UC San Diego

CSE UCSD

How Scientists Use Jupyter Notebooks: Goals, Quality Attributes, and Opportunities

Ruanqianqian (Lisa) Huang, Savitha Ravi,
Michael He, Boyu Tian, Sorin Lerner, Michael Coblenz

JupyterCon 2025 | November 4, 2025

Jupyter Notebooks Widely Used Among Scientists

NEWS FEATURE | 20 January 2021 | Correction 22 January 2021 |
Update 19 February 2021 | Correction 08 April 2021

Ten computer codes that transformed science

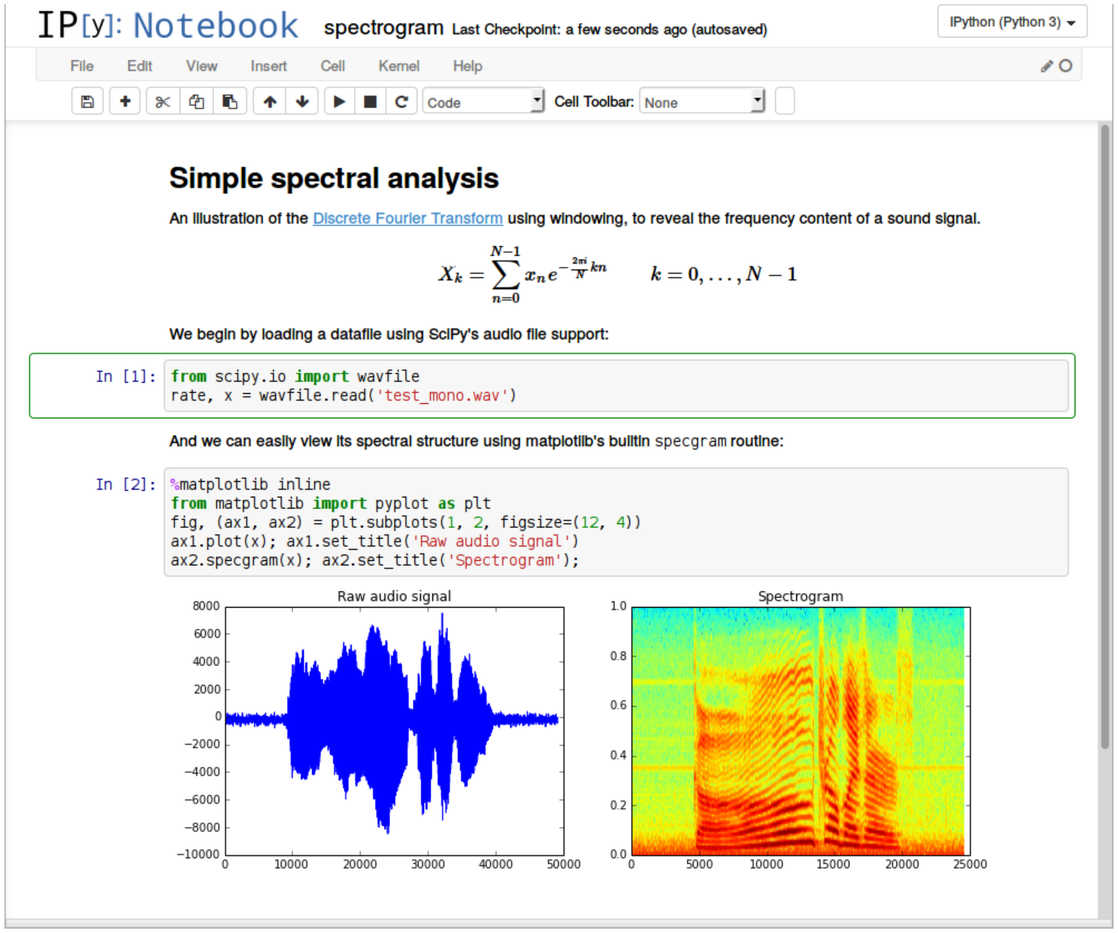
From Fortran to arXiv.org, these advances in programming and platforms sent biology, climate science and physics into warp speed.

By Jeffrey M. Perkel

Data explorer: IPython Notebook (2011)

Nature 2021

Jupyter Notebooks Widely Used Among Scientists



Explanatory text

Re-executable code cells

Cell outputs

B. E. Granger and F. Pérez, "Jupyter: Thinking and Storytelling With Code and Data," in *Computing in Science & Engineering*, vol. 23, no. 2, pp. 7-14, 1 March-April 2021, doi: 10.1109/MCSE.2021.3059263.

A. Rogozhnikov, “Jupyter (IPython) notebooks features,” https://arogozhnikov.github.io/2016/09/10/jupyter-features.html

Prior Results of Jupyter in General Data Analysis

- “Exploratory” [Kery and Myers 2017]
- “Throw-away” [Kandel et al. 2012]
- “Messy” [Kery et al. 2018, Rule et al. 2018]
- Too much effort to clean up the mess [Rule et al. 2018]

FAIR Principles

For scientific data and code

Findable

Accessable

Interoperable

Reusable

A Thin Layer over the Interpreter

A Thin Layer over the Interpreter

```
[1]: a = 10
```

```
[2]: a += 1
     a
```

```
[2]: 11
```

All cells share a global state

A Thin Layer over the Interpreter

```
[1]: a = 10

[2]: a += 1
     a

[2]: 11
```

All cells share a global state

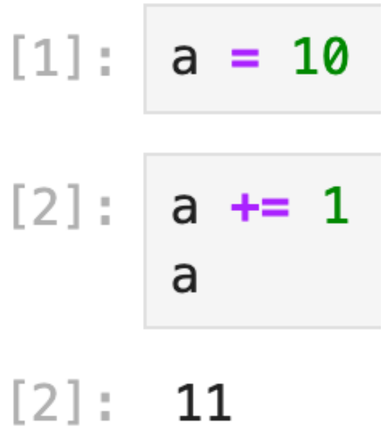
```
[1]: a = 10

[3]: a += 1
     a

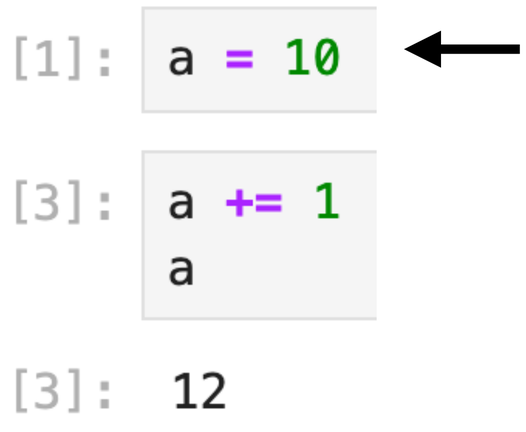
[3]: 12
```

Cells have side effects

A Thin Layer over the Interpreter



All cells share a global state



Cells have side effects

A Thin Layer over the Interpreter

```
[1]: a = 10

[2]: a += 1
     a

[2]: 11
```

All cells share a global state

```
[1]: a = 10

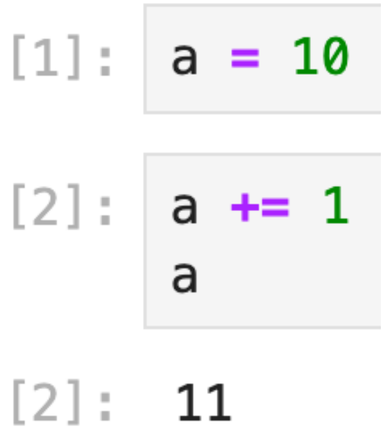
[3]: a += 1
     a

[3]: 12
```

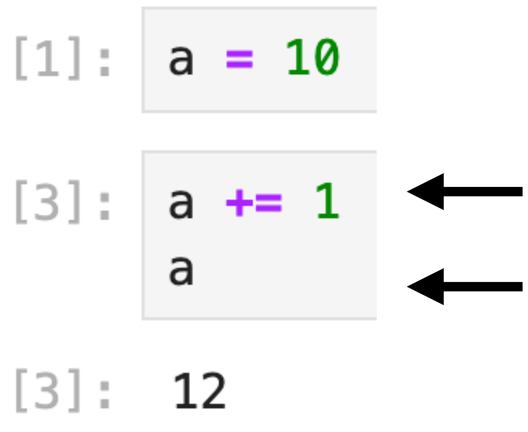


Cells have side effects

A Thin Layer over the Interpreter

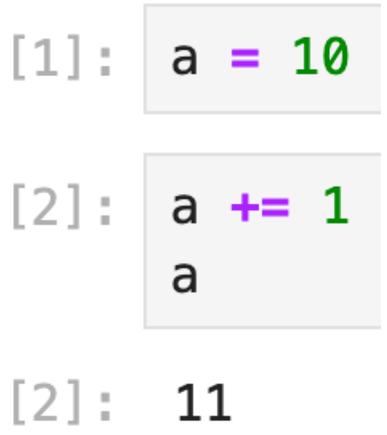


All cells share a global state

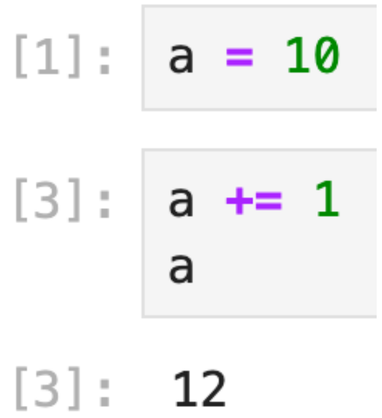


Cells have side effects

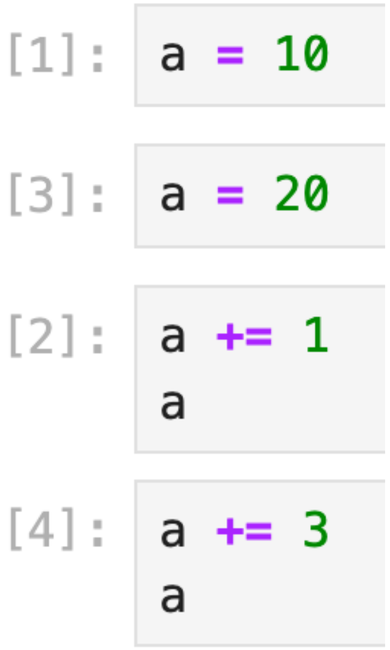
A Thin Layer over the Interpreter



All cells share a global state



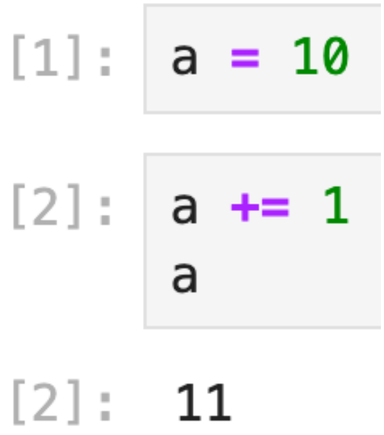
Cells have side effects



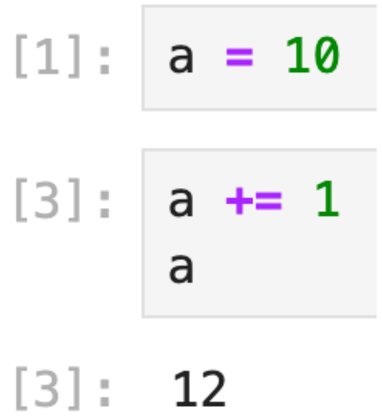
a = 24

Cells can go out of order

A Thin Layer over the Interpreter

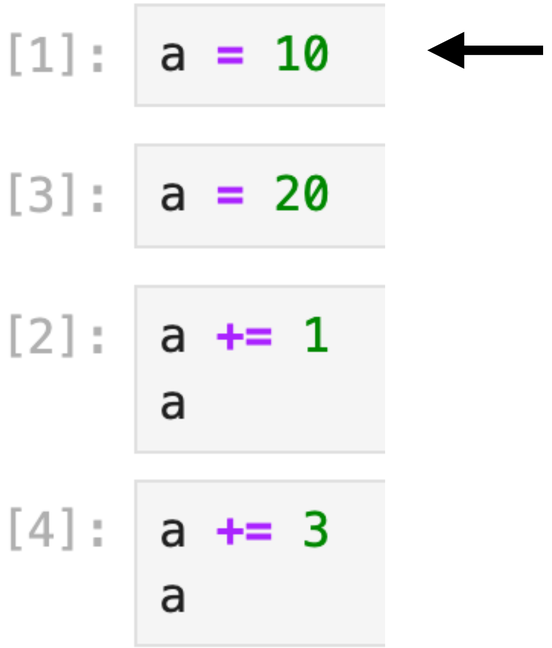


All cells share a global state



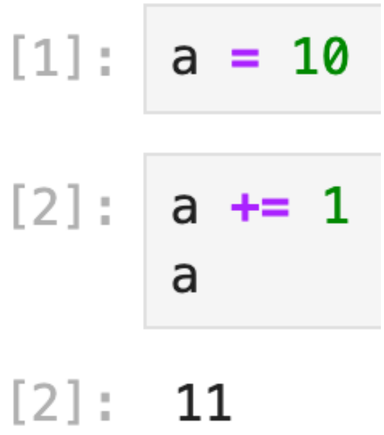
Cells have side effects

a = 10

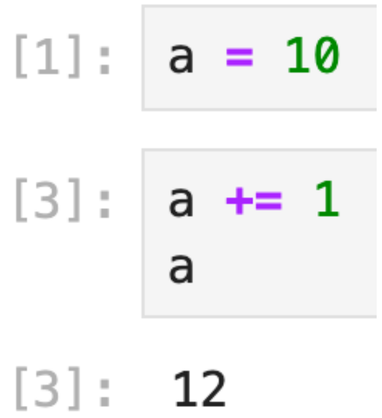


Cells can go out of order

A Thin Layer over the Interpreter

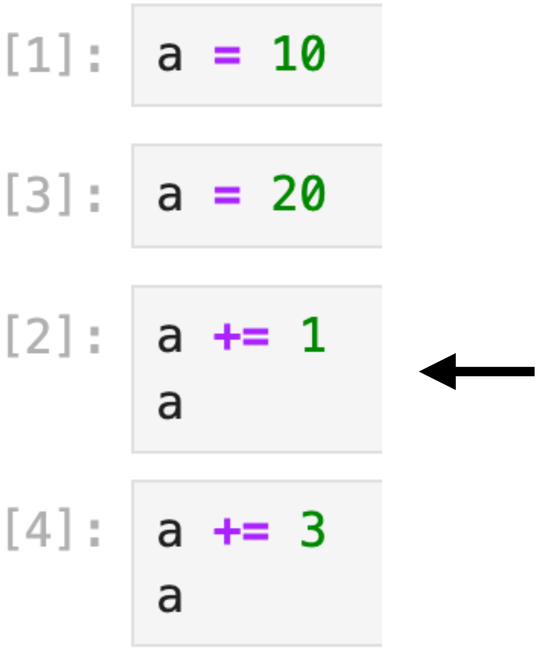


All cells share a global state



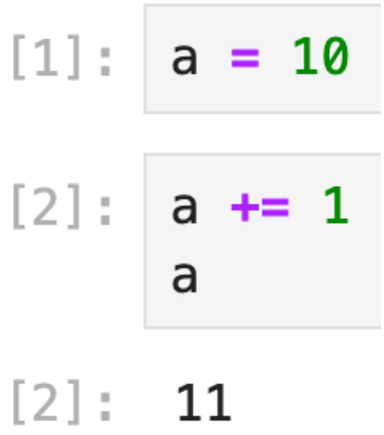
Cells have side effects

a = 11

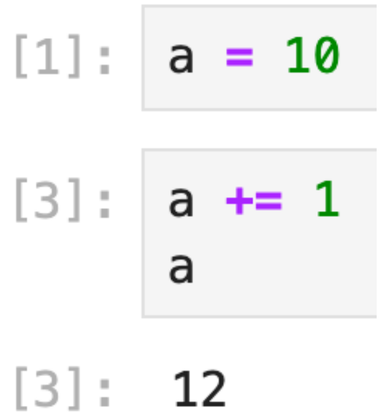


Cells can go out of order

A Thin Layer over the Interpreter

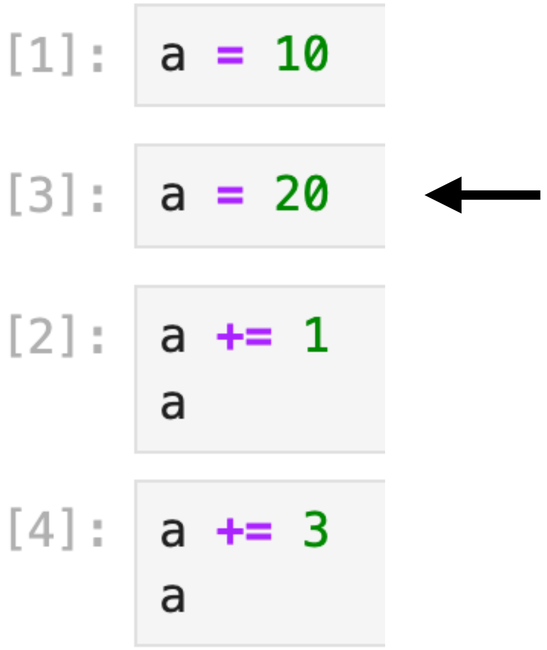


All cells share a global state



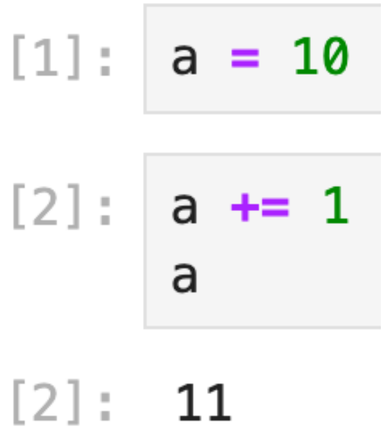
Cells have side effects

a = 20

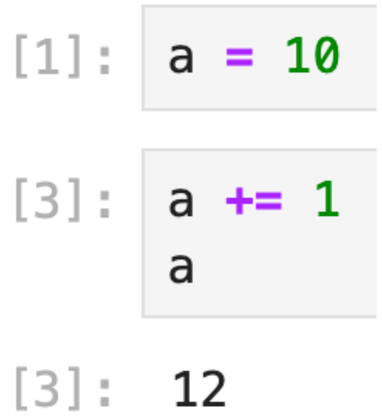


Cells can go out of order

A Thin Layer over the Interpreter

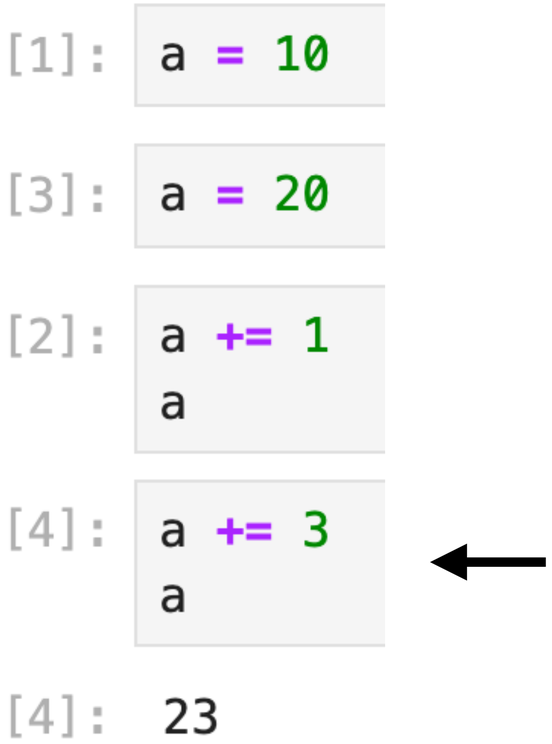


All cells share a global state



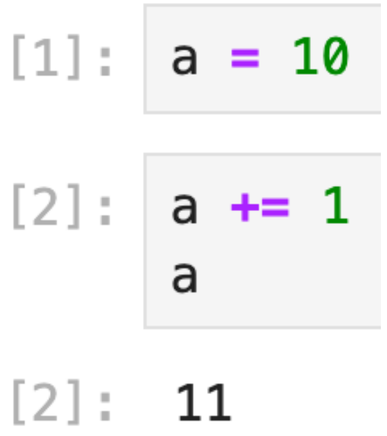
Cells have side effects

a = 23

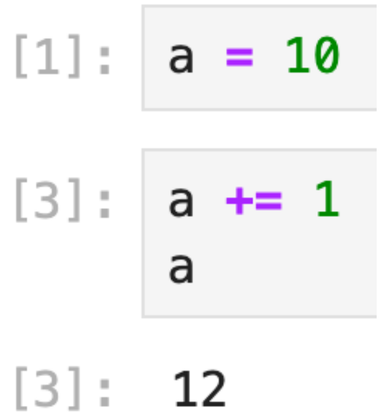


Cells can go out of order

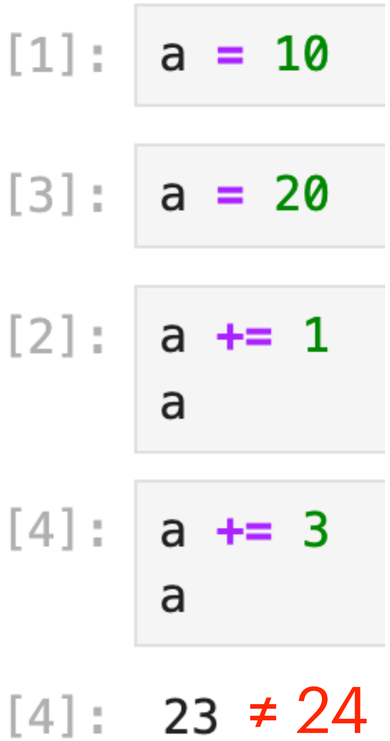
A Thin Layer over the Interpreter



All cells share a global state



Cells have side effects



Cells can go out of order

A Thin Layer over the Interpreter

```
In [1]:  1  a = 10

In [2]:  1  a += 1
         2  a

Out[2]: 11
```

All cells share a global state

```
In [1]:  1  a = 10

In [3]:  1  a += 1
         2  a

Out[3]: 12
```

Cells have side effects

```
[7]:  a = 20

[6]:  a += 1
      a

[6]: 11

[8]:  a += 3
      a

[8]: 23
```

Cells can go out of order

A Thin Layer over the Interpreter

```
In [1]:  1  a = 10

In [2]:  1  a += 1
         2  a

Out[2]: 11
```

All cells share a global state

```
In [1]:  1  a = 10

In [3]:  1  a += 1
         2  a

Out[3]: 12
```

Cells have side effects

```
[7]:  a = 20

[6]:  a += 1
      a

[6]: 11

[8]:  a += 3
      a

[8]: 23
```

Cells can go out of order

How does Jupyter *support* the needs of scientific users?

What do scientists value in software?

What do scientists value in software?

Quality Attributes

What do scientists value in software?

Quality Attributes

e.g., reusability, maintainability, scalability

What do scientists value in software?

Quality Attributes

e.g., reusability, maintainability, scalability

What tactics do scientists use to promote those qualities?

Study Details

- 60min observation of **a participant's own task** and 30min interview

Study Details
Qualitative Analysis Process

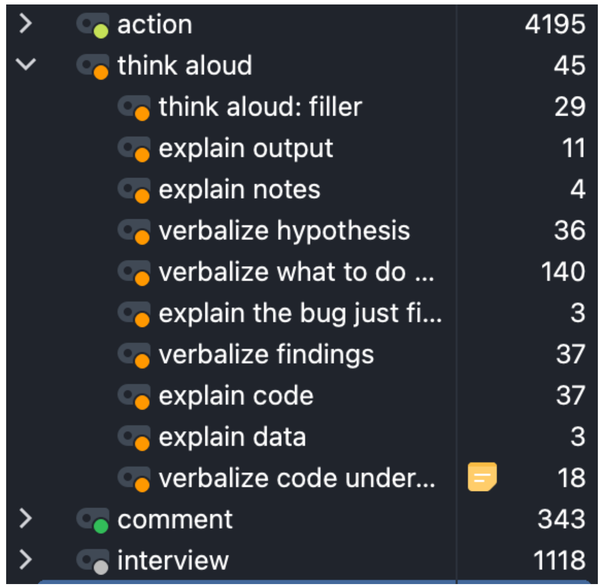
- Grounded theory analysis process
  - **Memos** to determine saturation of findings

P04

- *PhD student in Computer Science working on wireless communication with knowledge about SWE; task is an exploratory analysis for understanding the quality of geographical data and assess whether the data processing or visualization algorithms need adjustments*
- **Readability**
  - When reusing code written by themselves, participant feels confident about understanding the code if they have left themselves proper documentation earlier
  - Meaningful (variable and function) naming is crucial
- **Reproducibility**
  - Participant feels confident for the others to reproduce their output if linearity is manually enforced
- **Incremental execution:** participant sometimes feels hesitant about “restart and runall” when they want to avoid reloading large data / recomputing something time consuming
- **Cleanup:** happens iteratively from idea to idea
  - Merge small cells of one-liners into bigger code blocks
  - If code will be later reused, then further wrap the code blocks into functions
  - If code will not be reused, then will not waste the effort of refactoring into functions
- **Navigation / Structure:** Add markdown headings throughout the notebook for segmentation

Study Details
Qualitative Analysis Process

- Grounded theory analysis process
  - **Memos** to determine saturation of findings
  - **Low-level codes** from recordings, transcripts, and comments



Study Details
Qualitative Analysis Process

- Grounded theory analysis process
  - **Memos** to determine saturation of findings
  - **Low-level codes** from recordings, transcripts, and comments
  - **High-level codes** for notebook goals, quality attributes, and user tactics

low-level code

B	C	H
Code	**Code Comment**	**Tactic**
comment > comment: inspecting code within a function definition	nable to inspect statements within a function definition via execution to understand their behavior, so pulling them out to he top level for inspection	out of order execution

high-level code

Study Details

- 60min observation of **a participant's own task** and 30min interview
- Grounded theory qualitative analysis process
  - **High-level codes** for notebook goals, quality attributes, and user tactics

Study Details

- 60min observation of **a participant's own task** and 30min interview
- Grounded theory qualitative analysis process
  - **High-level codes** for notebook goals, quality attributes, and user tactics

- 20 participants (9 female):

Study Details

- 60min observation of **a participant's own task** and 30min interview
- Grounded theory qualitative analysis process
  - **High-level codes** for notebook goals, quality attributes, and user tactics

- 20 participants (9 female):
  - 11 computing, 3 biology, 3 social sciences, 3 earth sciences

Study Details

- 60min observation of **a participant's own task** and 30min interview
- Grounded theory qualitative analysis process
  - Derived notebook goals, quality attributes, and user tactics

- 20 participants (9 female):
  - 11 computing, 3 biology, 3 social sciences, 3 earth sciences
  - 21 notebooks total (one participant worked on two notebooks)

Research Questions

1. What goals do scientific users pursue in notebooks?

Research Questions

1. What goals do scientific users pursue in notebooks?
2. What non-functional requirements (quality attributes) do they prioritize?

Research Questions

1. What goals do scientific users pursue in notebooks?
2. What non-functional requirements (quality attributes) do they prioritize?
3. What tactics do they use to meet the quality requirements?

Research Questions

1. What goals do scientific users pursue in notebooks?
2. What non-functional requirements (quality attributes) do they prioritize?
3. What tactics do they use to meet the quality requirements?
   - Direct use of existing notebook features, OR

Research Questions

1. What goals do scientific users pursue in notebooks?
2. What non-functional requirements (quality attributes) do they prioritize?
3. What tactics do they use to meet the quality requirements?
   - Direct use of existing notebook features, OR
   - Workarounds for the limitations of existing features

Results

Results

Goals
(RQ1)

Disposable Explorations

Findings

Artifact Construction

Results

Goals (RQ1)

- Disposable Explorations
- Findings
- Artifact Construction

Quality Attributes (RQ2)

- Clarity
- Reusability
- Reproducibility
- Explorability
- Correctness
- Performance
- Debuggability
- Collaboration

Results

Goals (RQ1)

- Disposable Explorations
- Findings
- Artifact Construction

Quality Attributes (RQ2)

- Clarity
- Reusability
- Reproducibility
- Explorability
- Correctness
- Performance
- Debuggability
- Collaboration

Additional Findings

- Use of AI

Results

This Talk

Quality Attributes (RQ2)

- Explorability
- Clarity
- Performance

Results

This Talk

Quality Attributes (RQ2)

Explorability

Cell-output correspondences

Cell-based inspection

Clarity

Markdown cells

Saving outputs

Performance

Saved state

Cell-by-cell execution

Tactics (RQ3)

Results

This Talk

Quality Attributes (RQ2)

- **Explorability**
  - Cell-output correspondences
  - Cell-based inspection
- **Clarity**
  - Markdown cells
  - Saving outputs
- **Performance**
  - Saved state
  - Cell-by-cell execution

\+ Design Opportunities

Tactics (RQ3)

Results

Explorability

Quality Attribute (RQ2)

Explorability

Tactics (RQ3)

Using cell-output correspondences

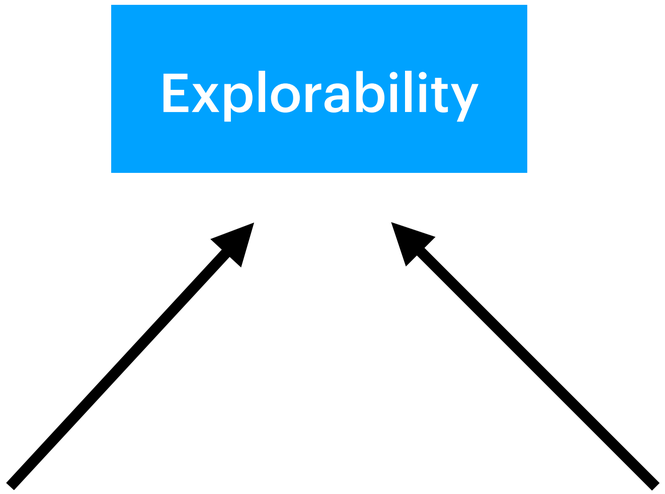
Cell-based inspection

+ Design Opportunities

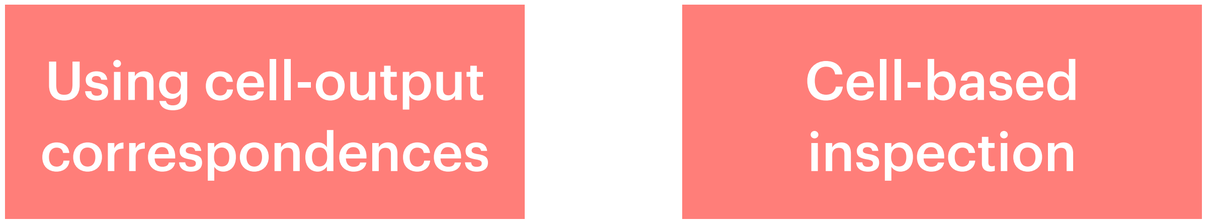
Results

Explorability

Quality Attribute (RQ2)



Tactics (RQ3)



+ Design Opportunities

Explorability

Tactic: Using cell-output correspondences

“It’s really easy to see intermediate values in your code because you have them stored”
— P12

Explorability

Tactic: Using cell-output correspondences

"It's really easy to see intermediate values in your code because you have them stored" — P12

[9]:
```
df
```

[9]:

	num_legs	num_wings	num_specimen_seen
falcon	2	2	10
dog	4	0	2
spider	8	0	1
fish	0	0	8

[10]:
```
df["num_wings_seen"] = df["num_wings"] * df["num_specimen_seen"]
df
```

[10]:

	num_legs	num_wings	num_specimen_seen	num_wings_seen
falcon	2	2	10	20
dog	4	0	2	0
spider	8	0	1	0
fish	0	0	8	0

Explorability

Workaround: Cell-based inspection

Inspection code: temporary code snippets that help the user understand the current state of the notebook

Explorability

Workaround: Cell-based inspection

```
[26]: for i in range(2999, 2000, -1):
          print(i)
```

```
2999
2998
2997
2996
2995
```

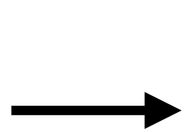
P19

Explorability

Workaround: Cell-based inspection

```
[26]: for i in range(2999, 2000, -1):
          print(i)
```

```
2999
2998
2997
2996
2995
```

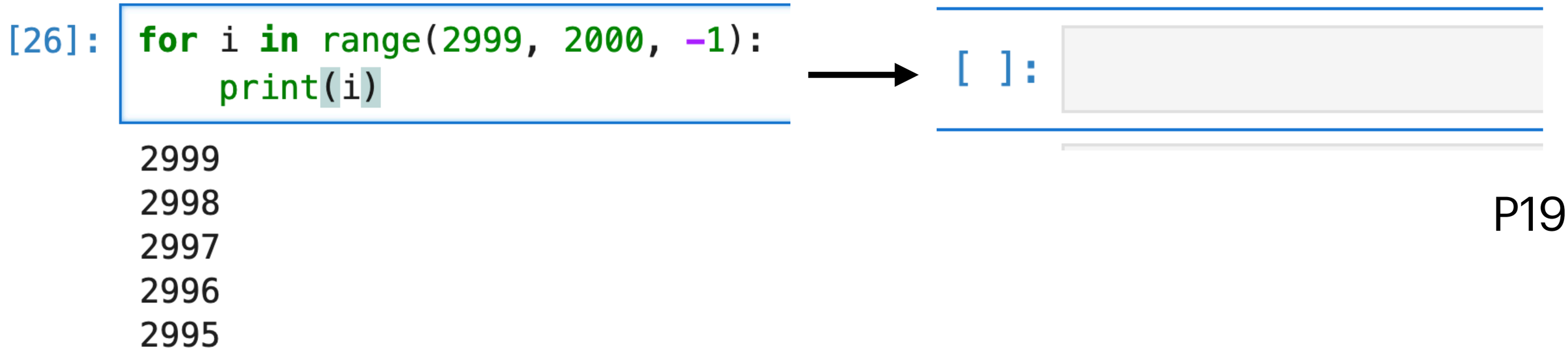


```
[ ]:
```

P19

Explorability

Workaround: Cell-based inspection



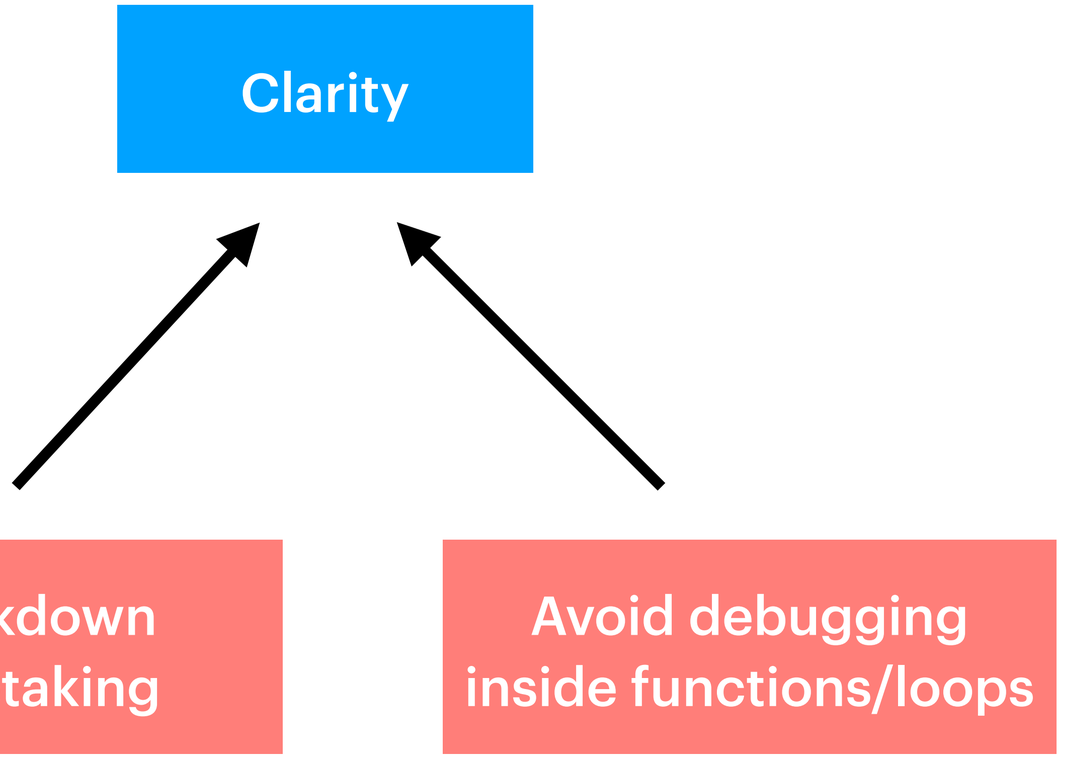
P19

Design Opportunity: incorporating inspection code without clutter

Results

Clarity

Quality Attribute (RQ2)



Tactics (RQ3)

Clarity

Tactic: Markdown Notetaking

“The Markdown function is extremely important to me.” — P20

Clarity

Tactic: Markdown Notetaking

“The Markdown function is extremely important to me.” — P20

P17

```
# What destination ASes systematically filter out ICMP Packets?
```

```
Fundação Carlos Chagas Filho de Amparo a Pesquisa          38
Fujitsu Limited                                            35
Rackspace, Ltd.                                            30
Name: count, dtype: int64
```

granularity of dying traceroutes (on which AS on which prefix) across probes

Clarity

Workaround: Saving outputs

P4

```
for i in range(len(interp_server_ble_lat_arr)):
    print(i)
    lat = interp_server_ble_lat_arr[i]
    lon = interp_server_ble_lon_arr[i]
    filtered_df = pop_df.loc[(pop_df["latitude"] > lat
                             (pop_df["longitude"] > lon
    print(len(filtered_df))

    # for j in range(len(filtered_df)):
    #     h3_hex = filtered_df.iloc[j]['geometry']
    #     print(h3_hex)
    #     print(Point(lon, lat))
```

```
0
5
1
2
```

Clarity

Workaround: Saving outputs

P4

```
for i in range(len(interp_server_ble_lat_arr)):
    print(i)
    lat = interp_server_ble_lat_arr[i]
    lon = interp_server_ble_lon_arr[i]
    filtered_df = pop_df.loc[(pop_df["latitude"] > lat
                             (pop_df["longitude"] > lon
    print(len(filtered_df))

    # for j in range(len(filtered_df)):
    #     h3_hex = filtered_df.iloc[j]['geometry']
    #     print(h3_hex)
    #     print(Point(lon, lat))
```

```
0
5
1
2
```

A **tension** between grouping related code (**clarity**) and breaking apart a cell to inspect inside it (**debuggability**)

Clarity

Workaround: Saving outputs

P4

```
for i in range(len(interp_server_ble_lat_arr)):
    print(i)
    lat = interp_server_ble_lat_arr[i]
    lon = interp_server_ble_lon_arr[i]
    filtered_df = pop_df.loc[(pop_df["latitude"] > lat
                             (pop_df["longitude"] > lon
    print(len(filtered_df))

    # for j in range(len(filtered_df)):
    #     h3_hex = filtered_df.iloc[j]['geometry']
    #     print(h3_hex)
    #     print(Point(lon, lat))
```

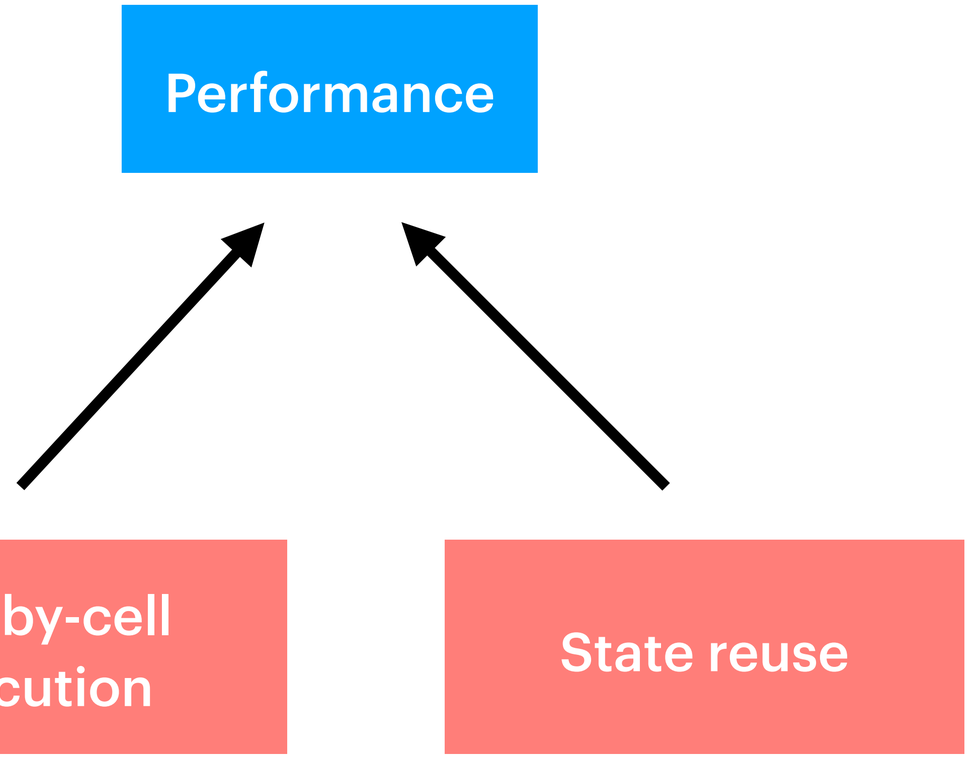
A **tension** between grouping related code (**clarity**) and breaking apart a cell to inspect inside it (**debuggability**)

DO: liveness affordances that work inside loops and functions

Results

Performance

Quality Attribute (RQ2)



Tactics (RQ3)

Performance

Tactic: Cell-by-cell execution

“The nice thing about Jupyter notebook for me is just loading the data, and not having to load it every time I run a script.” — P4

Performance

Tactic: Cell-by-cell execution

“The nice thing about Jupyter notebook for me is just loading the data, and not having to load it every time I run a script.” — P4



P2

Performance

Workaround: State reuse

State reuse + Manual state management → Incorrect code

Performance

Workaround: State reuse

State reuse + Manual state management → Incorrect code

```
[10]:
# Go through the JSON file and get the URLs, bu
# The goal is to see if new_parse_censys() gives
# old_parse_censys()
def get_old_censys_data_without_filtering():
    filename = "censys_domains_truncated.json"
    urls = []
    with open(filename, "r") as f:
        for line in f:
            data = json.loads(line)
            urls.extend(data["names"])
    return urls
```

P2

Performance

Workaround: State reuse

State reuse + Manual state management → Incorrect code

```
[10]:
# Go through the JSON file and get the URLs, bu
# The goal is to see if new_parse_censys() give
# old_parse_censys()
def get_old_censys_data_without_filtering():
    filename = "censys_domains_truncated.json"
    urls = []
    with open(filename, "r") as f:
        for line in f:
            data = json.loads(line)
            urls.extend(data["names"])
    return urls
```

```
[37]:
# Go through the JSON file and get the URLs, but
# The goal is to see if new_parse_censys() gives
# old_parse_censys()
def get_old_censys_data_without_filtering():
    filename = "censys_domains_truncated.json"
    urls = []
    with open(filename, "r") as f:
        for line in f:
            data = json.loads(line)
            urls.extend(data["names"])
    return urls
```

P2

Performance

Workaround: State reuse

State reuse + Manual state management → Incorrect code

```
[10]:
# Go through the JSON file and get the URLs, bu
# The goal is to see if new_parse_censys() gives
# old_parse_censys()
def get_old_censys_data_without_filtering():
    filename = "censys_domains_truncated.json"
    urls = []
    with open(filename, "r") as f:
```

```
[37]:
# Go through the JSON file and get the URLs, but
# The goal is to see if new_parse_censys() gives
# old_parse_censys()
def get_old_censys_data_without_filtering():
    filename = "censys_domains_truncated.json"
    urls = []
    with open(filename, "r") as f:
```

DO: context-aware tracking of outdated code

Results

Quality Attributes (RQ2)	Tactics (RQ3)	Design Opportunities
Explorability	Cell-output correspondences Cell-based inspection	Incorporating inspection code without clutter
Clarity	Markdown cells Saving outputs	Liveness affordances that work inside loops and functions
Performance	Caching data Cell-by-cell execution	Context-aware tracking of outdated code

Ongoing Work: Peapod

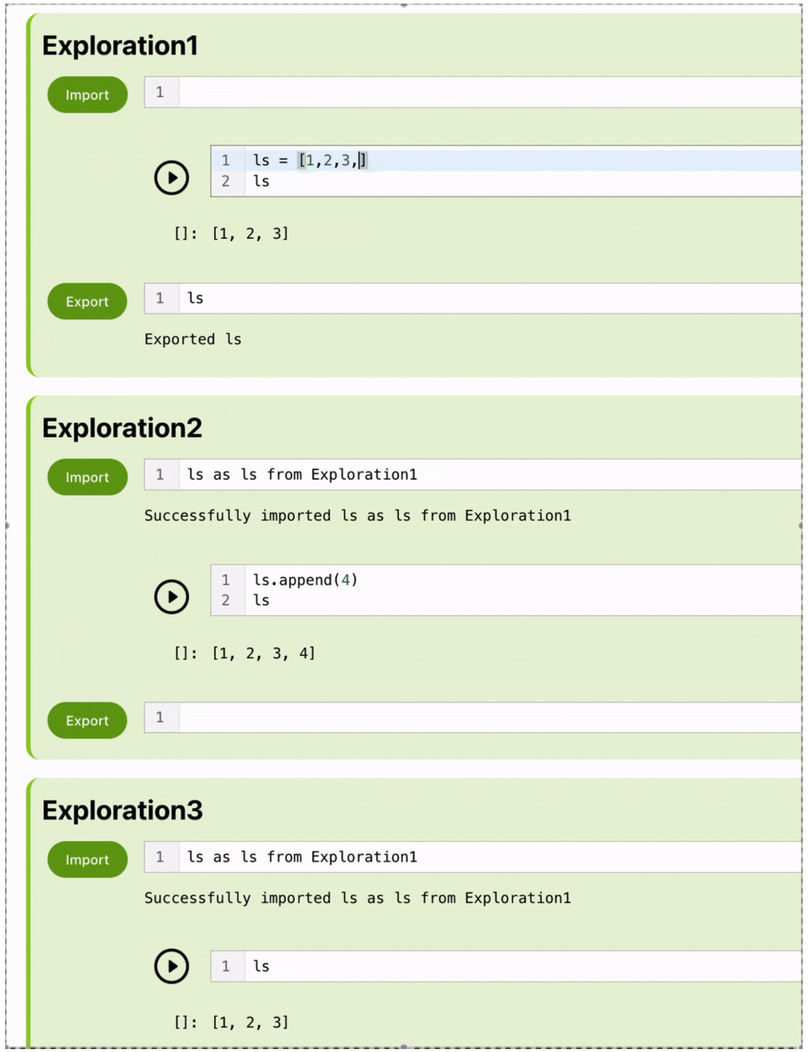
Abstractions for modular notebooks

```
[1]: a = 10

[2]: a += 1
     a

[2]: 11
```

Out of date exploration



All cells share a global state

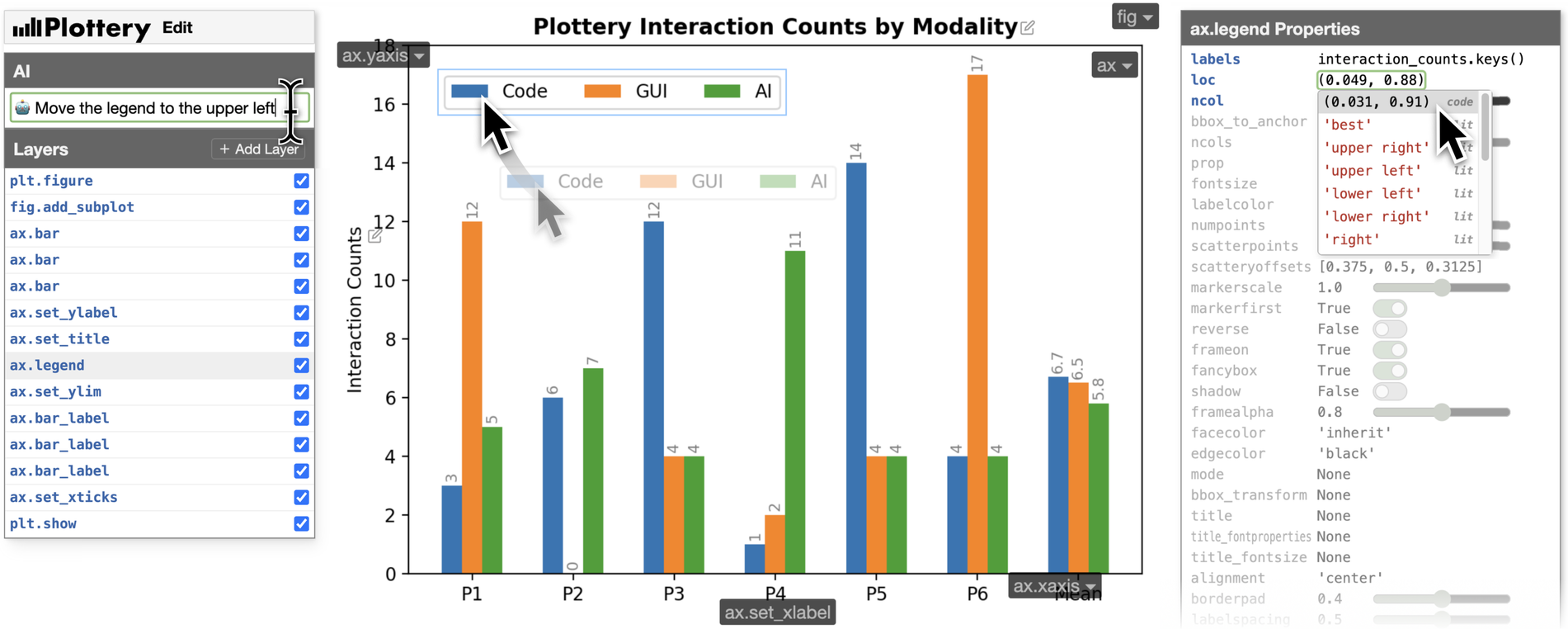
Contact: Ayla Kurdak akurdak@ucsd.edu

Ongoing Work: Plottery

Multimodal creation of Matplotlib plots

GUI + Code + AI

```
...
legend = ax.legend(labels=counts.keys(), loc=(0.049, 0.88), ncol=3)
...
```



Contact: Brian Hempel brian@brianhempel.com

Ongoing Work: TidyNote

A notebook extension that supports organizing exploration at various levels

Notebook

In [10]:
```
# Load CSV file data into DataFrame with pandas.
# Newly defined variable: df (DataFrame from CSV file)

df = pd.read_csv('../dataset/netflix_titles.csv')
```

Part 1: Count the total number of rows

- Get the number of rows of the given dataframe
- Print the value
- (Hint: use `len()`)

In [ ]:
```
# Output number of rows in DataFrame df.
# Newly defined variable: total_items (number of rows in df)

total_items = len(df)
total_items
```

Out[7]: 8807

In [ ]:

Scratchpad

In [ ]:
```
# Retrieves column names from the DataFrame
# No new variables are defined in this cell

df.columns
```

Out[5]:
```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added',
       'release_year', 'rating', 'duration', 'listed_in', 'description'],
      dtype='object')
```

In [ ]:
```
# Lists unique values in 'type' column of DataFrame
# No new variables are defined in this cell

df['type'].unique()
```

Out[8]: array(['Movie', 'TV Show'], dtype=object)

Contact: Lisa Huang r6huang@ucsd.edu

Conclusion

An **observational study** of 20 scientific users of Jupyter notebooks showing:

Conclusion

An **observational study** of 20 scientific users of Jupyter notebooks showing:

- **Goals** for notebooks are **not just for messy explorations**

Conclusion

An **observational study** of 20 scientific users of Jupyter notebooks showing:

- **Goals** for notebooks are **not just for messy explorations**
- Scientific users value **many quality attributes (QAs), some in conflict**

Conclusion

An **observational study** of 20 scientific users of Jupyter notebooks showing:

- **Goals** for notebooks are **not just for messy explorations**
- Scientific users value **many quality attributes (QAs), some in conflict**
- **Tactics** used to meet one QA **often inhibit other QAs**

Conclusion

An **observational study** of 20 scientific users of Jupyter notebooks showing:

- **Goals** for notebooks are **not just for messy explorations**
- Scientific users value **many quality attributes (QAs), some in conflict**
- **Tactics** used to meet one QA **often inhibit other QAs**

New designs for notebooks that help scientists meet their QAs

Paper: https://doi.org/10.1109/ICSE55347.2025.00232

Contact: Savitha Ravi, s2ravi@ucsd.edu

UC San Diego

CSE UCSD

ArXiv Paper

Extra Slides

Use of AI Tools

- 11 participants used AI tools: 3 Copilot*, 9 ChatGPT* (P5 used both)

ChatGPT Usage Scenario	Participants	Success Rate

* early 2024 versions

Use of AI Tools

- 11 participants used AI tools: 3 Copilot*, 9 ChatGPT* (P5 used both)

ChatGPT Usage Scenario	Participants	Success Rate
Explaining code	P7	2/2
Fixing errors	P13, P17	2/2
Generating visualization code	P9, P13	3/3
Using unfamiliar libraries	P5, P9, P15, P19	1/5
Other programming tasks	P10, P13, P20	3/4

(e.g., brainstorming a solution)

* early 2024 versions

Use of AI Tools

- 11 participants used AI tools: 3 Copilot*, 9 ChatGPT* (P5 used both)

ChatGPT Usage Scenario	Participants	Success Rate
Explaining code	P7	2/2
Fixing errors	P13, P17	2/2
Generating visualization code	P9, P13	3/3
Using unfamiliar libraries	P5, P9, P15, P19	1/5
Other programming tasks	P10, P13, P20	3/4

(e.g., brainstorming a solution)

low success rate

- **Scientific users rely on AI tools for notebook programming, but they sometimes lack the knowledge to successfully use AI suggestions**

* early 2024 versions