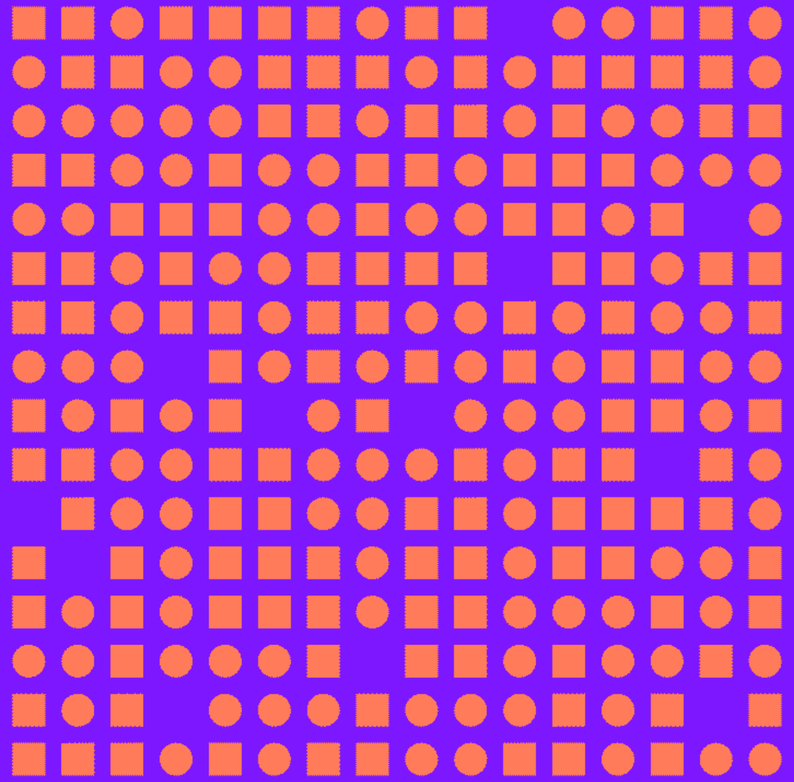
TypeFox

Notebook Internals in Theia: A Deep Dive

Mark Sujew – TheiaCon ‘25



Who am I?

Mark Sujew

TypeFox

HAW HAMBURG

THEIA

langium



Agenda

1. Explaining Notebooks
2. Initial IDE Integrations
3. Native IDE Integration
4. Technical Challenges



Explaining Notebooks

Interactive Python (IPython)

```
Python 3.6.3 | packaged by conda-forge | (default, Nov  4 2017, 10:13:32)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.0.0.dev -- An enhanced Interactive Python. Type '?' for help.

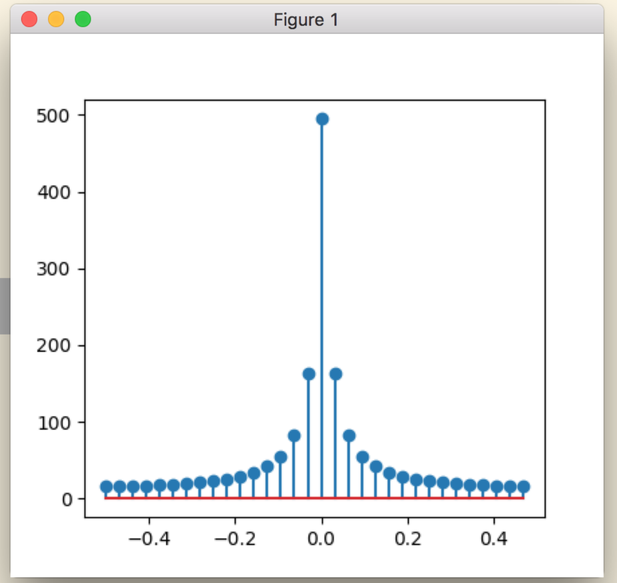
In [1]: from numpy.fft import *
   ...: from numpy import arange
   ...: a = arange(32)
   ...: A = fft(a)
   ...: f = fftfreq(32)

In [2]: %matplotlib tk

In [3]: from matplotlib.pyplot import stem

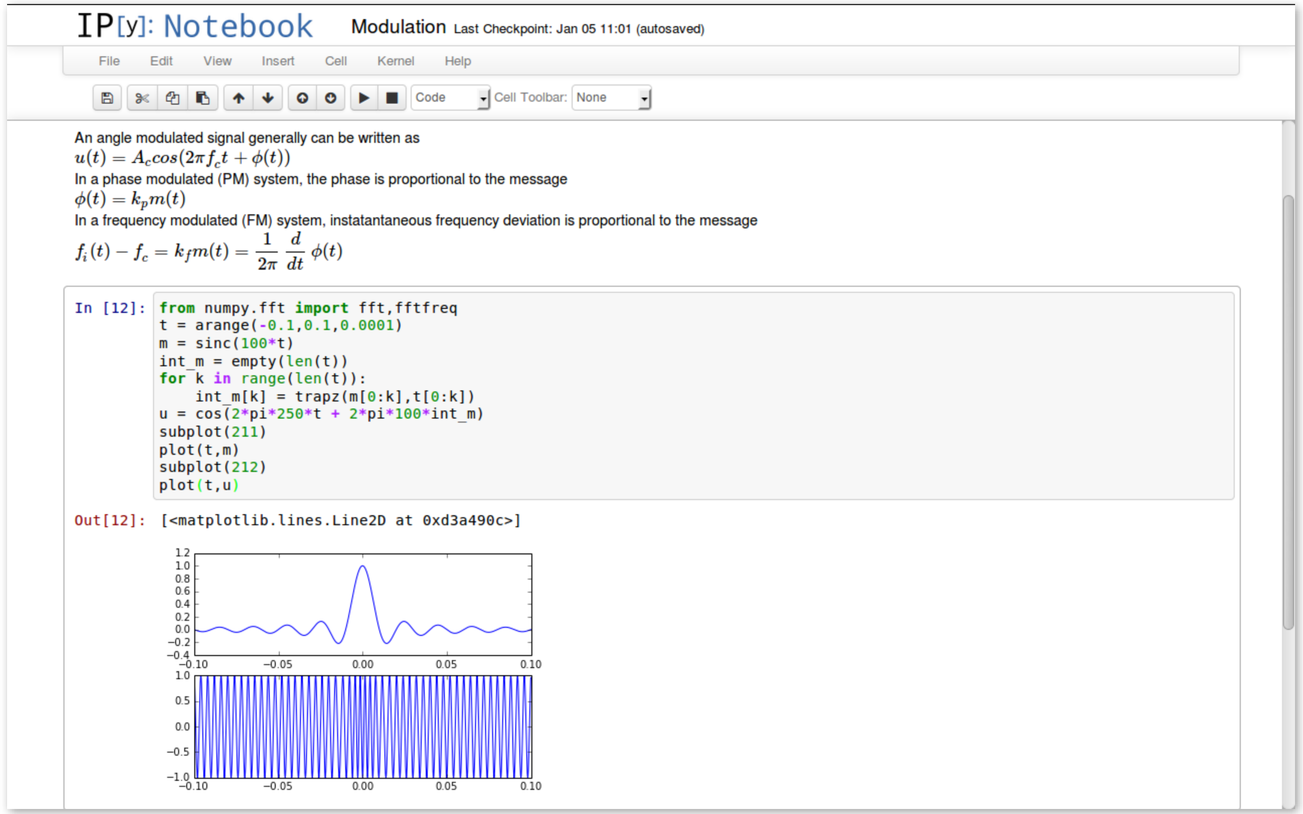
In [4]: stem(f, abs(A))
Out[4]: <Container object of 3 artists>

In [5]: _.
          add_callback        eventson
          baseline            get_children
          count()             get_label
```



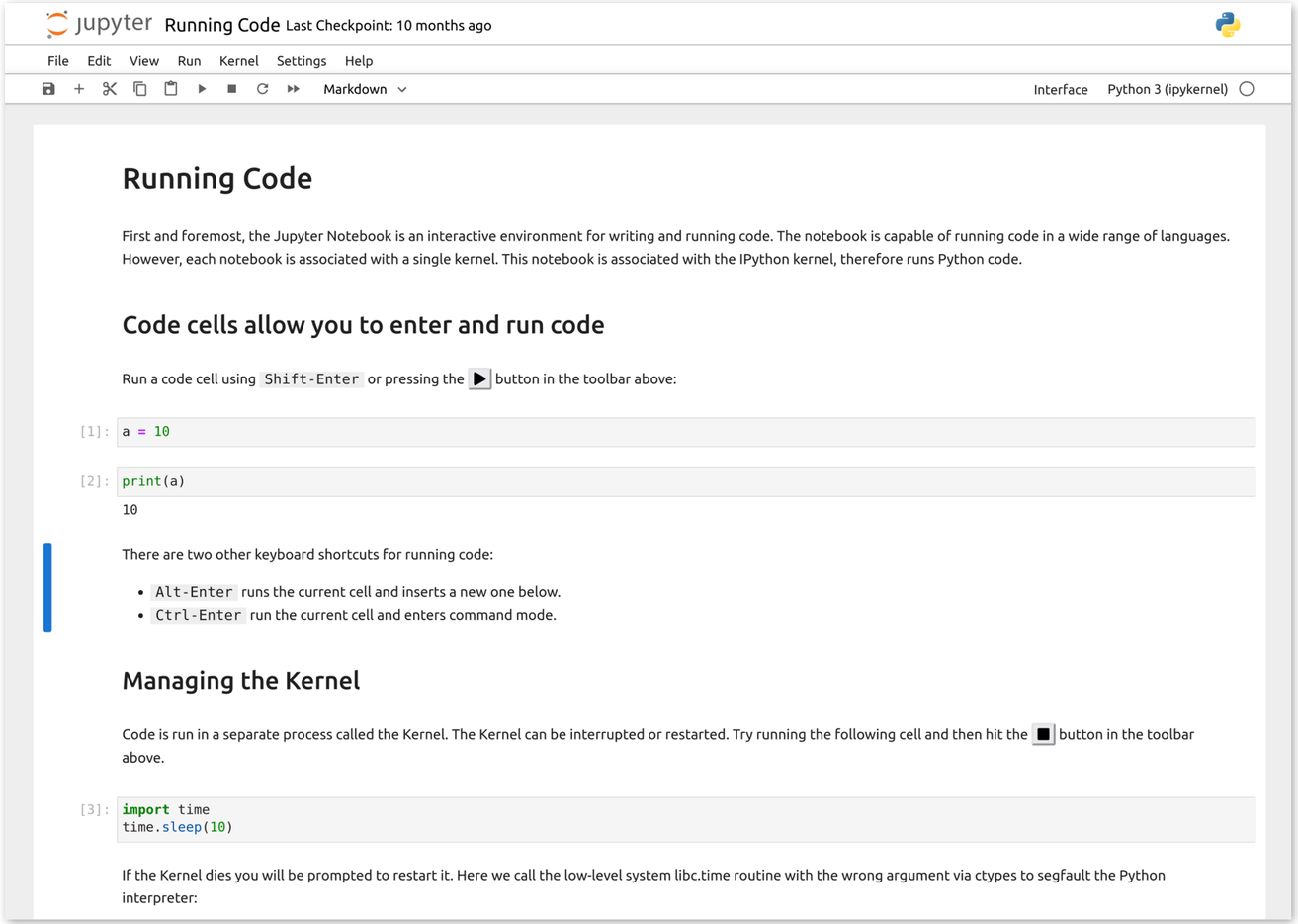
- Started in 2001
- Execute Python code snippets
- Multimedia Output

Python Notebooks



- Released in 2011
- Main Features from IPython
- Code **and documentation**
- Presented as web app

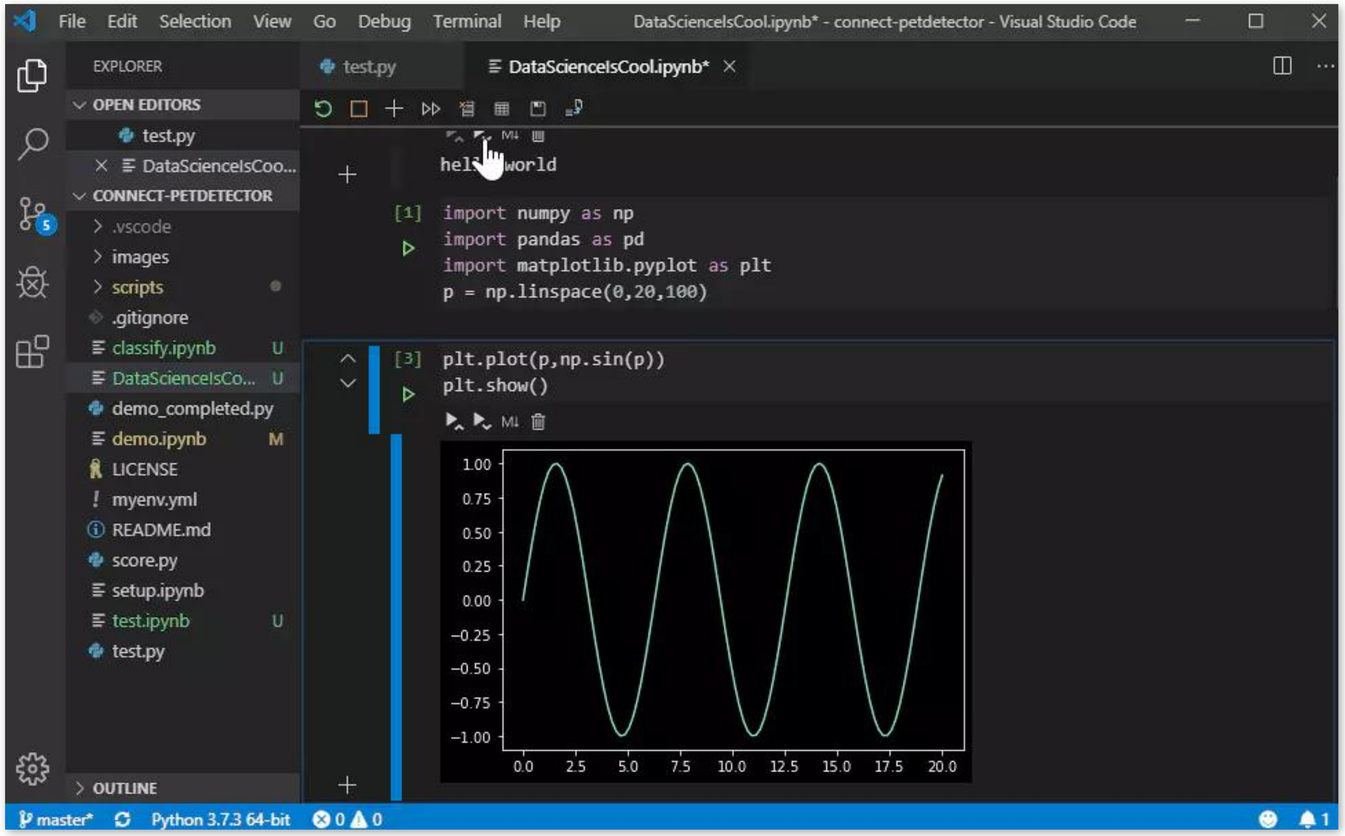
Jupyter Notebook



- Split off in 2014
- Language agnostic
- Uses IPython as a *Kernel*
- Hugely popular

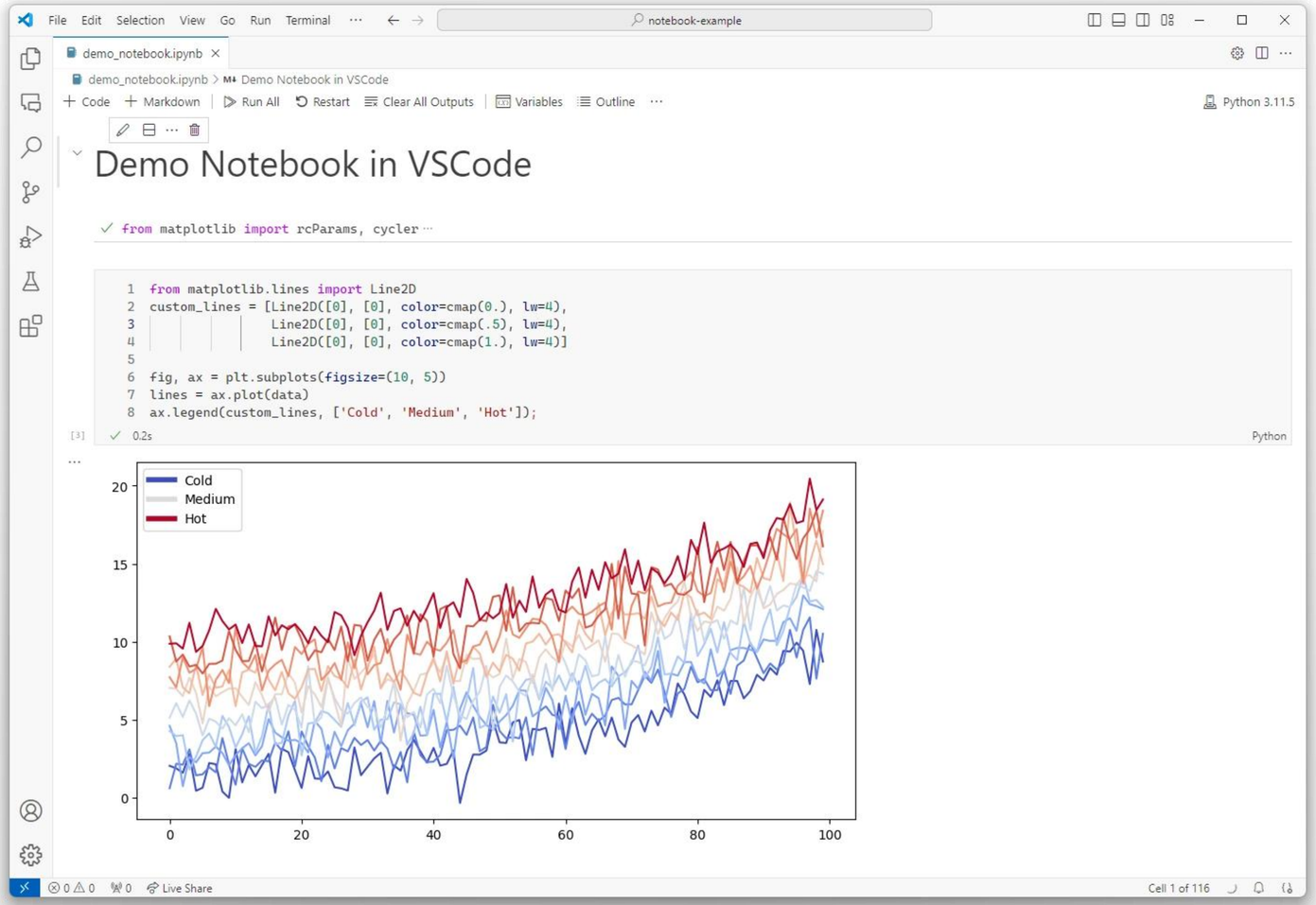


Jupyter Extension for VS Code + Theia



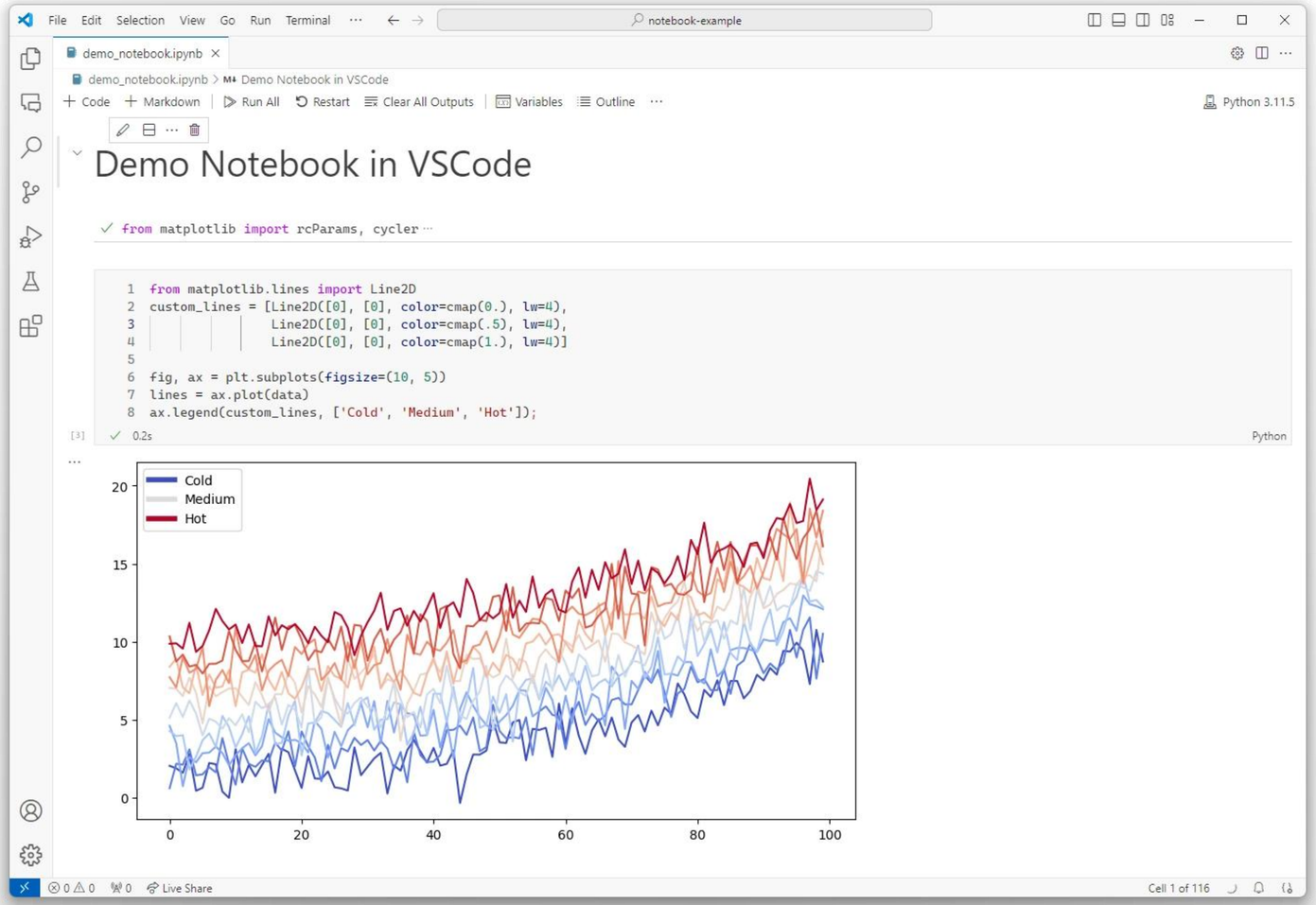
- Created by Microsoft in 2019
- Displays as a *Webview*
- **Works in Theia out-of-the-box**
- Complicated Language support
- Fairly slow

Native Notebooks in VS Code



- New UI feature in 2021
- Language agnostic
- Support for custom kernels
- VS Code API

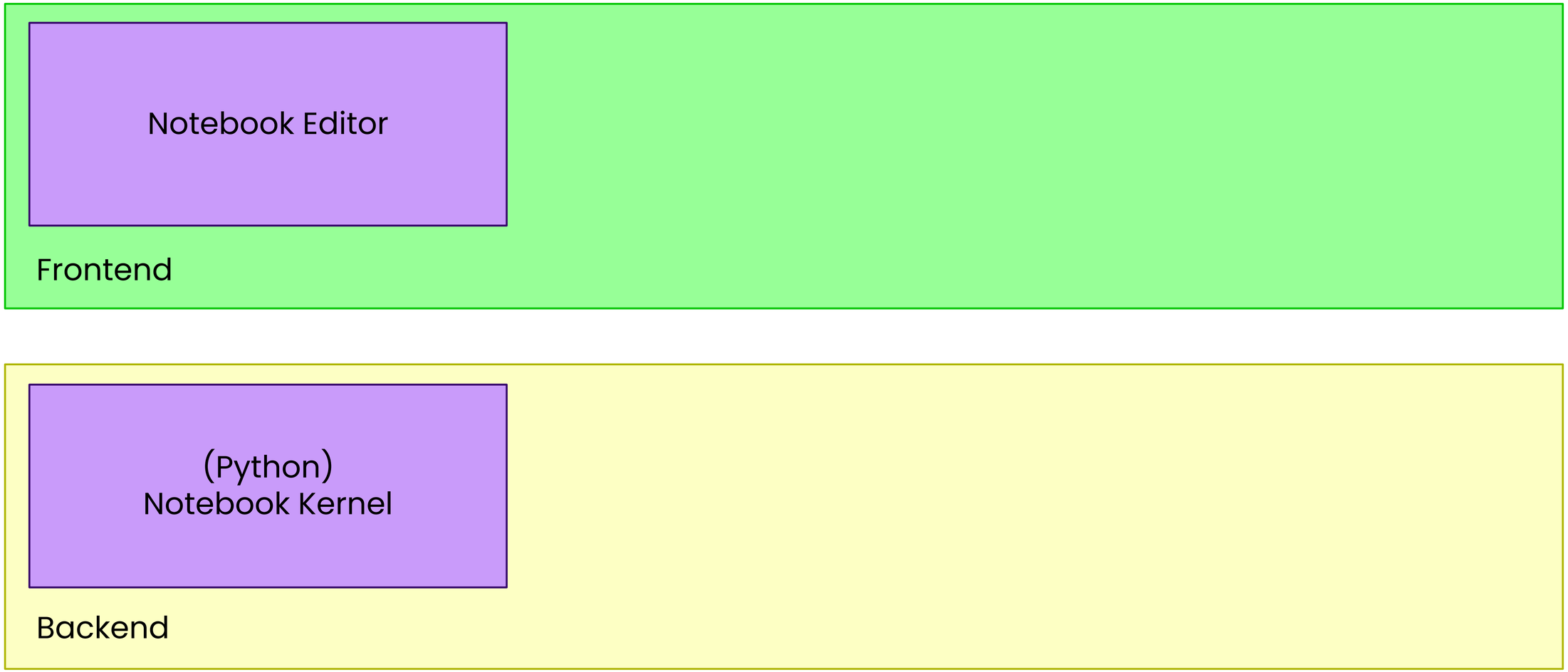
Native Notebooks in VS Code + Theia (?)



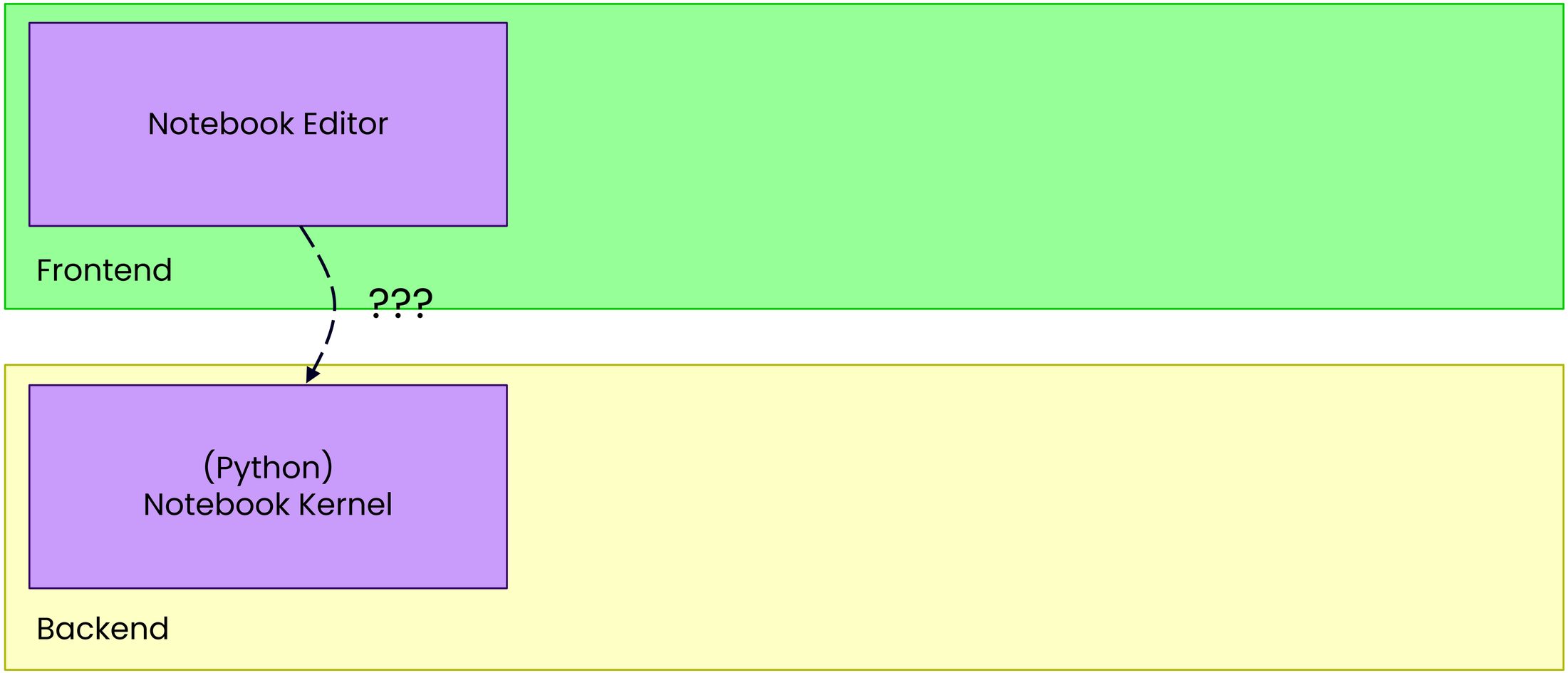
- Not based on the webview API
- Completely new frontend UI
- Additional VS Code API

... a lot of work for Theia

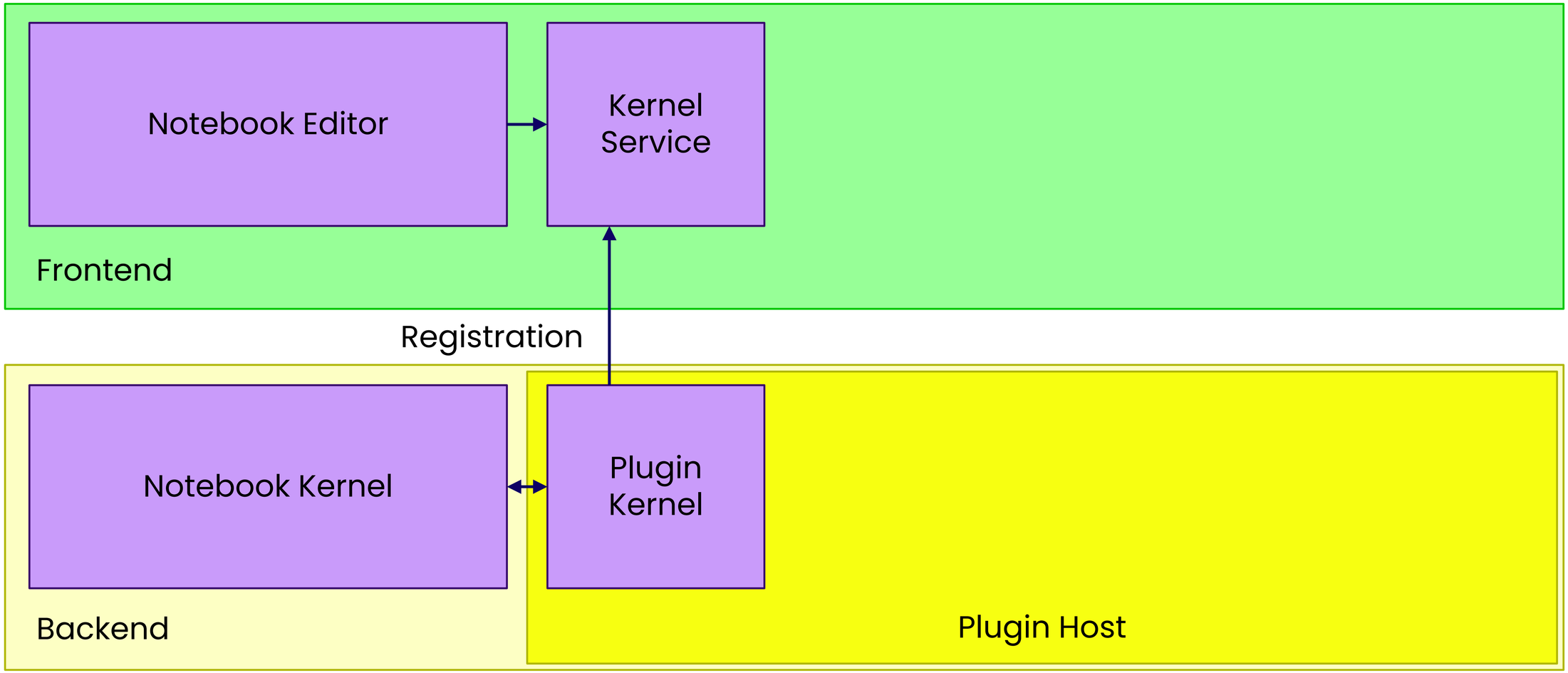
Notebook Communication



Notebook Communication

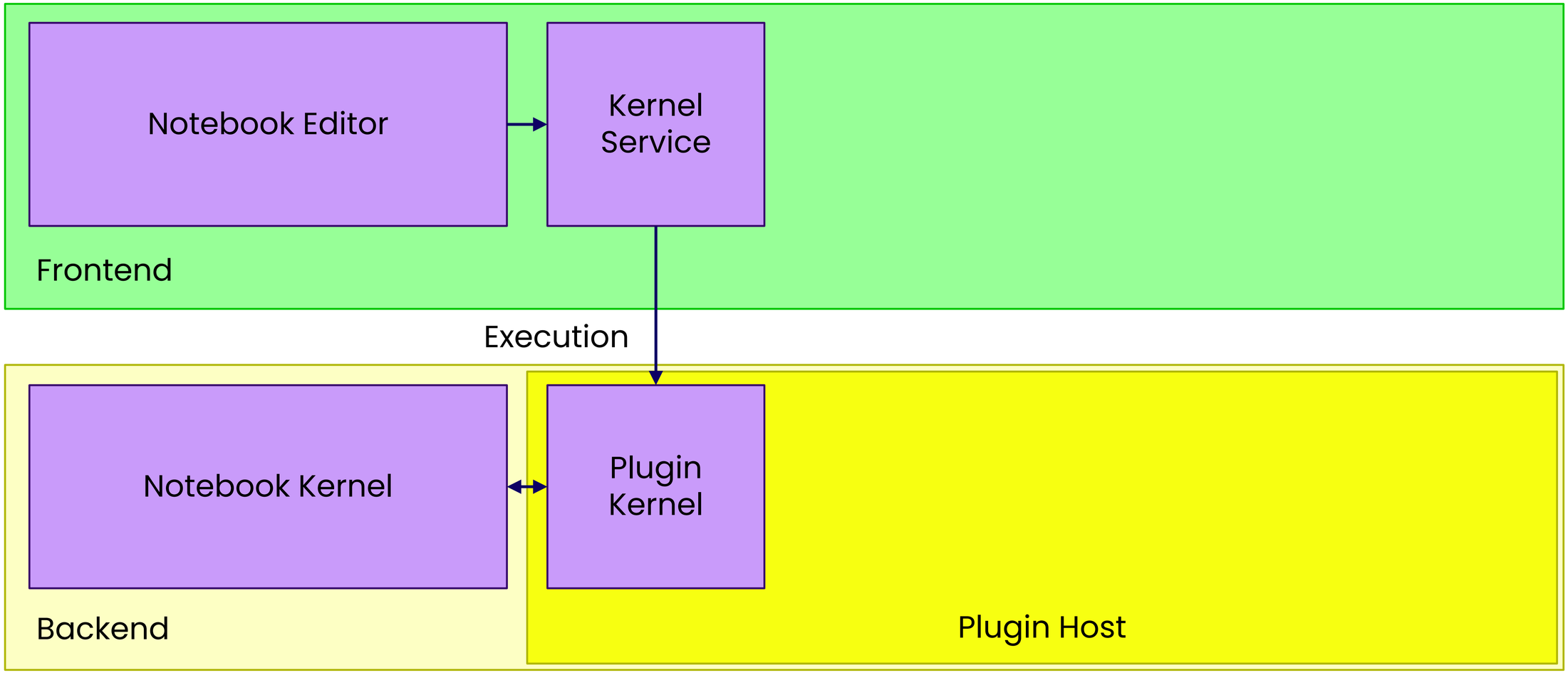


Notebook Communication



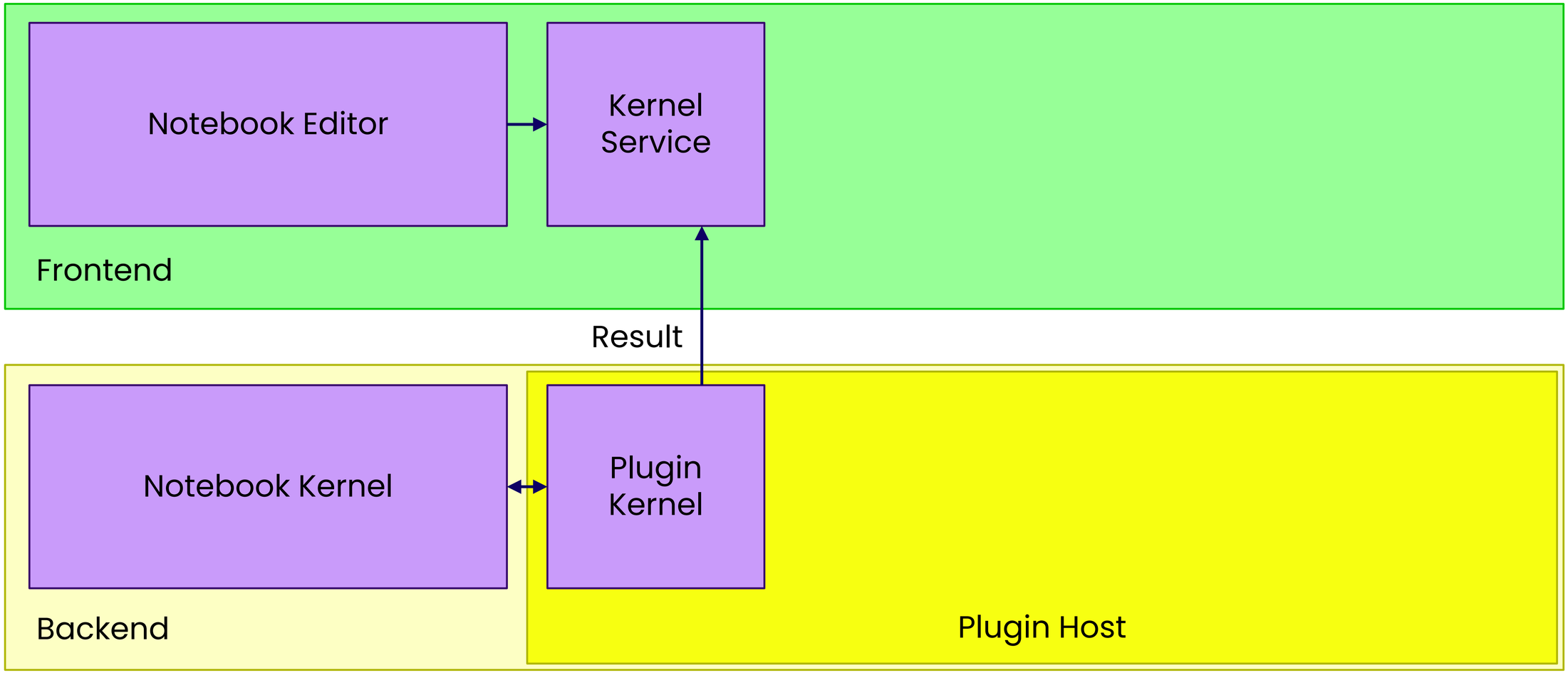


Notebook Communication

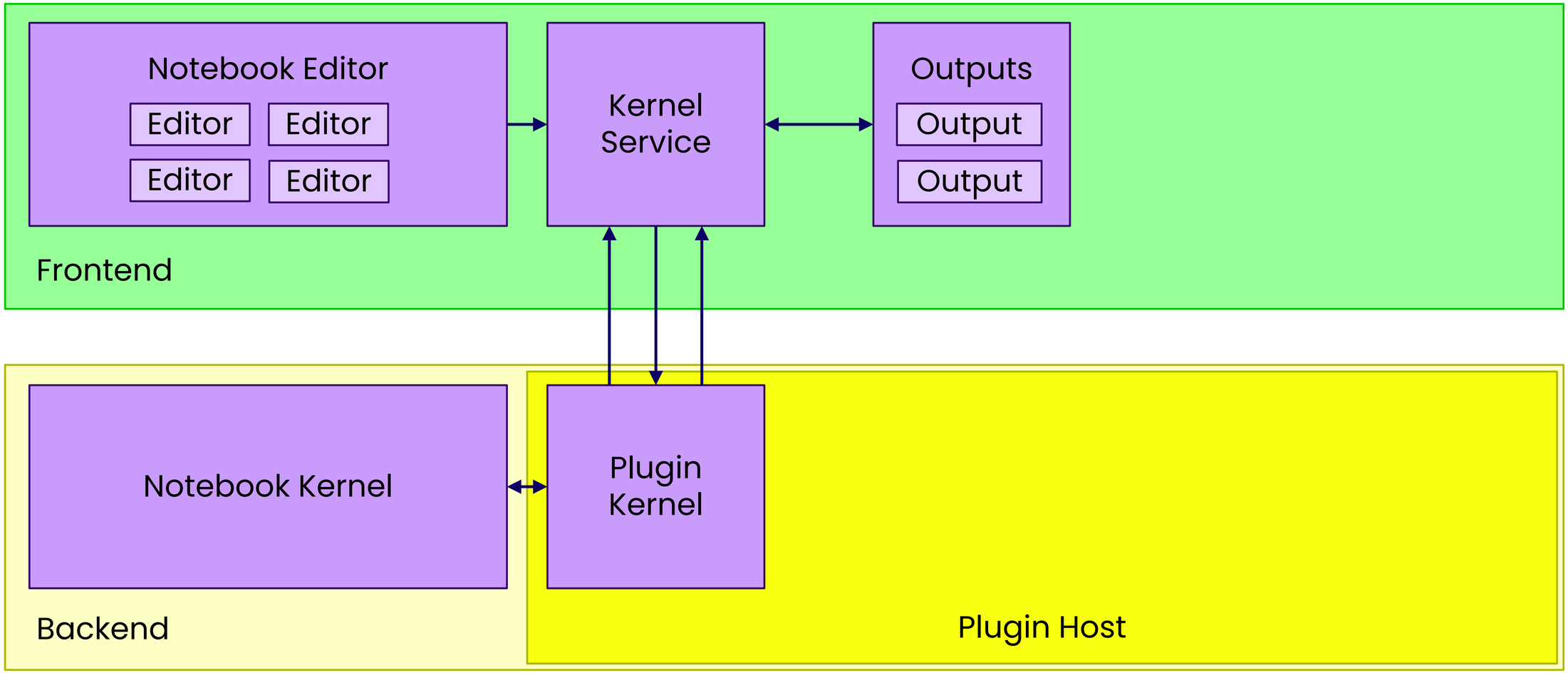




Notebook Communication

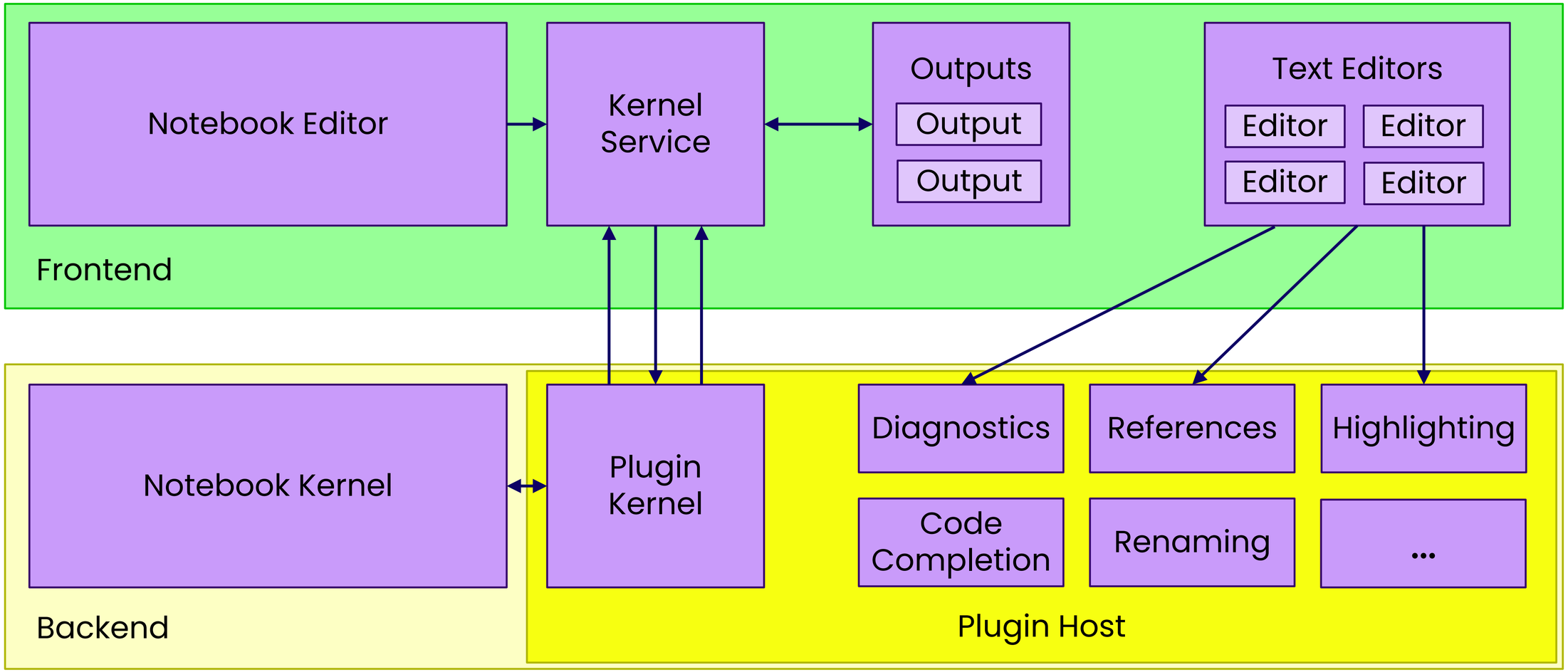


Notebook Communication



Notebook Communication

~25 000 LoC



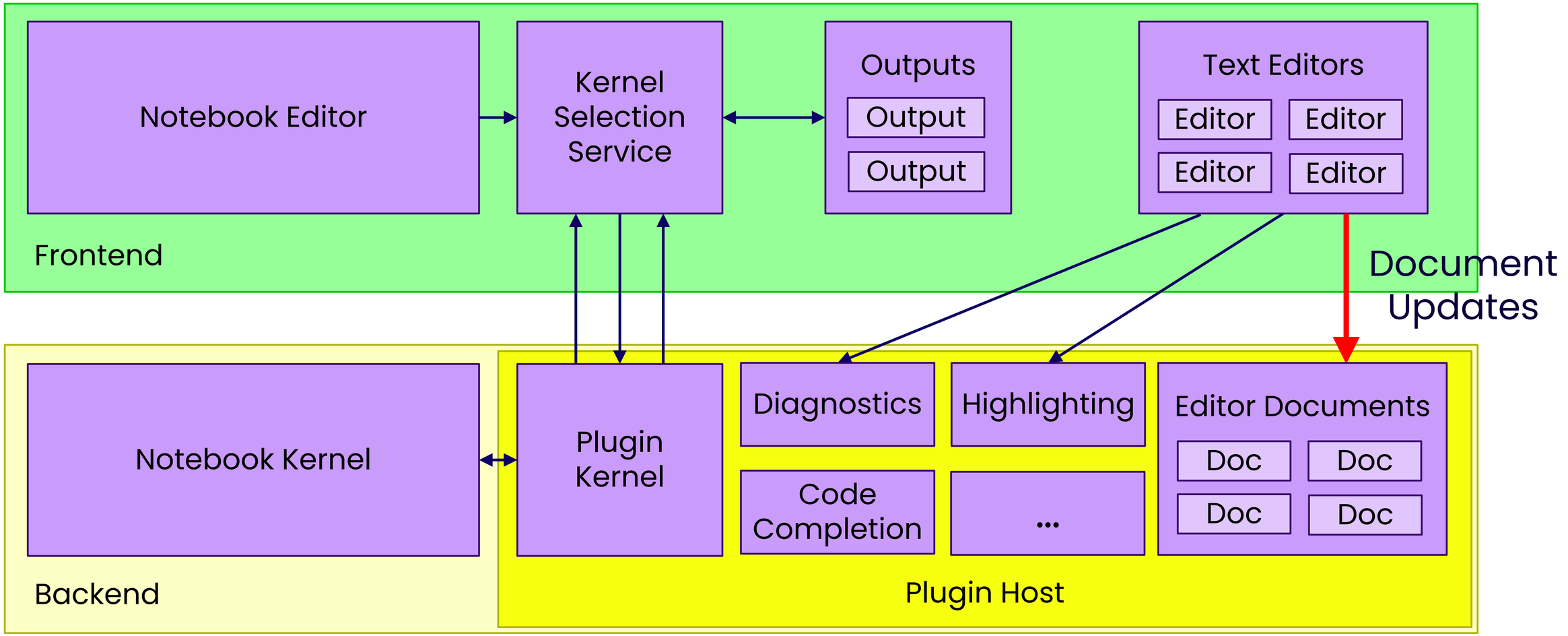
Various Performance Issues...

- Rendering a large amount of Monaco editors
  → Lazy rendering in DOM
- RPC bottleneck when opening a large notebook
  → Batching RPC calls
- Rendering a lot of outputs
  → Uhm, let's see...

- Browser specific issues...



Notebook Communication - Performance



Notebook Layouting

```
1 pop_dict
```

Python

```
{'Hovedstaden': 1.84,
 'Sjælland': 0.84,
 'Syddanmark': 1.22,
 'Midtjylland': 1.32,
 'Nordjylland': 0.59}
```

```
1 pop_series = pd.Series(pop_dict)
2 pop_series
```

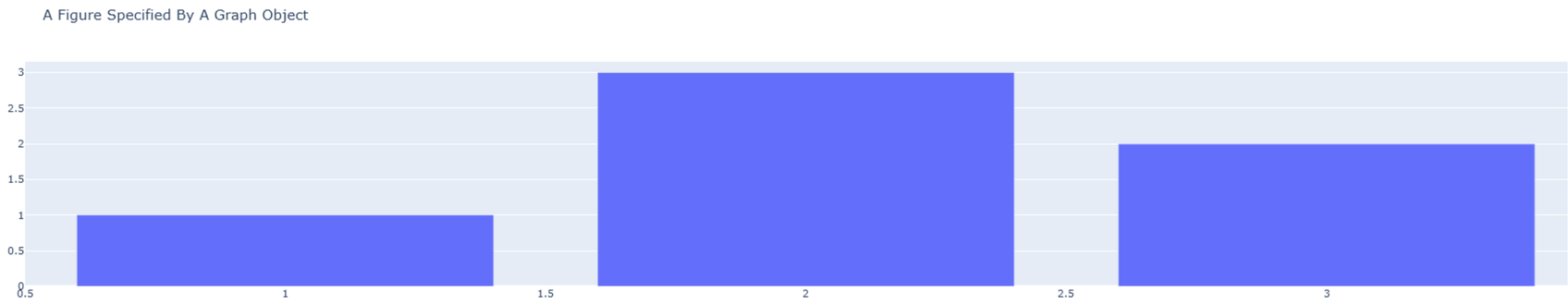
Python

```
Hovedstaden    1.84
Sjælland       0.84
Syddanmark     1.22
Midtjylland    1.32
Nordjylland    0.59
dtype: float64
```

```
1 import plotly.graph_objects as go
2
3 fig = go.Figure(
4     data=[go.Bar(x=[1, 2, 3], y=[1, 3, 2])],
5     layout=dict(title=dict(text="A Figure Specified By A Graph Object"))
6 )
7
8 fig.show()
```

0,0s

Python



Notebook Layouting – Naive View

Individual Iframes

```
1 pop_dict
```

Python

```
{'Hovedstaden': 1.84,
 'Sjælland': 0.84,
 'Syddanmark': 1.22,
 'Midtjylland': 1.32,
 'Nordjylland': 0.59}
```

```
1 pop_series = pd.Series(pop_dict)
2 pop_series
```

Python

```
Hovedstaden    1.84
Sjælland       0.84
Syddanmark     1.22
Midtjylland    1.32
Nordjylland    0.59
dtype: float64
```

```
1 import plotly.graph_objects as go
2
3 fig = go.Figure(
4     data=[go.Bar(x=[1, 2, 3], y=[1, 3, 2])],
5     layout=dict(title=dict(text="A Figure Specified By A Graph Object"))
6 )
7
8 fig.show()
```

0,0s

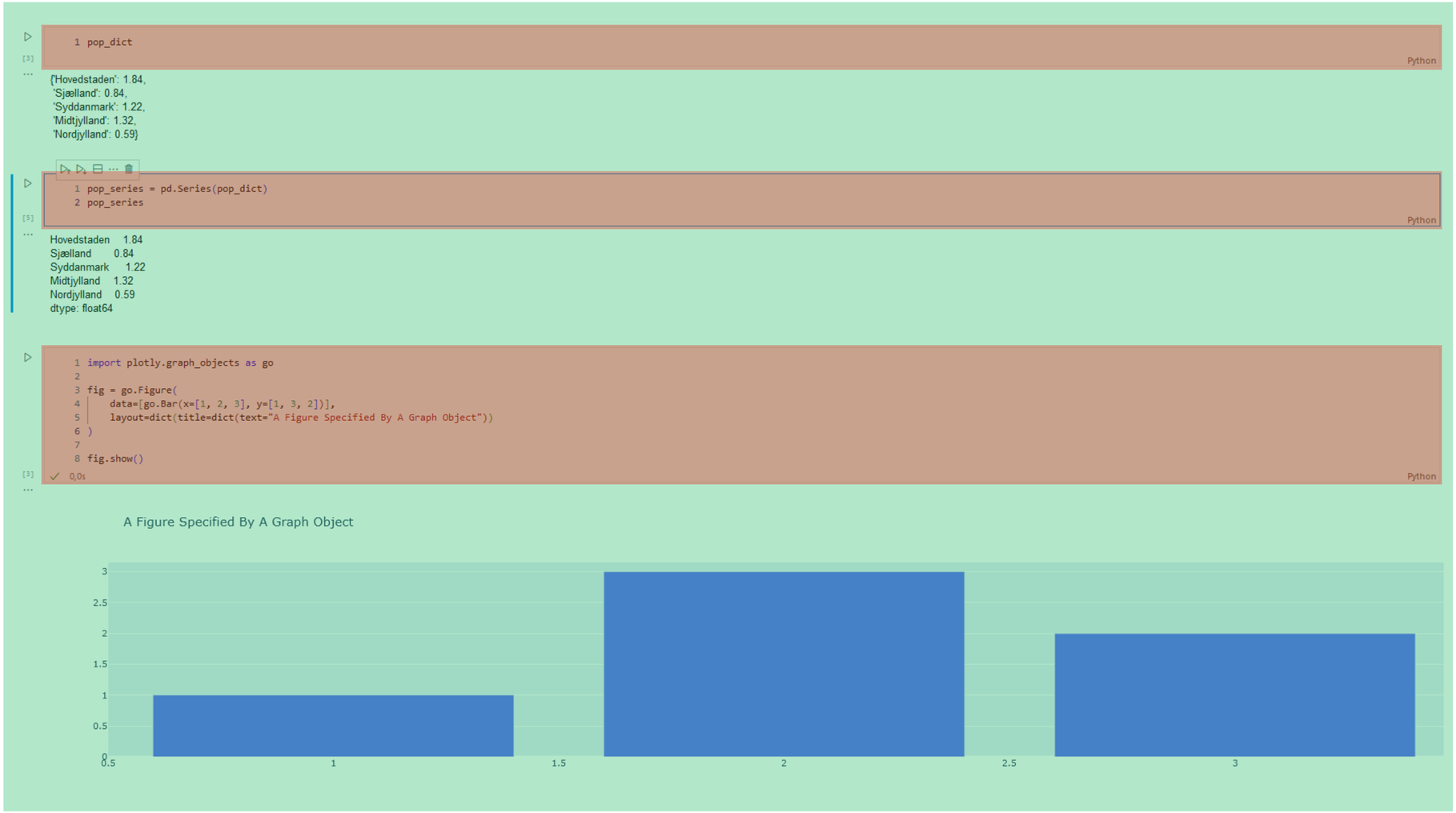
Python



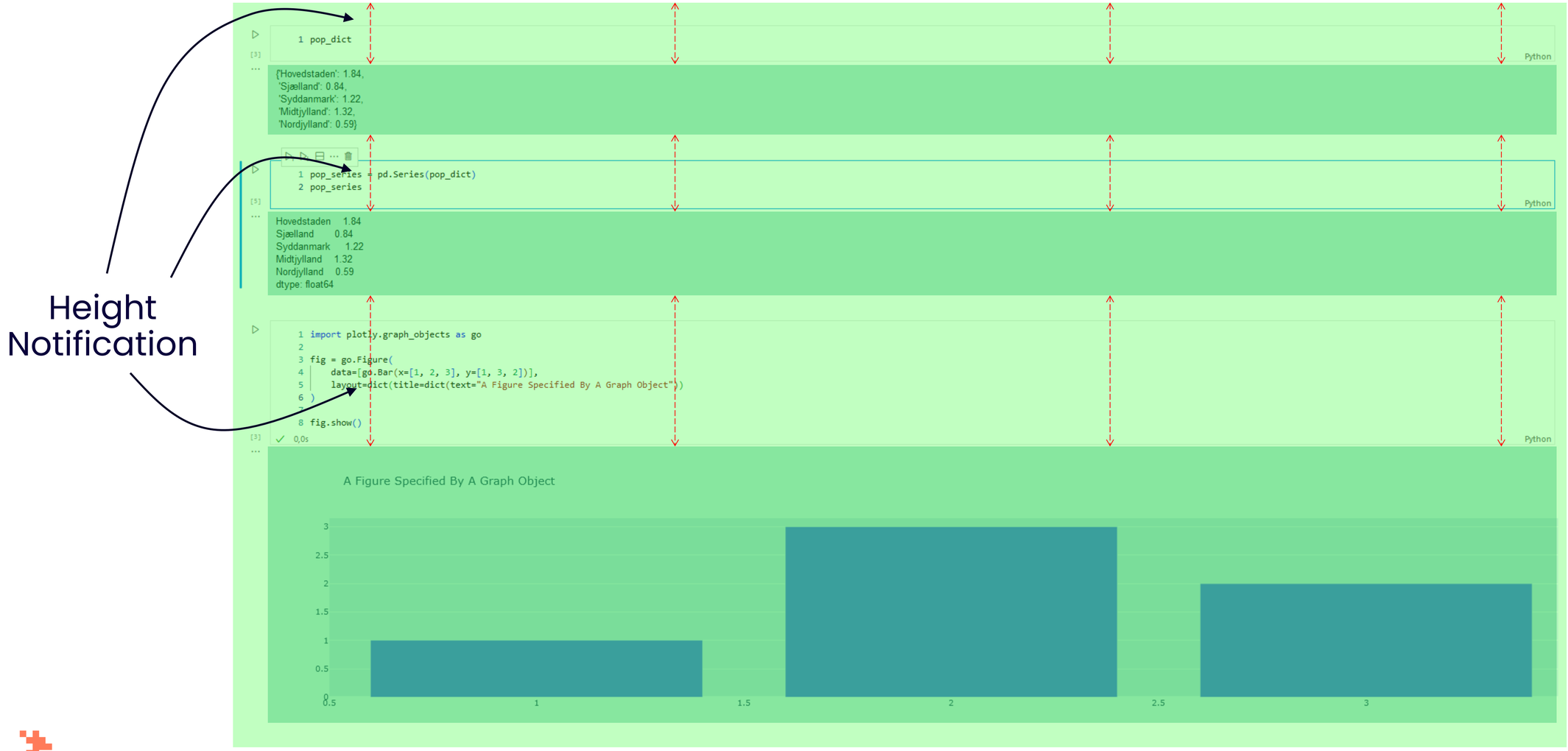
Notebook Layouting – Backlayer Webview

Singleton

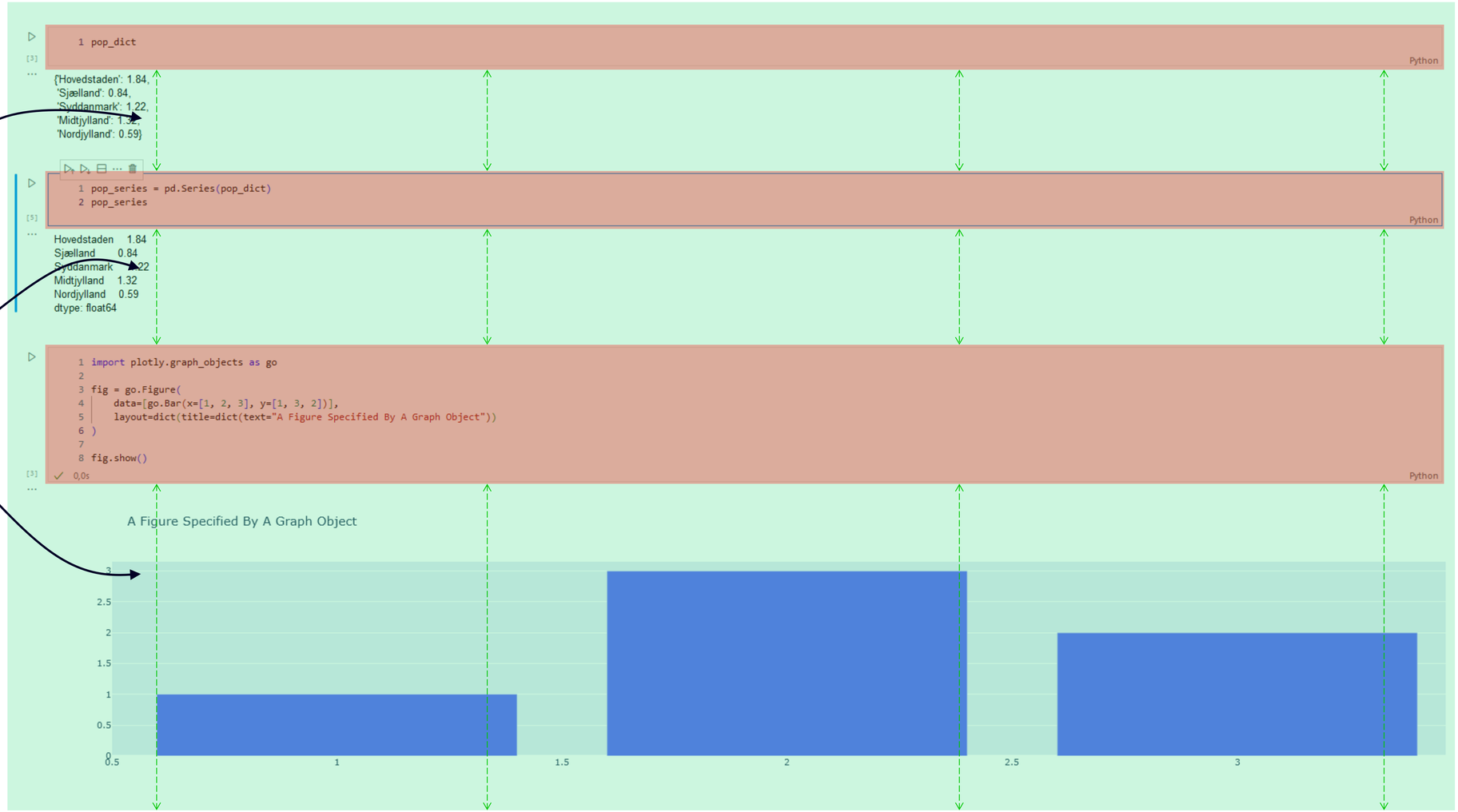
Iframe



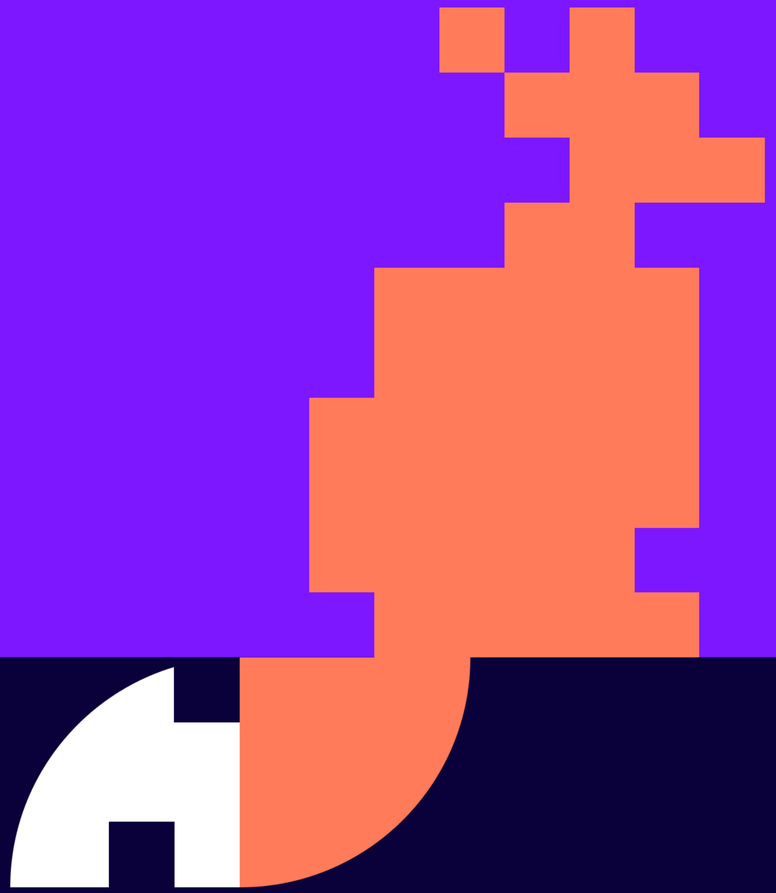
Notebook Layouting – Backlayer Webview



Notebook Layouting – Editor Overlay



TypeFox



Questions?