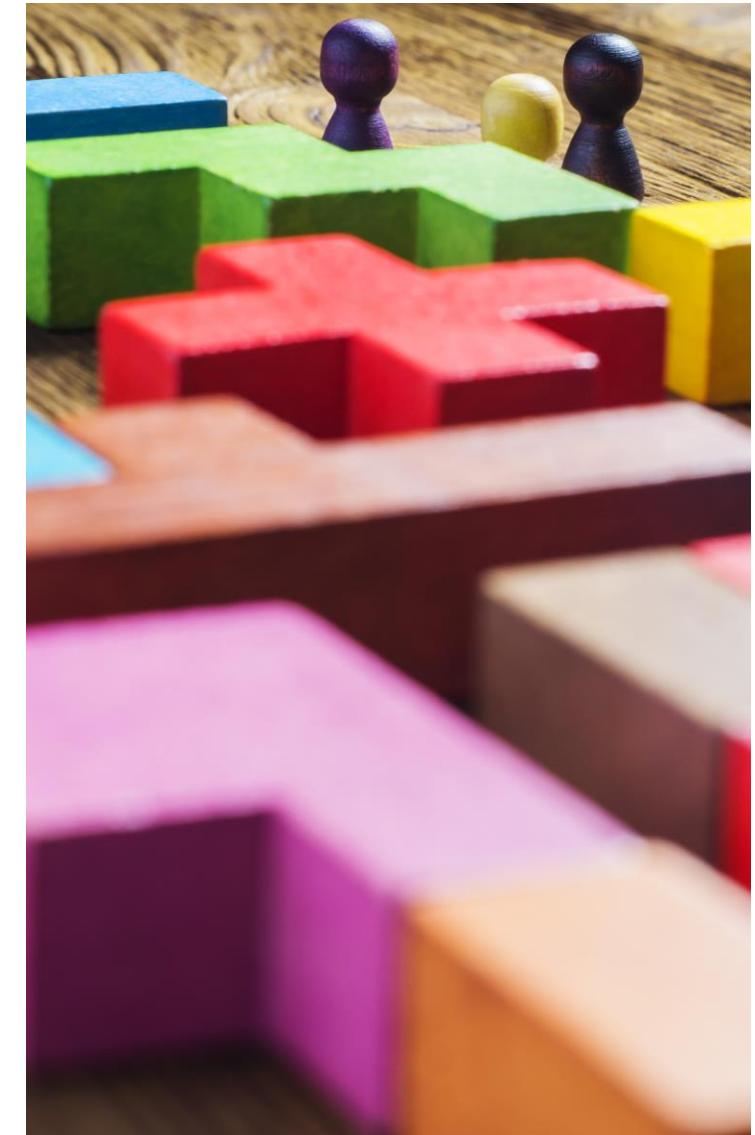
"It works on my machine. Must be some dependency."

Learning Process – can disrupt state of flow when context-switching across tools.

"I went to Google it – and lost time going down a rabbit hole."

Knowledge Gaps – can hamper the intuition that only comes with experience.

"I don't know what I don't know – what if I missed something?"



What I want to cover today

Transferable Learning
with [Jupyter Notebooks](#)

Consistent Dev Environment
with [GitHub Codespaces](#)

Focused Learning (in-context)
with [GitHub Copilot](#)

Building AI-assisted intuition
using [Microsoft LIDA](#)





<https://aka.ms/pydata-workshop-2023>

What You Need

- GitHub Account
- Github Codespaces (within free quota)
- Kaggle Dataset (or bring your own)
- GitHub Copilot (free trial available)*
- Open AI API Key (paid subscription)*

* require paid accounts – optional exercises if you don't have one.



Part 1:

Get A Reproducible Development Environment with GitHub Codespaces + Jupyter Notebooks

Exercise 1: Fork Repo, Launch Codespaces

Based on the official GitHub Codespaces for Jupyter template.

Fork the repo to your own profile, then launch a codespace from the menu



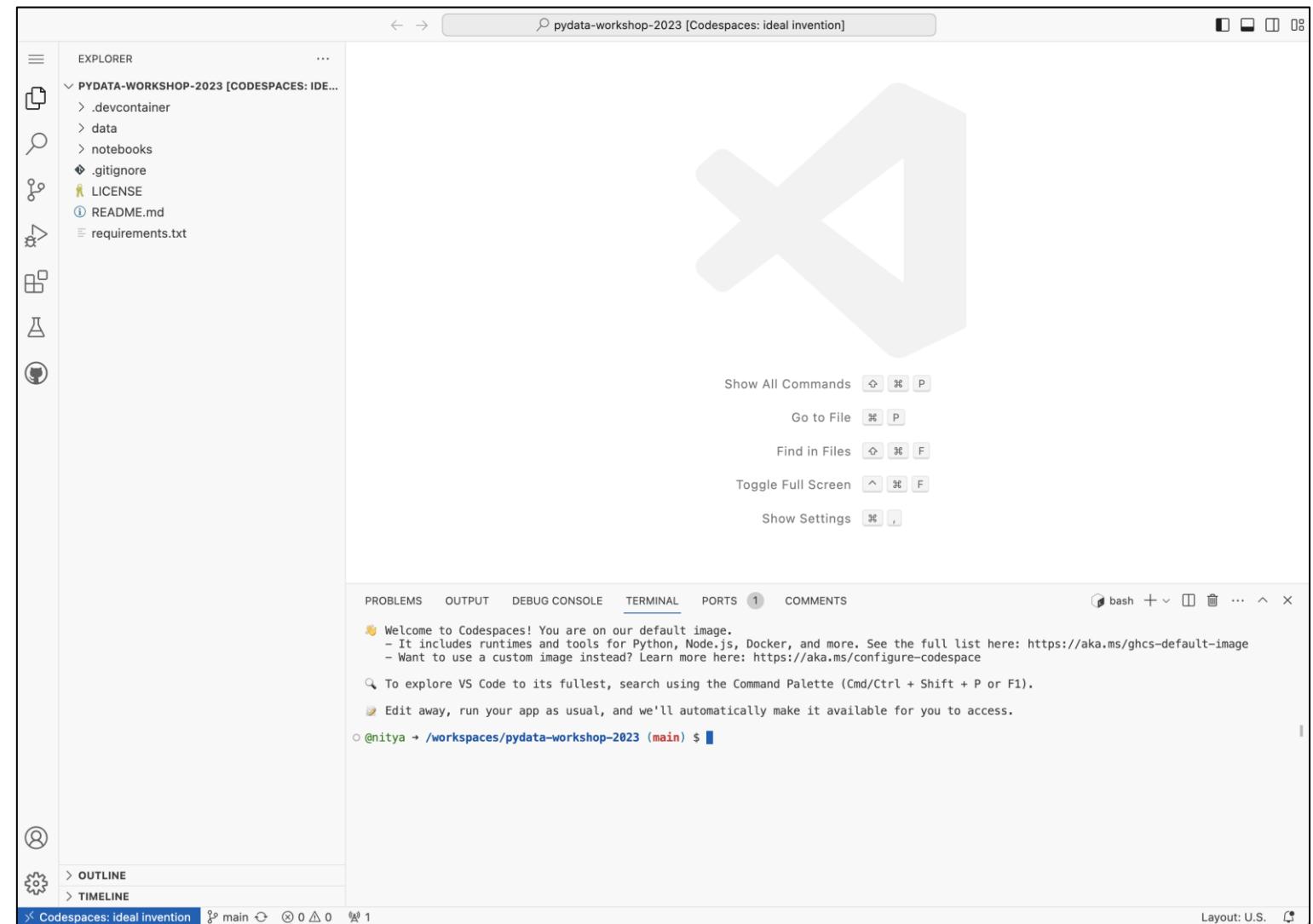
The screenshot shows the GitHub repository page for 'pydata-workshop-2023'. The repository is public and was generated from [github/codespaces-jupyter](#). The main interface includes a sidebar with navigation links like Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The central area displays the repository's contents, including files like .devcontainer, data, notebooks, .gitignore, LICENSE, README.md, and requirements.txt, all with an 'Initial commit' status. A prominent feature is the 'Codespaces' section, which is currently empty ('No codespaces') and includes a 'Create codespace on main' button. Below this, there's a banner for 'GitHub Codespaces ❤️ Jupyter Notebooks'. The right side of the page contains sections for About (repository details), Releases (no releases), Packages (no packages), and Languages (Python 100%).

Exercise 1: Fork Repo, Launch Codespaces

Based on the official GitHub [Codespaces for Jupyter](#) template.

Fork the repo to your own profile, then launch a codespace from the menu

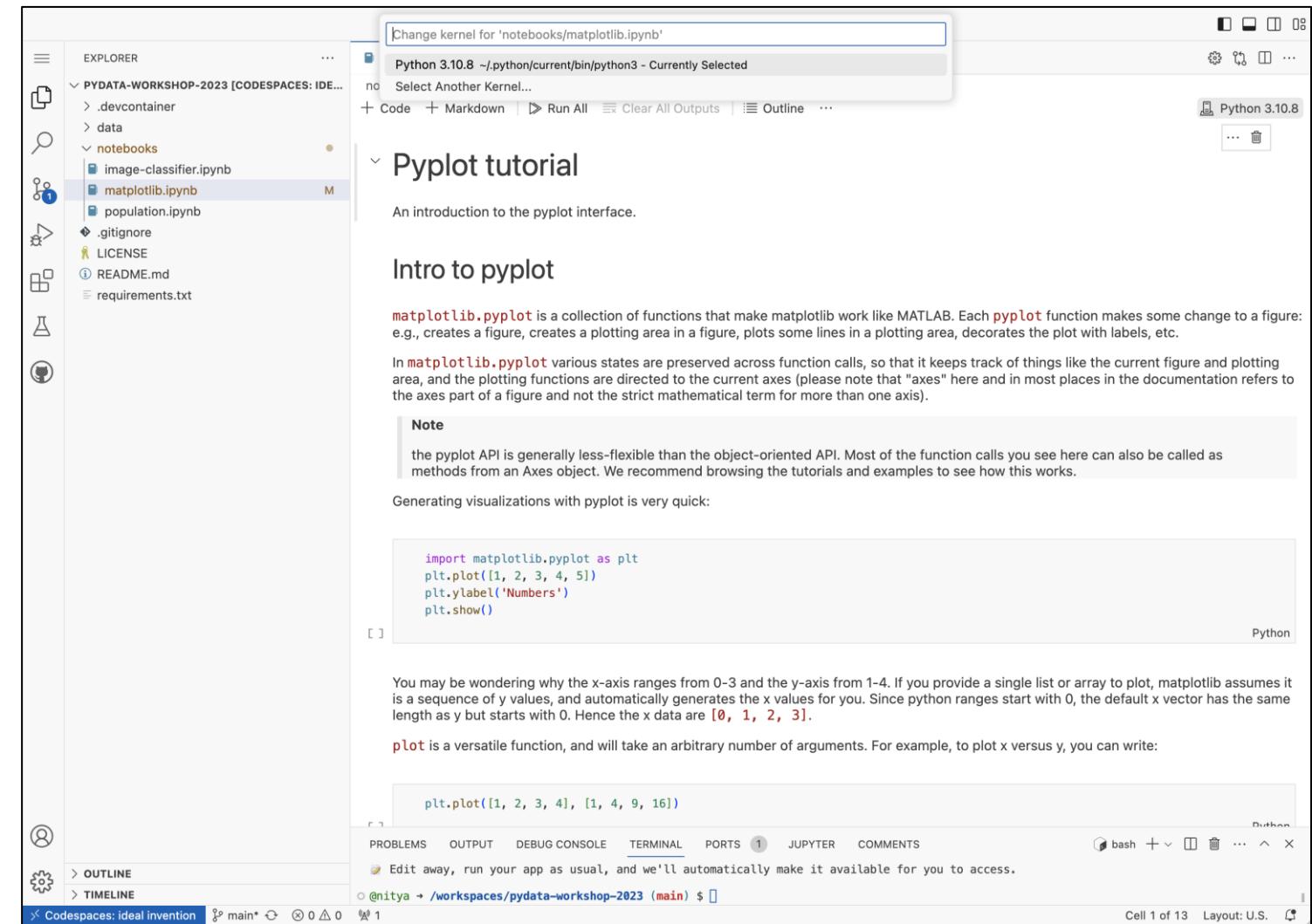
You should see a Visual Studio Code IDE + a dev environment ready to use!



Exercise 2: Let's Run the Default Notebook!

Open the *notebooks*/ folder and look for the *matplotlib.ipynb* file.

If not already set, 'Select Kernel' pick Python 3.10.8 target, 'Clear All Outputs'



Exercise 2: Let's Run the Default Notebook!



Open the *notebooks*/ folder and look for the *matplotlib.ipynb* file.

If not already set, 'Select Kernel' pick Python 3.10.8 target, 'Clear All Outputs'

"Run All". You just setup a Python env with Jupyter runtime with a few clicks.

The screenshot shows a Jupyter Notebook interface within a CodeSpaces workspace. The left sidebar displays a file tree for 'PYDATA-WORKSHOP-2023' containing notebooks like 'image-classifier.ipynb' and 'matplotlib.ipynb'. The main area shows a code cell for 'matplotlib.ipynb' with Python code using NumPy and Matplotlib to plot several data series. The output cell displays a scatter plot with red dashed lines, blue squares, and green triangles. The bottom navigation bar includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, JUPYTER, and COMMENTS, along with a terminal prompt and status indicators.

```
import numpy as np

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```

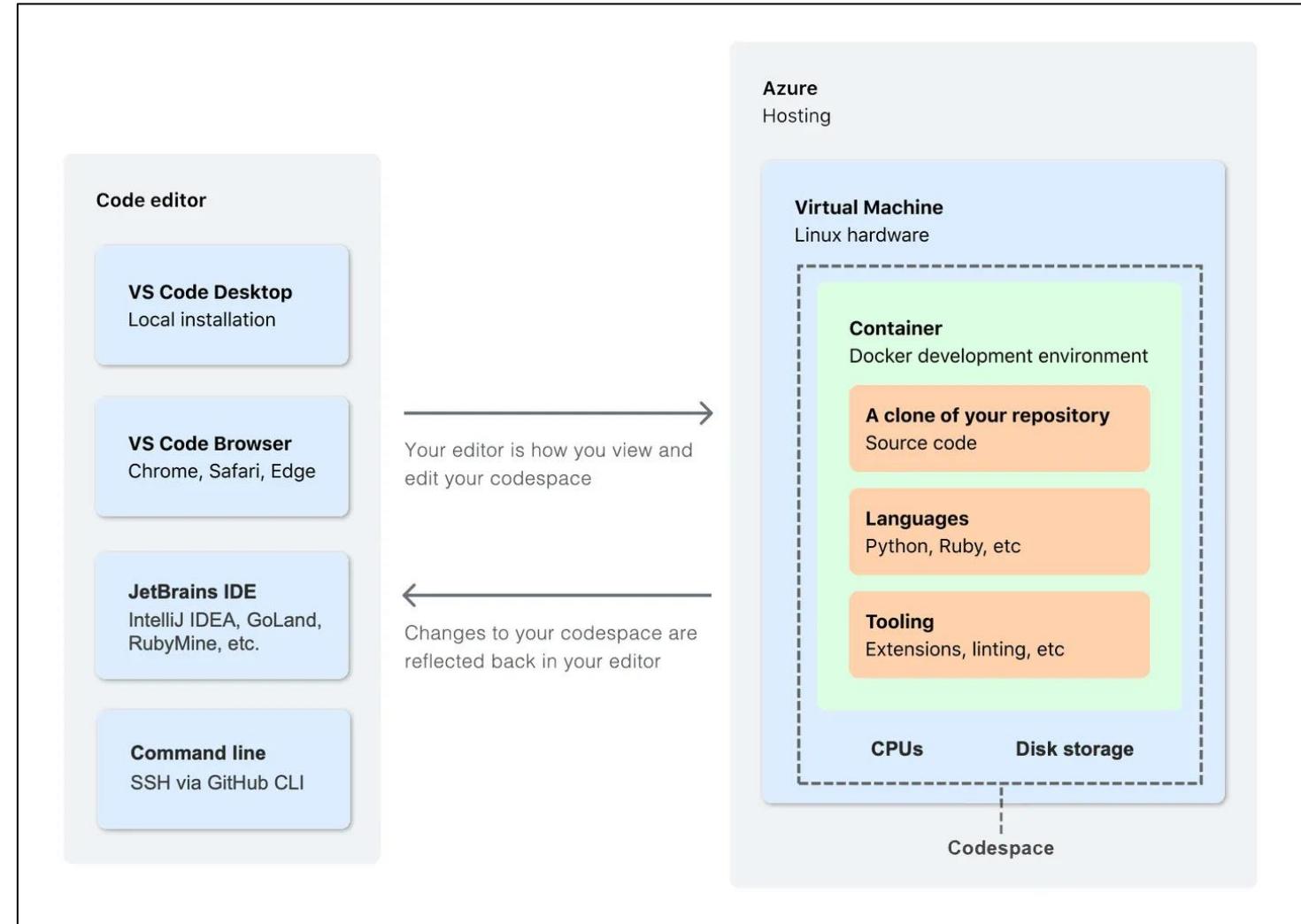
The plot shows three data series: a red dashed line (y=x), blue square markers (y=x²), and green triangle markers (y=x³). The x-axis ranges from 0 to 5, and the y-axis ranges from 0 to 100.

Under The Hood: What's GitHub Codespaces?

A GitHub Codespace is a “[Development Container](#)” that runs in a dedicated VM in the Azure Cloud.

You can set a different default editor – but VS Code gets us Extensions!

Editor preference	
<input type="radio"/>	Visual Studio Code Connect to the cloud from your local desktop client. Requires Visual Studio Code with the GitHub Codespaces extension.
<input checked="" type="radio"/>	Visual Studio Code for the Web Edit and preview changes straight from the browser.
<input type="radio"/>	JetBrains Gateway Connect to the cloud from your local desktop client. Requires the JetBrains Gateway with the GitHub Codespaces plugin, and a JetBrains license.
<input type="radio"/>	JupyterLab Edit and run notebooks from the browser with JupyterLab.

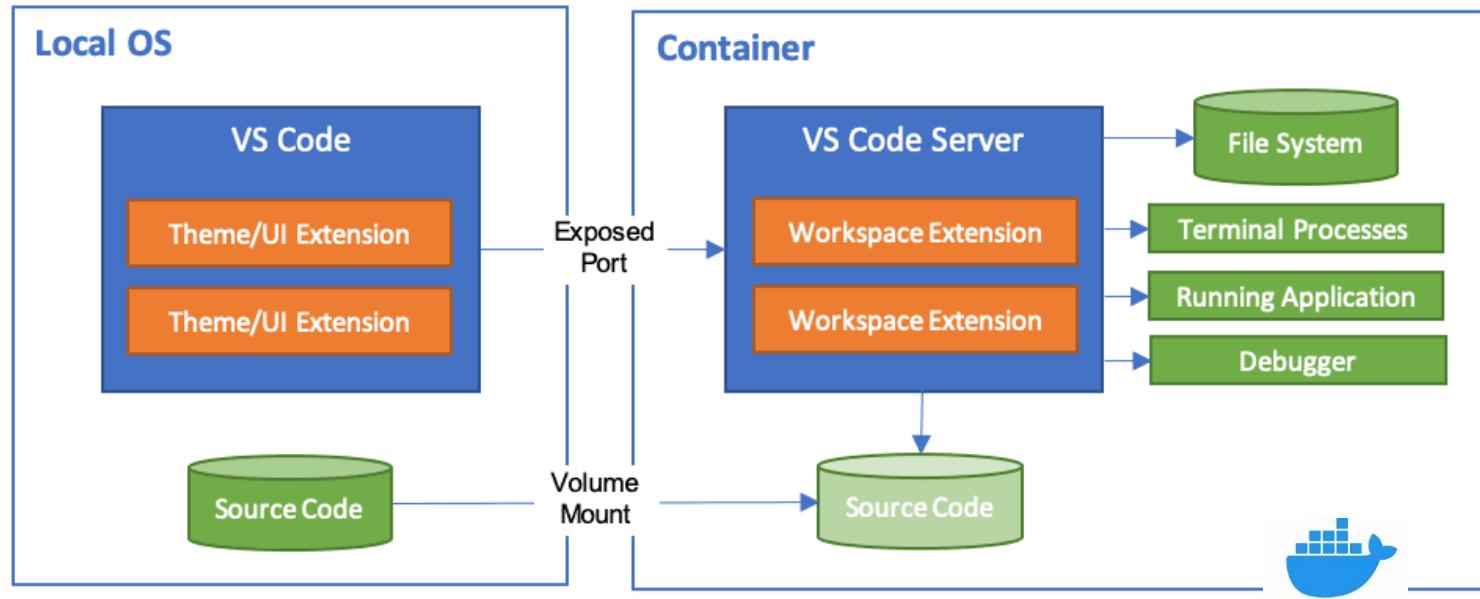


Under The Hood: What's a “Development Container”?

“Configuration as Code”
using *devcontainer.json* –
think version controlled!

Lifecycle hooks – update
env from *requirements.txt*

```
devcontainer.json x
1
2  "image": "mcr.microsoft.com/devcontainers/universal:2",
3  "hostRequirements": {
4    "cpus": 4
5  },
6  "waitFor": "onCreateCommand",
7  "updateContentCommand": "python3 -m pip install -r requirements.txt",
8  "postCreateCommand": "",
9  "customizations": {
10    "codeSpaces": {
11      "openFiles": []
12    },
13    "vscode": {
14      "extensions": [
15        "ms-toolsai.jupyter",
16        "ms-python.python"
17      ]
18    }
19  }
20
21 }
```



Your Visual Studio
Code editor runs in
the host machine
outside the container

Your code (filesystem)
is mounted by the
container – and stays
in sync with changes

A dev container is a Docker
container with a predefined
image that has all your dev
environment dependencies

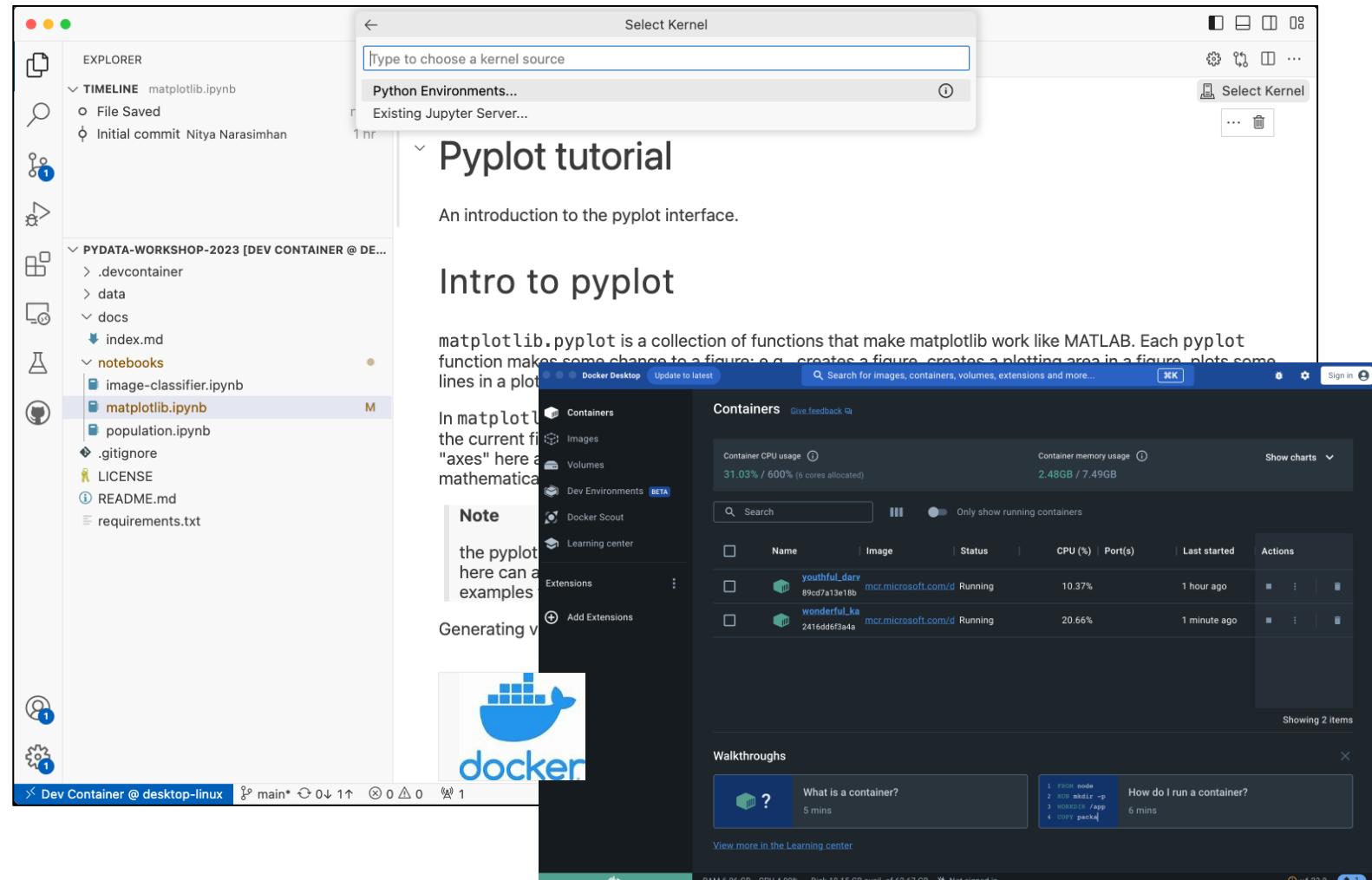
Everyone gets the **same dev environment** instantly

Under The Hood: Run it locally - with Docker Desktop!

Just “Open in Remote Container” – status shows Dev Container (local) or Codespaces (cloud) in blue

No code changes needed to move between the two.
It just works!

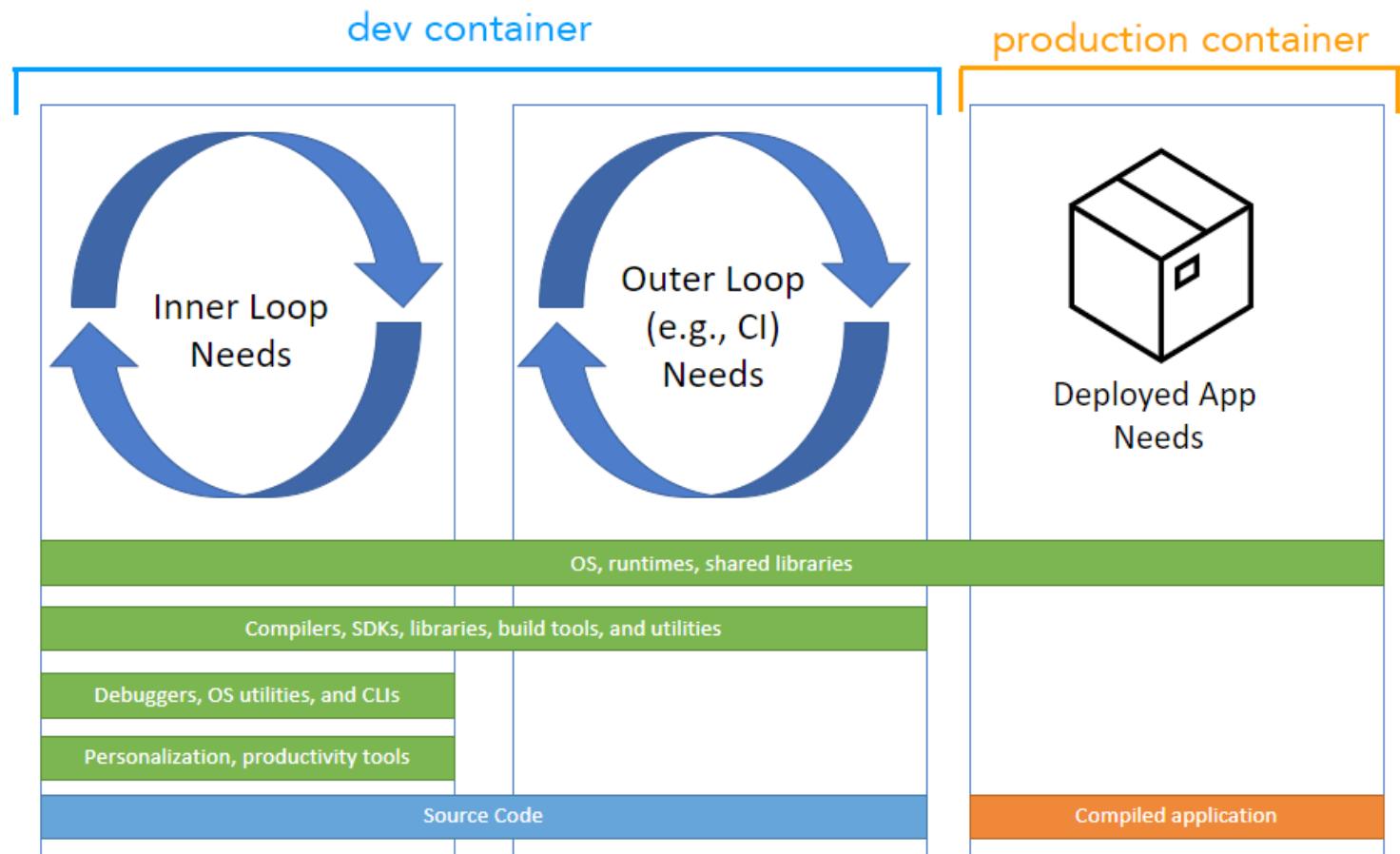
Local usage saves your quota – but has some limitations (e.g, “secrets”)



Under The Hood: Production vs. Dev Containers

As an app developer, I get a **consistent runtime** for the entire cycle from dev to staging to production

And I can debug & test in development (with richer tools) & be confident that deployed experiences will function as expected





Part 2: Get Focused Learning With GitHub Copilot

Exercise 3: Install GitHub Copilot Chat Extension

Add it on demand – or save to *devcontainer.json* for reusability by all.

I've chosen to save this – users will now get this in their env automatically.

The screenshot shows the VS Code interface with the Extensions Marketplace open. A context menu is displayed over the "GitHub Copilot Chat" extension, with the option "Add to devcontainer.json" highlighted. To the right, a code editor window displays a JSON file named "devcontainer.json". The file contains configuration for a development container, including the image to use, host requirements, and a list of extensions to install. The "GitHub Copilot Chat" extension is listed under the "extensions" section. The bottom of the screen shows the terminal output, which includes a welcome message for Codespaces and instructions for using the environment.

```
1  "image": "mcr.microsoft.com/devcontainers/universal:2",
2  "hostRequirements": {
3      "cpus": 4
4  },
5  "waitFor": "onCreateCommand",
6  "updateContentCommand": "python3 -m pip install -r requirements.txt",
7  "postCreateCommand": "",
8  "customizations": {
9      "codeSpaces": {
10          "openFiles": []
11      },
12      "vscode": {
13          "extensions": [
14              "ms-toolsai.jupyter",
15              "ms-python.python"
16          ]
17      }
18  },
19  "volumes": [
20      {
21          "containerPath": "/home/codespace/.local/share/jupyter",
22          "hostPath": "/home/codespace/.local/share/jupyter"
23      }
24  ],
25  "networking": {
26      "networkMode": "host"
27  }
28 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 3

Welcome to Codespaces! You are on our default image.
- It includes runtimes and tools for Python, Node.js, Docker, and more. See the full list here: <https://aka.ms/ghcs-default-image>
- Want to use a custom image instead? Learn more here: <https://aka.ms/configure-codespace>

To explore VS Code to its fullest, search using the Command Palette (Cmd/Ctrl + Shift + P or F1).

Edit away, run your app as usual, and we'll automatically make it available for you to access.

When using "conda" from outside of GitHub Codespaces, note the Anaconda repository contains restrictions on commercial use that may impact certain organizations. See <https://aka.ms/ghcs-conda>

codespace → /workspaces/pydata-workshop-2023 (main) \$

Ln 19, Col 4 Spaces: 2 UTF-8 LF {} JSON with Comments

Exercise 3: Save config, then open chat window



You get the 'chat' icon in your sidebar, with a slide-out chat window ...

And your config file is updated (rebuild now!)

```
(1) devcontainer.json M
1  {
2    "image": "mcr.microsoft.com/devcontainers/universal:2",
3    "hostRequirements": {
4      "cpus": 4
5    },
6    "waitFor": "onCreateCommand",
7    "updateContentCommand": "python3 -m pip install -r requirements.txt",
8    "postCreateCommand": "",
9    "customizations": {
10      "codespaces": {
11        "openFiles": []
12      },
13      "vscode": {
14        "extensions": [
15          "ms-toolsai.jupyter",
16          "ms-python.python",
17          "GitHub.copilot-chat"
18        ]
19      }
20    }
21  }
```

The screenshot shows the VS Code interface with the GitHub Copilot Chat extension installed. The sidebar features a 'CHAT: GITHUB COPILOT' icon. A slide-out chat window from 'GitHub Copilot' is visible, displaying a welcome message and information about its AI-powered capabilities. The main workspace shows the 'devcontainer.json' configuration file with the 'GitHub.copilot-chat' extension listed under 'customizations'. The bottom right corner shows the GitHub Copilot Chat extension settings page, which includes categories like Programming Languages, Snippets, Machine Learning, and Education.

CHAT: GITHUB COPILOT

GitHub Copilot

Welcome @nitya, I'm your Copilot and I'm here to help you get things done faster.

I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and [share feedback](#) so that we can learn and improve. Check out the [Copilot documentation](#) to learn more.

devcontainer.json

Extension: GitHub Copilot Chat

GitHub Copilot Chat v0.10.0 Preview

GitHub [github.com](#) | ⚡ 3,968,058 | ★★★★☆ (74)

AI chat features powered by Copilot

Disable ⚡ Uninstall ⚡ ⚡

Extension is enabled on 'Dev Container @ desktop-linux'

DETAILS FEATURE CONTRIBUTIONS EXTENSION PACK RUNTIME STATUS

GitHub Copilot Chat

GitHub Copilot Chat is a companion extension to [GitHub Copilot](#) that houses preview Chat features:

- **Chat view:** Ask Copilot for help with any task or question in the Chat view, bringing in code from your current files.
- **Inline Chat:** Apply Copilot's AI suggestions directly to your code, staying

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 3

Welcome to Codespaces! You are on our default image.
- It includes runtimes and tools for Python, Node.js, Docker, and more. See the full list here: <https://aka.ms/ghcs-default-image>
- Want to use a custom image instead? Learn more here: <https://aka.ms/configure-codespace>

To explore VS Code to its fullest, search using the Command Palette (Cmd/Ctrl + Shift + P or F1).

Edit away, run your app as usual, and we'll automatically make it available for you to access.

When using "conda" from outside of GitHub Codespaces, note the Anaconda repository contains restrictions on commercial use that may impact certain organizations. See <https://aka.ms/ghcs-conda>

codespaces → /workspaces/pydata-workshop-2023 (main) \$

Categories

- Programming Languages
- Snippets
- Machine Learning
- Education

Under The Hood: What's GitHub Copilot (Chat)?

- Code explanations
- Code assistance
- Code refinement
- Unit testing
- Code profiling
- Code debugging

Ask questions in natural language – get responses inline, and in context!

About GitHub Copilot

GitHub Copilot is an AI pair programmer that offers autocomplete-style suggestions as you code. You can receive suggestions from GitHub Copilot either by starting to write the code you want to use, or by writing a natural language comment describing what you want the code to do. GitHub Copilot analyzes the context in the file you are editing, as well as related files, and offers suggestions from within your text editor. GitHub Copilot is powered by a generative AI model developed by GitHub, OpenAI, and Microsoft.

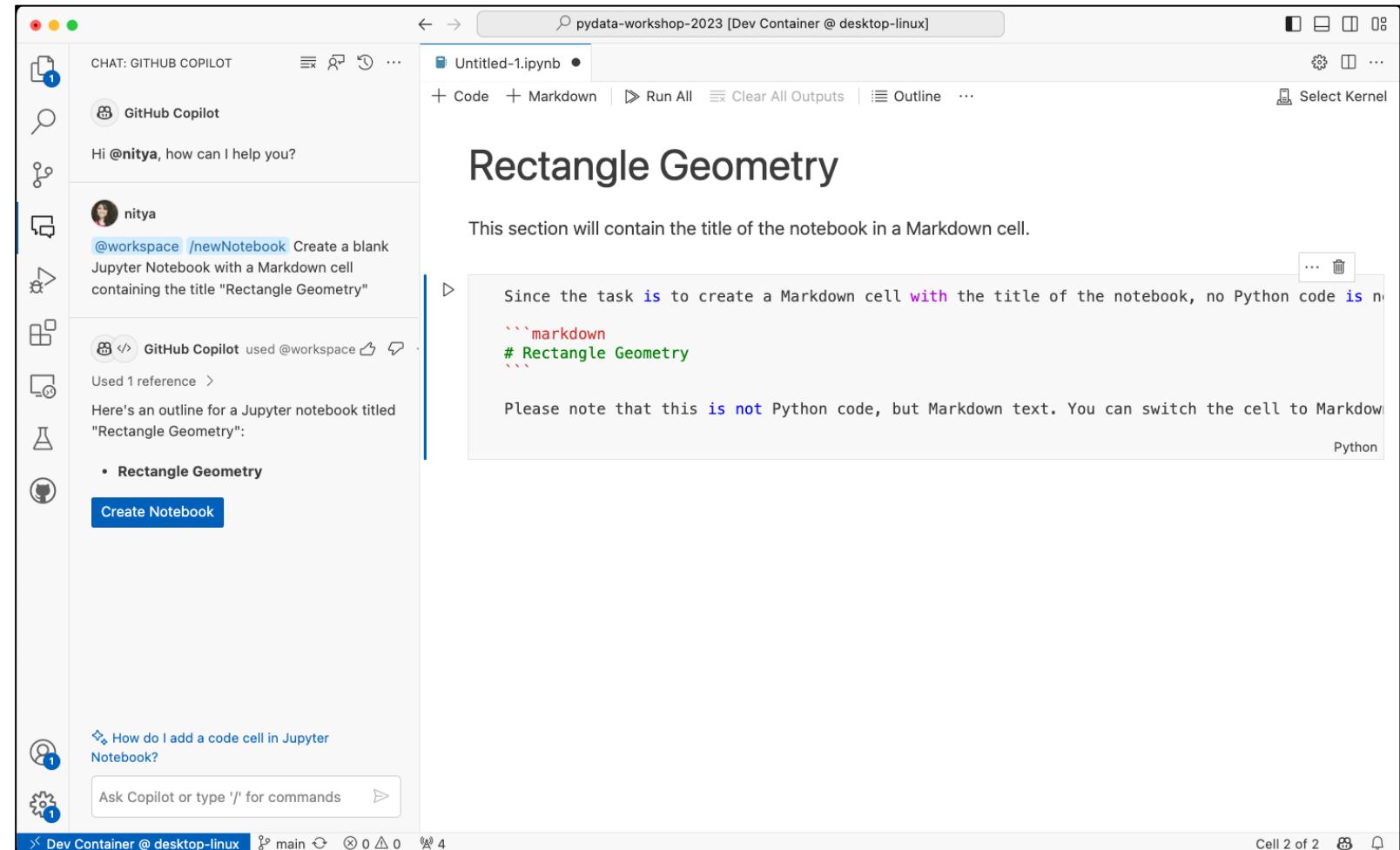
GitHub Copilot is trained on all languages that appear in public repositories. For each language, the quality of suggestions you receive may depend on the volume and diversity of training data for that language. For example, JavaScript is well-represented in public repositories and is one of GitHub Copilot's best supported languages. Languages with less representation in public repositories may produce fewer or less robust suggestions.

GitHub Copilot is available as an extension in Visual Studio Code, Visual Studio, Vim, Neovim, the JetBrains suite of IDEs, and Azure Data Studio. For more information on using GitHub Copilot in Visual Studio Code, Visual Studio, Vim, Neovim, and JetBrains, see "[Getting started with GitHub Copilot](#)." For more information on using GitHub Copilot in Azure Data Studio, see [GitHub Copilot extension](#) in Microsoft Learn.

Exercise 4: Have it create a `/newNotebook` for me

Copilot Chat provides a chat experience that uses the current window and history as context.

It's not perfect – see the Python cell – but it's a quick start to my task.



Exercise 4: Let's setup the notebook with our problem

I have a USACO problem called "[Blocked Billboard](#)" that has a [published solution](#).

Let's first add the problem into our Notebook as Markdown. This becomes a workspace where I can try coding my answers.

The screenshot shows a Jupyter Notebook interface with the following details:

- EXPLORER** panel on the left:
 - TIMELINE**: Shows activity: "edit" (1 min ago), "Insert Cell" (2 mins ago), "edit" (3 mins ago).
 - PYDATA-WORKSHOP-2023 [DEV ...]**:
 - .devcontainer
 - devcontainer.json
 - icon.svg
 - data
 - docs
 - notebooks
 - pydata**:
 - 01-newNotebook.ipynb
 - billboard.in
 - billboard.out
 - .gitignore
 - LICENSE
 - README.md
 - requirements.txt
- Code Editor** panel on the right:
 - Title Bar**: 01-newNotebook.ipynb U, billboard.in U, Python 3.10.8
 - Toolbar**: Code, Markdown, Run All, Restart, Clear All Outputs, Variables, Outline, ...
 - Section Header**: Rectangle Geometry
 - Description**: Let's look at the [Blocked Billboard](#) problem from USACO 2017 December Contest, Bronze
 - Text Content**: During long milking sessions, Bessie the cow likes to stare out the window of her barn at two huge rectangular billboards across the street advertising "Farmer Alex's Amazingly Appetizing Alfalfa" and "Farmer Greg's Great Grain". Pictures of these two cow feed products on the billboards look much tastier to Bessie than the grass from her farm. One day, as Bessie is staring out the window, she is alarmed to see a huge rectangular truck parking across the street. The side of the truck has an advertisement for "Farmer Smith's Superb Steaks", which Bessie doesn't quite understand, but she is mostly concerned about the truck potentially blocking the view of her two favorite billboards.
 - Text Content**: Given the locations of the two billboards and the location of the truck, please calculate the total combined area of both billboards that is still visible. It is possible that the truck obscures neither, both, or only one of the billboards.
 - Section Header**: INPUT FORMAT (file billboard.in):
 - Description**: The first line of input contains four space-separated integers: $x_1 \ y_1 \ x_2 \ y_2$, where (x_1, y_1) and (x_2, y_2) are the coordinates of the lower-left and upper-right corners of the first billboard in Bessie's 2D field of view.
 - Description**: The next line contains four more integers, similarly specifying the lower-left and upper-right corners of the second billboard. The third and final line of input contains four integers specifying the lower-left and upper-right corners of the truck. All coordinates are in the range -1000 to +1000. The two billboards are guaranteed not to have any positive area of overlap between themselves.
 - Bottom Navigation**: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS (5), JUPYTER, bash - pydata, ...
 - Bottom Status Bar**: Dev Container @ desktop-linux, main*, 0 △ 0, 5, Ln 3, Col 1 (24 selected), Spaces: 4 LF, Cell 5 of 8

Exercise 4: Copy solution – and run it (provide inputs)

Now we can use Copilot Chat to ask questions to help us understand the code and complexity

 nitya

what is time complexity mean?

 GitHub Copilot ◀ ▶ ⋮

Used 1 reference ▾

 01-newNotebook.ipynb:1-29 pydata

Time complexity is a concept in computer science that describes the amount of time an algorithm takes to run as a function of the size of the input to the program. It is usually expressed using Big O notation, which describes the upper bound of the time complexity in the worst-case scenario. For example, a linear search algorithm has a time complexity of $O(n)$ because the time it takes to find an item (or determine it's not there) grows linearly with the size of the list being searched.

The screenshot shows a Jupyter Notebook interface with the following details:

- EXPLORER** sidebar:
 - TIMELINE**: 01-newNotebook.ipynb
 - edit (now)
 - edit (34 secs)
 - edit (1 min)
 - Insert Cell
 - edit
- PYDATA-WORKSHOP-2023 [DEV CONTAINER @ DE...]** folder:
 - .devcontainer
 - devcontainer.json
 - icon.svg
 - > data
 - > docs
 - > notebooks
 - pydata**
 - 01-newNotebook.ipynb
 - billboard.in
 - billboard.out
 - .gitignore
 - LICENSE
 - README.md
 - requirements.txt
- Code Editor**: A Python script titled "01-newNotebook.ipynb". The code defines a function `main()` that reads input coordinates from "billboard.in", processes them to find visible points, and prints the count to "billboard.out".

```
import sys
MAX_POS = 2000

def main():
    sys.stdin = open("billboard.in", "r")
    sys.stdout = open("billboard.out", "w")

    visible = [[False for _ in range(MAX_POS)] for _ in range(MAX_POS)]

    for i in range(3):
        x1, y1, x2, y2 = map(int, input().split())
        x1 += MAX_POS // 2
        y1 += MAX_POS // 2
        x2 += MAX_POS // 2
        y2 += MAX_POS // 2
        for x in range(x1, x2):
            for y in range(y1, y2):
                visible[x][y] = i < 2

    ans = 0
    for x in range(MAX_POS):
        for y in range(MAX_POS):
            ans += visible[x][y]
    print(ans)

main()
```
- Terminal**: bash - pydata
- Bottom Status Bar**: Dev Container @ desktop-linux, main*, 0 0 △ 0, Ln 6, Col 12, Tab Size: 4, LF, Cell 9 of 10

Exercise 4: Copy solution – and run it (provide inputs)

Notice how I stay in flow
(no external visits) – and
am prompted to learn
more in context ...

- explain this code to me
- how can I improve the solution
- can you simplify it using a class

The “[Fast Solution](#)” from USACO agrees on O(1) but uses classes. I can ask about that too..

The image shows three separate screenshots of a GitHub Copilot conversation window. Each screenshot has a user profile picture of a woman named 'nitya' on the left and a GitHub Copilot AI icon on the right.

- Screenshot 1:** nitya asks "explain this code to me". GitHub Copilot responds with a file reference "Used 1 reference" to "01-newNotebook.ipynb:1-29 pydata". It then explains that the Python script reads input from "billboard.in" and writes to "billboard.out". It provides a step-by-step explanation of initializing a 2D boolean array "visible" and adjusting coordinates by adding `MAX_POS // 2`. At the bottom, there is a question "What is the purpose of the MAX_POS constant in the billboard code?" with a blue diamond icon.
- Screenshot 2:** nitya asks "What is the purpose of the MAX_POS constant in the billboard code?". GitHub Copilot responds by explaining that `MAX_POS` defines the size of the 2D `visible` array and represents a grid of points. It also notes that `MAX_POS` is used to adjust coordinates to ensure they are non-negative.
- Screenshot 3:** nitya asks "how can I improve this solution?". GitHub Copilot replies that the current solution is inefficient due to nested loops over a large `visible` array. Instead, it suggests calculating the area of rectangles directly, which has a time complexity of $O(1)$. It then provides an improved code snippet:

```
def main():
    sys.stdin = open("billboard.in",
    sys.stdout = open("billboard.out"
    read(f.readline())
    re
    to
```

nitya then asks "can you simplify this by using a class". GitHub Copilot responds that yes, the code can be simplified using a class. It provides a sample class definition:

```
class Billboard:
    def __init__(self, x1, y1, x2, y2)
```

Exercise 4: I can also explore this more intuitively



This time let's use Copilot inline, so it generates code for us

We don't know "how" to visualize it – but since the goal was to explore our intuition visually (vs. code the solution), the end result really helps!

Can You visualize the billboards and the truck for me

Accept Discard ⚡ Changed 20 lines

```
1 import matplotlib.pyplot as plt
2
3 # billboard 1 coordinates
4 x1, y1, x2, y2 = 1, 2, 3, 5
5
6 # billboard 2 coordinates
7 x3, y3, x4, y4 = 6, 0, 10, 4
8
9 # truck coordinates
10 x5, y5, x6, y6 = 2, 1, 8, 3
11
12 # plot the billboards and the truck
13 fig, ax = plt.subplots()
14 ax.add_patch(plt.Rectangle((x1, y1), x2-x1,
15 ax.add_patch(plt.Rectangle((x3, y3), x4-x3,
16 ax.add_patch(plt.Rectangle((x5, y5), x6-x5,
17 plt.xlim(-10, 20)
18 plt.ylim(-10, 20)
19 plt.show()
```

A scatter plot showing three rectangles representing billboards and a truck. Billboard 1 (blue) is at (1, 2) with dimensions 2x3. Billboard 2 (green) is at (6, 0) with dimensions 4x4. The truck (red) is at (2, 1) with dimensions 6x2. All objects are plotted on a coordinate system from -10 to 20 on both axes.



Part 3:

Explore Data Visualization the traditional way

(I ask what I know about)

Exercise 5: Get a dataset to explore (e.g., from Kaggle)

ARAVIND A S - UPDATED A YEAR AGO

70 New Notebook Download (4 kB) ...

IPL 2022-Dataset

Matchwise data of Tata IPL 2022 (IPL Season 15)

Data Card Code (6) Discussion (0)

About Dataset

Complete Match-wise data of IPL 2022

MAKE AN EDA with this dataset

Content

This dataset contains Matchwise data of IPL matches 2022 (March26 - May29 2022) ,The complete data of group stage matches.

Attribute Information

1. Match Number
2. Date of the match
3. Name of Venue
4. Playing team 1
5. Playing team 2
6. Stage of the tournament

Usability ① 10.00

License Database: Open Database, Cont...

Expected update frequency Daily

Tags Tabular Sports Beginner Cricket pandas



ASHISH JANGRA - UPDATED 2 YEARS AGO

258 New Notebook Download (305 kB) ...

TED Talks

This datasets is having all the ted talks available on Ted.com

Data Card Code (58) Discussion (1)

About Dataset

Context

This dataset is created for beginner students of Data Analysis who can explore the field with real-life data. Using TED talk data will help them to analyze the talks and they can also watch the talks of their favorite author with the help of the dataset as well.

Content

This dataset contains 6 different features of each talk available on TED's website which you can find below

- title - Title of the Talk
- author - Author of Talk
- date - Date when the talk took place
- views - Number of views of the Talk
- likes - Number of likes of the Talk
- link - Link of the talk from ted.com

Acknowledgements

Usability ① 10.00

License CC BY-NC-SA 4.0

Expected update frequency Quarterly

Tags Earth and Nature Education Beginner Data Analytics Data Visualization Exploratory Data Analysis Python



[IPL2022 Dataset](#) - shared with [Open Data Commons License](#)
↗ downloaded Oct 2023 · [Example EDA](#) on Kaggle.

[TED Talks Dataset](#) - shared with [Creative Commons License](#)
↗ downloaded Oct 2023 · [Example EDA](#) on Kaggle.

Exercise 5: Can I have it teach me to visualize it?



nitya

@workspace /newNotebook Create a notebook that explains how to use the seaborn package for data visualizations - use the data/kaggle/IPL-2022.csv file as your input example

GitHub Copilot used @workspace

Used 1 reference >

Here's an outline for a Jupyter notebook that explains how to use the seaborn package for data visualizations using the IPL-2022.csv file:

- Import Required Libraries
- Load the IPL-2022.csv Dataset
- Data Overview
- Visualize the Distribution of Runs
- Visualize the Relationship between Variables
- Visualize Categorical Data

[Create Notebook](#)

Use pandas to load the IPL-2022.csv dataset.

```
1
2 # Load the IPL-2022.csv dataset using pandas
3 ipl_data = pd.read_csv('../data/kaggle/IPL-2022.csv')
4
5 # Display the first 5 rows of the dataset
6 ipl_data.head()
7
```

3] ✓ 0.0s Python

	match_id	date	venue	team1	team2	stage	toss_winner	toss_decision	first_ings_s
0	1	March 26,2022	Wankhede Stadium, Mumbai	Chennai	Kolkata	Group	Kolkata	Field	
1	2	March 27,2022	Brabourne Stadium, Mumbai	Delhi	Mumbai	Group	Delhi	Field	
2	3	March 27,2022	Dr DY Patil Sports Academy, Mumbai	Banglore	Punjab	Group	Punjab	Field	
3	4	March 28,2022	Wankhede Stadium, Mumbai	Gujarat	Lucknow	Group	Gujarat	Field	
4	5	March 29,2022	Maharashtra Cricket Association Stadium,Pune	Hyderabad	Rajasthan	Group	Hyderabad	Field	

Exercise 5: Can I have it fix mistakes it made?

nitya

@workspace /newNotebook Create a notebook that explains how to use the seaborn package for data visualizations - use the data/kaggle/IPL-2022.csv file as your input example

GitHub Copilot used @workspace    

Used 1 reference >

Here's an outline for a Jupyter notebook that explains how to use the seaborn package for data visualizations using the IPL-2022.csv file:

- Import Required Libraries
- Load the IPL-2022.csv Dataset
- Data Overview
- Visualize the Distribution of Runs
- Visualize the Relationship between Variables
- Visualize Categorical Data

[Create Notebook](#)

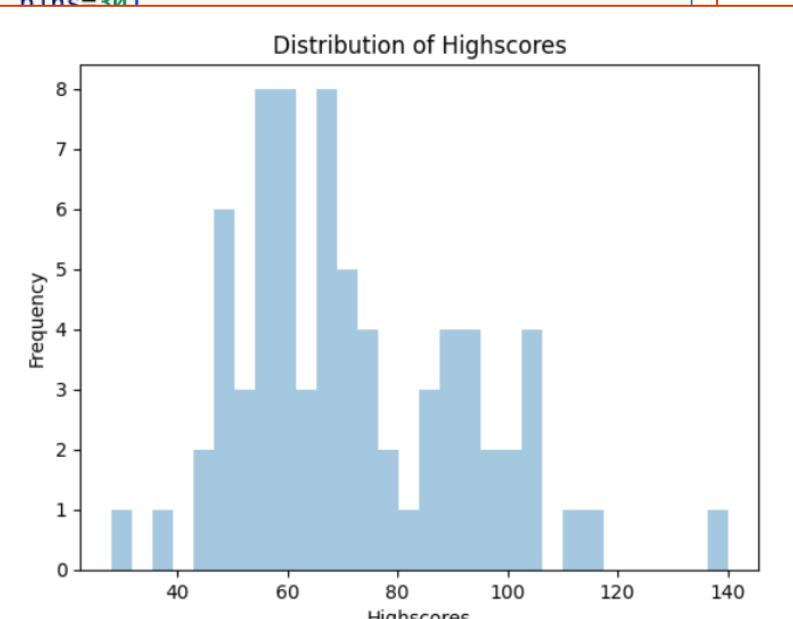
fix this to use highscore column for runs 

Copilot generated code may be incorrect

```
1 # Visualize the Distribution of Runs
2
3 # Check if the 'runs' column exists in the dataset
4 if 'runs' in ipl_data.columns:
5     # Use seaborn's distplot() function to visualize the distribution of runs in the data
6     sns.distplot(ipl_data['runs'], kde=False, bins=20)
7
8     # Set the title for the plot
9     plt.title('Distribution of Runs')
10
11    # Set the x-label for the plot
12    plt.xlabel('Runs')
13
14    # Set the y-label for the plot
15    plt.ylabel('Frequency')
16
17    # Display the plot
18    plt.show()
19
20 else:
21     print("'runs' column does not exist in the dataset.")
```

[5] 0.0s

... 'runs' column does not exist in the dataset.



Highscore Range	Frequency
35-40	1
40-45	1
45-50	2
50-55	6
55-60	8
60-65	8
65-70	5
70-75	4
75-80	3
80-85	2
85-90	3
90-95	4
95-100	2
100-105	4
105-110	1
110-115	1
115-120	0
120-125	0
125-130	0
130-135	0
135-140	1

Exercise 5: Can it visualize something specific?

nitya

@workspace /newNotebook Create a notebook that explains how to use the seaborn package for data visualizations - use the data/kaggle/IPL-2022.csv file as your input example

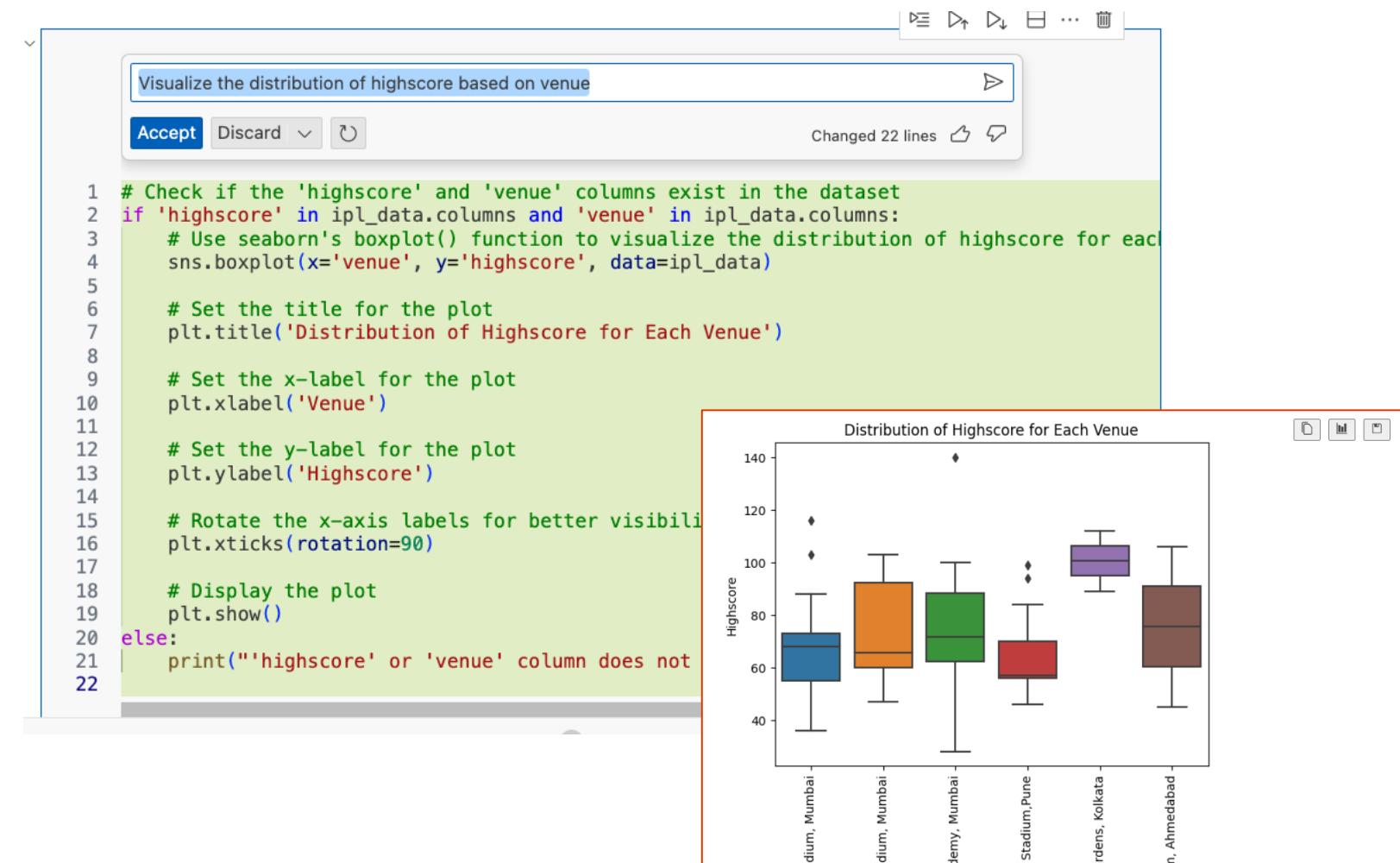
GitHub Copilot used @workspace

Used 1 reference >

Here's an outline for a Jupyter notebook that explains how to use the seaborn package for data visualizations using the IPL-2022.csv file:

- Import Required Libraries
- Load the IPL-2022.csv Dataset
- Data Overview
- Visualize the Distribution of Runs
- Visualize the Relationship between Variables
- Visualize Categorical Data

Create Notebook



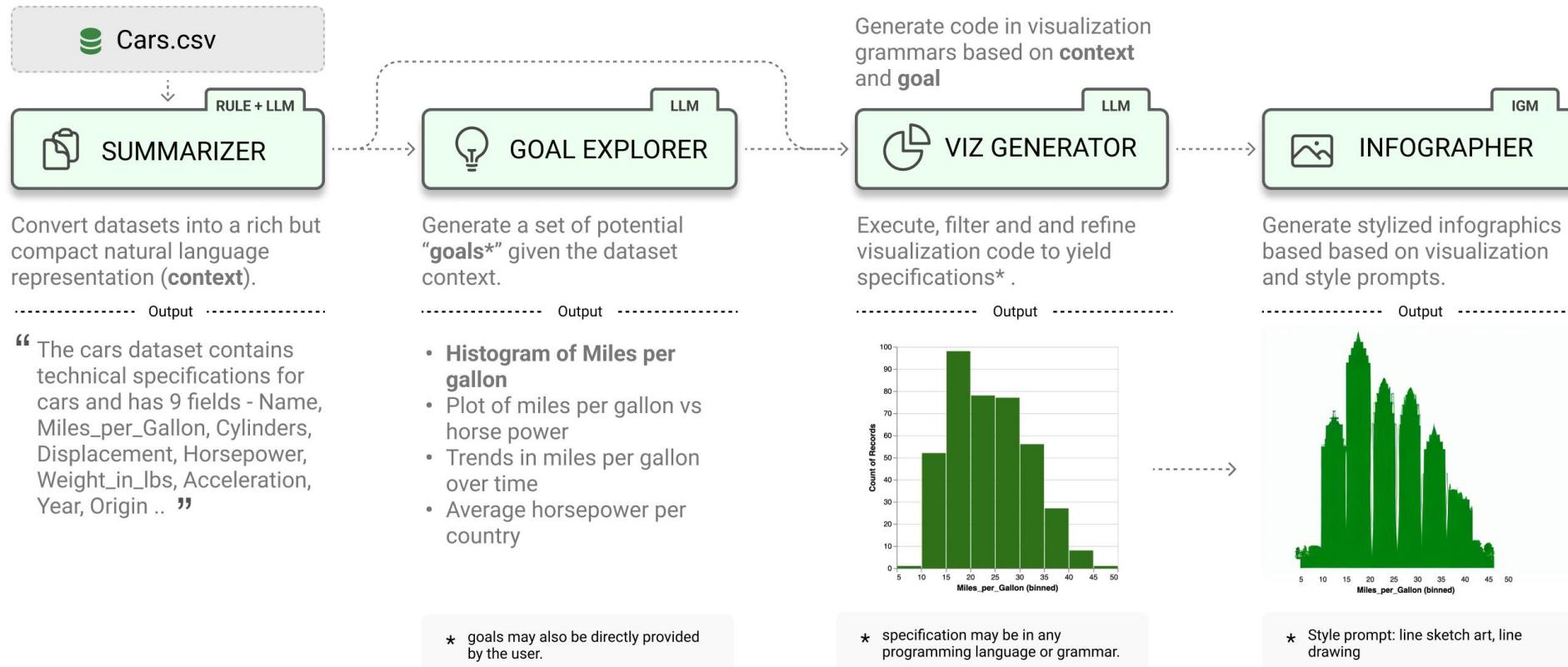


Part 4:

Explore Data Visualization without intuition (Using Open AI – with Microsoft LIDA)

LIDA: Generate Data Visualizations & Infographics

LIDA is a library for generating data visualizations and data-faithful infographics. LIDA is grammar agnostic (will work with any programming language and visualization libraries e.g. matplotlib, seaborn, altair, d3 etc) and works with multiple large language model providers (OpenAI, Azure OpenAI, PaLM, Cohere, Huggingface). Research Paper: <https://arxiv.org/abs/2303.02927>



Open AI API Key: Set it as env var (protect it as secret)



```
# Setup
from lida import Manager, TextGenerationConfig , llm

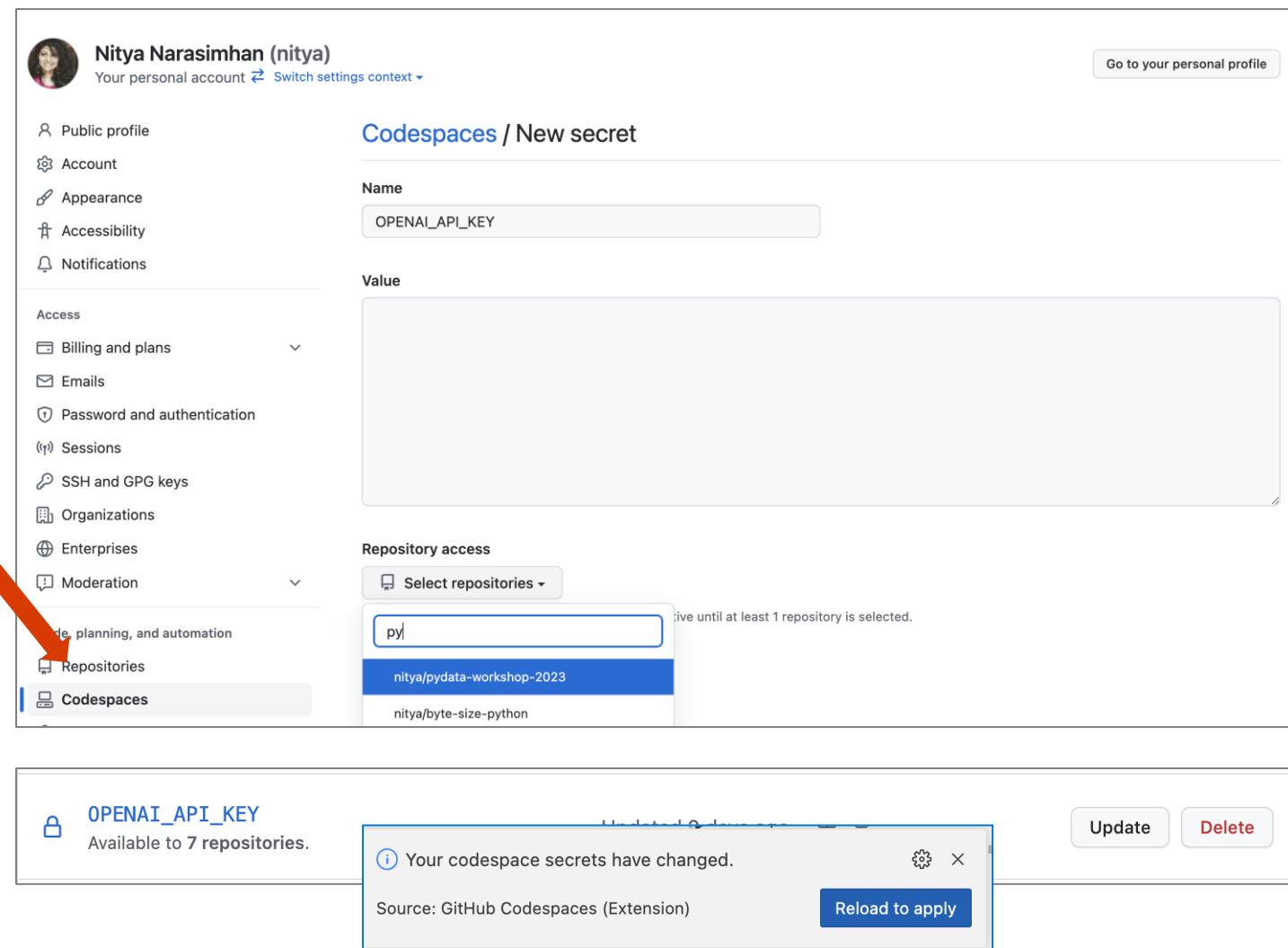
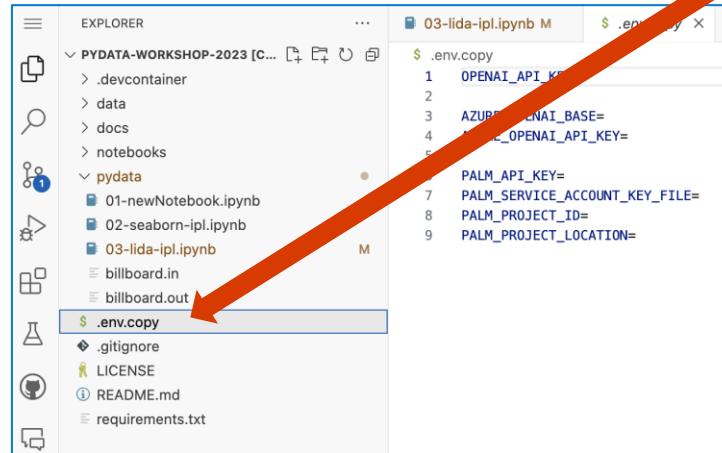
csvfile = "./../data/kaggle/IPL-2022.csv"
lida = Manager(text_gen = llm("openai")) # palm, cohere .
textgen_config = TextGenerationConfig(n=1, temperature=0.5, model="gpt-3.5-turbo-0301", use_cache=True)

[2] 0.0s

ValueError
Traceback (most recent call last)
File "/workspaces/pydata-workshop-2023/pydata/03-lida-ipynb Cell 3 line 5
  2 from lida import Manager, TextGenerationConfig , llm
  3 csvfile = "./../data/kaggle/IPL-2022.csv"
  4 lida = Manager(text_gen = llm("openai")) # palm, cohere .
  5 textgen_config = TextGenerationConfig(n=1, temperature=0.5, model="gpt-3.5-turbo-0301", use_cache=True)

File "/usr/lib/python3.10/site-packages/llms/generators/text/textgen.py:26, in llm(provider, **kwargs)
   23     api_key = api_key or os.environ.get("OPENAI_API_KEY", None)
   24     if api_key is None:
   25         raise ValueError(
   26             "OpenAI API key is not set. Please set the OPENAI_API_KEY environment variable."
   27         )
   28     openai.api_key = api_key
   29
   30 if organization:

ValueError: OpenAI API key is not set. Please set the OPENAI_API_KEY environment variable.
```



The screenshot shows the GitHub Codespaces 'New secret' configuration page for 'Nitya Narasimhan (nitya)'. The 'Name' field is set to 'OPENAI_API_KEY' and the 'Value' field is empty. The 'Access' section includes options for Billing and plans, Emails, Password and authentication, Sessions, SSH and GPG keys, Organizations, Enterprises, and Moderation. A dropdown menu for 'Select repositories' shows 'py' selected, with 'nitya/pydata-workshop-2023' highlighted. A modal at the bottom right indicates that 'Your codespace secrets have changed.' and provides 'Update' and 'Delete' buttons, along with a 'Reload to apply' button.

Exercise 6: Data Summarization with LIDA (foundation)

The screenshot shows a Jupyter Notebook interface within a GitHub repository. The repository name is 'pydata-workshop-2023'. The notebook is titled 'Step 4. Visualize Data Using Traditional Means'.

The code in the notebook is as follows:

```
# Setup
from lida import Manager, TextGenerationConfig , llm

csvfile = "./../data/kaggle/IPL-2022.csv"
lida = Manager(text_gen = llm("openai")) # palm, cohere .
textgen_config = TextGenerationConfig(n=1, temperature=0.5, model="gpt-3.5-turbo-0301", use_cache=True)

[1]    ✓ 1.0s                                         Python
```

```
# Summarize
summary = lida.summarize(csvfile)
summary_data = list(summary.keys())
for keys in summary_data:
    print(keys, ":", summary[keys])

[2]    ✓ 0.0s                                         Python
```

```
... name : IPL-2022.csv
file_name : IPL-2022.csv
dataset_description :
fields : [{"column': 'match_id', 'properties': {'dtype': 'number', 'std': 21, 'min': 1, 'max': 74, 'samples': [5, 64, 19], 'num_unfield_names : ['match_id', 'date', 'venue', 'team1', 'team2', 'stage', 'toss_winner', 'toss_decision', 'first_ings_score', 'first_
```

Given a dataset, generate a compact summary of that data in a compact natural language representation that serves as context for subsequent tasks. The goal of the summarizer is to produce an dense-but-compact information summary for a given dataset that is useful as grounding context for visualization tasks. The grounding context is defined as one that contains information an analyst would need to understand the dataset and the tasks that can be performed on it.

See paper for details

Exercise 6: Goal Generation with LIDA (build intuition)

```
▶ 
1 # generate 10 goals from the summary with default persona
2 goals = lida.goals(summary, n=10, textgen_config=textgen_config,) # exploratory data analysis
3
4 # create a list of dictionaries containing the goal information
5 import pandas as pd
6 goal_list = []
7 for goal in goals:
8     goal_dict = {'Question': goal.question, 'Visualization': goal.visualization, 'Rationale': goal.rationale}
9     goal_list.append(goal_dict)
10 df = pd.DataFrame(goal_list)
11 display(df)
12
```

[4] ✓ 16.1s Python

...	Question	Visualization	Rationale
0	What is the distribution of first innings scor...	Histogram of first_ings_score	This visualization will help us understand the...
1	What is the distribution of second innings sco...	Histogram of second_ings_score	This visualization will help us understand the...
2	What is the average margin of victory in the m...	Bar chart of margin	This visualization will help us understand the...
3	Which team has won the most matches in the tou...	Bar chart of match_winner	This visualization will help us understand whi...
4	What is the distribution of scores for the top...	Histogram of highscore	This visualization will help us understand the...
5	What is the distribution of best bowling figur...	Histogram of best_bowling_figure	This visualization will help us understand the...
6	What is the distribution of the number of wick...	Bar chart of first_ings_wkts	This visualization will help us understand the...
7	What is the distribution of the number of wick...	Bar chart of second_ings_wkts	This visualization will help us understand the...
8	What is the distribution of player of the matc...	Bar chart of player_of_the_match	This visualization will help us understand the...
9	What is the distribution of matches played in ...	Bar chart of venue	This visualization will help us understand the...

Exercise 6: Goal Generation with LIDA (add persona)

```
1 # generate 5 goals from the summary - with the persona is a fan of the Mumbai team
2 goals = lida.goals(summary, n=5, textgen_config=textgen_config, persona="fan of the Mumbai team who wants to see t
3
4 # create a list of dictionaries containing the goal information
5 import pandas as pd
6 goal_list = []
7 for goal in goals:
8     display(goal)
```

[13] ✓ 7.9s

Python

... Goal 0

Question: What is the performance of Mumbai team in the first innings compared to other teams?

Visualization: Bar chart of first_ings_score for Mumbai and other teams

Rationale: This visualization will help us understand how Mumbai team is performing in the first innings compared to other teams. It will give us an idea of whether Mumbai team is scoring more or less in the first innings compared to other teams.

... Goal 1

Question: What is the performance of Mumbai team in the second innings compared to other teams?

Visualization: Bar chart of second_ings_score for Mumbai and other teams

Rationale: This visualization will help us understand how Mumbai team is performing in the second innings compared to other teams. It will give us an idea of whether Mumbai team is scoring more or less in the second innings compared to other teams.

Exercise 6: Data Visualization Generation with LIDA

```
1 # Visualize A Goal
2 charts = lida.visualize(summary=summary, goal=goals[0]) # exploratory data analysis
3 print("Charts length:", len(charts))
4 charts[0]
[5] ✓ 5.5s
...
... Charts length: 1
...


What is the distribution of first innings scores across all matches?



A histogram showing the frequency distribution of first innings scores. The x-axis ranges from 80 to 220, and the y-axis (Frequency) ranges from 0.0 to 20.0. A blue normal distribution curve is overlaid on the histogram bars. A vertical red dashed line indicates the mean score is 171.12.


```

```
1 # Visualize it again - and specify a different library and textgen_config (change te
2 target = goals[2]
3 library = "seaborn"
4 textgen_config = TextGenerationConfig(n=1, temperature=0.2, use_cache=True)
5 charts = lida.visualize(summary=summary, goal=target,library=library,textgen_config=
6 charts[0]
[7] ✓ 4.2s
...


What is the average margin of victory in the matches?



A histogram showing the distribution of margins of victory. The x-axis represents the margin values, and the y-axis (Margin) ranges from 0 to 80. A horizontal red dashed line indicates the mean margin is 16.97.


```

Exercise 6: Data Visualization Generation (user query)

```
1 # Use natural language user query instead of pre-formulated goal
2 user_query = "What is the frequency of toss decisions based on team ?"
3 textgen_config = TextGenerationConfig(n=1, temperature=0.2, use_cache=True)
4 charts = lida.visualize(summary=summary, goal=user_query, textgen_config=textgen_config)
5 charts[0]
```

[8] ✓ 5.4s Python

...

What is the frequency of toss decisions based on team ?

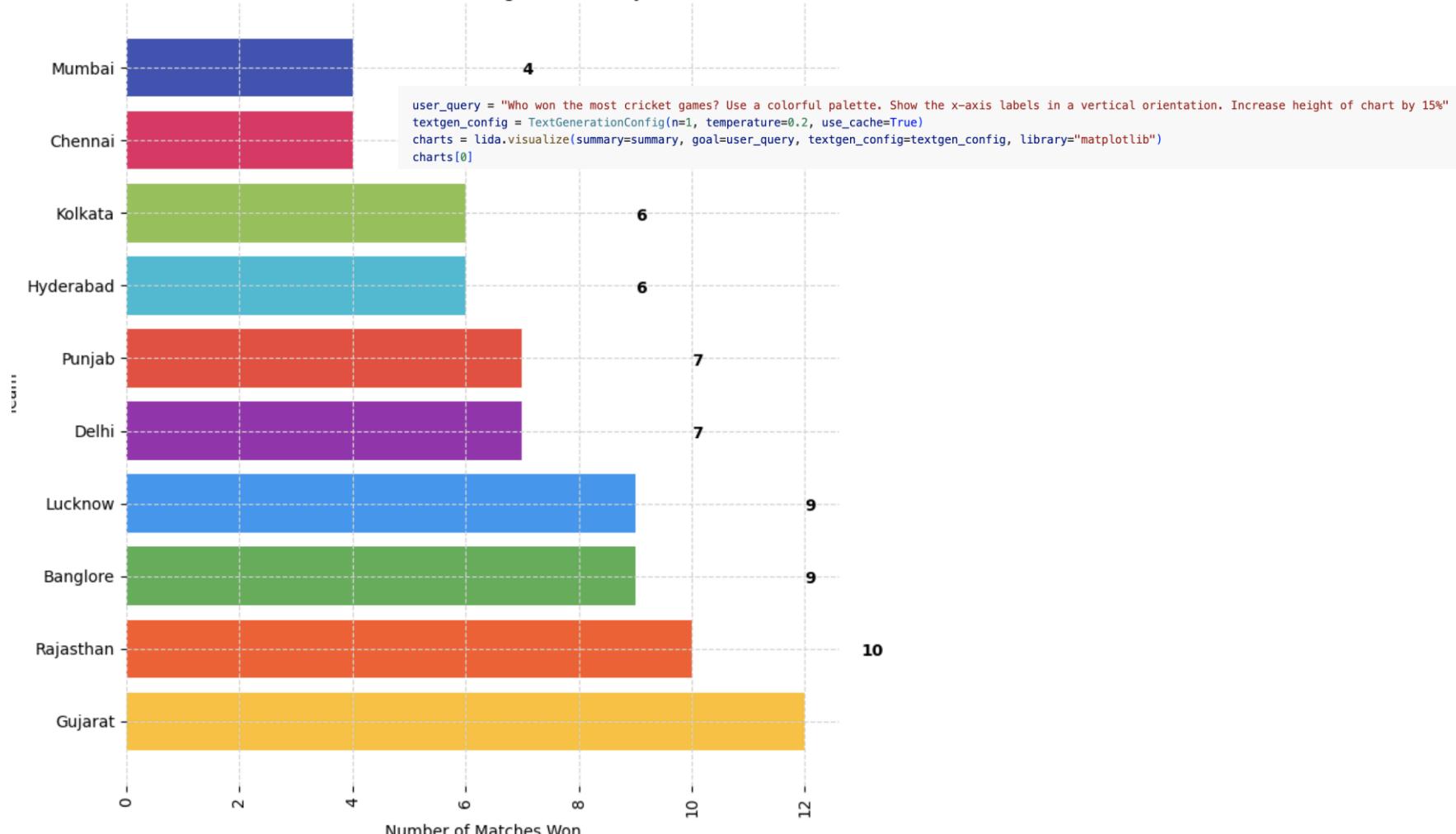
City	Bat	Field
Bangalore	3	13
Chennai	2	10
Delhi	1	11
Gujarat	3	7
Hyderabad	2	5
Kolkata	1	6
Lucknow	2	4
Mumbai	0	3
Punjab	1	0

GitHub Copilot
can do this too ..

But you can vary
the parameters
and customize the
base prompt here
programmatically

Exercise 6: Data Visualization Generation (user query)

Who won the most cricket games? Use a colorful palette. Show the x-axis labels in a vertical orientation. Increase height of chart by 15%



GitHub Copilot
can do this too ..

But you can vary
the parameters
and customize the
base prompt here
programmatically

Provides flexibility
for trial-and-error
experiments to
build intuition.

Exercise 6: Data Visualization Explanation with LIDA

```
▶ 1 # Explain visualization
  2 explanation = lida.explain(code=charts[0].code)
  3 for obj in explanation[0]:
  4     display(obj)
  5
[9] ✓ 9.8s
...
... {'section': 'accessibility',
  'code': 'None',
  'explanation': 'The goal of the chart is to show the frequency of toss decisions based on team. It uses a stacked bar chart with the x-axis representing the teams, the y-axis representing the frequency of toss decisions. The chart uses a white grid style and a Set2 color palette. The legend indicates two categories: "bat" and "field".'}
...
... {'section': 'transformation',
  'code': "data = data[['team1', 'toss_decision']]\\n\n    data = data.groupby(['team1', 'toss_decision']).size().reset_index(name='count')",
  'explanation': "The code filters the data to include only the 'team1' and 'toss_decision' columns. It then groups the data by 'team1' and 'toss_decision' and counts the frequency of each group. The resulting DataFrame has columns 'team1', 'toss_decision', and 'count'.}
...
... {'section': 'visualization',
  'code': "sns.set_style('whitegrid')\\nsns.set_palette('Set2')\\nax = sns.barplot(x='team1', y='count',
hue='toss_decision', data=data)\\nax.set(xlabel='Team', ylabel='Frequency')\\nplt.title('What is the frequency of toss
decisions based on team ?', wrap=True)\\nplt.legend(title='Toss Decision', loc='upper right')\\nplt.xticks(rotation=45,
ha='right')\\nreturn plt;",
  'explanation': "The code sets the style to white grid and the color palette to Set2. Then, it creates a bar plot using the seaborn library, with 'team1' on the x-axis, 'count' on the y-axis, and 'toss_decision' as the hue. The resulting chart is a stacked bar chart showing the frequency of toss decisions for each team. The x-axis labels are rotated 45 degrees for better readability. The chart has a title and a legend showing the two possible toss decisions: bat or field."}
```

system_prompt =

"""
You are a helpful assistant highly skilled in providing helpful, structured explanations of visualization of the plot(data: pd.DataFrame) method in the provided code. You divide the code into sections and provide a description of each section and an explanation. The first section should be named "accessibility" and describe the physical appearance of the chart (colors, chart type etc), the goal of the chart, as well the main insights from the chart.

You can explain code across the following 3 dimensions:

1. **accessibility:** the physical appearance of the chart (colors, chart type etc), the goal of the chart, as well the main insights from the chart.
2. **transformation:** This should describe the section of the code that applies any kind of data transformation (filtering, aggregation, grouping, null value handling etc)
3. **visualization:** step by step description of the code that creates or modifies the presented visualization.

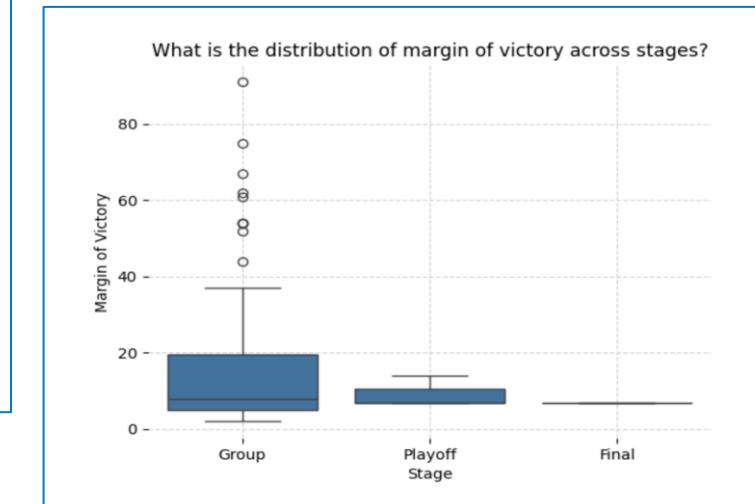
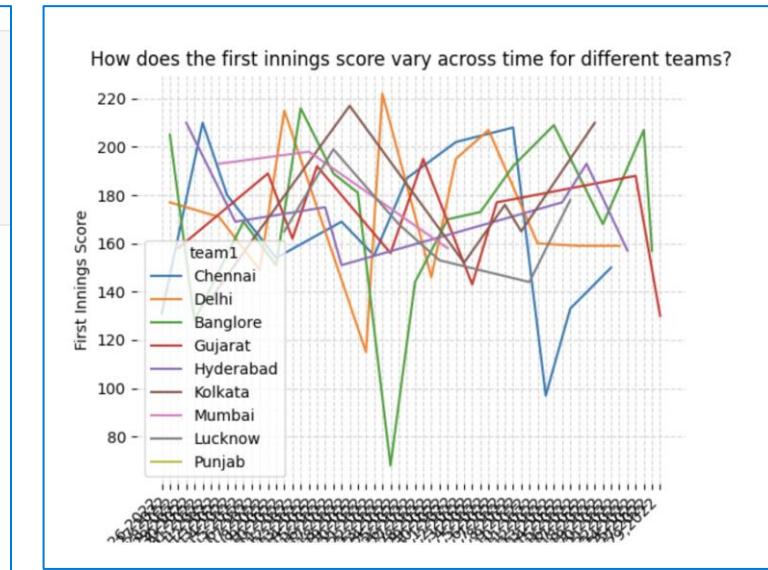
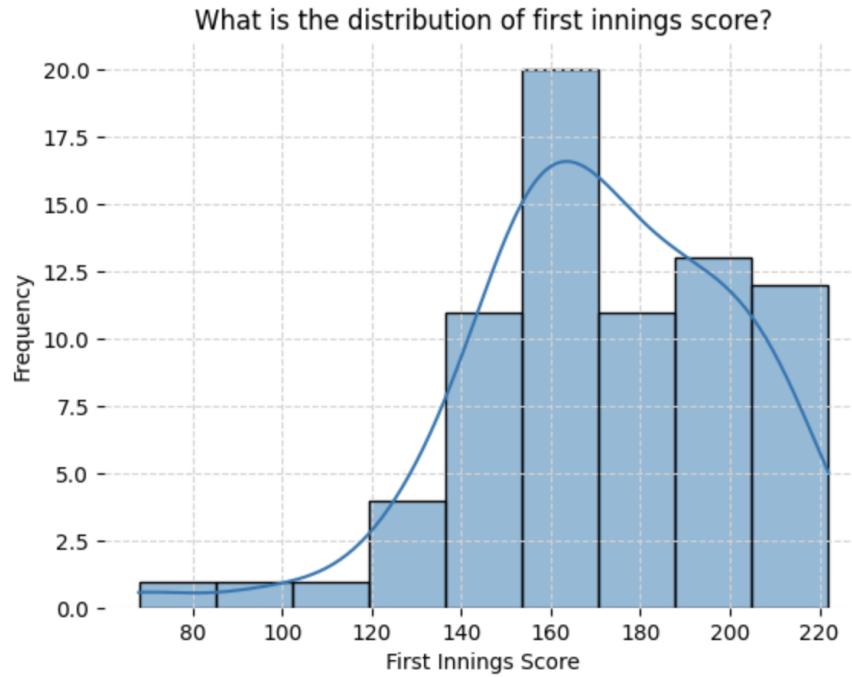
[lida/components/viz/vizexplainer.py](#)

Exercise 6: Visualization Recommendation (for a goal)

```
▷ ▾ # Recommend 3 visualizations
recommendations = lida.recommend(code=charts[0].code, summary=summary, n=3, textgen_config=textgen_config)

for chart in recommendations:
    display(chart)
[11] ✓ 5.3s
...

```

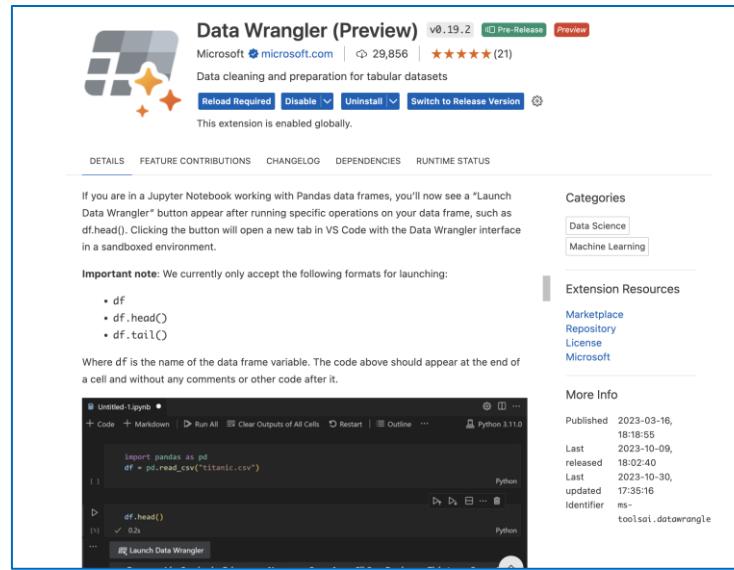




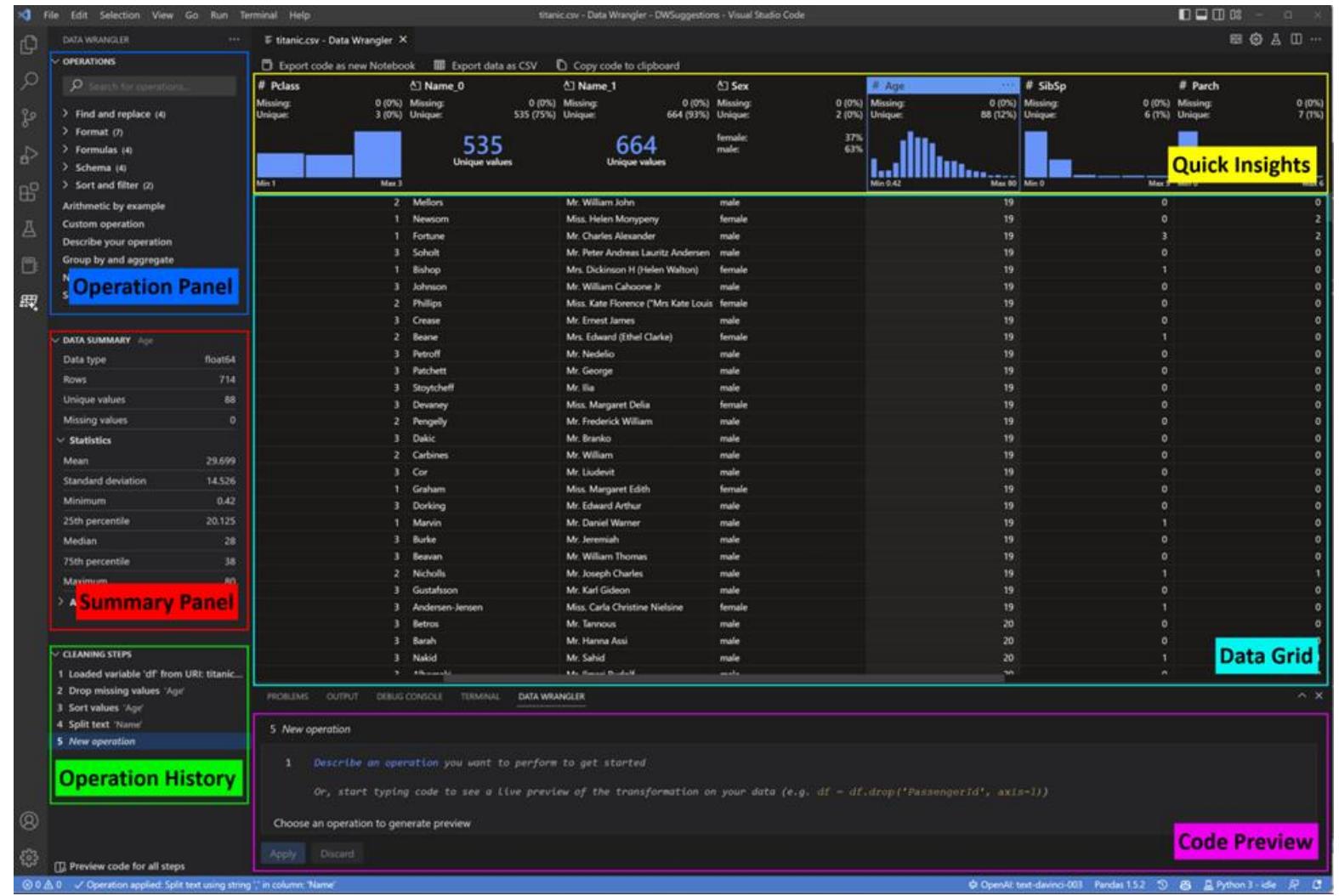
Wrap-Up: Where do we go from here?

Things To Try: Activate Data Wrangler (Preview)

<https://aka.ms/datawrangler>.



Data Wrangler is a code-centric data cleaning tool aims to increase the productivity by providing a rich UI that automatically generates Pandas code for insightful column statistics and visualizations.

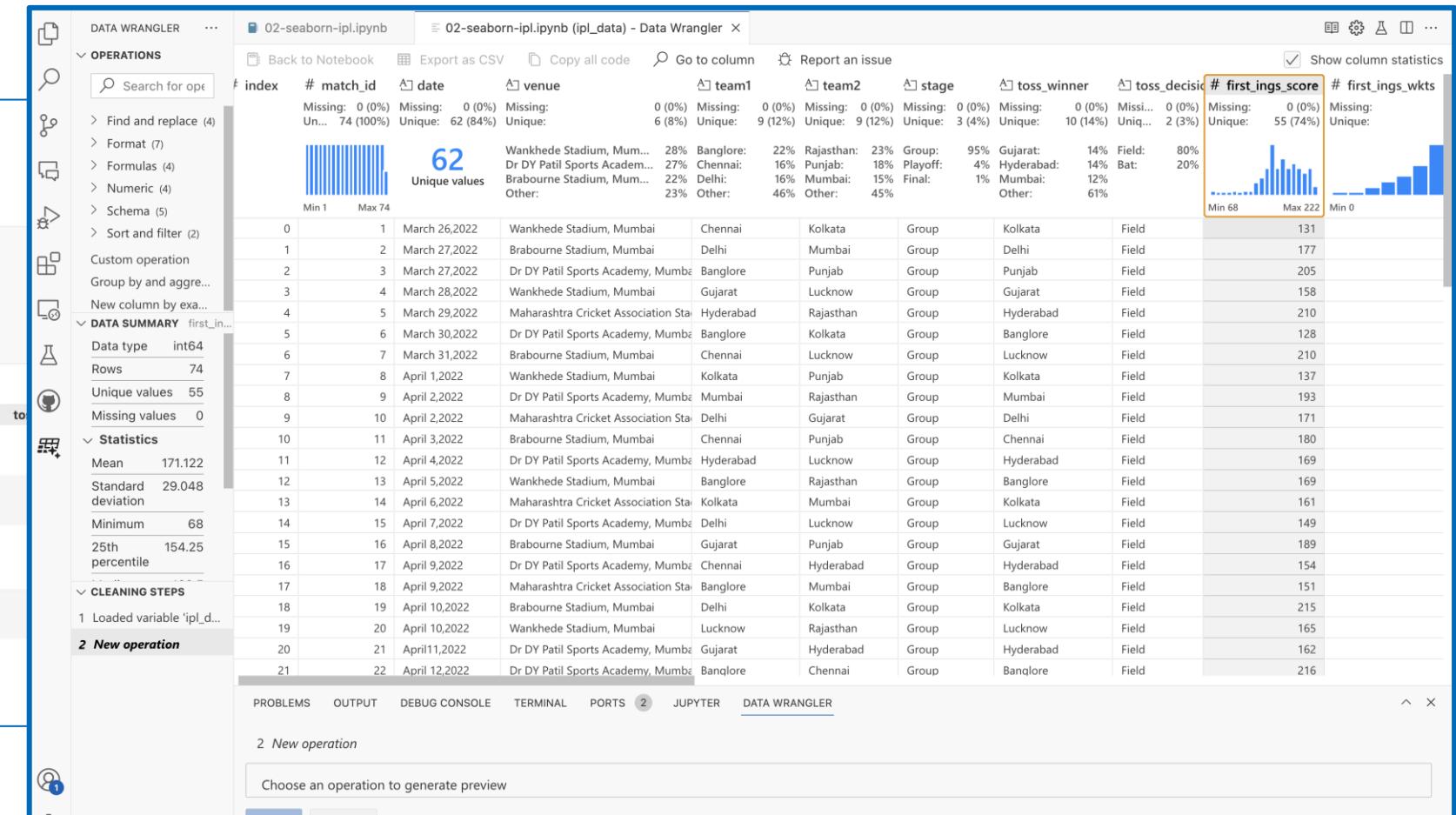


Things To Try: Activate Data Wrangler (Preview)

Load the IPL-2022.csv Dataset

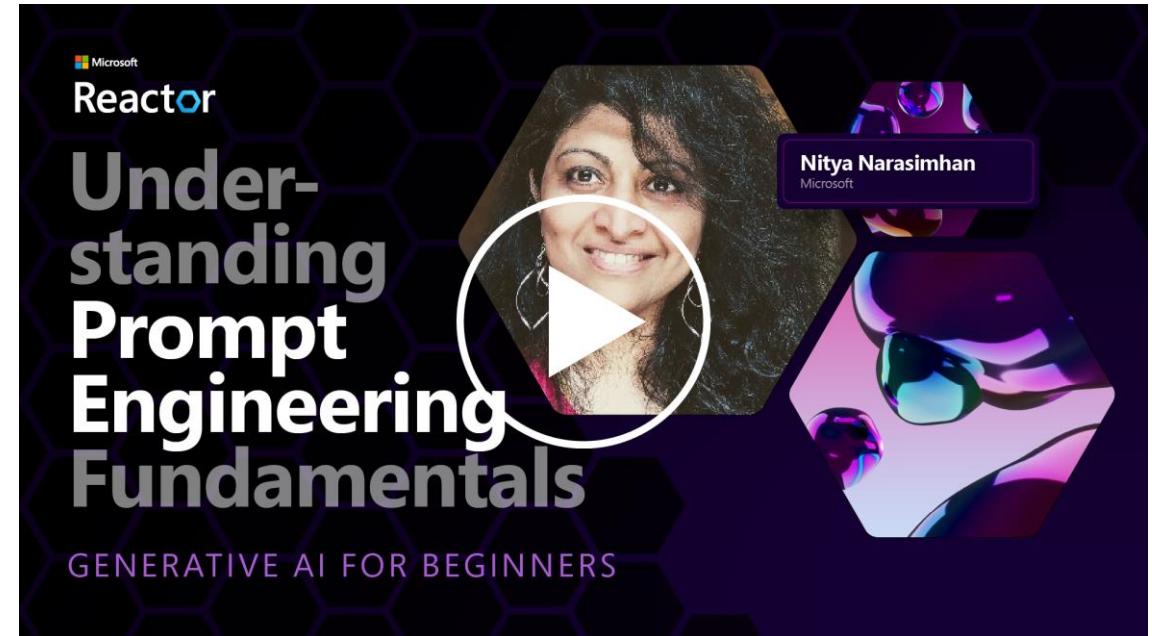
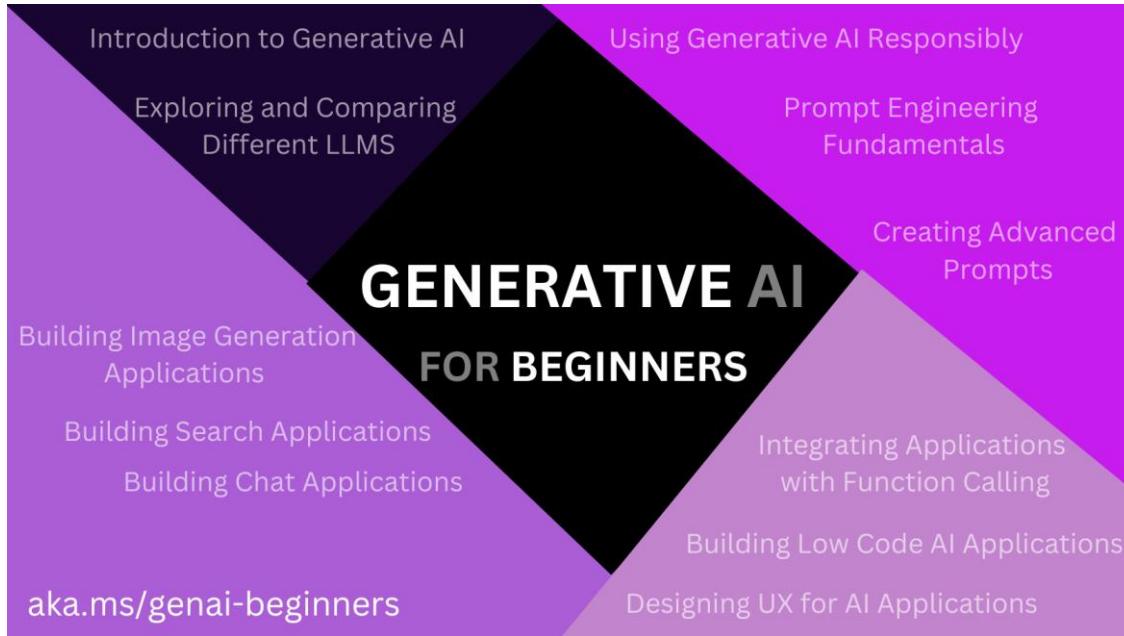
Use pandas to load the IPL-2022.csv dataset.

```
1 # Load the IPL-2022.csv dataset using pandas
2 ipl_data = pd.read_csv('../data/kaggle/IPL-2022.csv')
3
4 # Display the first 5 rows of the dataset
5 ipl_data.head()
6
7 0.0s
```



Activates on `<dataframe>.head()` inside Jupyter Notebook

Generative AI Course – From Principles to Prototypes



12 Lessons to start with (free, open-source)

Jupyter Notebooks (assignments)

Github Codespaces (environment)

OpenAI or Azure OpenAI (API key)

Practice Prompt Engineering

Learn by doing more ..

Learning Resources

The dev container can be used with either GitHub Codespaces (online, in the cloud) or with Docker Desktop (offline, in local device). It is set up for default usage with a Visual Studio Code editor frontend. With this in mind, here are some learning resources for beginners:

1. [Python Developer Roadmap](#) - step-by-step topics guidance for learning
2. [Get Started with Python in VS Code](#) - learn fundamentals.
3. [Data Science in VS Code](#) - skill up on Jupyter Notebooks!
4. [GitHub Codespaces for Machine Learning](#) - build classifiers & more.
5. [OpenAI Cookbook](#) - example code, guides for prompt engineering
6. [Getting Started With GitHub Copilot](#) - understand capabilities
7. [GitHub Copilot: Fly with Python ...](#) - RealPython community article
8. [How to use GitHub Copilot: Prompts, tips, use cases](#) - Github blog
9. [Create Dynamic Data Viz with OpenAI & React](#) - OSS Dashboard
10. [Chat with your CSV..](#) - OSS Demo with Langchain/Streamlit
11. [PandasAI](#) - Analyze data using natural lang (OpenAI, HuggingFace)

Contribute to help others ..



LIDA Project

Organization for sharing projects and resources that build on the open-source microsoft/lida tool (LLMs for Automated Visualization)

Follow

README.md

About LIDA Project

LIDA Project is an organization for support community-facing projects and experiments using [Microsoft/Lida](#). Here is a brief summary of what LIDA is, and its core features.

What is LIDA?

LIDA is a library for generating data visualizations and data-faithful infographics.

LIDA is grammar agnostic (will work with any programming language and visualization libraries e.g. matplotlib, seaborn, altair, d3 etc) and works with multiple large language model providers (OpenAI, Azure OpenAI, PaLM, Cohere, Huggingface).

- 📖 | Read [the paper](#) for technical details on how LIDA is implemented.
- 📈 | See [this notebook](#) for a sample tutorial.
- 🌐 | Check out [this website](#) for project updates.
- ⭐ | Star [this repo](#) if you like the project.
- 🎥 | Watch [this video](#) for a quick walkthrough of LIDA in action.

 View as: Public

You are viewing the README and pinned repositories as a public user.

[Get started with tasks](#) that most successful organizations complete.

Discussions

Set up discussions to engage with your community!

[Turn on discussions](#)

People



[Invite someone](#)



Watch **this** repo for more examples and tutorials for #30Days learning journeys

<https://aka.ms/pydata-for-jsdevs>

Follow **this** org, codespaces template, streamlit application
<https://aka.ms/lida/org>



For the non-Python
Programmer

Simplifying Data Analysis With GitHub Codespaces, Jupyter Notebooks & Open AI

- Transferable Learning with Jupyter Notebooks
- Consistent Dev Environment with GitHub Codespaces
- Focused Learning (in-context) with GitHub Copilot
- AI-assisted intuition using Microsoft LIDA

Thank you

<https://aka.ms/pydata-workshop-2023>

