

# mage: Fluid Moves Between Code and Graphical Work in Computational Notebooks

Archaeologist  
Sahil Bhatia

# Overview

1. API that allows users to move fluidly between code in GUI and notebook

standard notebook

```
df.head()
```

	age	workclass	fnlwgt
0	90	?	77053
1	82	Private	132870
2	66	?	186061
3	54	Private	140359
4	41	Private	264663

1 mage : user edits table

```
%summon table df
```

	age	workclass	fnlwgt	occupation
0	90	?	77053	?
1	82	Private	132870	?
2	66	?	186061	?
3	54	Private	140359	?
4	41	Private	264663	?

2 mage : edits reflect in code

```
# -- generated code --  
column_names = list(df)  
column_names.pop(6)  
column_names.insert(1, "occupation")  
df = df.reindex(columns=column_names)  
%summon table df
```

	age	occupation	workclass
0	90	?	?
1	82	Exec-managerial	Private

```
%summon image dogs[1]
```



# Influenced By: ipywidgets

1. Interactive widgets in the notebook environment
  - a. Notebooks come **alive** when widgets are used
2. Core widgets supported are: sliders, progress bars, text boxes, display areas

## IntSlider

Slider:  8

## BoundedIntText

Bounded Int:

## Text

String:

## Textarea

String:

## RadioButtons

Options:  option 1  
 option 2  
 option 3

## SelectMultiple

Options:

## Dropdown

Number:

## Checkbox


Check me

## Button

## DatePicker

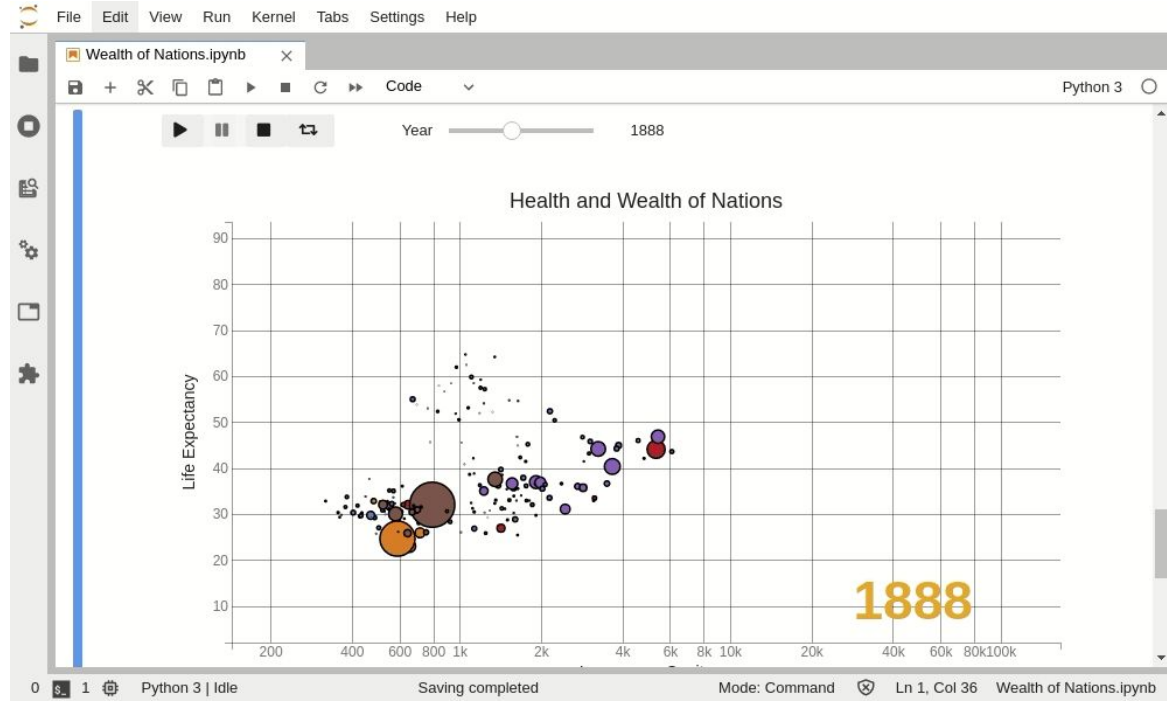
Pick a Date:

## IntProgress

Progress: 

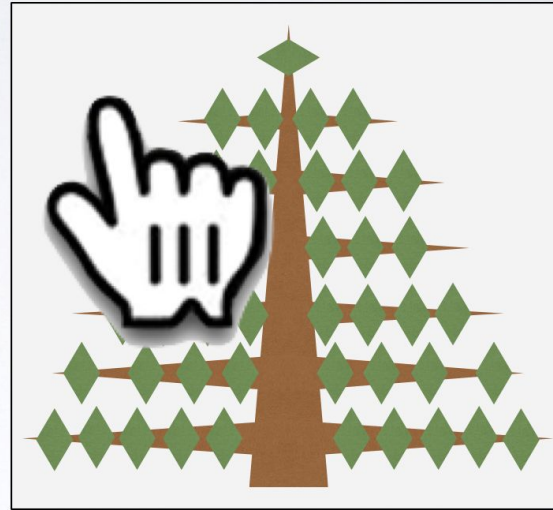
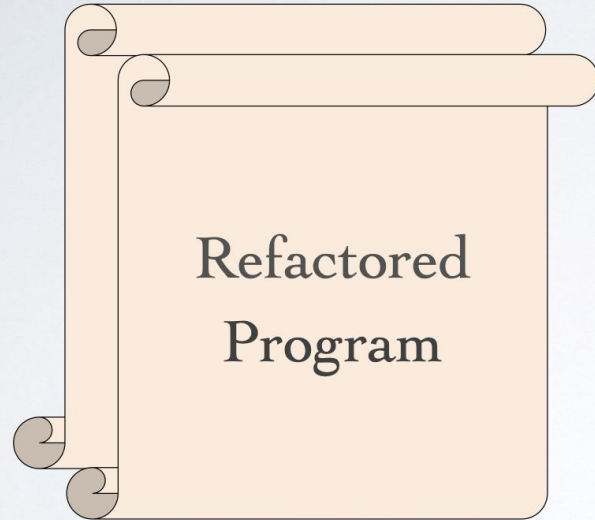
# Influenced By: ipywidgets

1. Even allows users to build their own widgets using a template
  - a. Cookiecutter templates to build widgets : typescript and javascript

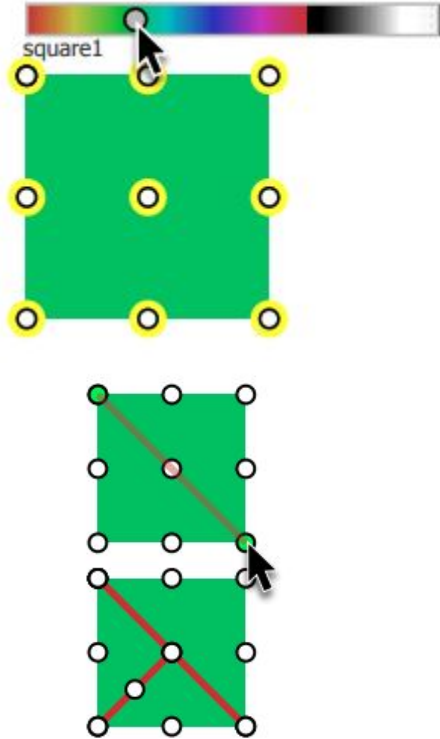


# Influenced by: Sketch-n-Sketch

Programming + Direct Manipulation?



# Influenced by: Sketch-n-Sketch



```
square1 = square 0 [158, 127] 156
```

```
svg (concat [  
  [square1]  
])
```

```
y = 127
```

```
x = 158
```

```
topLeft = [x, y]
```

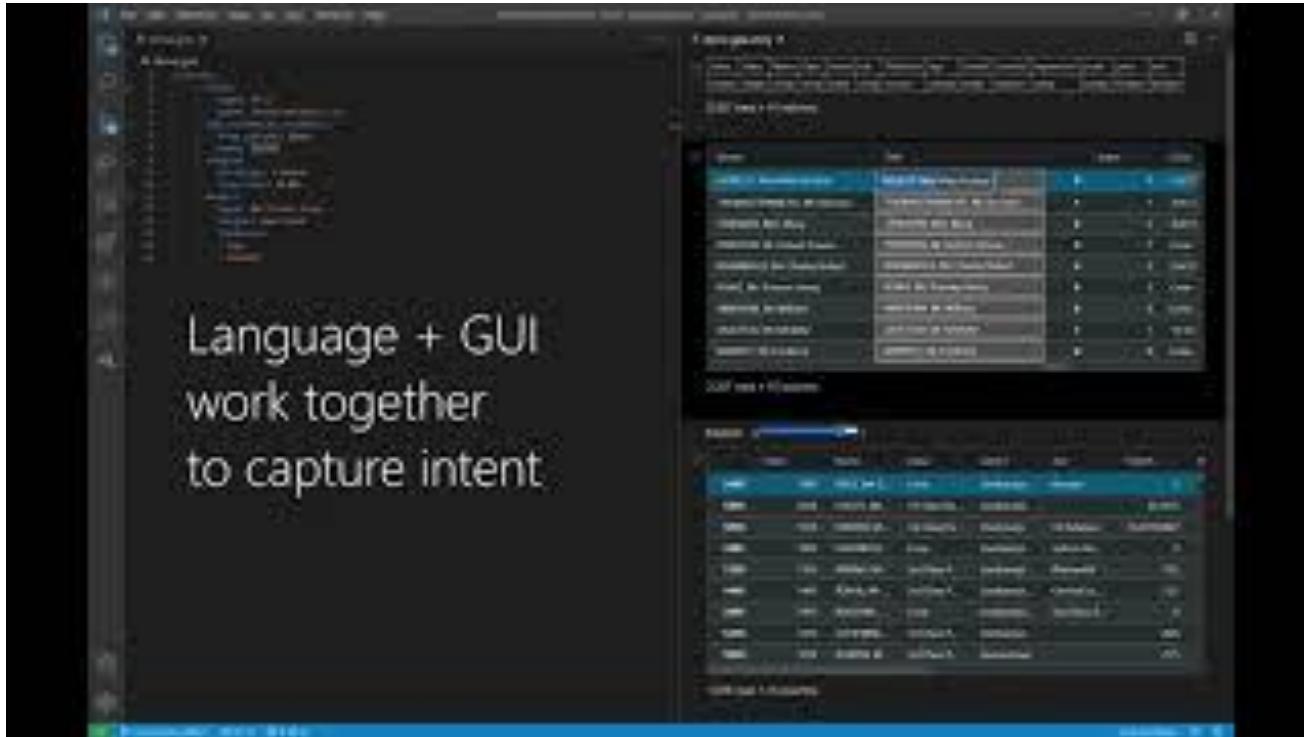
```
w = 156
```

```
square1 = square 140 topLeft w
```

```
y2 = y + w
```

```
line1 = line 0 5 topLeft [x + w, y2]
```

# Influences: Glinda: Supporting Data Science with Live Programming, GUIs and a Domain-specific Language



# Influences: Glinda: Supporting Data Science with Live Programming, GUIs and a Domain-specific Language

1. Introduce a user experience that extends an existing IDE with exploratory features to support data science workflows.
2. DSL for data science workflows



# Influences: Glinda: Supporting Data Science with Live Programming, GUIs and a Domain-specific Language

The screenshot shows the Glinda GUI with a file configuration on the left and a data table on the right. The configuration is for a file named 'titanic.csv' located at '/data/titanic.csv'. A menu is open over the configuration, showing options like 'filter', 'group', 'index', 'merge', 'model (decision\_tree)', and 'model (deepnet)'. The data table has columns: Index, Name, Class, Joined, Job, and Ticket. It displays 9 rows of data. At the bottom, it indicates '2,207 rows x 14 columns'.

Index	Name	Class	Joined	Job	Ticket
0	HEWLETT, ...	2nd Class P...	Southampt...		
1	THOMAS/T...	3rd Class P...	Cherbourg	Scholar	
2	CANAVAN, ...	3rd Class P...	Queenstown		
3	PERRITON, ...	Crew	Southampt...	Saloon Ste...	
4	BAINBRIGG...	2nd Class P...	Southampt...	Horse Trainer	
5	BLAKE, Mr ...	Crew	Southampt...	Fireman	
6	MINTRAM, ...	Crew	Southampt...	Fireman	
7	SAALFELD, ...	1st Class Pa...	Southampt...	Businessman	
8	BARRETT, ...	Crew	Southampt...	Leading Fir...	

The screenshot shows the Glinda GUI with a model configuration on the left and a data table with a performance graph on the right. The configuration is for a decision tree model with a target of 'Survived' and features 'Age' and 'Gender'. The data table has columns: index, Index, Name, Class, Joined, Job, TicketCost, Age, Gender, Survived, Department, probs, and pred. It displays 1,766 rows of data. A performance graph shows training and validation accuracy over 10 iterations. The training accuracy (blue line) starts at approximately 0.88 and decreases to about 0.80. The validation accuracy (orange line) starts at approximately 0.72 and increases to about 0.78.

index	Index	Name	Class	Joined	Job	TicketCost	Age	Gender	Survived	Department	probs	pred
integer	integer	string	string	string	string	number	number	string	boolean	string	number	boolean

1,766 rows x 14 columns

strategy: random

fraction: 0.8

type: decision\_tree  
target: Survived  
features: Age, Gender

training: 0.88, 0.83, 0.82, 0.81, 0.80  
validation: 0.72, 0.74, 0.76, 0.77, 0.78

# Influences: Glinda: Supporting Data Science with Live Programming, GUIs and a Domain-specific Language

## 1. Matching Recipes

A

```
def scatterplot(input, x, y, color, size, tooltip):  
    """  
    plot:  
        type: scatter_plot  
        x: { $type: columnname }  
        y: { $type: columnname }  
        color: { $type: columnname; $optional: true }  
        size: { $type: columnname; $optional: true }  
        tooltip: { $type: [columnname]; $optional: true }  
    """  
    args = {"x": x, "y": y, "tooltip": [x, y]}  
    if color in input:  
        args["color"] = color  
        args["tooltip"].append(color)  
    if size in input:  
        args["size"] = size  
        args["tooltip"].append(size)  
    if tooltip:  
        args["tooltip"] = tooltip  
    chart = alt.Chart(input).mark_circle().encode(**args)  
    respond_chart(chart)
```

B

```
spotify:  
- data:  
    type: file  
    path: /data/spotify.csv  
- sample:  
    strategy: random  
    size: 1000  
- plot:  
    type: scatter_plot  
    x: energy  
    y: tempo  
    color: playlist_genre
```

C

```
spotify1 = pandas_read_file('/data/spotify.csv')  
spotify2 = pandas_sample(spotify1, 'random', 1000)  
scatterplot(spotify2, 'energy', 'tempo',  
            'playlist_genre', None, None)  
spotify = spotify2
```