

DYALOG

Belfast 2018

# Jupyter Notebooks

Adám Brudzewsky

# What are notebooks?

A *notebook* combines the functionality of

- a word processor — handles formatted text

- a *shell* or *kernel* — executes statements in a programming language and includes output inline

- a rendering engine — renders HTML in addition to plain text



# Example notebook

using Python

global density  
of metal bands

Creating a world map of x

Not secure | ramiro.org/notebook/metal-bands-map/

Apps | APL | EKL | kdb - Interprocess Co | The APL Orchard | ch

## Plot the map

We'll use the handy `plot` method available on `GeoDataFrame` objects. To make sure the map shows all countries, including those without data on metal bands, we have to plot these two sets separately. If you like to learn why, check out this notebook on [creating choropleth maps using GeoPandas](#).

In the final code section, we create two separate data frames `known` and `unknown`. The `known` countries will be plotted using a colormap that seemed appropriate and the `Jenks classification method`, that reduces the variance within classes and maximize the variance between classes. There will be 9 different classes with darker colors indicating higher band ratios.

The `unknown` countries will be shown with a white background and a striped pattern. We also add some descriptive text, move the legend to the lower left part of the map and set the legend's size.

```
known = world.dropna(subset=['band_ratio'])
unknown = world[world['band_ratio'].isna()]

ax = known.plot(column='band_ratio', cmap='inferno_r', figsize=(20, 12), scheme='fisher_jenks', k=9, legend=True, edgecolor='#aaaaaa')
unknown.plot(ax=ax, color='#ffffff', hatch='///', edgecolor='#aaaaaa')

ax.set_title('Metal bands per 1 million people', fontdict={'fontsize': 20}, loc='left')
description = '''
Based on existing and split-up bands listed on metalstorm.net in 2017 made available in the dataset Metal Bands by Nation Kaggle.com/mrpantherson
and population estimates from naturalearthdata.com • Author: Ramiro Gómez - ramiro.org'''
ax.annotate(description, xy=(0.07, 0.1), size=12, xycoords='figure fraction')

ax.set_axis_off()
legend = ax.get_legend()
legend.set_bbox_to_anchor((-11, -4))
legend.prop.set_size(12)
```

### Metal bands per 1 million people

Color	Density Range (per 1 million people)
Light Yellow	0.00 - 1.08
Yellow	1.08 - 3.77
Orange	3.77 - 6.84
Dark Orange	6.84 - 12.73
Red	12.73 - 17.36
Dark Red	17.36 - 47.05
Purple	47.05 - 52.43
Dark Purple	52.43 - 57.71
Black	57.71 - 81.51

Based on existing and split-up bands listed on [metalstorm.net](#) in 2017 made available in the dataset Metal Bands by Nation [kaggle.com/mrpantherson/metal-by-nation](#) and population estimates from [naturalearthdata.com](#) • Author: Ramiro Gómez - [ramiro.org](#)

## Conclusion

The map above and the one [posted on reddit six years ago](#) show similar patterns regarding regions with high and low metal band ratios. Moreover, it is obvious that our dataset comprises less countries and, looking at the actual numbers, has a lot less records in total.

# Exam

## using

### global

### of m

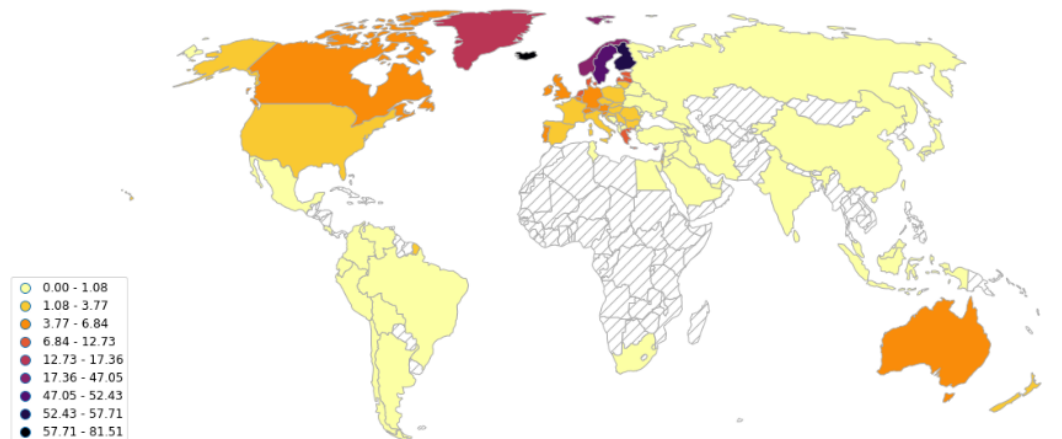
```
known = world.dropna(subset=['band_ratio'])
unknown = world[world['band_ratio'].isna()]

ax = known.plot(column='band_ratio', cmap='inferno_r', figsize=(20, 12), scheme='fisher_jenks', k=9, legend=True, edgecolor='#aaaaaa')
unknown.plot(ax=ax, color='#ffffff', hatch='//', edgecolor='#aaaaaa')

ax.set_title('Metal bands per 1 million people', fontdict={'fontsize': 20}, loc='left')
description = '''
Based on existing and split-up bands listed on metalstorm.net in 2017 made available in the dataset Metal Bands by Nation kaggle.com/mrpantherso
and population estimates from naturalearthdata.com • Author: Ramiro Gómez - ramiro.org'''
ax.annotate(description, xy=(0.07, 0.1), size=12, xycoords='figure fraction')

ax.set_axis_off()
legend = ax.get_legend()
legend.set_bbox_to_anchor((.11, .4))
legend.prop.set_size(12)
```

Metal bands per 1 million people



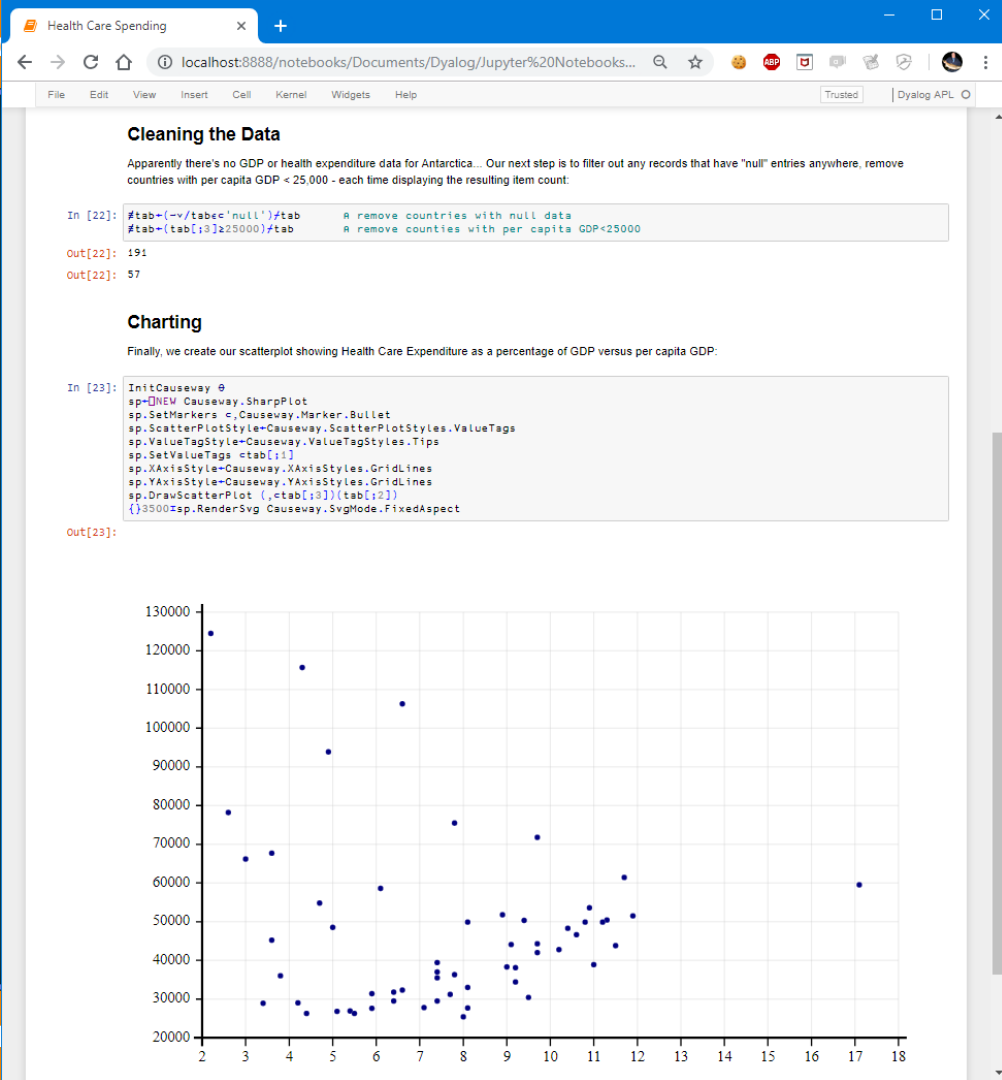
Based on existing and split-up bands listed on metalstorm.net in 2017 made available in the dataset Metal Bands by Nation kaggle.com/mrpantherson/metal-by-nation and population estimates from naturalearthdata.com • Author: Ramiro Gómez - ramiro.org



# Example notebook

using Dyalog APL

health care expenditure  
vs GDP per capita



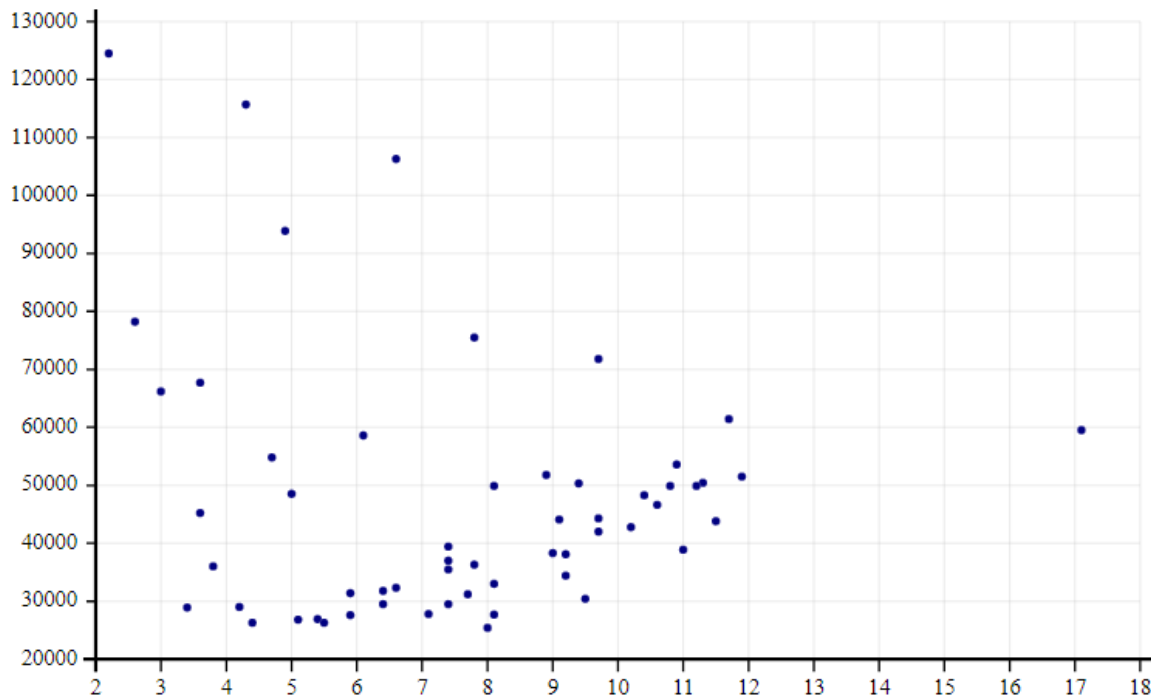
Exa

usin

hea  
vs G

```
In [23]: InitCauseway @
sp=NEW Causeway.SharpPlot
sp.SetMarkers c,Causeway.Marker.Bullet
sp.ScatterPlotStyle=Causeway.ScatterPlotStyles.ValueTags
sp.ValueTagStyle=Causeway.ValueTagStyles.Tips
sp.SetValueTags ctab[:1]
sp.XAxisStyle=Causeway.XAxisStyles.GridLines
sp.YAxisStyle=Causeway.YAxisStyles.GridLines
sp.DrawScatterPlot (,ctab[:3])(tab[:2])
{}3500zsp.RenderSvg Causeway.SvgMode.FixedAspect
```

Out[23]:



# Notebook benefits

A single document that combines explanations with executable code and its output — an ideal way to provide:

- reproducible research results

- documentation of processes

- instructions

- tutorials and training materials of all shapes and sizes

A digital learning environment for computational thinking



# What is *Jupyter* notebook?

First notebook: Mathematica 1.0 in '88

Jupyter notebook is a part of

Project Jupyter, a nonprofit to

*develop open-source software,  
standards, and services for  
interactive computing across  
dozens of programming languages*

beginning with **Julia**, **Python**, **R**, and now over 70  
languages, including Dyalog APL





# What is *Jupyter* notebook?

First notebook: Mathematica 1.0 in '88

Jupyter notebook is a part of

Project Jupyter, a nonprofit to

*develop open-source software,  
standards, and services for  
interactive computing across  
dozens of programming languages*

beginning with **Julia**, **Python**, **R**, and now over 70  
languages, including Dyalog APL

ONE OF THE MOST  
SIGNIFICANT ADVANCES  
IN THE SCIENTIFIC  
COMPUTING ARENA  
UNIVERSITY CORPORATION  
FOR ATMOSPHERIC  
RESEARCH



# Ways to use Jupyter notebooks

On your own PC after installing a Jupyter notebook server

With an online notebook server like [cocalc.com](https://cocalc.com)

Save notebook with output and use a notebook viewer

Export to HTML, PDF,  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ , etc.



# Local notebook server — Python

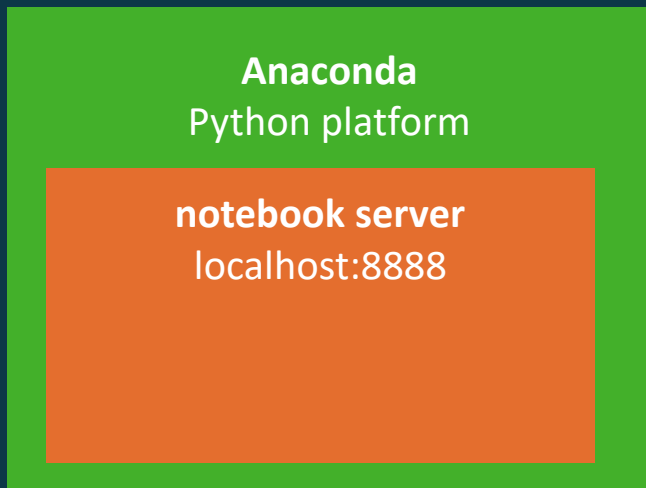


# Local notebook server — Python

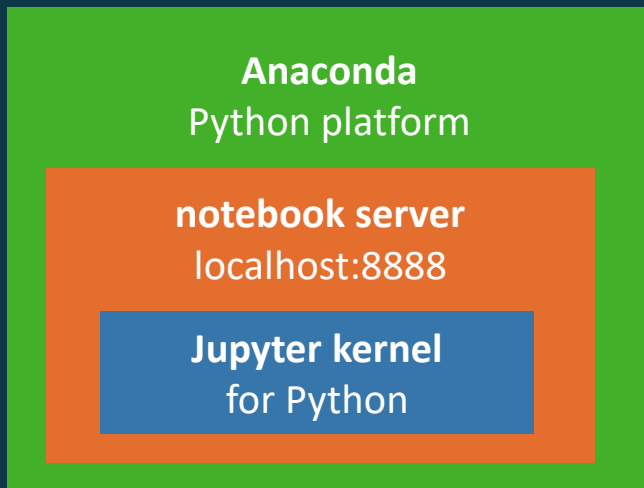
Anaconda  
Python platform



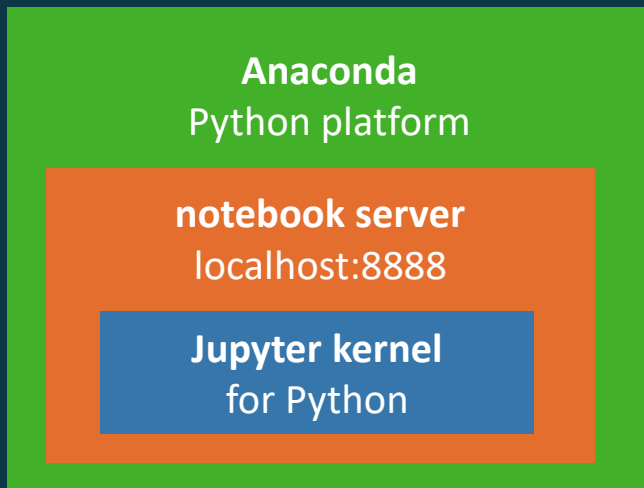
# Local notebook server — Python



# Local notebook server — Python



# Local notebook server — Python



# Local notebook server — Python

web browser

Anaconda  
Python platform

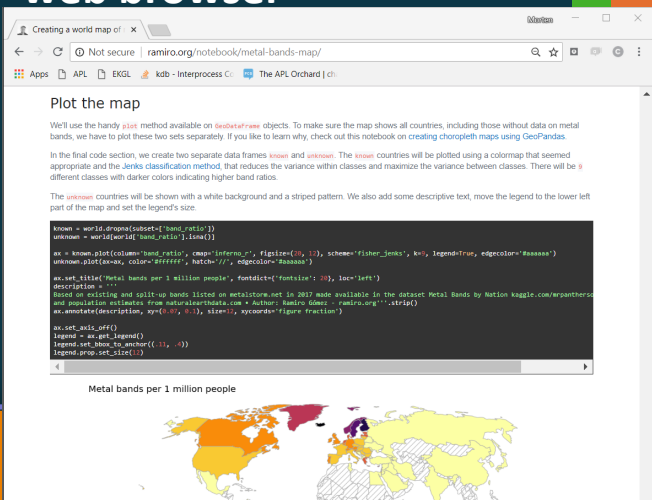
notebook server  
localhost:8888

interpreter  
Python

Jupyter kernel  
for Python



Jupyter Notebooks





# Local notebook server — Python

web browser

Anaconda  
Python platform

notebook server  
localhost:8888

interpreter  
Python

Jupyter kernel  
for Python



Jupyter Notebooks



# Local notebook server — Python

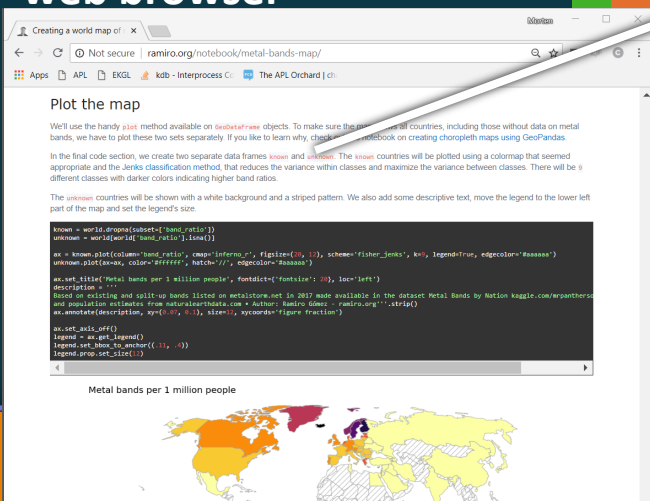
web browser

Anaconda  
Python platform

notebook server  
localhost:8888

Jupyter kernel  
for Python

interpreter  
Python



Jupyter Notebooks



# Local notebook server — Python

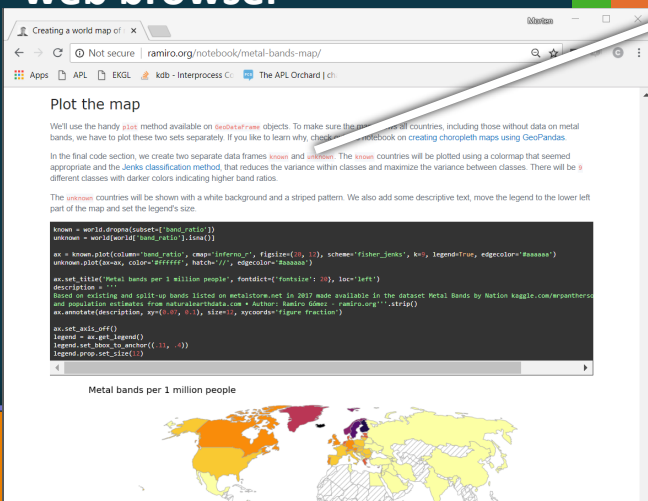
web browser

Anaconda  
Python platform

notebook server  
localhost:8888

Jupyter kernel  
for Python

interpreter  
Python



Jupyter Notebooks



# Local notebook server — Python

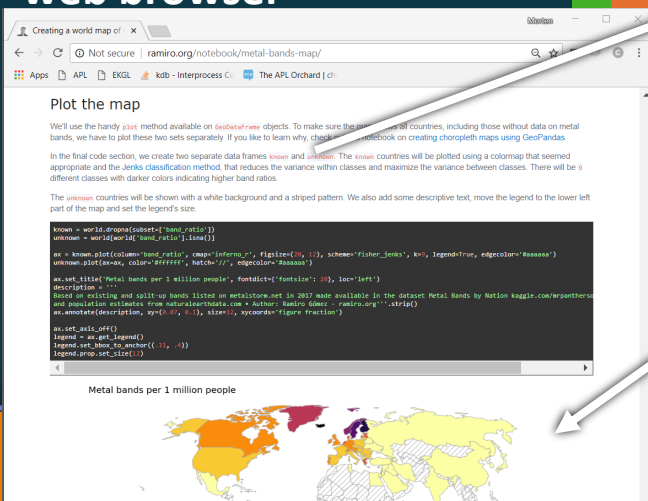
web browser

Anaconda  
Python platform

notebook server  
localhost:8888

Jupyter kernel  
for Python

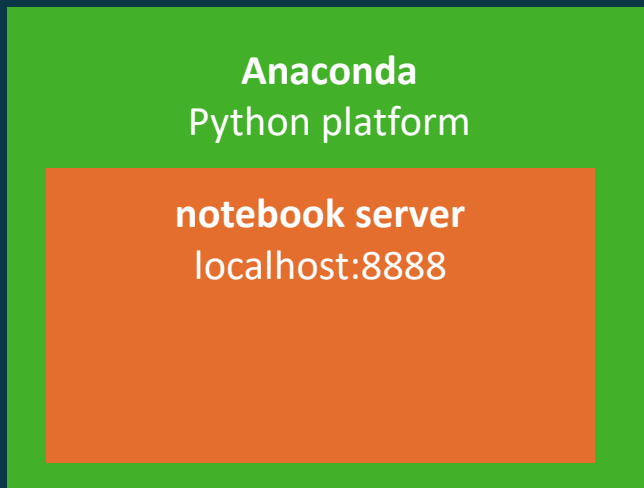
interpreter  
Python



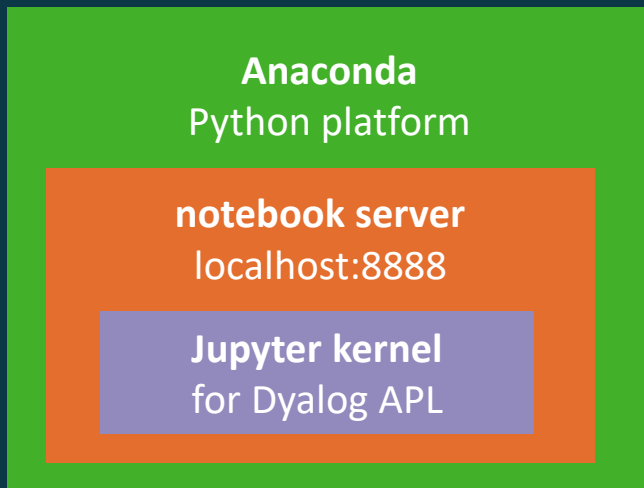
Jupyter Notebooks



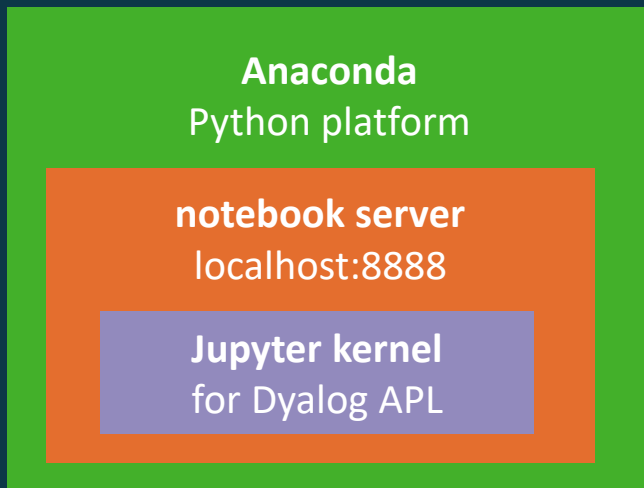
# Local notebook server — APL



# Local notebook server — APL

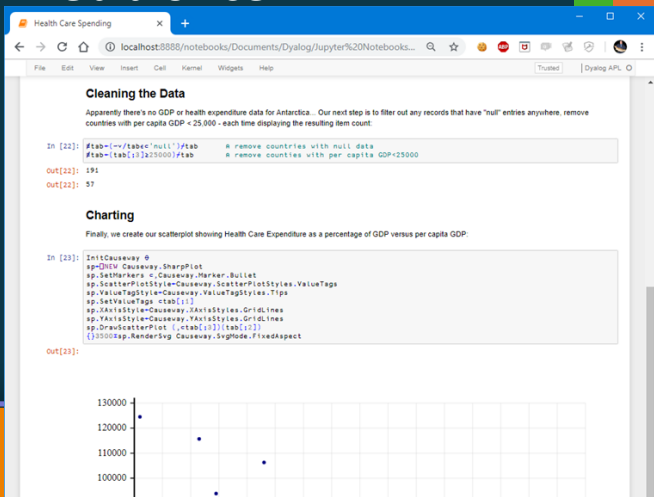


# Local notebook server — APL



# Local notebook server — APL

web browser



Anaconda  
Python platform

notebook server  
localhost:8888

Jupyter kernel  
for Dyalog APL

interpreter  
Dyalog APL

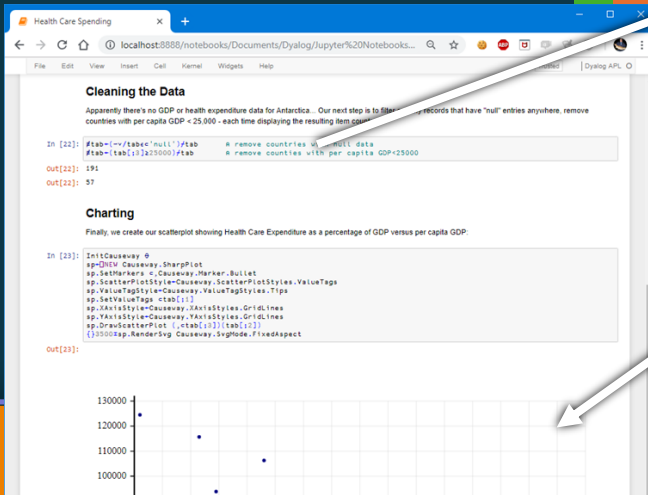
Jupyter Notebooks





# Local notebook server — APL

web browser



Anaconda  
Python platform

notebook server  
localhost:8888

Jupyter kernel  
for Dyalog APL

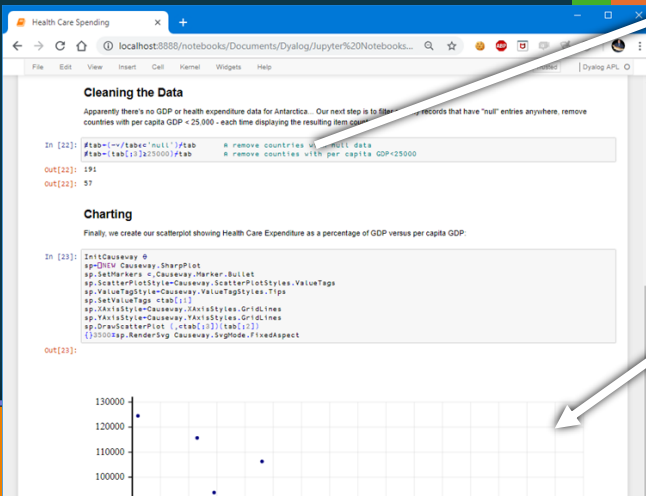
interpreter  
Dyalog APL

Jupyter Notebooks



# Local notebook server — APL

web browser



Anaconda  
Python platform

notebook server  
localhost:8888

Jupyter kernel  
for Dyalog APL

interpreter  
Dyalog APL

RIDE protocol

Jupyter Notebooks



# Setting up local notebook server

Install Dyalog 😊

Install Dyalog's Jupyter kernel

Install Anaconda

Launch Jupyter notebook server



# Setting up local notebook server

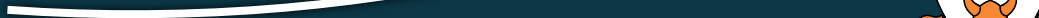
Install Dyalog 😊

Install Dyalog's Jupyter kernel

Install Anaconda

Launch Jupyter notebook server

installation instructions



# Demo

Installing Jupyter

Opening a notebook

Modifying content



# Online notebook servers



The image shows a browser window displaying the CoCalc website. The website has a navigation bar with links for Policies, Software, Pricing, API, and a Sign In button. The main content area features the CoCalc logo, a green 'Create Account!' button, and a link to 'sign in with your account'. Below this, the text 'Run Jupyter Notebooks Online' is displayed. At the bottom, there is a small inset image of a Jupyter Notebook interface showing a code editor with a Python script and a terminal window.

CoCalc - Run Jupyter Notebook

https://cocalc.com/doc/jupyter-notebook.html

COCALC Policies Software Pricing API Sign In

COCALC

Create Account!

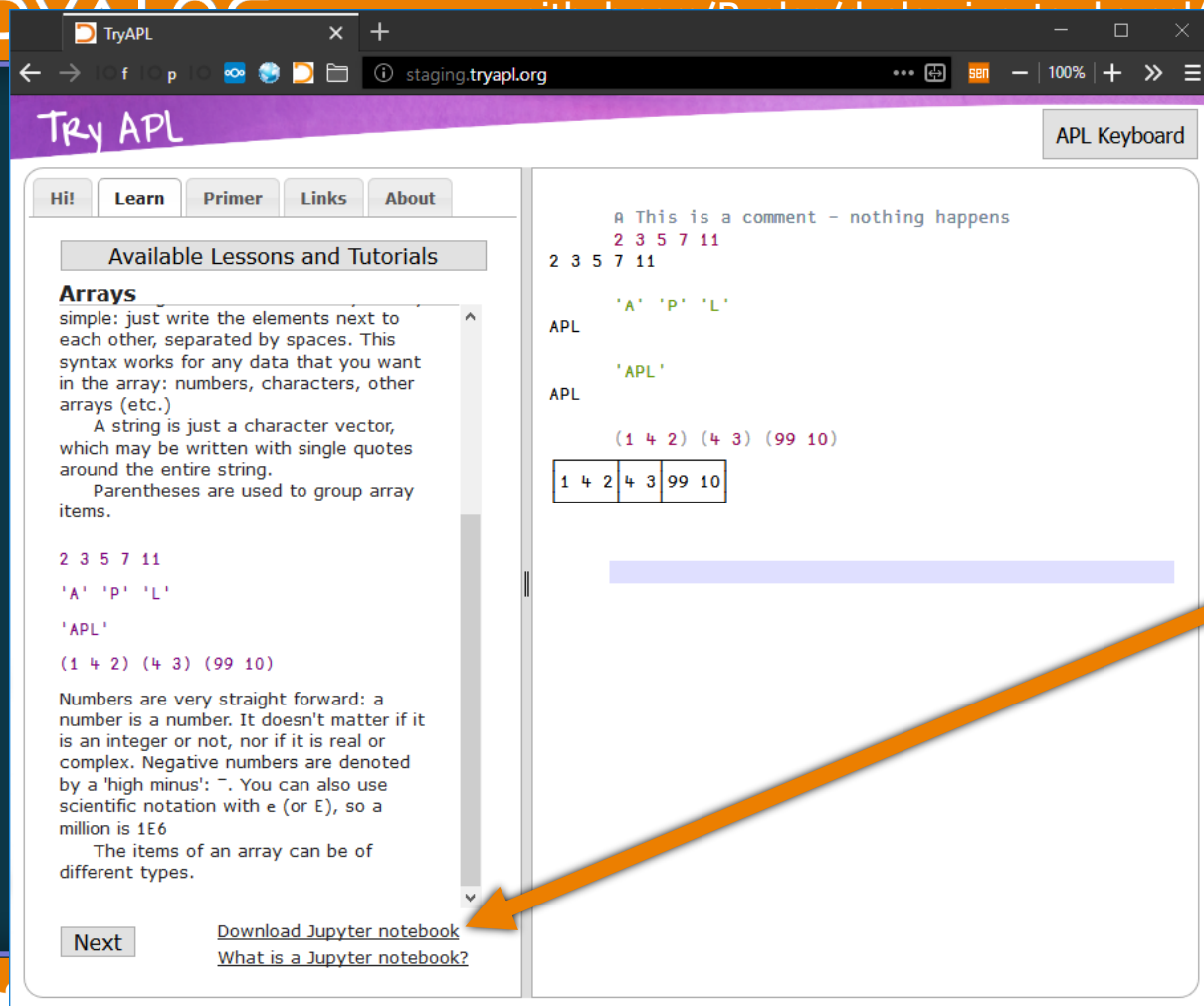
or sign in with your account

Run Jupyter Notebooks Online

CoCalc is an online web service where you can **run Jupyter notebooks right inside your browser**. It handles all the tedious details for you. You no longer

There are  
online services  
for various  
programming  
languages





The screenshot shows the TryAPL website in a web browser. The browser's address bar displays 'staging.tryapl.org'. The website has a purple header with the 'TryAPL' logo and a navigation menu with links: 'Hi!', 'Learn', 'Primer', 'Links', and 'About'. A 'TryAPL Keyboard' button is located in the top right corner. Below the navigation menu, there is a section titled 'Available Lessons and Tutorials'. The 'Arrays' lesson is selected and displayed on the left side of the interface. The right side of the interface features a code editor with APL code and its corresponding visual representation. An orange arrow points from the 'Download Jupyter notebook' link in the left sidebar to the code editor area.

TryAPL

APL Keyboard

Hi! Learn Primer Links About

Available Lessons and Tutorials

**Arrays**

simple: just write the elements next to each other, separated by spaces. This syntax works for any data that you want in the array: numbers, characters, other arrays (etc.)

A string is just a character vector, which may be written with single quotes around the entire string.

Parentheses are used to group array items.

2 3 5 7 11

'A' 'P' 'L'

'APL'

(1 4 2) (4 3) (99 10)

Numbers are very straight forward: a number is a number. It doesn't matter if it is an integer or not, nor if it is real or complex. Negative numbers are denoted by a 'high minus': -. You can also use scientific notation with e (or E), so a million is 1E6

The items of an array can be of different types.

Next

[Download Jupyter notebook](#)

[What is a Jupyter notebook?](#)

```
A This is a comment - nothing happens
2 3 5 7 11
2 3 5 7 11

'A' 'P' 'L'
APL

'APL'
APL

(1 4 2) (4 3) (99 10)
1 4 2 4 3 99 10
```

TryAPL's  
lessons  
are now  
Jupyter  
notebooks



TryAPL

APL Keyboard

[Creating and Editing a Notebook](#)

[Modifying Rank and Depth: ↑ ↓ ← →](#)

[Random Numbers: ?](#)

**Closer Looks at Some Operators**

[Reduce and Scan: / \](#)

**Highlights of Recent Releases**

[New in version 14.0](#)

[New in version 15.0](#)

[New in version 16.0](#)

**Interesting Explorations**

[Conway's Game of Life](#)

[Depth First Search](#)

[Lookup Without Replacement](#)

[Sudoku Solver](#)

**Introductory Course**

a) [Names and Expressions](#)

b) [Experiments](#)

c) [More Experiments](#)

d) [Characters \(Text\)](#)

e) [APL Errors](#)

f) [More Characters and Names and](#)

[Structure](#)

g) [Order of Evaluation](#)

h) [Direct Definition of Functions](#)

i) [Numbers as Text and Formatted Data](#)

j) [Select and Locate](#)

k) [Replace and Fill](#)

l) [Reading APL Expressions](#)

Or enter the URL to a Jupyter notebook:

<https://github.com/Dyalog/dy>

```
A This is a comment - nothing happens
2 3 5 7 11
2 3 5 7 11

'A' 'P' 'L'
APL

'APL'
APL

(1 4 2) (4 3) (99 10)

1 4 2 | 4 3 | 99 10
```

and you  
can up-load  
your own  
Jupyter  
notebooks  
as well





# Online notebook servers

Benefit: nothing to install

you may need to sign up for an account

To protect servers, host may place restrictions

or run in a sandbox with limited connectivity

Notebooks can execute any code

all code is run on the host server

same privileges as local execution



# Static notebook viewers

Notebooks are stored as .ipynb files

- .ipynb files are in JSON format

- each code cell may include output from the last execution

You can share an .ipynb file

- anyone with a local notebook server can view it

- ... but of course cannot execute anything new

Many online systems have viewers

- GitHub's file previewer

- Project Jupyter's [nbviewer.jupyter.org](https://nbviewer.jupyter.org)



# Exported notebooks

Notebooks can be exported to many standard formats

for example HTML, PDF, and  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$

Some formats require 3<sup>rd</sup> party plug-ins

Exported notebooks are static

that is expressions cannot be re-executed



# Demo

Creating a new notebook document

Running our own notebook under TryAPL

Generating rich output



## Ways to use notebooks — recap

Installing a Jupyter notebook server on your PC

Use an online notebook server like [cocalc.com](https://cocalc.com)

Store the notebook with output, then open in a notebook viewer

Export to HTML, PDF,  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ , ...



# Ask questions now!

**Wiki** [github.com/Dyalog/dyalog-jupyter-kernel/wiki](https://github.com/Dyalog/dyalog-jupyter-kernel/wiki)

**Email** [notebooks@dyalog.com](mailto:notebooks@dyalog.com) and [tryapl@dyalog.com](mailto:tryapl@dyalog.com)

## Thank you

*Technology Partnership (tp.rs)*: prototype APL kernel

*Will Robertson* (intern): kernel work and many notebooks

*Gil Athoraya* (of Optima Systems): syntax colouring



# Last day of Dyalog '18... Let's stay in touch!

[dyalog.com](https://dyalog.com)

Community

Chat Room

[chat.stackexchange.com/  
rooms/52405](https://chat.stackexchange.com/rooms/52405)

Search: "apl orchard"



#dyalog18

[adam@dyalog.com](mailto:adam@dyalog.com)



# Last day of Dyalog '18

dyalog.com

Community

Chat Room

chat.stackexchange.com/  
rooms/52405

Search: "apl orchard"



#dyalog18

adam@dyalog.com

## The APL Orchard

Learn and teach, questions about both golfing and general coding. Email support@dyalog.com for write access. Enter )about for chatbot info.



PCG

apl

array-manipulation

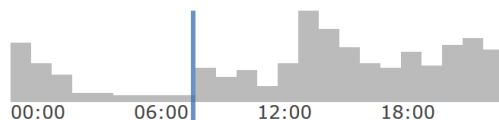
j

k

tips

first message 2017-01-24

last message 10 seconds ago



43

today

252

yesterday

177

per day



1.4k

this week

1.1k

last week

1.3k

per week

join this room

view transcript

search for messages containing