PLAID: Supporting Computing Instructors to Identify Domain-Specific Programming Plans at Scale

Yoshee Jain*

Mehmet Arif Demirtaş*

Kathryn Cunningham

*Indicates Equal Authorship

Meet Jane: Teaching Programming to Non-CS Majors

College of Liberal Arts & Sciences
Department of Psychology

Data Analysis for Psychology Research

pandas

However, her students **do not want** to be software developers.

```
import pandas as pd

df = pd.read_csv("EEG_data.csv")

def final_score(eeg1, eeg2):
  return (eeg1 * 0.2 + eeg2 * 0.8) if (eeg2 > eeg1) else eeg1

df['final_score'] = df.apply(lambda obs: final_score(obs.eeg1, obs.eeg2))
```

```
import pandas as pd

df = pd.DataFrame.from_dict({'rating': [6, 9, 4], 'duration': [2.5, 3.0, 1.2]})

df['attention_score'] = df.apply(lambda x: x['rating'] * x['duration']), axis='columns')
```

```
import pandas as pd

df = pd.read_csv("EEG_data.csv")

def final_score(eeg1, eeg2):
  return (eeg1 * 0.2 + eeg2 * 0.8) 

df['final_score'] = df.apply(lambda obs: final_score(obs.eeg1, obs.eeg2))
```

Goal: Create a new column by processing existing ones

```
import pandas as pd

df = pd.DataFrame.from_dict({'rating': [6, 9, 4], 'duration': [2.5, 3.0, 1.2]})

df['attention_score'] = df.apply(lambda x: x['rating'] * x['duration']), axis='columns')
```

Template that highlights **the goal** of **a common code pattern**

```
import pandas as pd

df = pd.read_csv("EEG_data.csv")

def final_score(eeg1, eeg2):
  return (eeg1 * 0.2 + eeg2 * 0.8)

df['final_score'] = df.apply(lambd
```

```
import pandas as pd

df = pd.DataFrame.from_dict({'rati

df['attention_score'] = df.apply(l
```

Name: Composite columns with lambda

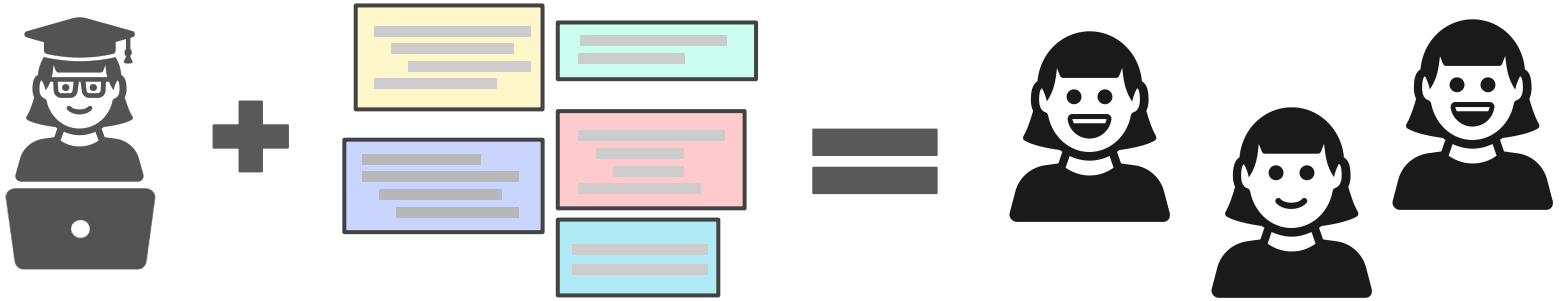
Goal: Create a new column by processing existing ones

Solution:

```
# Given a DataFrame object and a function

df['new'] = df.apply(lambda row: func(row['column']))
```

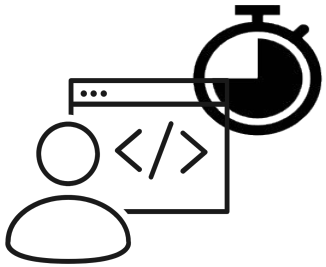
Changeable Areas: `Column to create` `Processor func` `Column to process`

Plan-Focused Pedagogies: Instruction That Uses Programming Plans



Plans can **improve motivation** [1] and **problem-solving skills** [2]

[1] K Cunningham, B J Ericson, R A Bejarano, and M Guzdial. 2021. Avoiding the Turing Tarpit: Learning Conversational Programming by Starting from Code's Purpose. **CHI '21.**
[2] N Weinman, A Fox, and MA Hearst. 2021. Improving Instruction of Programming Patterns with Faded Parsons Problems. **CHI '21.**



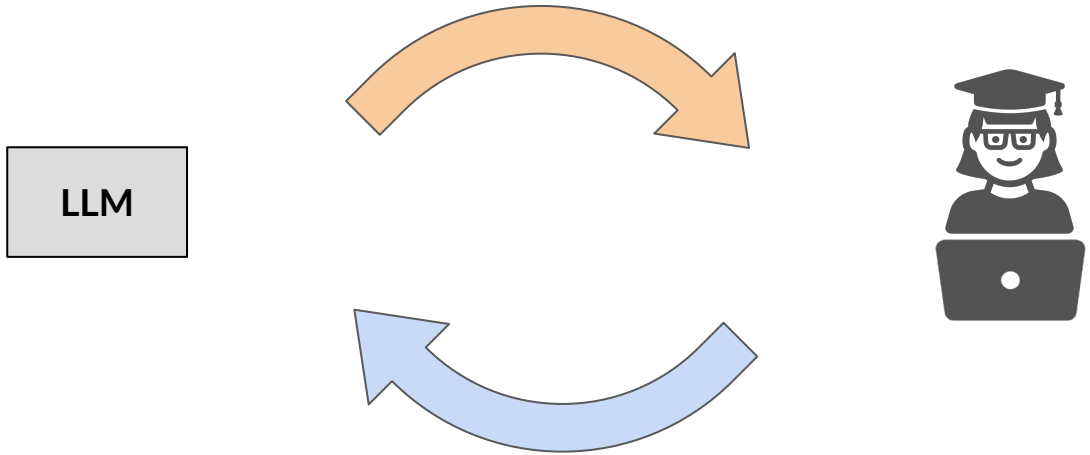
However, plan-focused pedagogies;

- have been mostly limited to **introductory programming content**
- can be **time-consuming** and **unclear** to apply as an instructor

Instructor-in-the-loop systems for interacting with LLMs can support this process.

In instructor-in-the-loop systems…

LLM generates various examples to provide many alternatives



Instructor chooses and refines content according to their learning objectives

Our Contributions:

1

Understand **instructor practices** and **challenges** for applying plan-based pedagogies

2

Present design goals for incorporating LLM-generated content into instructor workflows through iterative design workshops

3

Design and evaluate PLAID: a system combining LLMs' strengths with instructor expertise

Our Contributions:

1. Understand **instructor practices** and **challenges** for applying plan-based pedagogies

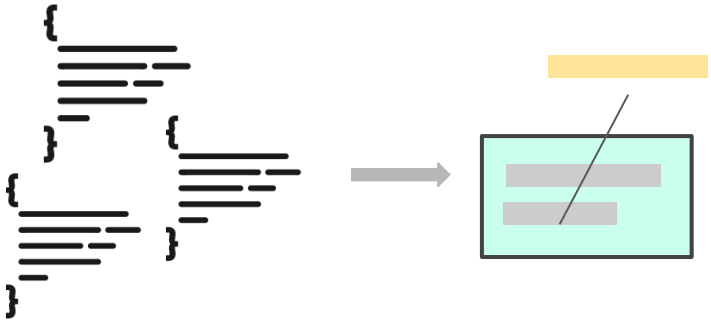
2. **Present design goals** for incorporating LLM-generated content into instructor workflows through iterative design workshops

3. **Design and evaluate PLAID**: a system combining LLMs' strengths with instructor expertise

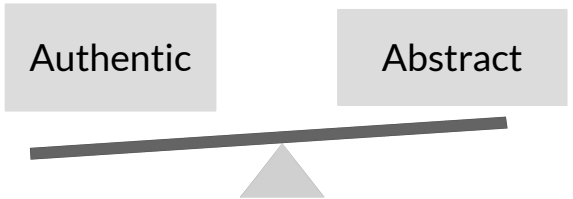
What are the challenges in applying plan-focused pedagogies?

Challenges Observed in Formative Study with Instructors (N=10)

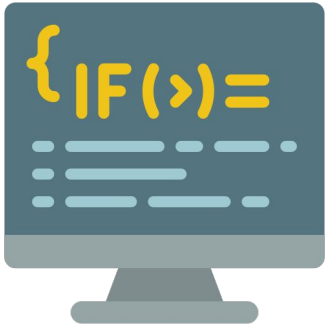
Instructors find it hard to **find patterns common in real-world programs.**

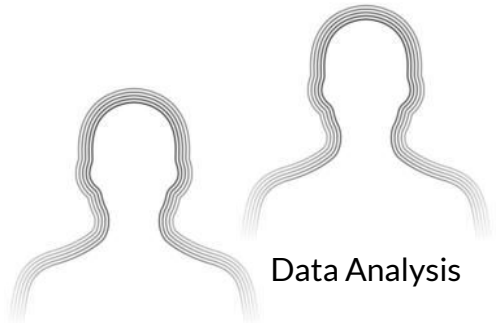


Finding the **right balance between real-world complexity and abstract templates** was challenging.



Due to a lack of prior work, instructors **may struggle to learn about plan-based pedagogies.**





Data Analysis

Web Development

Challenges	Opportunities
Tedious to find plans from practice	Generate **many authentic programs** and enable **interactions for quick exploration**
Difficult to find the right level of abstraction in plans	Provide interactions for comparing **similar content to recognize high-level patterns**
Challenging to learn how to apply plan-based pedagogies	Scaffold instructor experience by providing **structured components** derived from existing **best practices**.

How does PLAID support plan identification?

Generating Example Programs

Creating a Dataframe from dictionary

```
data = {
    'A': [1, 2, 3, 4],
    'B': [10, 20, 30, 40],
    'C': [100, 200, 300, 400]
}
df = pd.DataFrame(data)
```

Calculating the mean of a column

```
import pandas as pd

# Assuming df is your DataFrame
mean_value = df['col_name'].mean()
```

Calculating the median of a column

```
import pandas as pd

# Assuming df is your DataFrame
median_value =
df['col_name'].median()
print(median_value)
```

Segmenting Patterns from Programs

Clustering Patterns into Plans

Plan: Creating DataFrames

Plan: Summary Statistics

Plan: Plotting Data

Item Details

Name:

Summary Statistics

Goal:

Generate summary statistics for a single column

Solution:

```
import pandas as pd

# Assuming df is your DataFrame
mean_value = df['col_name'].mean()
```

Changeable Areas:

col_name - Name of the column
mean() - The aggregation function for the statistic

Potential Values Hide

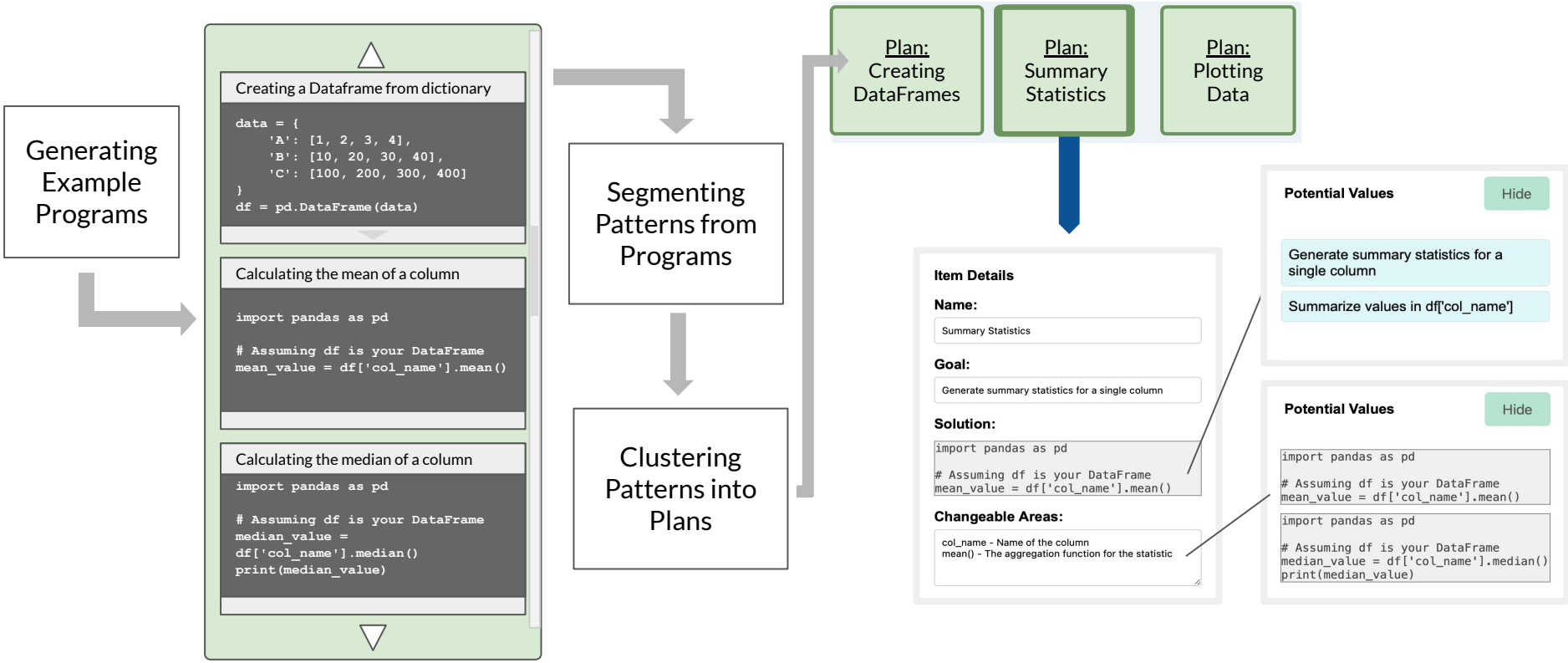
Generate summary statistics for a single column

Summarize values in df['col_name']

Potential Values Hide

```
import pandas as pd

# Assuming df is your DataFrame
mean_value = df['col_name'].mean()
```

```
import pandas as pd

# Assuming df is your DataFrame
median_value = df['col_name'].median()
print(median_value)
```


Generating Example Programs
Creating a Dataframe from dictionary
data = {
'A': [1, 2, 3, 4],
'B': [10, 20, 30, 40],
'C': [100, 200, 300, 400]
}
df = pd.DataFrame(data)
Calculating the mean of a column
import pandas as pd
# Assuming df is your DataFrame
mean_value = df['col_name'].mean()
Calculating the median of a column
import pandas as pd
# Assuming df is your DataFrame
median_value =
df['col_name'].median()
print(median_value)
Segmenting Patterns from Programs
Clustering Patterns into Plans
Plan: Creating DataFrames
Plan: Summary Statistics
Plan: Plotting Data
Item Details
Name:
Summary Statistics
Goal:
Generate summary statistics for a single column
Solution:
import pandas as pd
# Assuming df is your DataFrame
mean_value = df['col_name'].mean()
Changeable Areas:
col_name - Name of the column
mean() - The aggregation function for the statistic
Potential Values
Hide
Generate summary statistics for a single column
Summarize values in df['col_name']
Potential Values
Hide
import pandas as pd
# Assuming df is your DataFrame
mean_value = df['col_name'].mean()
import pandas as pd
# Assuming df is your DataFrame
median_value = df['col_name'].median()
print(median_value)

Generating Example Programs

Creating a Dataframe from dictionary

```
data = {
    'A': [1, 2, 3, 4],
    'B': [10, 20, 30, 40],
    'C': [100, 200, 300, 400]
}
df = pd.DataFrame(data)
```

Calculating the mean of a column

```
import pandas as pd

# Assuming df is your DataFrame
mean_value = df['col_name'].mean()
```

Calculating the median of a column

```
import pandas as pd

# Assuming df is your DataFrame
median_value =
df['col_name'].median()
print(median_value)
```

Segmenting Patterns from Programs

Clustering Patterns into Plans

Plan: Creating DataFrames

Plan: Summary Statistics

Plan: Plotting Data

Item Details

Name:

Summary Statistics

Goal:

Generate summary statistics for a single column

Solution:

```
import pandas as pd

# Assuming df is your DataFrame
mean_value = df['col_name'].mean()
```

Changeable Areas:

col_name - Name of the column
mean() - The aggregation function for the statistic

Potential Values Hide

Generate summary statistics for a single column

Summarize values in df['col_name']

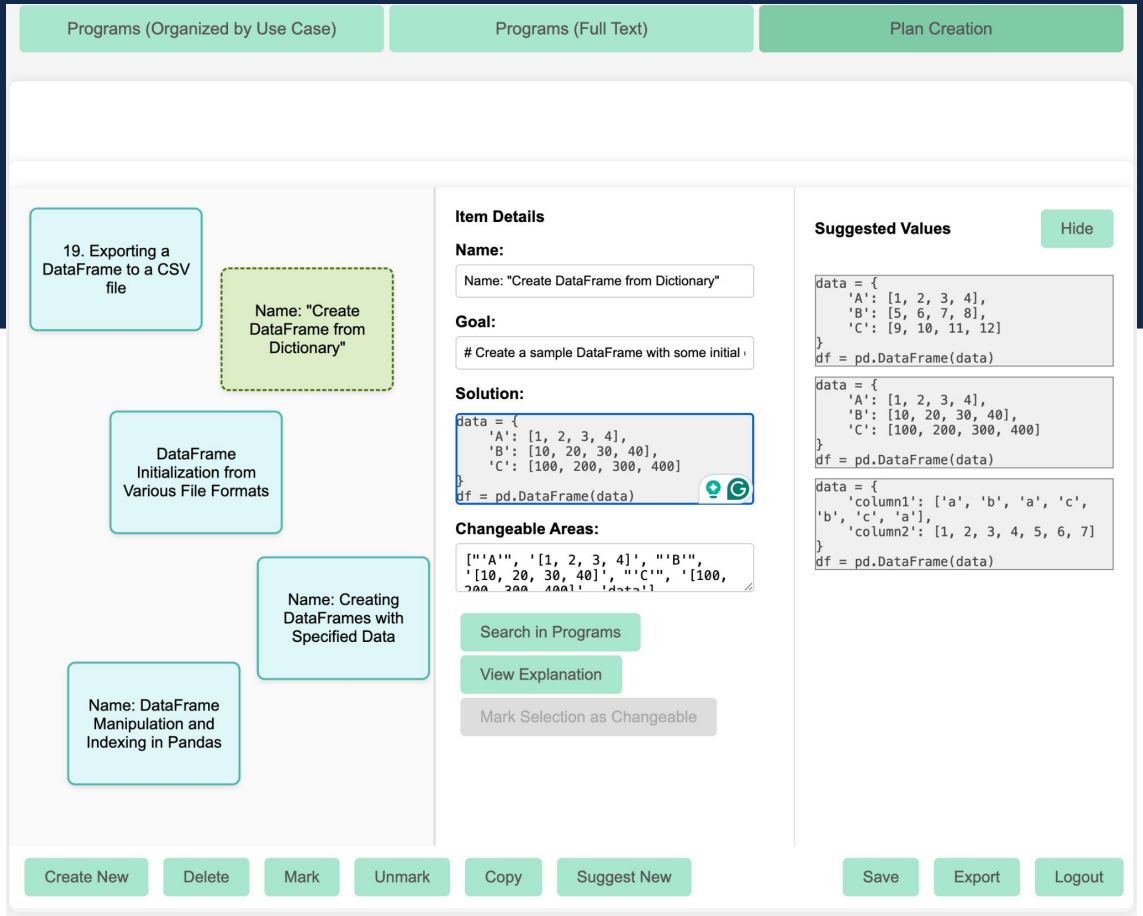
Potential Values Hide

```
import pandas as pd

# Assuming df is your DataFrame
mean_value = df['col_name'].mean()
```

```
import pandas as pd

# Assuming df is your DataFrame
median_value = df['col_name'].median()
print(median_value)
```

PLA ID

A system that **assists instructors in PLAn IDentification**

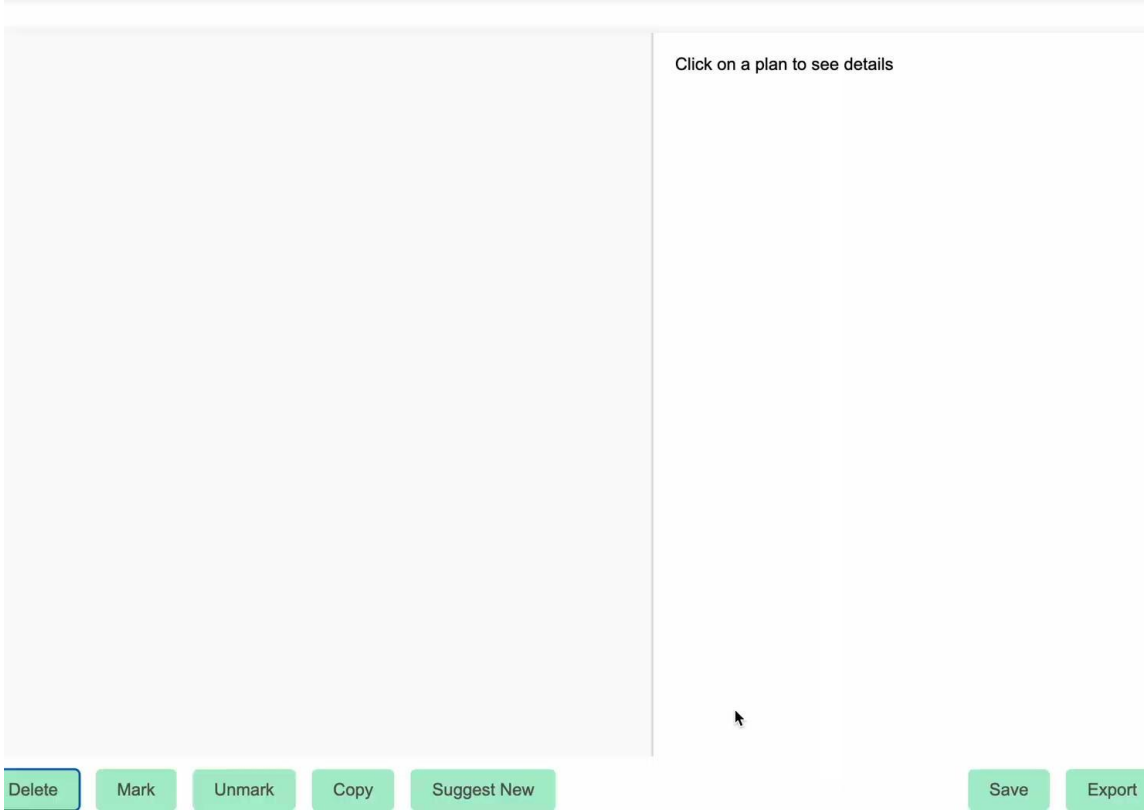


Jane wants to brainstorm ideas to create plans.

Suggests a plan

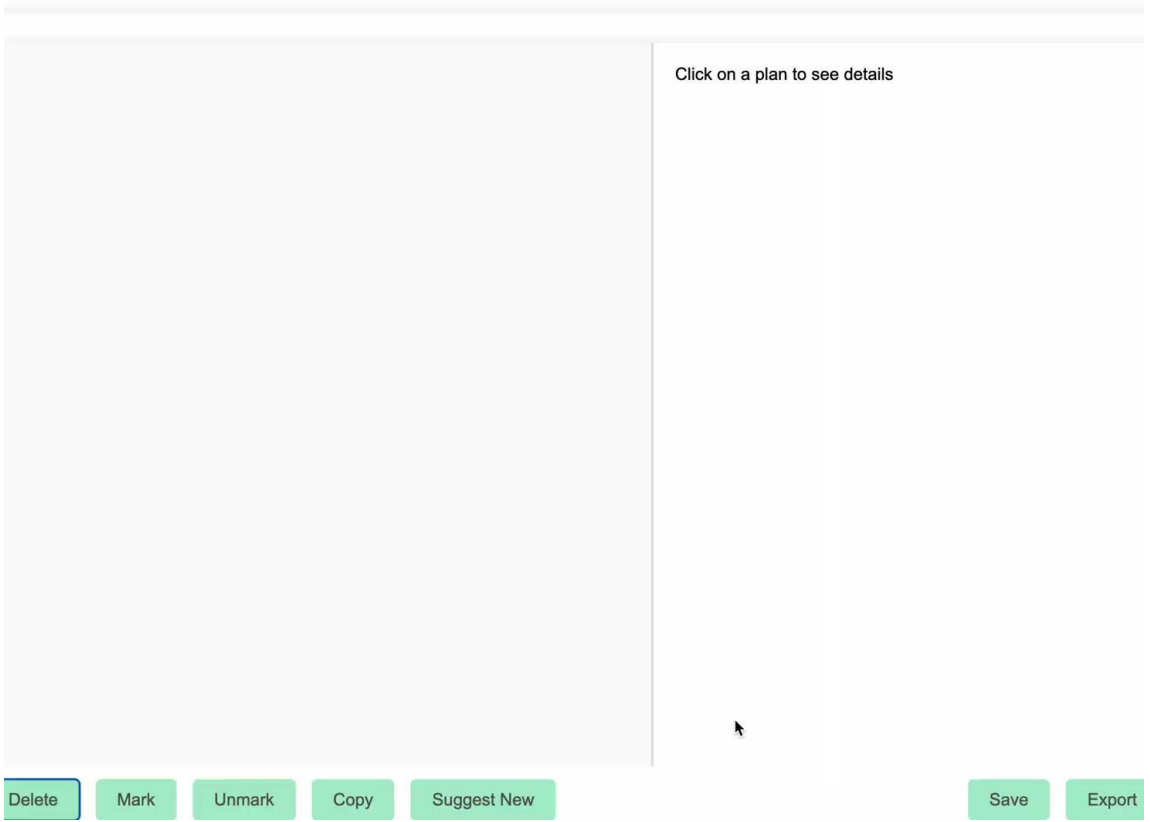
She finds the generated name and goal to be too generic.

Edit these fields



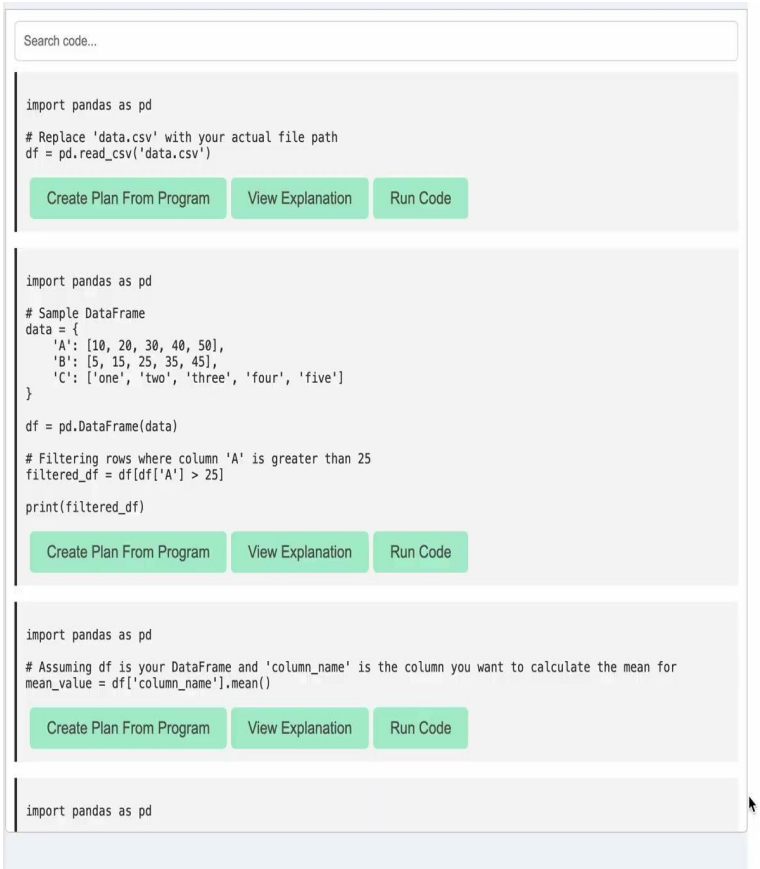
Tell students about reading data from different types of files.

Mark the file path as a changeable area



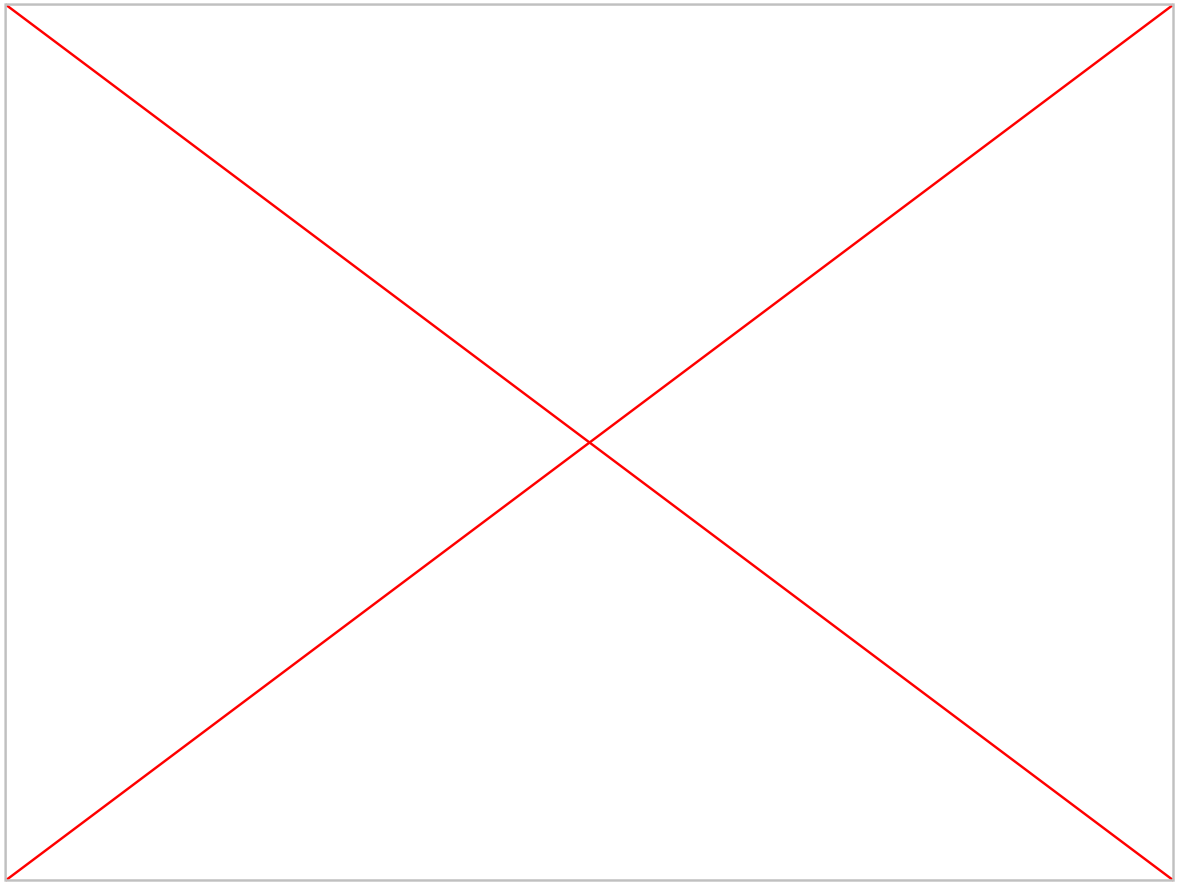
She wants to design a plan about creating histograms.

Explore examples and create plans from programs.



Creates multiple plans that achieve the same goal.

Organize plans into groups



Jane can use plans to…

- Reading Data from Files
- Statistical Operations
- Visualize Summary Statistics

Organize course slides around common goals

```
import pandas as pd

# "Check for equality in the column"
elem = Input element
col_name = Input column_name
df = Input dataframe

# share result
dataframe_name (df[col_name] == elem).sum()
print(result)
```

Create **worked examples**

```
import pandas as pd

df = pd._________("__________")

(A)   read_file, "EEG_data.csv"
(B)   read_json, "EEG_data.csv"
(C)   read_csv, "EEG_data.csv"
(D)   read_csv, "EEG_data.json"
```

Design **assessments**

Does PLAID make plan identification easier?

Results from Evaluation with Instructors (N=12)

Participants **created more plans** when using PLAID compared to the baseline condition.

Participants reported high scores on the **PSSUQ usability survey**.

Average task load for instructors was significantly **lower** with PLAID.

Qualitative Findings from Think-Aloud Sessions

Provides easily navigable reference material.

Initial ideas were hard to come up with:

- E8 stated, “I know the material for this on the Internet isn’t especially good”
- It is easier to “derive from an existing codebase...because the **sample code is the key part**” (E10)

Challenging to compare examples:

- E11 valued the **“condensed view”**, allowing them to look at multiple implementations on one screen

Provides easily navigable reference material.

Existing support does not meet instructors' needs:

Some participants compared our tool to ChatGPT:

- ChatGPT "was **verbose**" and that it was "quite an effort to ask even ChatGPT (for ideas)" (P9).
- ChatGPT output was **not appropriate for beginners**: "I don't even know if I fully understand (this concept)" (P5).

Creates learner-friendly material by providing structure

E10: "Stating explicit goals was useful for students to get more **motivated that [they] know the purpose** of learning about the code."

E9: "If I'm creating exercises in it, I'm specifying [to students] very clearly that **'This is the overall intuition of the coding flow, and these are the areas that you can play with.**'

Supports plan-based pedagogy in new domains

Instructors were interested:

- Most expressed interest in incorporating plan-based pedagogy in their courses.

Key Takeaways

For instructors:

- Presents LLM-generated content as **initial drafts** for refinement.
- Supports **interactions** to make this process easier and faster.

For designers:

- **Instructor-in-the-loop** approaches with LLMs are promising
- They can **automate repetitive work,** allowing instructors to focus on **refining** content.

PLAID: Supporting Computing Instructors to Identify Domain-Specific Programming Plans at Scale

Yoshee Jain*
yosheej2@illinois.edu

Mehmet Arif Demirtas*
mad16@illinois.edu

Katie Cunningham
katcun@illinois.edu

go.illinois.edu/PLAIDCHI

Takeaway | Instructors
PLAID enables using **plan-based pedagogies** in application-focused domains

Takeaway | Designers
LLMs should be combined with interactions for **exploration and refinement**