

Python Rgonomics

Emily Riederer
posit::conf(2024)
Seattle, WA

Why didn't you use python?

Why didn't you use python?

R is unparalleled for data wrangling

R is optimized for end-to-end data science communication

R has a better on-ramp and dev tools for non-developers

Why didn't you use python?

The magical flow state of R

“The” python stack



Picking tools - a tough choice?

Similar workflow and Rgonomics

or

Independently successful python tool

Picking tools - the criteria

Similar workflow and Ergonomics

and

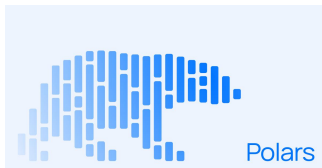
Independently successful python tool

- Functional
- Right level of abstraction
- Domain-specific

- Valuable, well-maintained, adopted
- Pythonic spirit (learn it, don't fight it!)
- Interoperable with broader ecosystem

An Rgonomic Stack

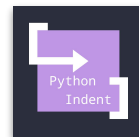
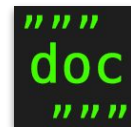
Wrangling



Communication



Dev Tools



Data

```
import polars as pl

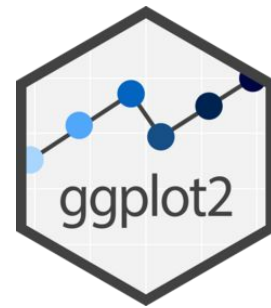
df = pl.read_csv('seattle-weather.csv')
df.glimpse()
```

```
Rows: 1461
Columns: 6
$ date          <date> 2012-01-01, 2012-01-02, 2012-01-03, 2012-01-04
$ precipitation <f64> 0.0, 10.9, 0.8, 20.3
$ temp_max      <f64> 12.8, 10.6, 11.7, 12.2
$ temp_min      <f64> 5.0, 2.8, 7.2, 5.6
$ wind          <f64> 4.7, 4.5, 2.3, 4.7
$ weather       <str> 'drizzle', 'rain', 'rain', 'rain'
```

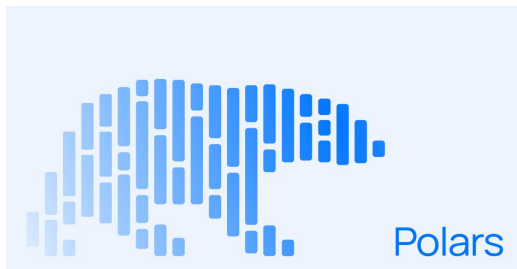
Data borrowed from [vega](#) - thanks!



R as in exploRation



Polars checks the boxes of an Rgonomic tool



✓ Similar workflow

- Functional paradigm
- Expansive and expressive API

✓ Independently successful

- Highly performant, zero-dependency Rust
- Wide and growing userbase
- Natively integrated into sns, sklearn, etc.

polars basics - similar verbs

dplyr	polars
df %>% select (a, b)	df. select ('a', 'b')
df %>% filter (a == 1)	df. filter (pl.col('a') == 1)
df %>% mutate (c = a + b)	df. with_columns (c = pl.col('a') + pl.col('b'))
df %>% group_by (a, b)	df. group_by ('a', 'b')
df %>% summarize (a = sum(a))	df. agg (pl.col('a').sum())

Expansive



Robust cleaning and manipulation for strings and date-times



Reshaping and structuring



List-columns, nested data frames, and data lists / dicts for easy iteration

dplyr (tidyverse) design principles

Composable

break complex problems into small pieces

Consistent

apply what you learn about one function to another

Human-Centered

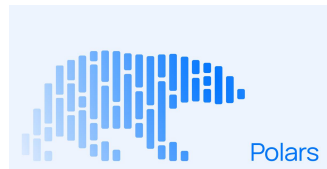
designed specifically to support a human data analyst

Composable



Across

```
df %>%  
  filter(b > 3) %>%  
  select(a,b)
```



Composable



Across

```
df %>%  
  filter(b > 3) %>%  
  select(a,b)
```



Across

```
(  
  df  
  .filter( pl.col('b') > 3 )  
  .select('a', 'b')  
)
```



Polars

Composable



Across

```
df %>%  
  filter(b > 3) %>%  
  select(a,b)
```



Within

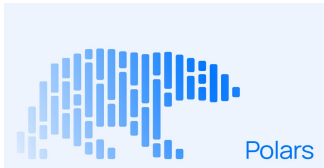
```
df %>%  
  mutate(c = a %>% length)
```

Across

```
(  
  df  
  .filter( pl.col('b') > 3 )  
  .select('a', 'b')  
)
```



Polars



Composable



Across

```
df %>%  
  filter(b > 3) %>%  
  select(a,b)
```



Within

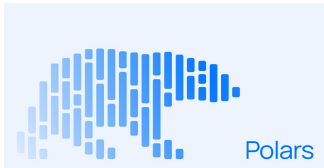
```
df %>%  
  mutate(c = length(a))
```

Across

```
(  
  df  
  .filter( pl.col('b') > 3 )  
  .select('a', 'b')  
)
```



Polars



Composable



Across

```
df %>%  
  filter(b > 3) %>%  
  select(a,b)
```



Within

```
df %>%  
  mutate(c = length(a))
```

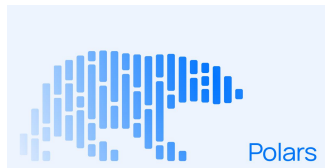
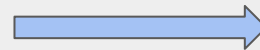
Across

```
(  
  df  
  .filter( pl.col('b') > 3 )  
  .select('a', 'b')  
)
```



Within

```
df.with_columns(  
  pl.col('a').str.len_chars()  
)
```



Consistency



```
df %>%  
  mutate(b = a + 1) %>%  
  filter(b > 3) %>%  
  select(a,b)
```

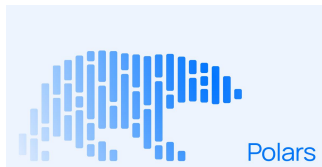
Consistency



```
df %>%  
  mutate(b = a + 1) %>%  
  filter(b > 3) %>%  
  select(a,b)
```

```
(df  
  .assign( b = lambda d: d['a'] + 1 )  
  .query( 'b > 3' )  
  [['a', 'b']]  
)
```

Consistency



```
df %>%  
  mutate(b = a + 1) %>%  
  filter(b > 3) %>%  
  select(a,b)
```

```
(df  
  .assign( b = lambda d: d['a'] + 1 )  
  .query( 'b > 3' )  
  [['a', 'b']]  
)
```

```
(df  
  .with_columns( b = pl.col('a') + 1 )  
  .filter( pl.col('b') > 3 )  
  .select('a', 'b')  
)
```

Human-Centered (Syntactic Sugar)

```
import polars.selectors as cs

(
  df
  .group_by('weather')
  .agg( cs.starts_with('temp').mean().round().name.prefix('avg_') )
)
```

date	temp_max	temp_min	weather
2012-01-01	12.8	5	"drizzle"
2012-01-02	10.6	2.8	"rain"
2012-01-03	11.7	7.2	"rain"
2012-01-04	12.2	5.6	"rain"
2012-01-05	8.9	2.8	"rain"

Human-Centered (Syntactic Sugar)

```
import polars.selectors as cs

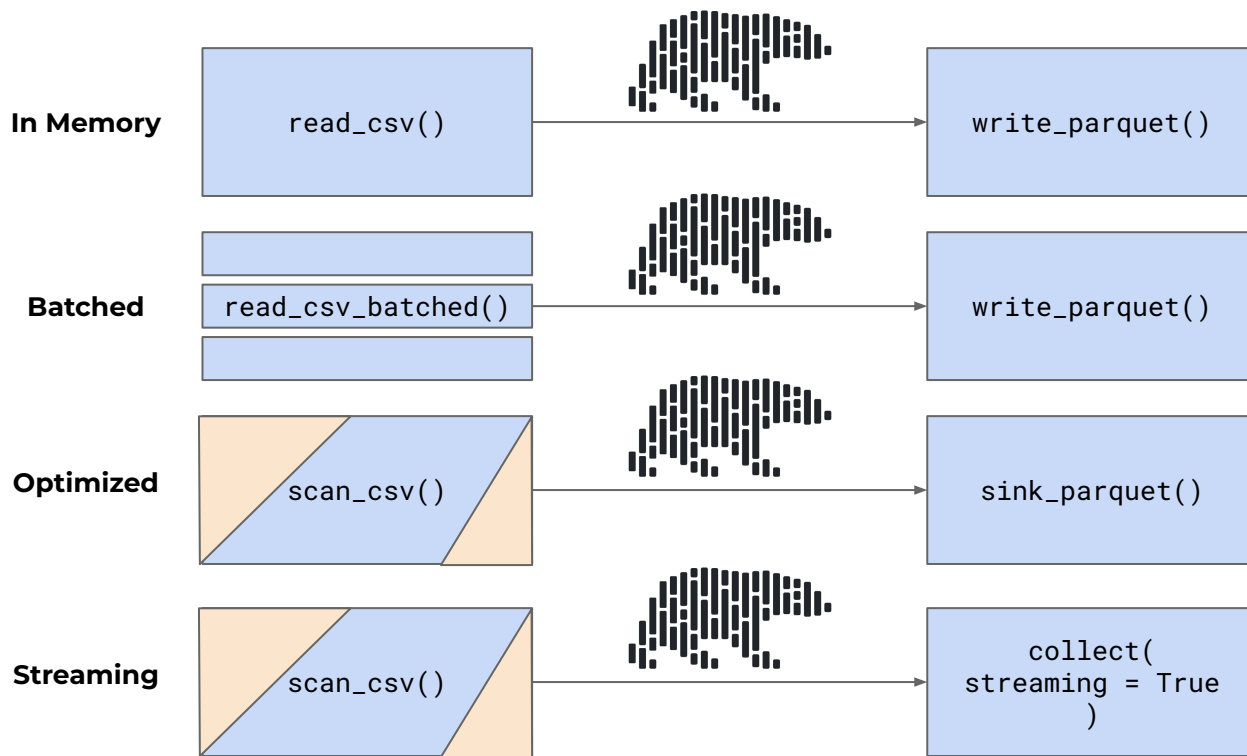
(
  df
  .group_by('weather')
  .agg( cs.starts_with('temp').mean().round().name.prefix('avg_') )
)
```

date	temp_max	temp_min	weather
2012-01-01	12.8	5	"drizzle"
2012-01-02	10.6	2.8	"rain"
2012-01-03	11.7	7.2	"rain"
2012-01-04	12.2	5.6	"rain"
2012-01-05	8.9	2.8	"rain"



weather	avg_temp_max	avg_temp_min
rain	13	8
drizzle	16	7
fog	17	8
sun	20	9
snow	6	0

Human-Centered (Domain Specific)



Visualization are the last mile, affording more freedom



High-fidelity clone of **ggplot2**

Well-supported and actively developed
(*thanks, Posit!*)

Growing adoption



Experimental OOP API with **ggplot2** flavor

Lives within popular python tools

“support[s]... specification and customization
without dropping down to matplotlib”

ggplot2 design principles

Grammar

mapping of data elements to aesthetics

Layers

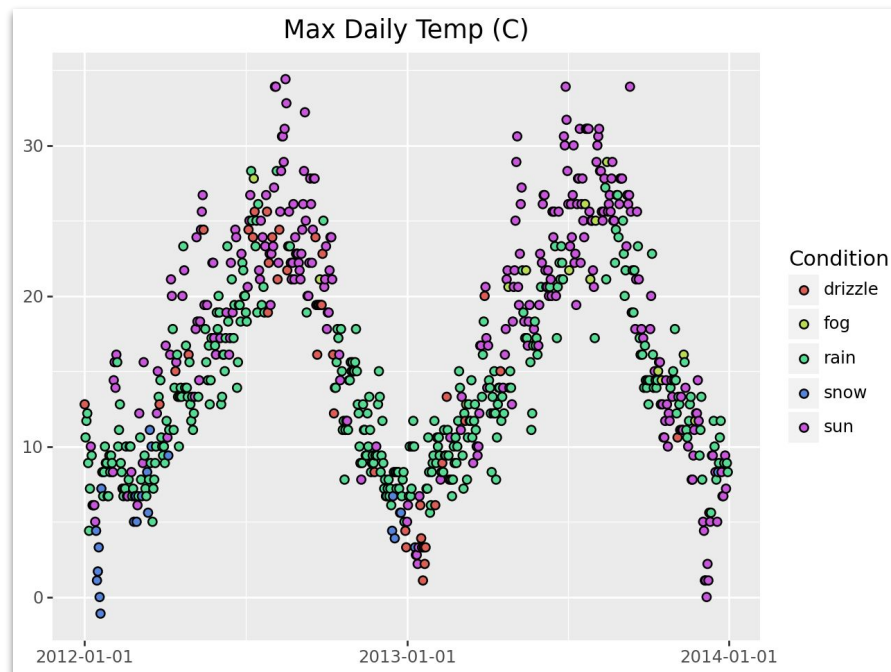
compose plots by iteratively adding layers

Manipulability

opportunity to control (optionally) nearly every aspect

plotnine provides a robust clone of ggplot2

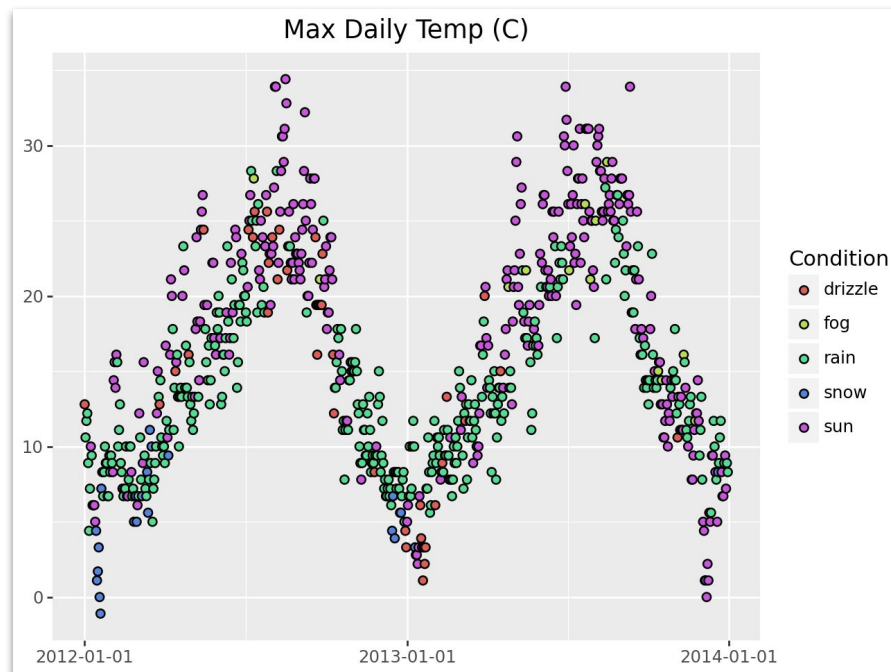
```
from plotnine import *  
  
(  
    ggplot(df) +  
    aes(x = 'date',  
        y = 'temp_max',  
        fill = 'weather') +  
    geom_point(size = 2) +  
    scale_x_date(date_breaks = '1 year') +  
    labs(  
        title = 'Max Daily Temp (C)',  
        x = '', y = '', fill = 'Condition')  
)
```



plotnine provides a robust clone of ggplot2

```
from plotnine import *

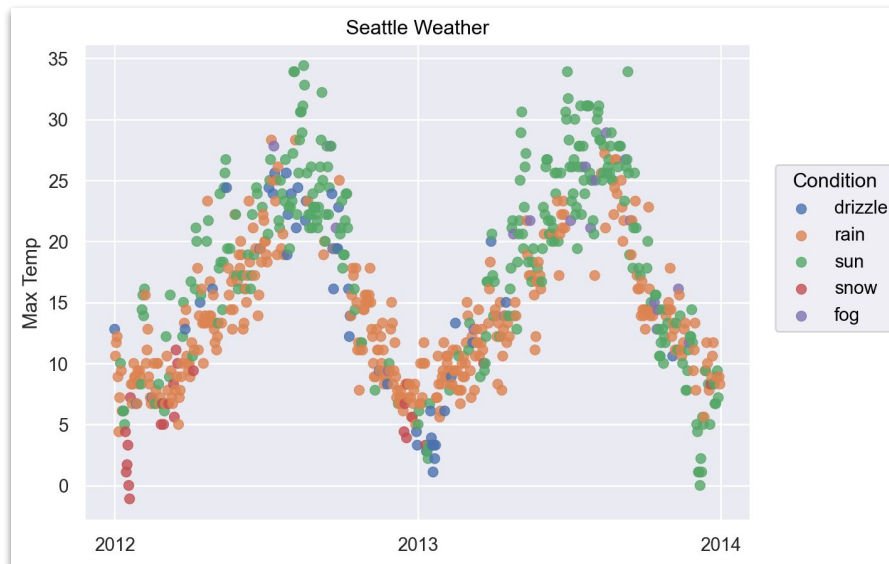
p = ggplot(df)
p += aes(x = 'date', y = 'temp_max',
         fill = 'weather')
p += geom_point(size = 2)
p += scale_x_date(date_breaks = '1 year')
p += labs(title = 'Max Daily Temp (C)',
         x = '', y = '', fill = 'Condition')
```



seaborn.objects experiments with a true pythonic alternative

```
import seaborn.objects as so

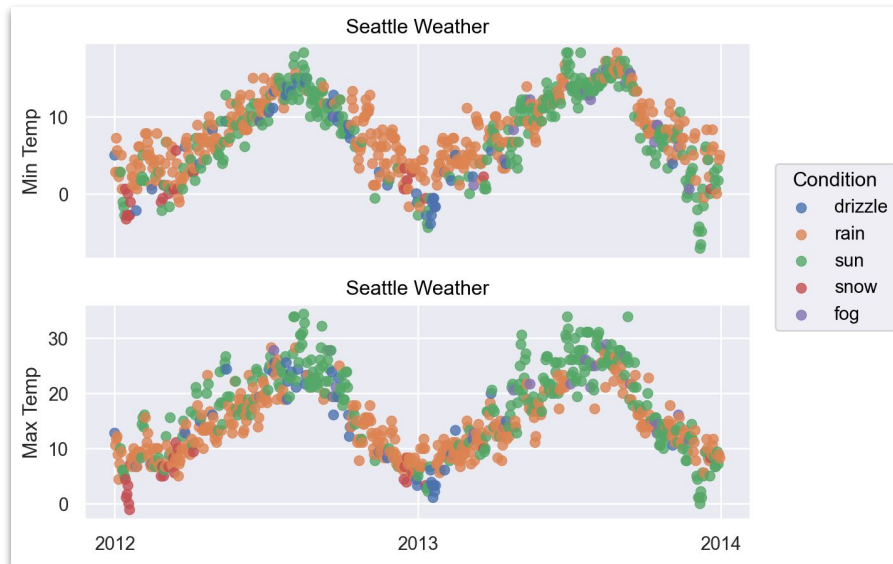
(
    so.Plot(df, x = 'date', y = 'temp_max')
        .add(so.Dot(alpha = 0.8), color = 'weather')
        .scale(x=so.Temporal())
        .label(x = '', y = 'Max Temp',
              title = 'Seattle Weather',
              color = 'Condition')
)
```



seaborn.objects experiments with a true pythonic alternative

```
import seaborn.objects as so

(
  so.Plot(df, x = 'date')
  .pair(y = ['temp_min', 'temp_max'])
  .add(so.Dot(alpha = 0.8), color = 'weather')
  .label(x = '',
         title = 'Seattle Weather',
         y0 = 'Min Temp',
         y1 = 'Max Temp',
         color = 'Condition')
  .scale(x=so.Temporal())
)
```



Both can return underlying matplotlib object when needed



```
type(p.draw())  
> matplotlib.figure.Figure
```

```
type(p.plot()._figure)  
> matplotlib.figure.Figure
```


R as in Reproducible Reporting



Quarto balances interactivity and reproducibility



```
wrangling.qmd X
wrangling.qmd
1 ---
2 jupyter: python3
3 ---
4
5 Run Cell | Run Next Cell
6 import polars as pl
7 ---
8
9 Run Cell | Run Next Cell | Run Above
10 df = pl.read_csv('seattle-weather.csv')
11 ---
12
13 Run Cell | Run Next Cell | Run Above
14 df_sub = df.filter( pl.col('date').str.slice(0,4) == '2012' )
15 ---
16
17 Run Cell | Run Above
18 df_sub.group_by('weather').agg( pl.col('temp_max').mean() )
19 ---
20
21
```

CONSOLE TERMINAL PROBLEMS OUTPUT PORTS DEBUG CONSOLE QUERY RESULTS LINEAGE DOCUMENTATION EDITOR ACTIONS

```
Python 3.11.0 (Envc:venv) * -D:\Desktop\pyqgo-post
>>> import polars as pl
>>> df = pl.read_csv('seattle-weather.csv')
>>> df_sub = df.filter( pl.col('date').str.slice(0,4) == '2012' )
>>> df_sub.group_by('weather').agg( pl.col('temp_max').mean() )
shape: (5, 2)
  weather  temp_max
  str      f64
  "sun"    20.234746
  "fog"    21.1
  "snow"   5.395238
  "rain"   12.88733
  "drizzle" 17.374194
```

```
wrangling.ipynb X wrangling.qmd
wrangling.ipynb > df = pl.read_csv('seattle-weather.csv')
+ Code + Markdown | Run All Restart Clear All Outputs Outline ...
```

```
import polars as pl
[16] ✓ 0.0s

df = pl.read_csv('seattle-weather.csv')
[17] ✓ 0.0s

df_sub = df.filter( pl.col('date').str.slice(0,4) == '2012' )
[18] ✓ 0.0s

df_sub.group_by('weather').agg( pl.col('temp_max').mean() )
[19] ✓ 0.0s
```

shape: (5, 2)

weather	temp_max
str	f64
"rain"	12.80733
"snow"	5.395238
"sun"	20.234746
"fog"	21.1
"drizzle"	17.374194

CONSOLE TERMINAL PROBLEMS OUTPUT PORTS DEBUG CONSOLE QUERY RESULTS LINEAGE DOCUMENTATION EDITOR ACTIONS

```
Python 3.11.0 (Envc:venv) * -D:\Desktop\pyqgo-post
```

✓ Plain text & version controllable

✓ Interactive given interactions with IDE

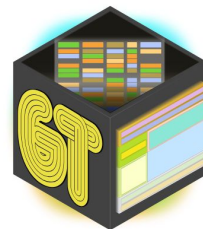
✓ Clean state

Great Tables beats out the... wait, is there even competition?

Great Tables



From the team behind **gt!**



Great Tables gives tables a grammar

```
(  
  GT(df_agg, rowname_col="mnth")  
  .tab_header(  
    title="Average Seattle Temperatures",  
    subtitle=html("Max Daily, &deg;C"),  
  )  
  .tab_spanner(label="By Year",  
               columns = cs.starts_with("2"))  
  .cols_align(align = "center")  
  .data_color(  
    domain=[30, 0],  
    palette=["darkred", "white", "lightblue"],  
    na_color="white",  
  )  
  .fmt_nanoplot("Trend", plot_type="line")  
)
```

Average Seattle Temperatures					
Max Daily, °C					
By Year					
	2012	2013	2014	2015	Trend
1	7.1	6.1	9.6	10.2	
2	9.3	9.5	8.2	12.5	
3	9.6	12.7	12.9	14.4	
4	14.9	14.2	15.5	15.5	
5	17.7	19.6	19.9	20.0	
6	18.7	23.3	21.6	26.1	
7	22.9	26.1	26.9	28.1	
8	25.9	26.1	26.4	26.1	
9	22.9	21.4	23.2	20.3	
10	15.8	14.2	18.0	17.5	
11	11.3	12.1	11.0	9.7	
12	7.2	7.0	10.1	8.4	

R as in dev expeRience



Ergonomics is about avoiding papercuts

Installation & Organization

- Helping computer find
- Managing multiple versions
- 'Protecting' system python

Dependency Management

- Isolating per project
- Adding / removing
- Documenting
- So many tools

Development Environment

- Creating a data-native experience
- Embracing 'advanced' tools

Software and dev environment ergonomics

Code over Clicks

Favor things that can be scripted and automated

Helpful

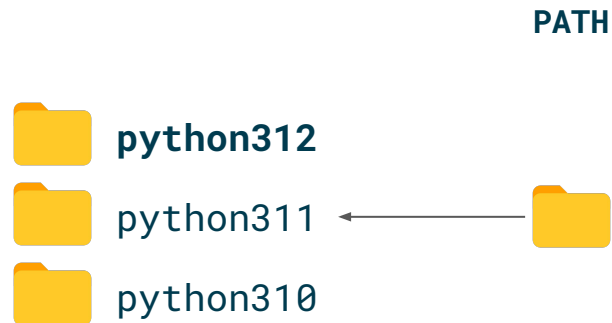
Opinionated tools promote the “pit of success”

Unsurprising

Tools aim to not have unsurprising side effects

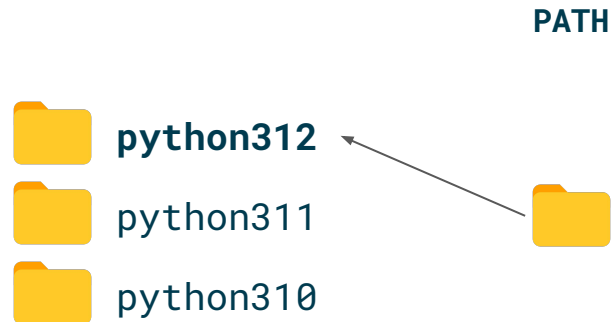
Installation with pyenv (pyenv-win)

```
> pyenv install 3.12.0
```

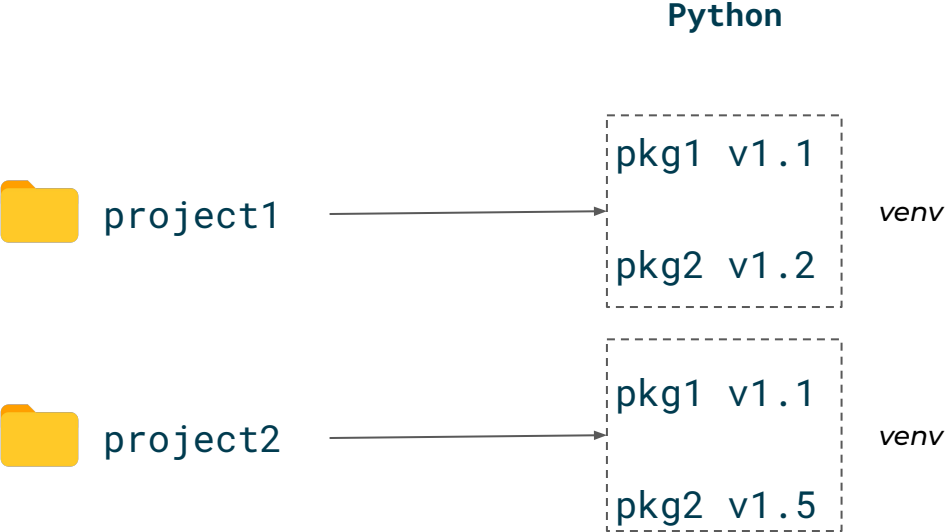


Installation with pyenv (pyenv-win)

```
> pyenv install 3.12.0  
> pyenv global 3.12.0
```



Dependency management woes



Dependency management with pdm



```
> pdm add polars
```



pyproject.toml

```
[project]
name = "my-project"
dependencies = [
    "ipykernel>=6.29.0",
    "polars>=0.20.7",
]
```

pdm.lock

```
<< dependencies of
dependencies>>
```

Environment Management with pdm



```
> pdm add polars  
> pdm remove polars
```



pyproject.toml

```
[project]  
name = "my-project"  
dependencies = [  
    "ipykernel>=6.29.0"  
]
```

pdm.lock

```
<< dependencies of  
dependencies>>
```

Environment Management with pdm



```
> pdm add polars
> pdm remove polars
> pdm export
```



pyproject.toml

```
[project]
name = "my-project"
dependencies = [
    "ipykernel>=6.29.0"
]
```

pdm.lock

```
<< dependencies of
dependencies>>
```

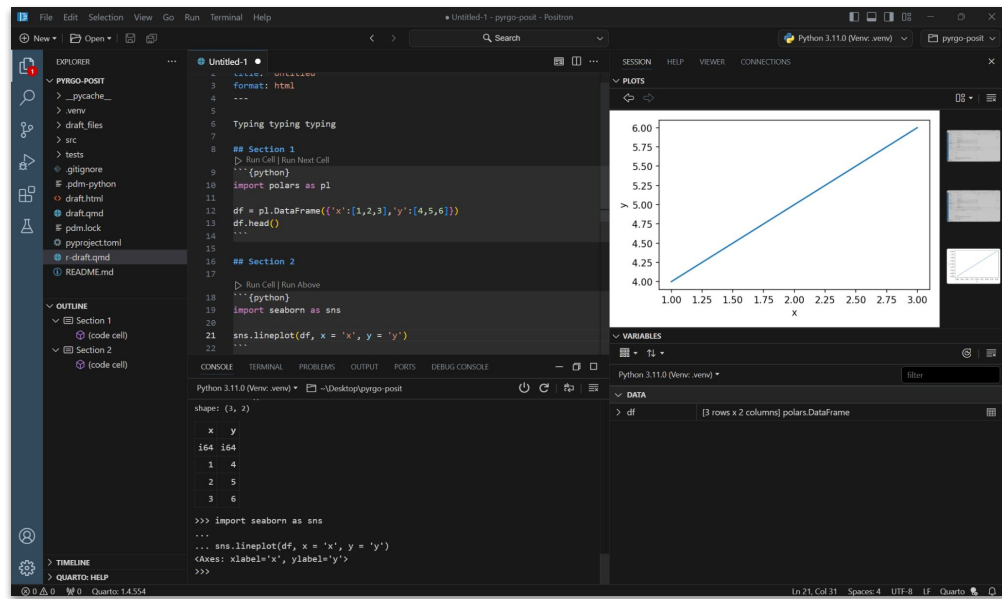
requirements.txt

VS Code and Positron create a customizable and 'data-first' developer experience



Editor

Plot Viewer



Terminal

Variable Explorer

Dev tools and extensions can help you fake the 'flow'

Cookiecutter



Structure project from templates

Keyboard Shortcuts



Prevent typos and limit boilerplate

ErrorLens



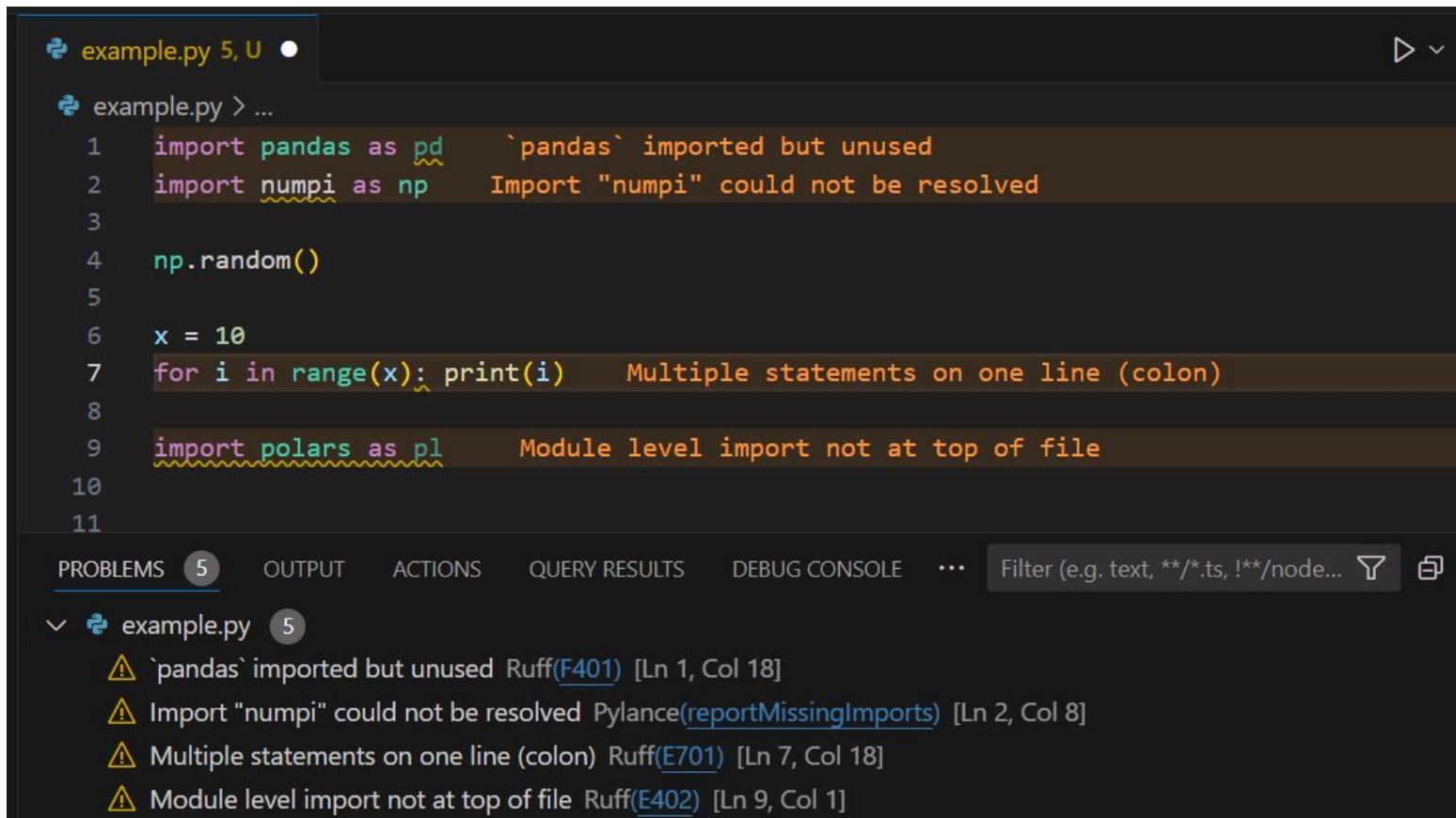
Aggressively highlight errors in IDE

Ruff



Note and fix style improvements

Dev tools and extensions can help you fake the 'flow'



```
example.py 5, U
example.py > ...
1  import pandas as pd    `pandas` imported but unused
2  import numpi as np    Import "numpi" could not be resolved
3
4  np.random()
5
6  x = 10
7  for i in range(x): print(i)    Multiple statements on one line (colon)
8
9  import polars as pl    Module level import not at top of file
10
11
```

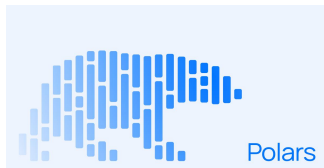
PROBLEMS 5 OUTPUT ACTIONS QUERY RESULTS DEBUG CONSOLE ... Filter (e.g. text, **/*.ts, !**/node...)

example.py 5

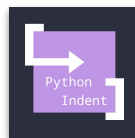
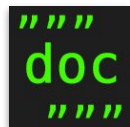
- ⚠️ `pandas` imported but unused Ruff(F401) [Ln 1, Col 18]
- ⚠️ Import "numpi" could not be resolved Pylance(reportMissingImports) [Ln 2, Col 8]
- ⚠️ Multiple statements on one line (colon) Ruff(E701) [Ln 7, Col 18]
- ⚠️ Module level import not at top of file Ruff(E402) [Ln 9, Col 1]

Why didn't you use python?

~~Why didn't you use python?~~ Why don't I check out python?



Great Tables



Questions?

↓ **Get in touch** ↓

@emilyriederer on [Web](#) | [Twitter](#) | [GitHub](#) | [LinkedIn](#) | Gmail

↓ **Check out these related posts** ↓

[Python Rgonomics](#)

[Polars' Rgonomic Patterns](#)

[Base Python Rgonomic Patterns](#)