Dominik Gresch

Better Autocomplete with Type Hints

Next: Simon Niederberger

Powering Innovation That Drives Human Advancement

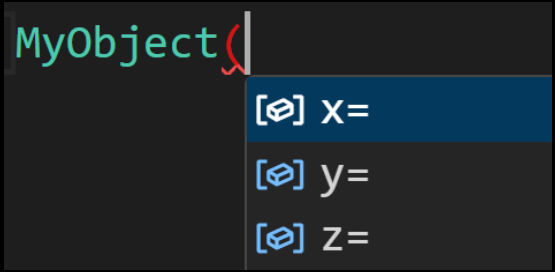
Better Autocomplete with Type Hints

Dominik Gresch

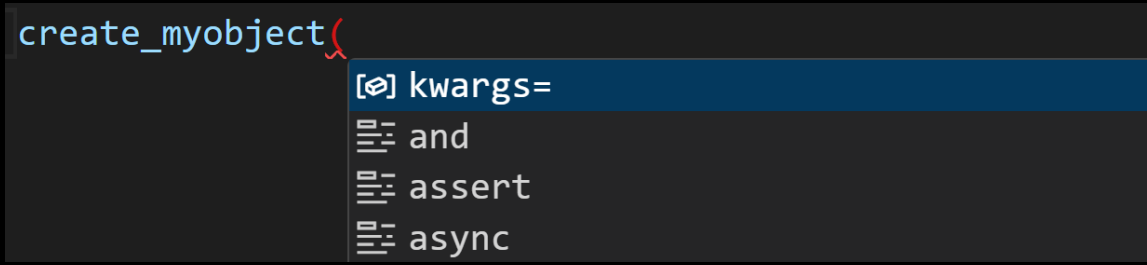
```
x: int = 4
def foo(y: str) -> int:
```

Static code checks… but can they do more?

```python
class MyObject:
    def __init__(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z
```



```python
def define_object_factory(object_type):
    def inner(*args, **kwargs):
        return object_type(*args, **kwargs)

    return inner

create_myobject = define_object_factory(MyObject)
```



```python
from collections.abc import Collection

class MyCollection(Collection):
    def __init__(self, values):
        self._values = list(values)

    def __contains__(self, value):
        return value in self._values

    def __iter__(self):
        return iter(self._values)

    def __len__(self):
        return len(self._values)


def create_collection(values):
    return MyCollection(values=values)
```

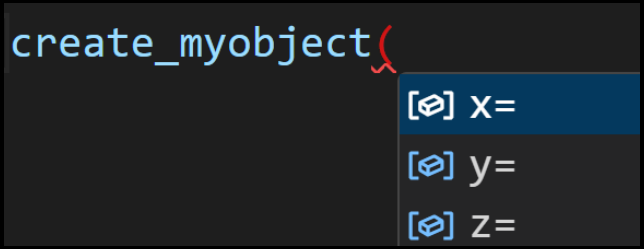
```
collection = create_collection(values=[MyObject(x=1, y="a", z=False)])

obj = next(iter(collection))

obj.
```

No suggestions.

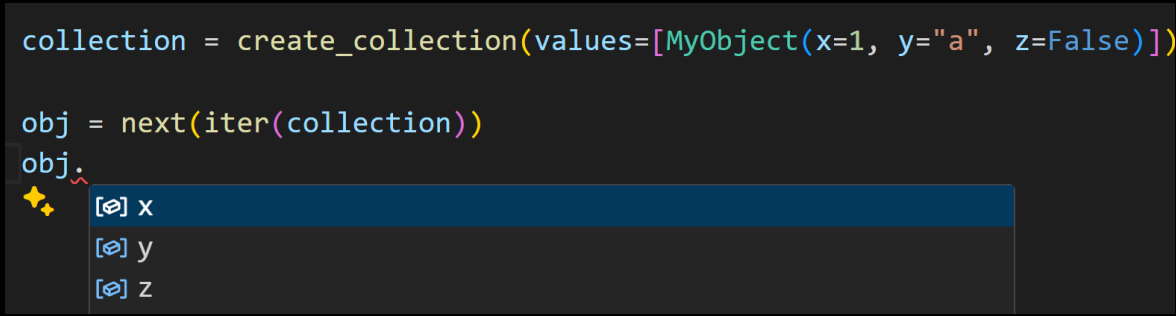
Add type hints!

```python
class MyObject:
    def __init__(self, x: int, y: str, z: bool):
        self.x = x
        self.y = y
        self.z = z
```

```python
from typing import ParamSpec, TypeVar


T = TypeVar("T")
P = ParamSpec("P")


def define_object_factory(object_type: Callable[P, T]) ->
Callable[P, T]:
    def inner(*args: P.args, **kwargs: P.kwargs) -> T:
        return object_type(*args, **kwargs)

    return inner
```



```python
from collections.abc import Callable, Collection, Iterable, Iterator
from typing import TypeVar

T = TypeVar("T")

class MyCollection(Collection[T]):
    def __init__(self, values: Iterable[T]):
        self._values = list(values)

    def __contains__(self, value: object) -> bool:
        return value in self._values

    def __iter__(self) -> Iterator[T]:
        return iter(self._values)

    def __len__(self) -> int:
        return len(self._values)


def create_collection(values: Iterable[T] = ()) -> MyCollection[T]:
    return MyCollection(values=values)
```



Happy Users!

Testing?

```
from typing_extensions import reveal_type

reveal_type(create_myobject)
```

```
Revealed type is "def (x: builtins.int, y: builtins.str, z:
builtins.bool) -> with_typehints.MyObject"
```

```python
from typing import Callable
from typing_extensions import assert_type
from mypy_extensions import Arg

assert_type(
    create_myobject,
    Callable[
        [Arg(int, 'x'), Arg(str, 'y'), Arg(bool, 'z’)],
        MyObject
    ]
)
```

Happy Developers!


Ansys

Simon Niederberger

Jupyter & Git

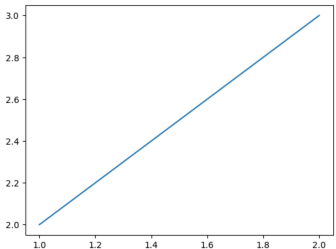
Next: Vita Midori

Jupyter Notebooks & Git

Simon Niederberger

```python
from time import sleep
import matplotlib.pyplot as plt

def make_plot() -> None:
    sleep(10)
    plt.plot([1,2], [2,3])

make_plot()
```

[1] ✓ 10.4s



I want to

- [ ] Frequently commit files to Git
- [ ] Maintain a clean and slim Git repository
- [ ] Keep Jupyter notebook outputs locally
- [ ] Don’t let merge conflicts brake the notebook

.ipynb files are JSON files

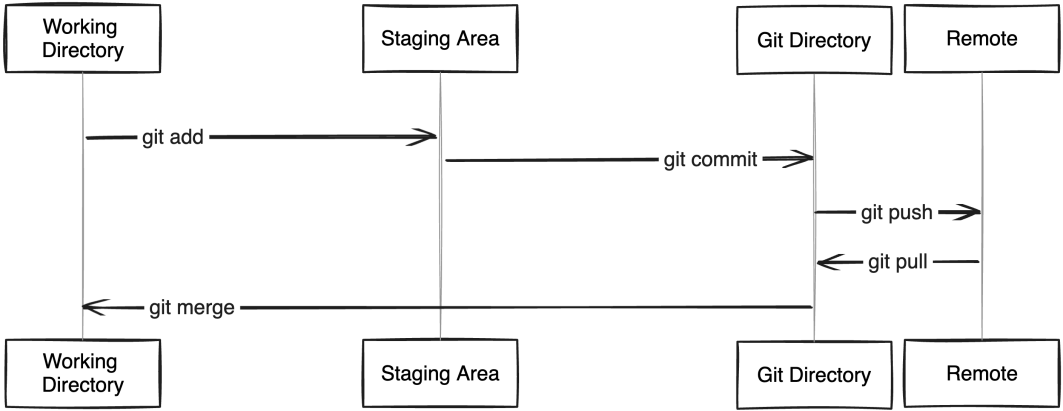
I want to

- [ ] Maintain a clean and slim Git repository:
  …
  “outputs”: [ ]
  …

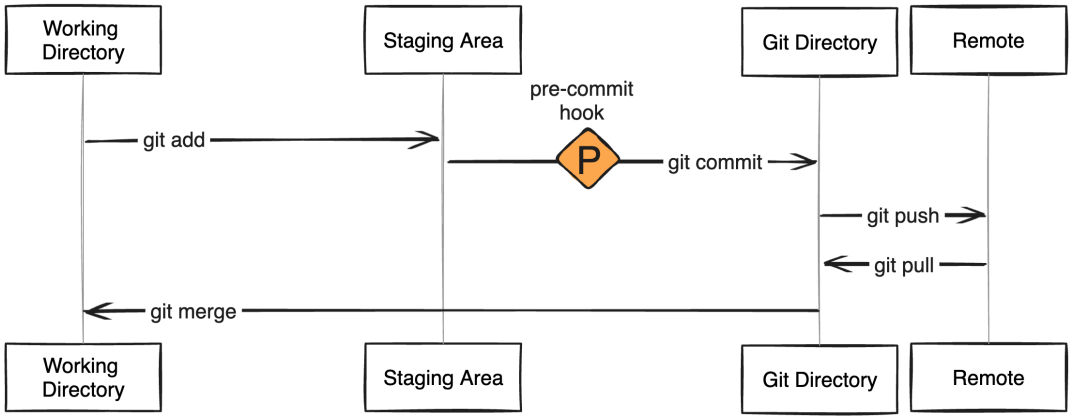
- [ ] Keep Jupyter notebook outputs locally:
  …
  “outputs”: [{ "data": { “image/png”: “iVBORw…
  …


```
{
    "cells": [
        {
            "cell_type": "code",
            "execution_count": 1,
            "metadata": { },
            "outputs": [
                {
                    "data": {
                        "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAiMAAAGdCAYAAADAnMpAAAAOXRFWHRTb2Z0d2FyZQBNY
BrUwzl2xRM7VSQMV9FxdQNHY4wDnX74/u8R7LBRC4OIf38/Hgj3N5feHNJZ7r7fW5OMdiG
LBYLGbHERERkUowDIOCggJatWqF1Xrp6x9OUUZOnjxJeHi42TFERESkGjIzMwkLC7vknzt
bt24fVar3ik75UXnFxMVbrhacsNzc3AAy9xdolGYZBYmIiS5Ys4ZtvvqFdu3ZXXGPGudCl
8gSlTppCTk8P48eN59NFHdUX1Mqp6nIcMGcLjjz/OnDlzGDRoEKdOnWLs2LFcf/31tGrVy
6Rf7a2c6hzn3bt3GwMGDDB8fHyMsLAwY9y4cUZxcXHdh3ci1TnOM2fONLp162b4+PgYISE
DSsiIiJiCt0zIiIiIqZSGRERERFTFqYyIiIiIqVRGRERERExlcqIiIqIyImIqlRERERExlcqIiI
                        "text/plain": [
                            "<Figure size 640x480 with 1 Axes>"
                        ]
                    },
                    "metadata": { },
                    "output_type": "display_data"
                }
            ],
            "source": [
                "from time import sleep\n",
                "import matplotlib.pyplot as plt\n",
                "\n",
                "def make_plot():\n",
                "    sleep(10)\n",
                "    plt.plot([1,2], [2,3])\n",
                "\n",
                "make_plot()"
            ]
        }
    ],
    "metadata": {
        "kernelspec": {
            "display_name": "Python 3",
            "language": "python",
            "name": "python3"
```

- [ ] Maintain a clean and slim Git repository



- [ ] Maintain a clean and slim Git repository



```python
def main(argv: Sequence[str] | None = None) -> int:
    parser = argparse.ArgumentParser()
    parser.add_argument("filenames", nargs="*", help="Filenames to check.")
    args = parser.parse_args(argv)
    retval = 0
    for filename in args.filenames:
        try:
            notebook = json.loads(git.Repo().git.show(f":{filename}"))
            for cell in notebook["cells"]:
                try:
                    if cell["metadata"] != {}:
                        retval += 1
                        logger.warning(
                            f"found cell with nonempty metadata in {filename}."
                        )
                    if cell["outputs"] != []:
                        retval += 1
                        logger.warning(
                            f"found cell with nonempty output in {filename}."
                        )
                except KeyError:
                    pass

        except ValueError as exc:
            logger.exception(f"{filename}: Failed to json decode ({exc})")
            retval = 1
    return retval
```

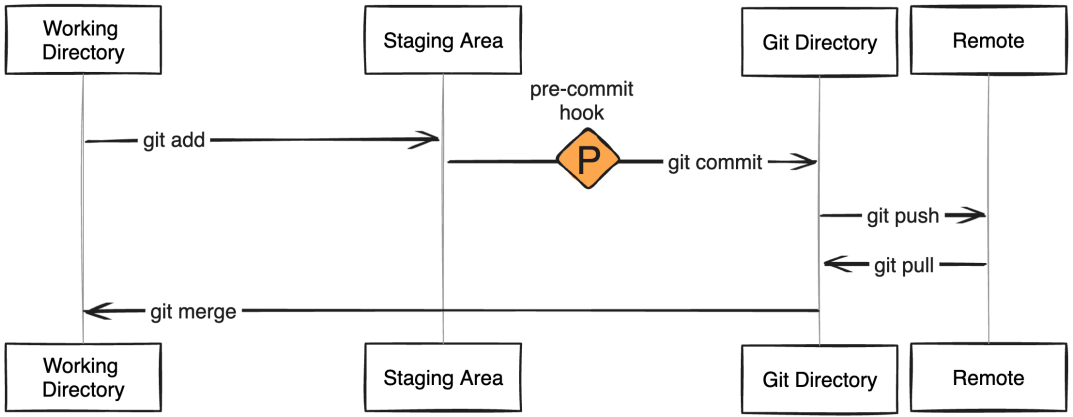
https://github.com/saemeon/ipynb_output_strip_pre_commit

.pre-commit-config.yaml

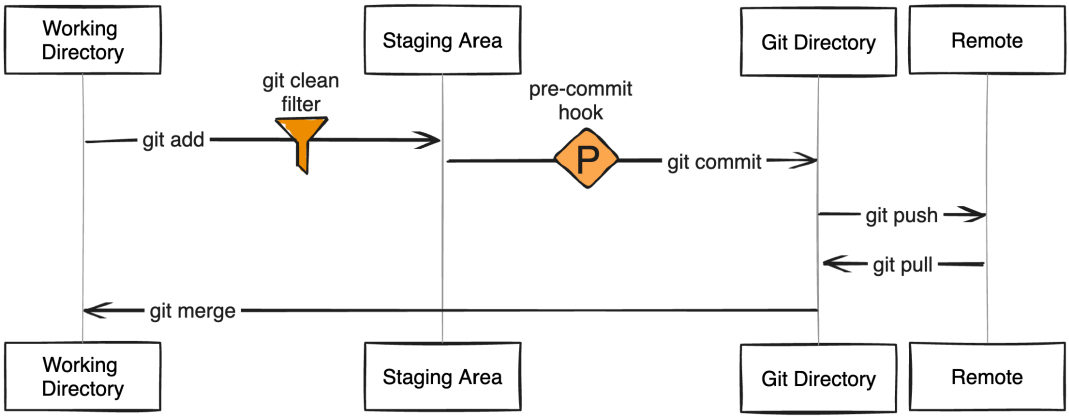
```
repos:
  - repo: https://github.com/saemeon/ipynb_output_strip_pre_commit
    rev: bedbd7195fd63d3f6e95c91f420245ca85b2adc5
    hooks:
        - id: check-ipynb-output-strip
```

$ pre-commit install

- [x] Maintain a clean and slim Git repository
- [ ] Keep Jupyter notebook outputs locally



- [x] Maintain a clean and slim Git repository
- [ ] Keep Jupyter notebook outputs locally



.gitconfig :

```
[filter "ipynb_filter"]
        clean = "jupyter nbconvert --ClearOutputPreprocessor.enabled=True —
ClearMetadataPreprocessor.enabled=True --to=notebook --stdin --stdout --log-level=ERROR"
```

.gitattributes:

```
*.ipynb filter=ipynb_filter
```

$ git config —local include.path ../.gitconfig

test.ipynb (Working Tree) M

test.ipynb

Input

```
from time import sleep
import matplotlib.pyplot a

def make_plot():
    sleep(10)
    plt.plot([1,2], [2,4])

make_plot()
```

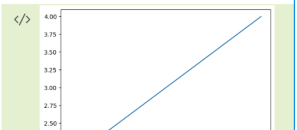
```
from time import sleep
import matplotlib.pyplot as plt

def make_plot():
    sleep(10)
    plt.plot([1,2], [2,4])

make_plot()
```

Metadata changed

```
{
    "language": "python",
    "metadata": {}
}
```

```
{
    "language": "python",
    "execution_count": 4,
    "metadata": {}
}
```

Outputs changed



test.ipynb (Index)

test.ipynb

Input changed

```
from time import sleep
import matplotlib.pyplot a

def make_plot():
    sleep(10)
    plt.plot([1,2], [2,3])

make_plot()
```

```
from time import sleep
import matplotlib.pyplot as plt

def make_plot():
    sleep(10)
    plt.plot([1,2], [2,4])

make_plot()
```

Metadata

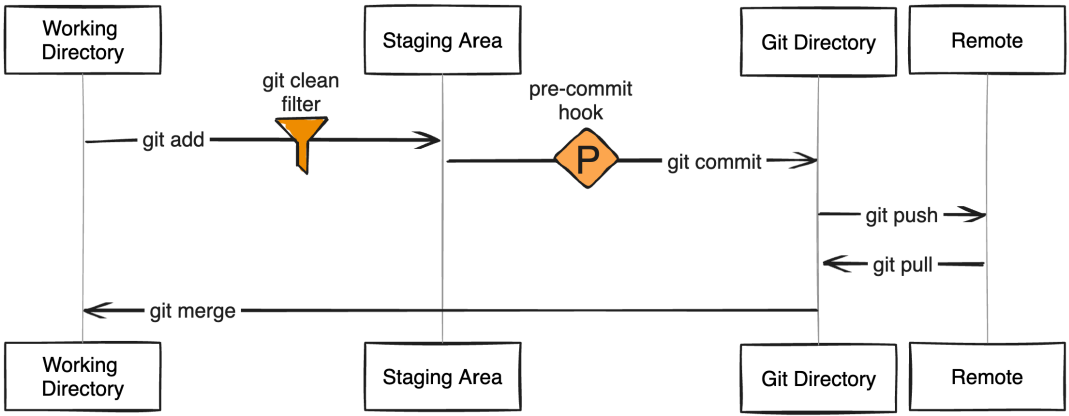
```
{
    "language": "python",
    "metadata": {}
}
```

```
{
    "language": "python",
    "metadata": {}
}
```

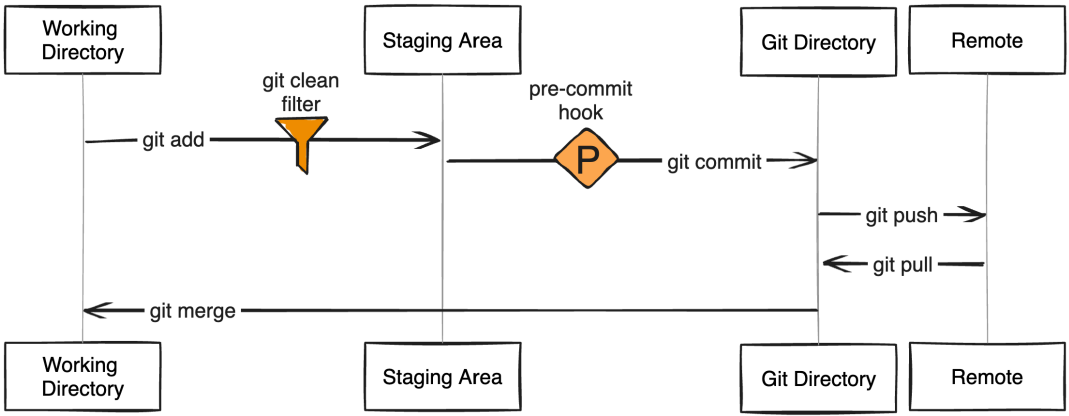
Outputs

No outputs to render

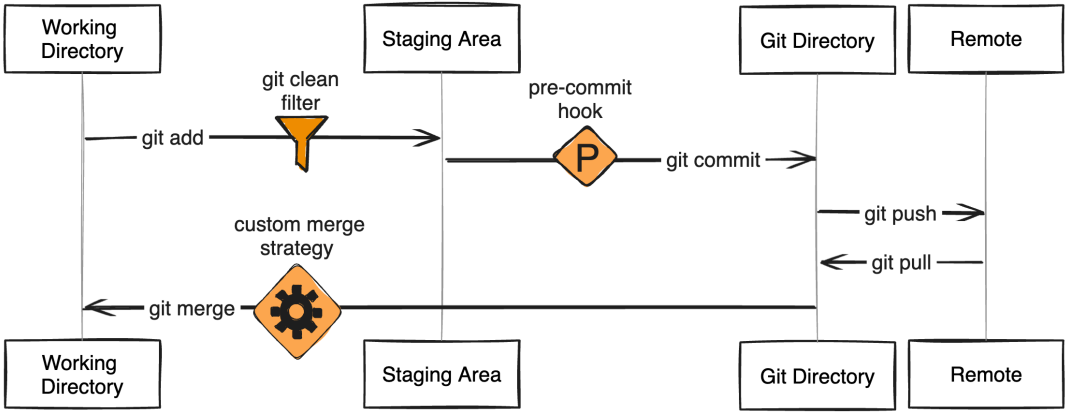
- [x] Maintain a clean and slim Git repository
- [x] Keep Jupyter notebook outputs locally



- [ ] Don’t let merge conflicts brake the notebook



- [ ] Don’t let merge conflicts brake the notebook



.gitconfig :

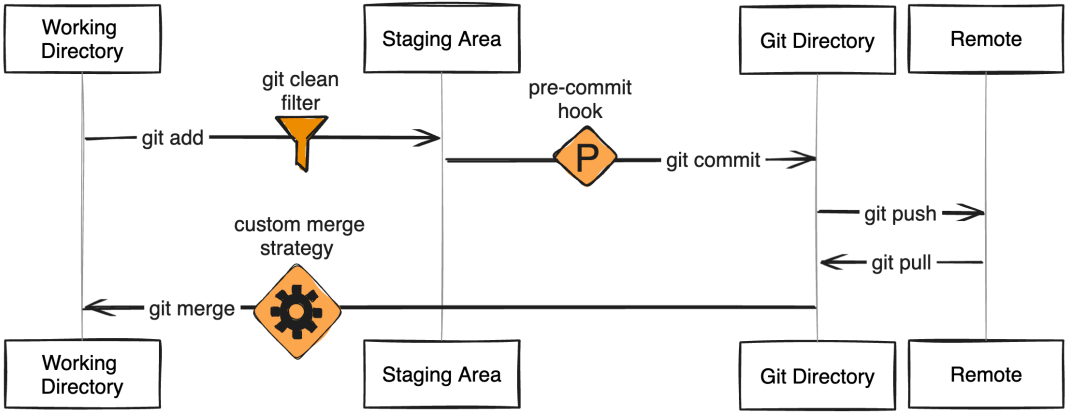
```
[filter "ipynb_filter"]
        clean = "jupyter nbconvert --ClearOutputPreprocessor.enabled=True —
ClearMetadataPreprocessor.enabled=True --to=notebook --stdin --stdout --log-level=ERROR"

[merge "ipynb_merge"]
        name = resolve conflicts with nbdev_fix
        driver = nbdev_merge %O %A %B %P
```

.gitattributes:

```
*.ipynb filter=ipynb_filter
*.ipynb merge=ipynb_merge
```

$ git config —local include.path ../.gitconfig

- [x] Don’t let merge conflicts brake the notebook



Setup:

1. Install relevant libraries

2. Setup pre-commit:
   .pre-commit-config.yaml

3. Configure Git filter and merge strategy:
   .gitconfig
   .gitattributes

.pre-commit-config.yaml > [ ] repos > {} 0 > [ ] hooks > {} 0

```yaml
repos:
  - repo: https://github.com/saemeon/ipynb_output_strip_pre_commit
    rev: bedbd7195fd63d3f6e95c91f420245ca85b2adc5
    hooks:
      - id: check-ipynb-output-strip
```

.gitconfig

```
[filter "ipynb_filter"]
    clean = "jupyter nbconvert --ClearOutputPreprocessor.enabled=True --ClearMetadataPreprocessor.ena
[merge "ipynb_merge"]
    name = resolve conflicts with nbdev_fix
    driver = nbdev_merge %O %A %B %P
```

.gitattributes

```
*.ipynb filter=ipynb_filter
*.ipynb merge=ipynb_merge
```

Used libraries

- pre-commit -> setup hook
- nbconvert -> git clean filter
- gitpython -> hook
- nbdev -> merge strategy

I want to

- [x] Frequently commit files to Git
- [x] Maintain a clean and slim Git repository
  > pre-commit hook
- [x] Keep Jupyter notebook outputs locally
  > git clean filter
- [x] Don’t let merge conflics brake the notebook
  > custom merge strategy

https://github.com/saemeon/ipynb_output_strip_pre_commit

Vita Midori

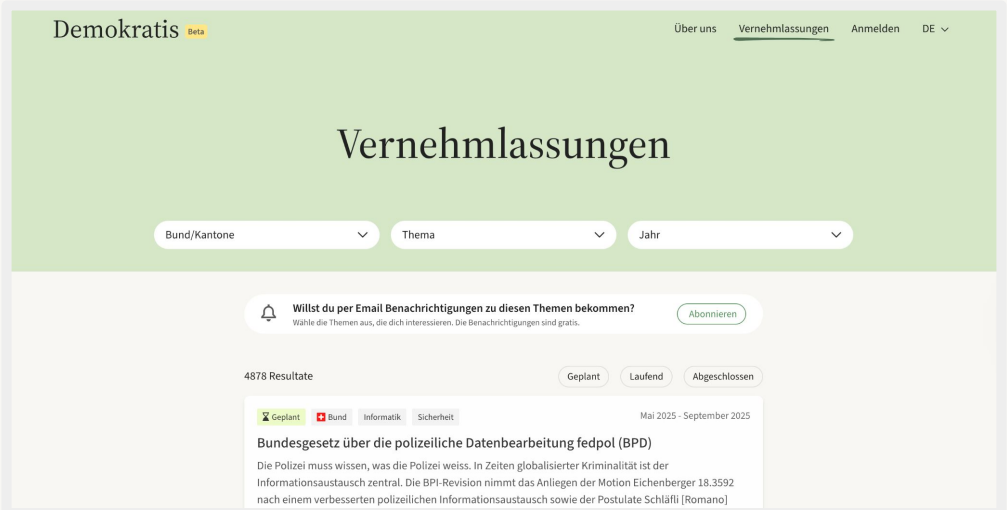
ML at Demokratis.ch

Next: Tim Head

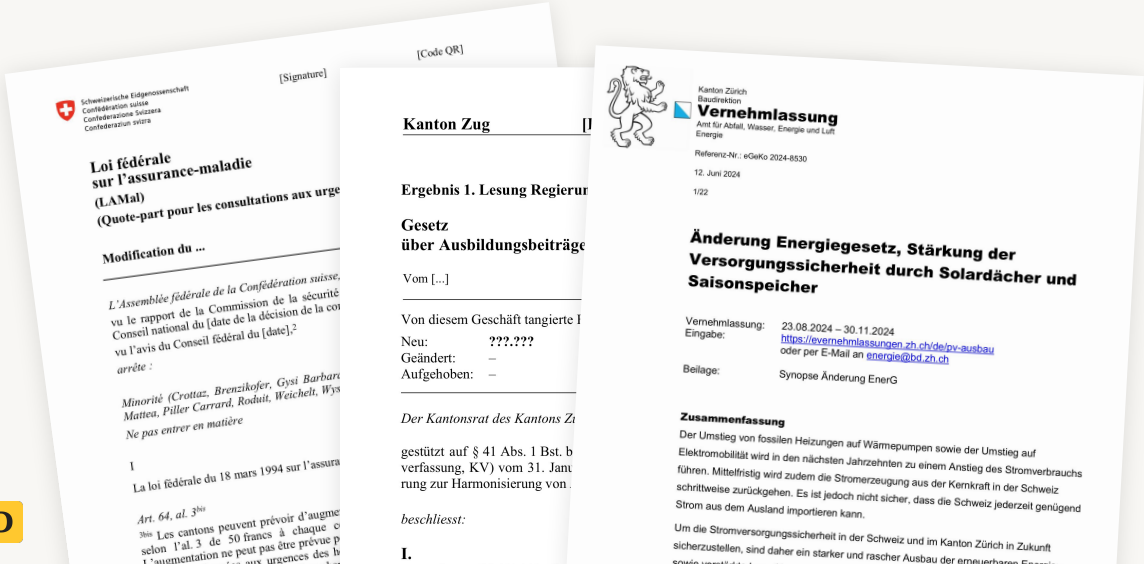
Demokratis

Digital infrastructure for
the consultation procedure

Tracking new law proposals and amendments



Anyone can give feedback and suggest changes



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

[Signature]

[Code QR]

Loi fédérale
sur l'assurance-maladie
(LAMal)
(Quote-part pour les consultations aux urge

Modification du ...

L'Assemblée fédérale de la Confédération suisse,
vu le rapport de la Commission de la sécurité
Conseil national du [date de la décision de la co
vu l'avis du Conseil fédéral du [date],[2]
arrête :

Minorité (Crottaz, Brenzikofer, Gysi Barbar
Mattea, Piller Carrard, Roduit, Weichelt, Wys
Ne pas entrer en matière

I
La loi fédérale du 18 mars 1994 sur l'assura

Art. 64, al. 3bis
[3bis] Les cantons peuvent prévoir d'augme
selon l'al. 3 de 50 francs à chaque e
L'augmentation ne peut pas être prévue p

Kanton Zug

Ergebnis 1. Lesung Regierun

Gesetz
über Ausbildungsbeiträge

Vom [...]

Von diesem Geschäft tangierte I

Neu:	**???.???**
Geändert:	–
Aufgehoben:	–

Der Kantonsrat des Kantons Z

gestützt auf § 41 Abs. 1 Bst. b
verfassung, KV) vom 31. Janu
rung zur Harmonisierung von

beschliesst:

I.

Kanton Zürich
Baudirektion
Vernehmlassung
Amt für Abfall, Wasser, Energie und Luft
Energie

Referenz-Nr.: eGeKo 2024-8530

12. Juni 2024

1/22

Änderung Energiegesetz, Stärkung der Versorgungssicherheit durch Solardächer und Saisonspeicher

Vernehmlassung:	23.08.2024 – 30.11.2024
Eingabe:	https://evernehmlassungen.zh.ch/de/pv-ausbau oder per E-Mail an energie@bd.zh.ch
Beilage:	Synopse Änderung EnerG

Zusammenfassung

Der Umstieg von fossilen Heizungen auf Wärmepumpen sowie der Umstieg auf Elektromobilität wird in den nächsten Jahrzehnten zu einem Anstieg des Stromverbrauchs führen. Mittelfristig wird zudem die Stromerzeugung aus der Kernkraft in der Schweiz schrittweise zurückgehen. Es ist jedoch nicht sicher, dass die Schweiz jederzeit genügend Strom aus dem Ausland importieren kann.

Um die Stromversorgungssicherheit in der Schweiz und im Kanton Zürich in Zukunft sicherzustellen, sind daher ein starker und rascher Ausbau der erneuerbaren Energie

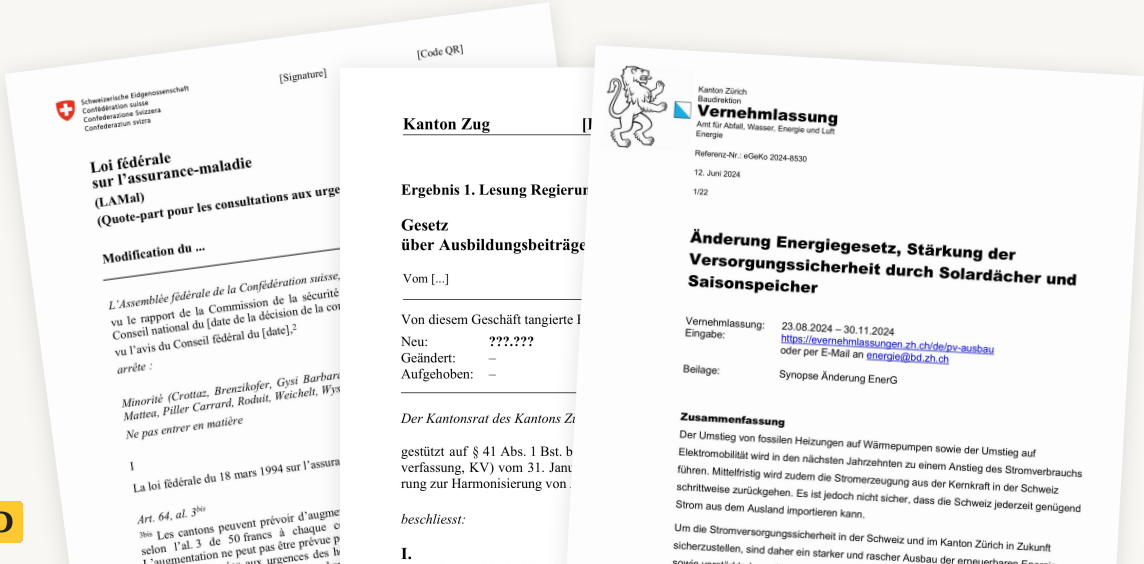
D

Problems we are currently working on

Topic classification

Document type recognition

Document structure



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

[Signature]

[Code QR]

Loi fédérale
sur l'assurance-maladie
(LAMal)
(Quote-part pour les consultations aux urge

Modification du ...

L'Assemblée fédérale de la Confédération suisse,
vu le rapport de la Commission de la sécurité
Conseil national du [date de la décision de la co
vu l'avis du Conseil fédéral du [date],[2]
arrête :

Minorité (Crottaz, Brenzikofer, Gysi Barbar
Mattea, Piller Carrard, Roduit, Weichelt, Wys
Ne pas entrer en matière

I
La loi fédérale du 18 mars 1994 sur l'assura

Art. 64, al. 3bis
[3bis] Les cantons peuvent prévoir d'augme
selon l'al. 3 de 50 francs à chaque e
L'augmentation ne peut pas être prévue p

Kanton Zug

Ergebnis 1. Lesung Regierun

Gesetz
über Ausbildungsbeiträge

Vom [...]

Von diesem Geschäft tangierte I

Neu:	**???.???**
Geändert:	–
Aufgehoben:	–

Der Kantonsrat des Kantons Z

gestützt auf § 41 Abs. 1 Bst. b
verfassung, KV) vom 31. Janu
rung zur Harmonisierung von

beschliesst:

I.

Kanton Zürich
Baudirektion
Vernehmlassung
Amt für Abfall, Wasser, Energie und Luft
Energie

Referenz-Nr.: eGeKo 2024-8530

12. Juni 2024

1/22

Änderung Energiegesetz, Stärkung der Versorgungssicherheit durch Solardächer und Saisonspeicher

Vernehmlassung:	23.08.2024 – 30.11.2024
Eingabe:	https://evernehmlassungen.zh.ch/de/pv-ausbau oder per E-Mail an energie@bd.zh.ch
Beilage:	Synopse Änderung EnerG

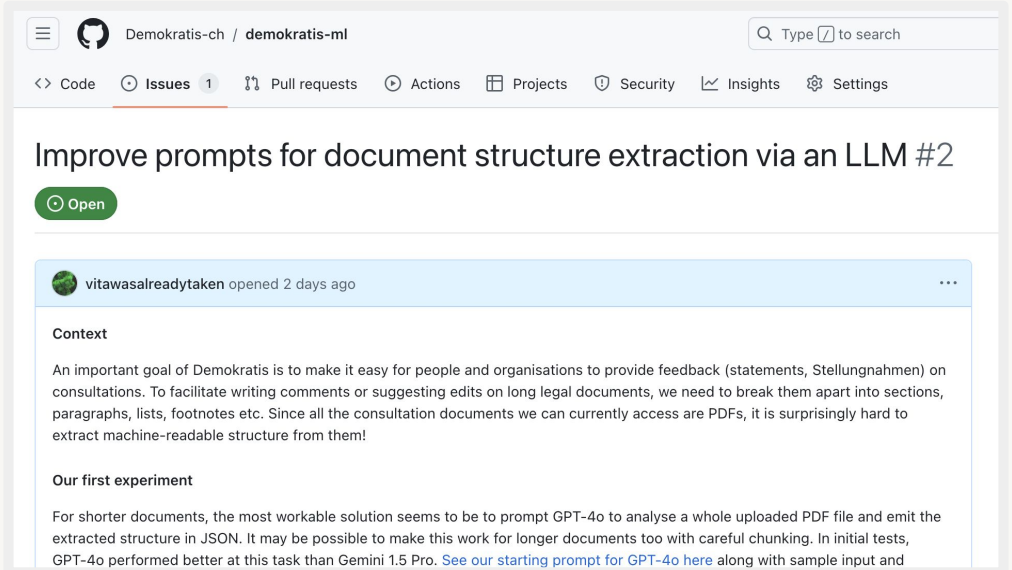
Zusammenfassung

Der Umstieg von fossilen Heizungen auf Wärmepumpen sowie der Umstieg auf Elektromobilität wird in den nächsten Jahrzehnten zu einem Anstieg des Stromverbrauchs führen. Mittelfristig wird zudem die Stromerzeugung aus der Kernkraft in der Schweiz schrittweise zurückgehen. Es ist jedoch nicht sicher, dass die Schweiz jederzeit genügend Strom aus dem Ausland importieren kann.

Um die Stromversorgungssicherheit in der Schweiz und im Kanton Zürich in Zukunft sicherzustellen, sind daher ein starker und rascher Ausbau der erneuerbaren Energie

D

```
{
  "document_title": "Vorentwurf: Bundesgesetz über Radio und Fernsehen (RTVG)",
  "amendment": "Abgabenanteile für lokale Radio- und regionale Fernsehveranstalter und Fördermassnahmen zugunsten der elektronischen Medien",
  "structure": [
    {
      "section": "I",
      "content": [
        {
          "article": "1",
          "title": "Gegenstand und Geltungsbereich",
          "paragraphs": [
            {
              "number": "1",
              "text": "Dieses Gesetz regelt:",
              "list": [
                { "index": "a", "text": "die Veranstaltung, die Aufbereitung, die Übertragung und den Empfang von Radio- und Fernsehprogrammen;" },
                { "index": "b", "text": "die Fördermassnahmen zugunsten der elektronischen Medien." }
              ]
            },
            {
              "number": "1bis",
              "text": "Soweit in diesem Gesetz nichts anderes vorgesehen ist, richtet sich die fernmeldetechnische Übertragung von Programmen nach dem Fernmeldegesetz vom 30. April 1997 (FMG)."
            }
          ]
        },
        {
          "article": "2",
          "title": "In diesem Gesetz bedeuten",
          "paragraphs": [
            {
              "index": "abis",
              "text": "Elektronische Medien: Medienangebote, die fernmeldetechnisch übertragen werden, für die Allgemeinheit bestimmt sind und nach redaktionellen Kriterien zusammengestellt werden."
            }
          ]
```


Demokratis-ch / demokratis-ml
Type / to search
Code
Issues 1
Pull requests
Actions
Projects
Security
Insights
Settings
Improve prompts for document structure extraction via an LLM #2
Open
vitawasalreadytaken opened 2 days ago
Context
An important goal of Demokratis is to make it easy for people and organisations to provide feedback (statements, Stellungnahmen) on consultations. To facilitate writing comments or suggesting edits on long legal documents, we need to break them apart into sections, paragraphs, lists, footnotes etc. Since all the consultation documents we can currently access are PDFs, it is surprisingly hard to extract machine-readable structure from them!
Our first experiment
For shorter documents, the most workable solution seems to be to prompt GPT-4o to analyse a whole uploaded PDF file and emit the extracted structure in JSON. It may be possible to make this work for longer documents too with careful chunking. In initial tests, GPT-4o performed better at this task than Gemini 1.5 Pro. See our starting prompt for GPT-4o here along with sample input and

demokratis.ch/ml

Tim Head

Zero Code Change Acceleration

Next: Josua Schmid

We love pandas,
but it can be slow

Accelerated pandas

cudf.pandas: the zero code change GPU accelerator for pandas built on cuDF

```
%%time
import pandas as pd

weekday_names = {
    0: "Monday",
    1: "Tuesday",
    2: "Wednesday",
    3: "Thursday",
    4: "Friday",
    5: "Saturday",
    6: "Sunday",
}

df = pd.read_parquet("nyc_parking_violations_2022.parquet")
df["Issue Date"] = df["Issue Date"].astype("datetime64[ms]")
df["issue_weekday"] = df["Issue Date"].dt.weekday.map(weekday_names)
df.groupby(["issue_weekday"])["Summons Number"].count().sort_values()
```

```
CPU times: user 48.6 s, sys: 10.4 s, total: 59 s
Wall time: 36.7 s
issue_weekday
Sunday         462992
Saturday      1108385
Monday        2488563
Wednesday     2760088
Tuesday       2809949
Friday        2891679
Thursday      2913951
Name: Summons Number, dtype: int64
```

```
%load_ext cudf.pandas
```

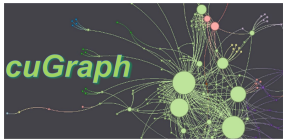
```
%%time
import pandas as pd

weekday_names = {
    0: "Monday",
    1: "Tuesday",
    2: "Wednesday",
    3: "Thursday",
    4: "Friday",
    5: "Saturday",
    6: "Sunday",
}

df = pd.read_parquet("nyc_parking_violations_2022.parquet")
df["Issue Date"] = df["Issue Date"].astype("datetime64[ms]")
df["issue_weekday"] = df["Issue Date"].dt.weekday.map(weekday_names)
df.groupby(["issue_weekday"])["Summons Number"].count().sort_values()
```

```
CPU times: user 601 ms, sys: 141 ms, total: 742 ms
Wall time: 720 ms
issue_weekday
Sunday         462992
Saturday      1108385
Monday        2488563
Wednesday     2760088
Tuesday       2809949
Friday        2891679
Thursday      2913951
Name: Summons Number, dtype: int64
```

Accelerated NetworkX

Zero code change acceleration for NetworkX, powered by cuGraph

NetworkX
Network Analysis in Python




```python
import pandas as pd
import networkx as nx

url = "https://data.rapids.ai/cugraph/datasets/cit-Patents.csv"
df = pd.read_csv(url, sep=" ", names=["src", "dst"], dtype="int32")
G = nx.from_pandas_edgelist(df, source="src", target="dst")

%time result = nx.betweenness_centrality(G, k=10)
```

```
user@machine:/# ipython bc_demo.ipy
CPU times: user 7min 38s, sys: 5.6 s, total: 7min 44s
Wall time: 7min 44s

user@machine:/# NETWORKX_BACKEND_PRIORITY=cugraph ipython bc_demo.ipy
CPU times: user 18.4 s, sys: 1.44 s, total: 19.9 s
Wall time: 20 s
```

pip install nx-cugraph-cu12 --extra-index-url https://pypi.nvidia.com
conda install -c rapidsai -c conda-forge -c nvidia nx-cugraph

Accelerated Polars

Zero code change acceleration for Polars, powered by cuDF


```python
query = (
    transaction.group_by("cust_id").agg(
        pl.sum("amount")
    ).sort(by="amount", descending=True)
    .head()
)

# Run on the CPU
result_cpu = query.collect()

# Run on the GPU
result_gpu = query.collect(engine="gpu")

# assert both result are equal
pl.testing.assert_frame_equal(result_gpu, result_cpu)
```

pip install polars[gpu] -U --extra-index-url=https://pypi.nvidia.com

Zero code change

Your existing code, but faster

Josua Schmid

Funny Slides for Data Scientist

Next: Ricardo Pereira

Renuo

Funny Slides for Data Scientists

or calling bullshit

Lightning talk at Swiss Python Summit 2024

callingbullshit.org

Based on the video recordings of a University of Washington's class called "Calling Bullshit" in Seattle:

"[...] the aim of this course is to help students navigate the bullshit-rich modern environment by identifying bullshit [...]"

10 hours of video:
https://callingbullshit.org/videos.html

More people like fresh juice!

Fresh

From Concentrate

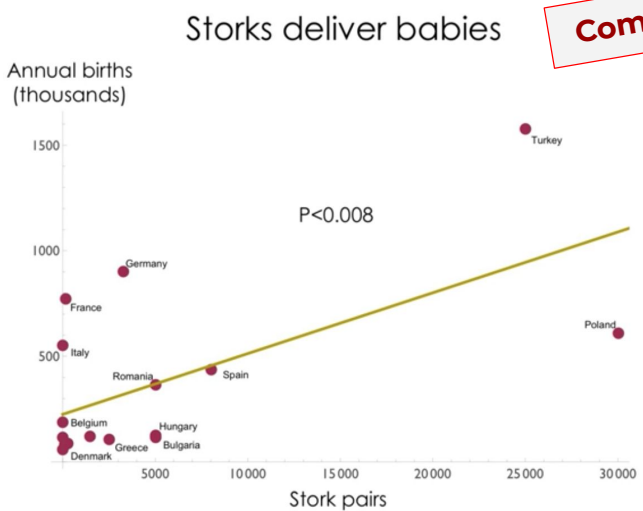
Apple vs Oranges

21 people

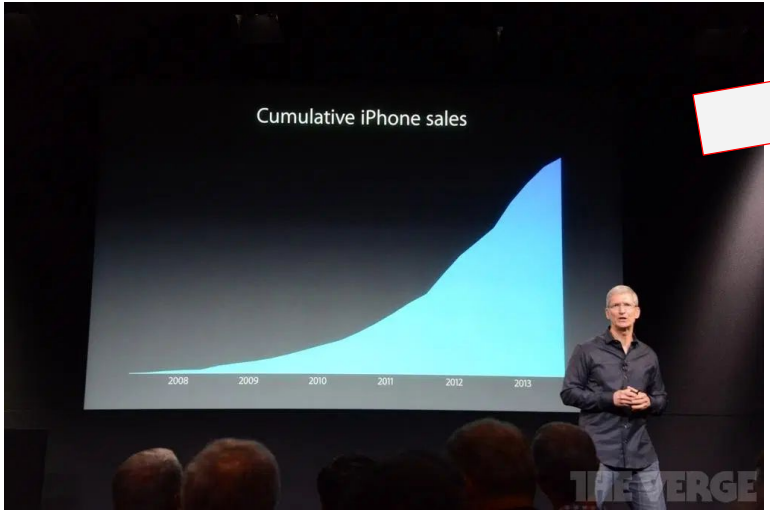
4 people

Causation vs Relation

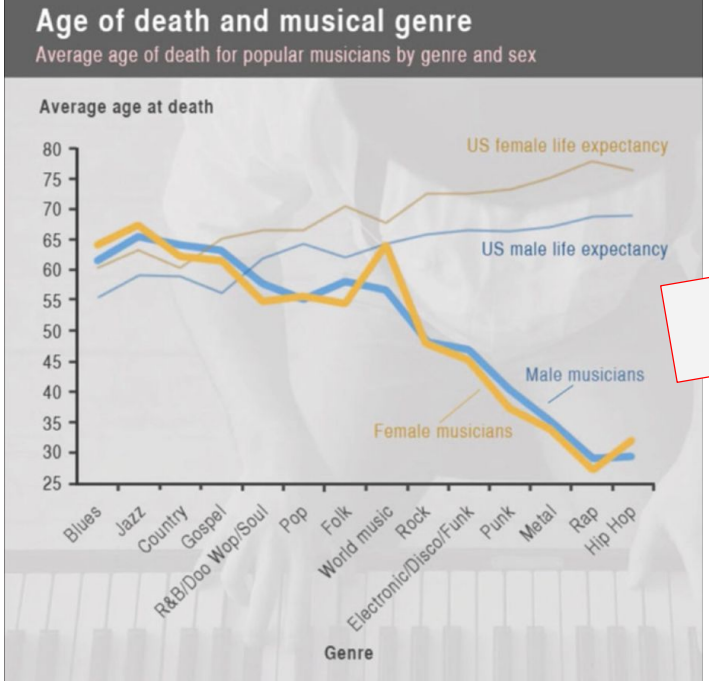
Common Cause "Country Size"



Much growth



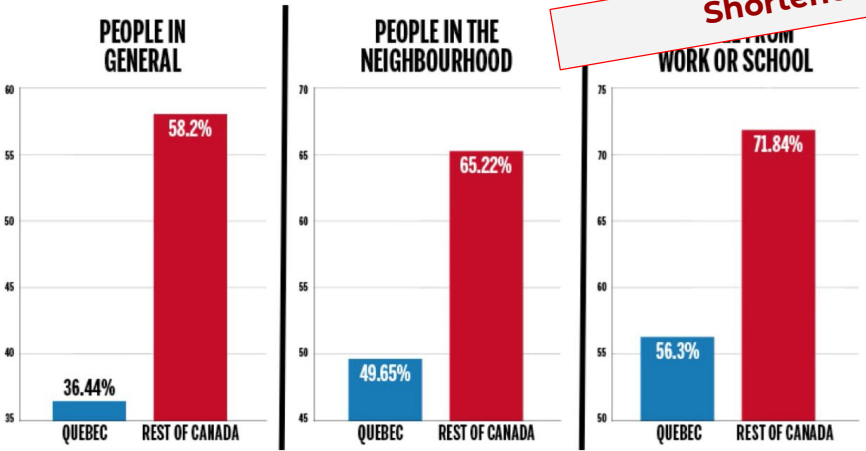
"Cumulative"



Right Censoring
They are still alive

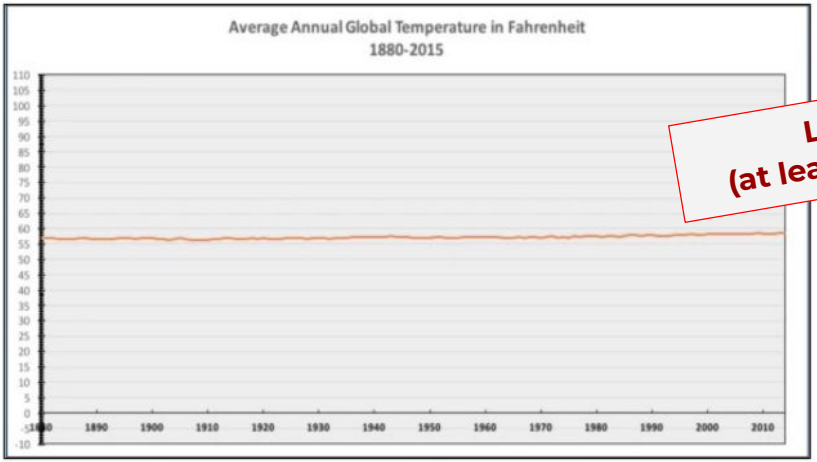
Do you trust...

Shortened Axis



MACLEAN'S

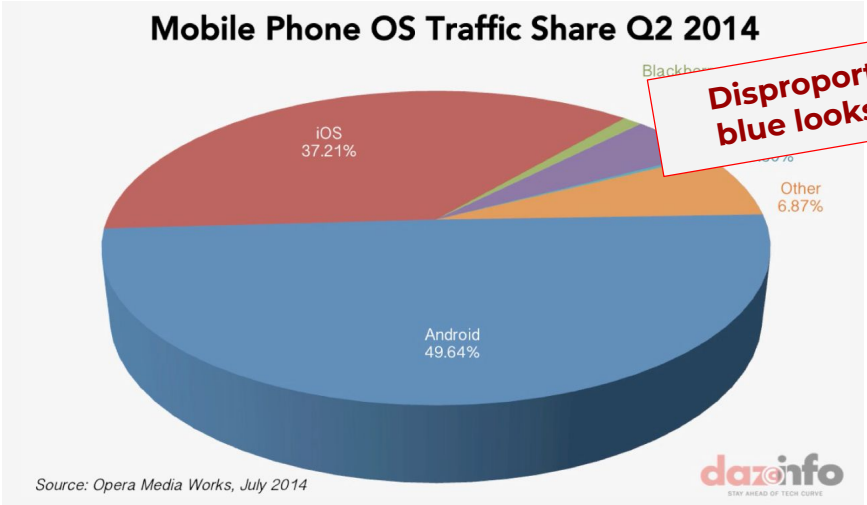
"THE ONLY GLOBAL WARMING CHART YOU NEED FROM NOW ON"



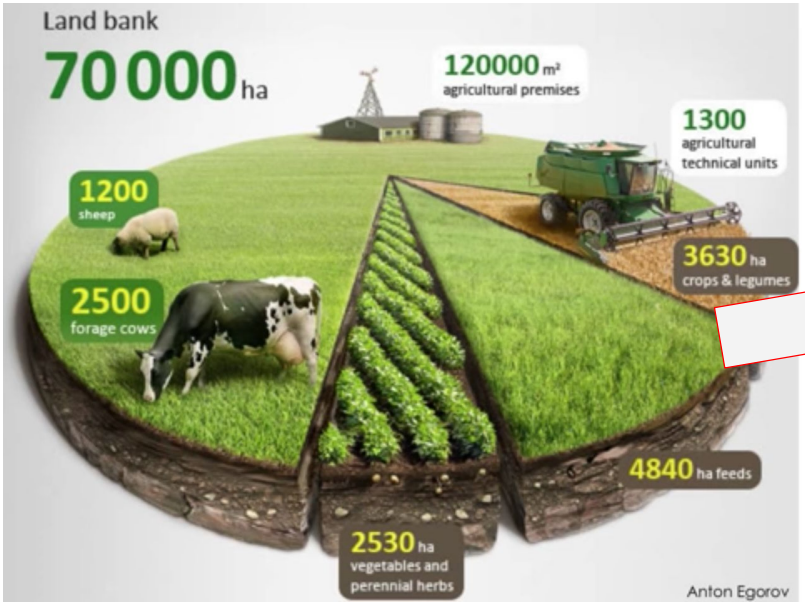
Long Axis (at least Fahrenheit)

Powerline blog

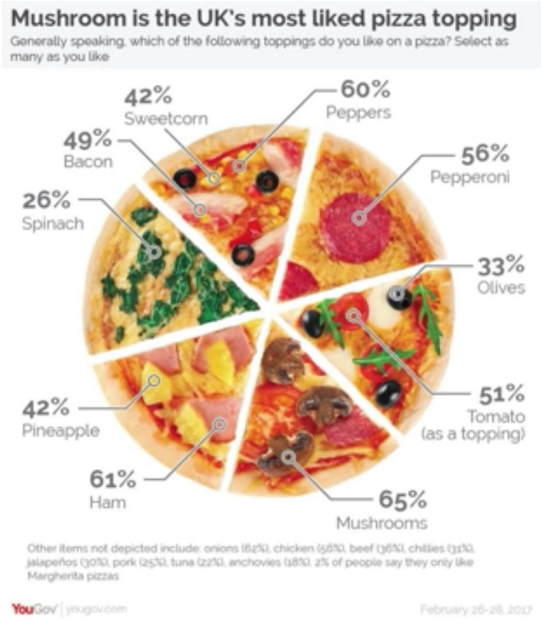
Proportional ink



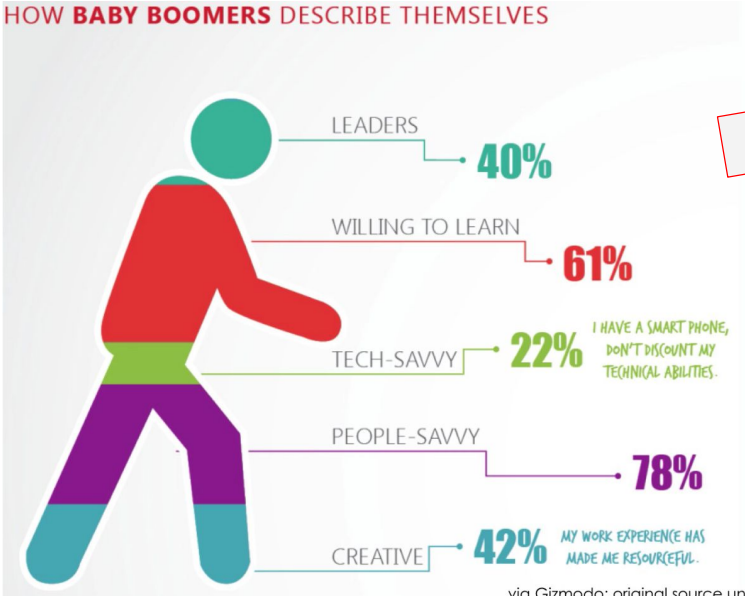
Disproportional ink blue looks like 70%


Land bank
70 000 ha
120000 m²
agricultural premises
1300
agricultural
technical units
1200
sheep
2500
forage cows
3630 ha
crops & legumes
4840 ha feeds
2530 ha
vegetables and
perennial herbs
Anton Egorov

Units?



Pizza chart have slice-sized slices

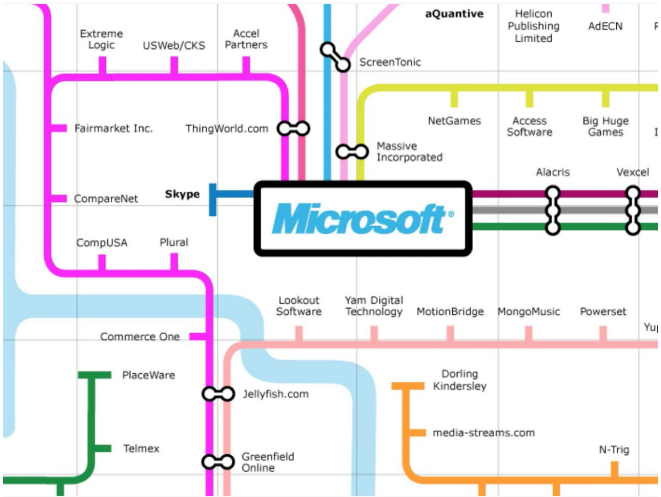


via Gizmodo; original source unknown

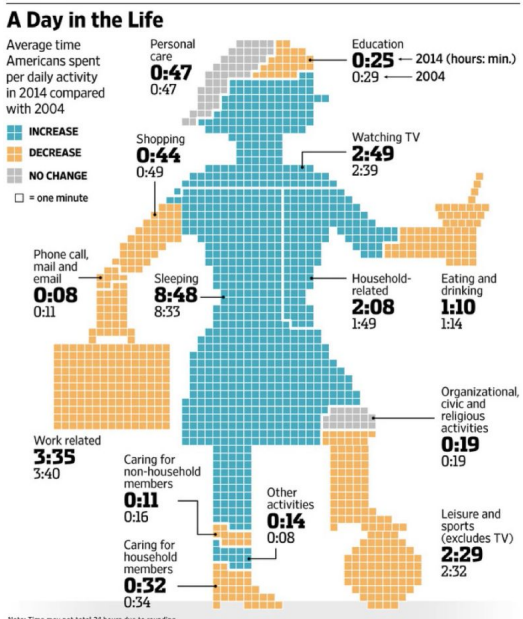
Is that even a graph?

"glass slipper"

"glass slipper"




via Washington Post
Trust
Partnership
Innovation
Performance
OUR VALUES
THOMSON REUTERS



These are rates: increase / decrease

The Wall Street Journal

Real House Prices over the Past Year

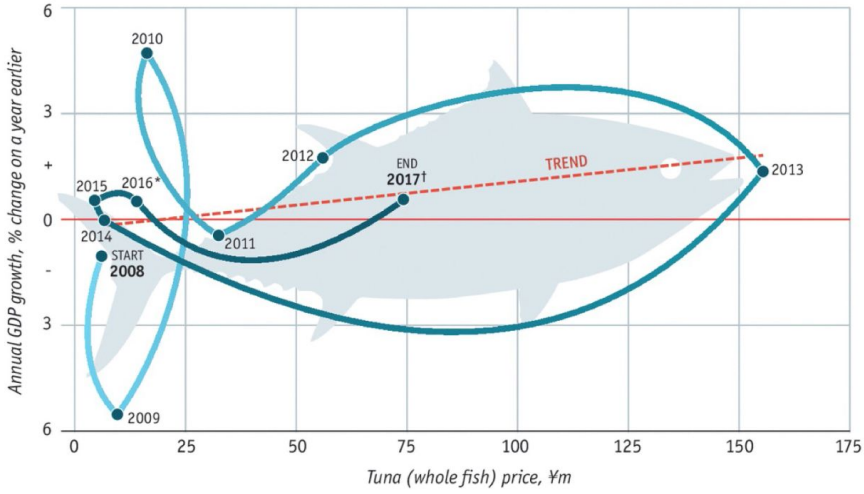
Real house prices increased over the past year in most countries.

(2016:Q4 or latest, annual percent change)

Sources: Bank of International Settlements, European Central Bank, Federal Reserve Bank of Dallas, Savills, Sinyi Real Estate Planning and Research, and national sources.

Economy of scales

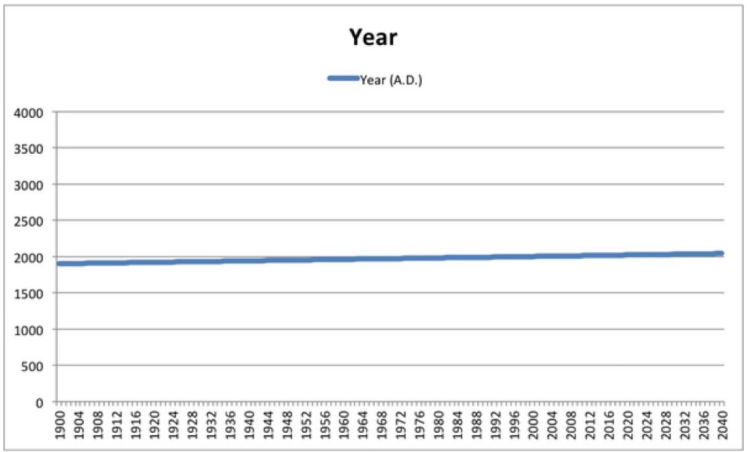
Tsukiji market, first tuna prices v Japan GDP growth, 2008-17



Sources: IMF; press reports

*GDP estimate †GDP forecast

I hope time did actually pass



Bloomberg's @Bizweekgraphics

Thank You

Renuo

Renuo AG
Industriestrasse 44
8304 Wallisellen

I stole everything from
https://callingbullshit.org/videos.html

www.renuo.ch
hello@renuo.ch
+41 44 500 55 66

Ricardo Pereira

Python Superset

Next: Stefan Keller

Data Context

- Meaning
- Quality
- Decision-making

Data Visualization

- Remove outliers
- Discover trends
- Yield insights

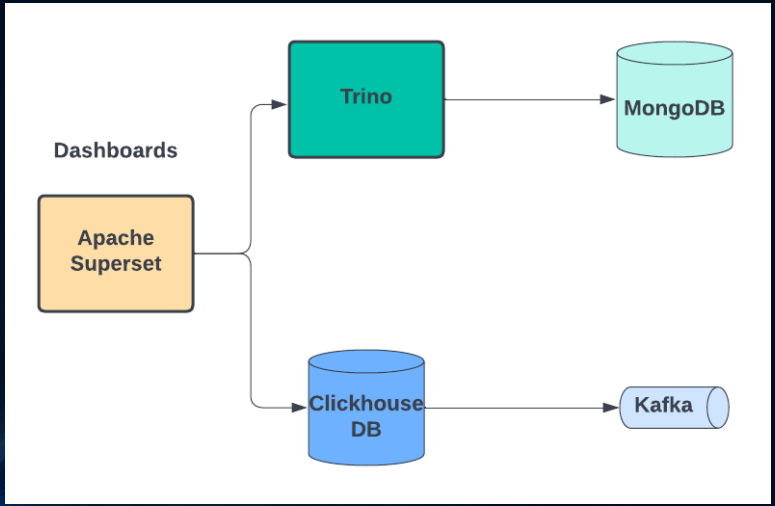
Paid

- Tableau
- Power BI

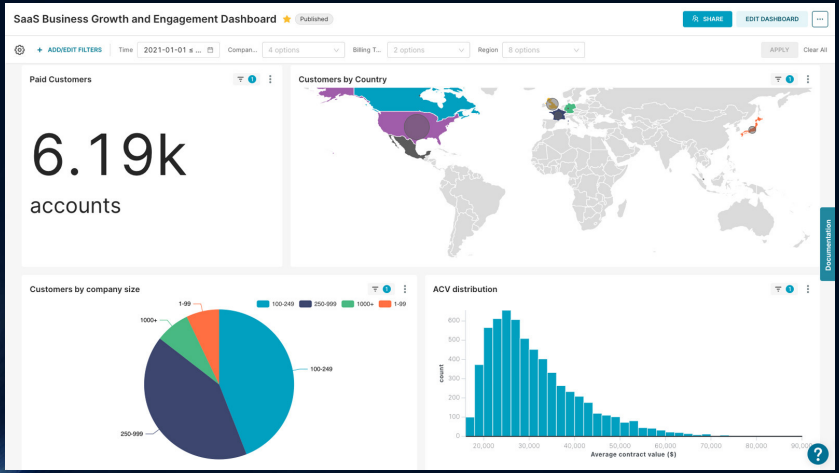
Free

- Apache superset

Infra


Trino
MongoDB
Dashboards
Apache
Superset
Clickhouse
DB
Kafka

Example



https://superset.apache.org/

Stefan Keller

Pythons and Ducks!

Next: Tamara Reuveni Mazig

Swiss
Python
Summit

DuckDB

Pythons and Ducks!

Lightning Talk #2

Prof. Stefan Keller

Institute for Software, FH OST Campus Rapperswil, ost.ch/ifs

Pythons and Ducks …

or: Underengineering and Overengineering are the Root of all Evil – and SQL could save us all!

Disclaimer: I'm a Geospatial Data Engineer and an evangelist for modern SQL, Postgres, OpenStreetMap and “Open Whatever”!

Underengineering is the Root of all Evil

- Python code directly analyzing CSV files ☹
  - Use at least (Geo)Parquet or (Geo)Arrow

- Pandas 
  - is the go-to tool for small to mid-sized datasets
  - But, as the dataset grows, plain DataFrames hit their limits: memory consumption, slow performance, and lack of parallelism

And Overengineering is the Root of all Evil too

- Why throwing cloud-solutions and NoSQL at every problem?

- Distributed storage and cloud solutions
  - are powerful, but not always necessary (and often a vendor-lock-in)
- Overkill for small or medium workloads:
  - slower development cycle, higher costs, and unnecessary complexity
- «Big Data is dead»
  - … except for few big companies (Source: https://motherduck.com/blog)
  - Do you really need all your data as «hot data»?

Solution 1: SQL …

- A word about SQL …

  → Who dares to admit they hate SQL ☺?

- A lot of people still think of SQL as being SQL:92 – especially marketeers
  - That's Windows 3 and the 90's vibes →
- Modern SQL is Window Functions, Common Table Expressions (CTEs), JSON Functions, Complex Types like arrays, maps, and structs
- SQL:2023 covers even Graphs!

Solution 1: SQL … cont.

- SQL saves pages of Python code
- SQL means processing close to the data
- SQL stays performant even with growing data

  → Who from you did ever rewrite code because of increased datasets?
- The “Elephants” are NoSQL too:
  - They are “Multi-Model DBMS”!
  - Support even LLM vectors

DB-Engines Ranking of Document Stores

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

This is a partial list of the complete ranking showing only document stores.

Read more about the method of calculating the scores.

☑ include secondary database models — 96 systems in ranking, S

Rank Oct 2024	Rank Sep 2024	Rank Oct 2023	DBMS	Database Model	Score Oct 2024
1.	1.	1.	Oracle	Relational, Multi-model	1309.45
2.	2.	2.	MySQL	Relational, Multi-model	1022.76
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	802.09
4.	4.	4.	PostgreSQL	Relational, Multi-model	652.16
5.	5.	5.	MongoDB	Document, Multi-model	405.21

Solution 2: In-process databases!

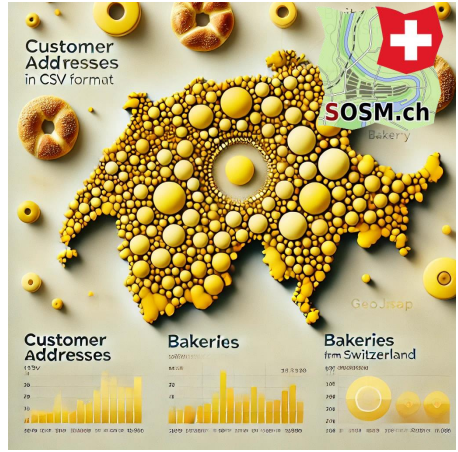
DuckDB

- DuckDD
  - eliminates (sometimes!) the need for centralized databases and the need for over-the-top cloud infrastructure
- DuckDB
  - is the “SQLite for Analytics”
  - is fast, lightweight, and designed for analytics workloads
  - supports modern SQL
- SQL and DuckDB fit perfectly in your Python environment

Customers and Bakeries Example

- Rank customers by number of nearby bakeries

- Customers are in CSV format: Use a geocoder like Nominatim from OSM to convert postal addresses to lat/lon
- Bakeries are in GeoJSON format and come from OpenStreetMap Switzerland (OSM.ch): Use e.g. Overpass Turbo to download



Python and SQL code ahead!

Install DuckDB
first https://duckdb.org/
then % pip install duckdb

Rank customers by number of nearby bakeries (Slide 1 of 2)

```
# Inspired by https://motherduck.com/blog/getting-started-gis-duckdb/

import duckdb

# Initialize DuckDB connection and load spatial extension
con = duckdb.connect('md:')
con.sql('INSTALL spatial;')
con.sql('LOAD spatial;')

# Create table for customers by reading the CSV file
customers_file = 'customers.csv'
con.sql(f"""
CREATE TABLE customers AS SELECT * FROM read_csv_auto('{customers_file}')  -- Schema detection: creates automatically a table;
""")

# Ensure the customers table contains latitude and longitude, and convert it to a geometry attribute called "geom"
con.sql("""
ALTER TABLE customers ADD COLUMN geom GEOMETRY;
UPDATE customers SET geom = ST_Point(longitude, latitude);
""")
```


```
# Create table for bakeries Switzerland from GeoJSON file; GeoJSON extends JSON with types Point, LineString, Polygon
con.sql("""
CREATE TABLE bakeries_ch AS
    SELECT properties->>'name' AS name, properties->>'addr:city' AS city, ST_GeomFromGeoJSON(geometry) AS geom
    FROM ST_Read('bakeries_ch.geojson')
""")

# Count bakeries within 1000m of each customer
con.sql("""
CREATE TABLE customers_with_bakeries AS
    SELECT c.*, COUNT(b.geom) AS bakeries_count
    FROM customers AS c
    LEFT JOIN bakeries_ch AS b
    ON ST_DWithin(c.geom, b.geom, 1000)  -- This is the magic spatial relationship function
    GROUP BY c.*, c.geom
    ORDER BY bakeries_count DESC
""")

# Export result to GeoJSON
con.sql("""
COPY (SELECT *, ST_AsGeoJSON(geom) AS geometry FROM customers_with_bakeries)
TO 'customers_with_bakeries.geojson'
WITH (FORMAT GDAL, DRIVER 'GeoJSON')  -- GDAL (OGR) is a read/write lib. for those weird spatial formats
""")
```

Even DuckDB has it's limitations…

PostgreSQL

When to use DuckDB

- Processing and storing tabular datasets, e.g., from CSV or Parquet files
- Interactive data analysis, e.g., join & aggregate multiple large tables
- Concurrent large changes, to multiple large tables, e.g., appending rows, adding/removing/updating columns
- Large result set transfer to client

When to not use DuckDB

- High-volume transactional use cases (e.g., tracking orders in a webshop)
- Large client/server installations for centralized enterprise data warehousing
- Writing to a single database from multiple concurrent processes
- Multiple concurrent processes reading from a single writable database

So use e.g. PostgreSQL for a reliable , extensible, scalable centralized store


SQL and DuckDB may save your day!
Fin
ost.ch/ifs
stefan.keller@ost.ch

Tamara Reuveni Mazig

Topic Modelling for Customer experience

Next: Jan Werth

Topic Modelling for Customer Expericence Radar

18.10.2024

BEKB | BCBE

BEKB | BCBE

What are our customers' main needs?

Challenges

How do we get meaningful customer needs from ~150 thousand protocols of customer interactions?

- Technical Limitations – Data Protection
- Fine grained topics required
- Many irrelevant words
- Indirect Speech
- Typos, abbreviations etc

Solution

Create synonyms with Word2Vec.

- Remove stopwords more effectively
- Reduce vocabulary size from ~20 thousand to ~3.5 thousand words
- Enable meaningful topic generation with LDA

Words mit synonym **bancomat**

- bankomat
- bankomaten
- bancomaten
- banci
- bacomat
- bankautomat
- fremdbankautomaten
- geldautomat
- bankautomaten
- eaa
- gaa
- automaten
- automat

Python

```python
def generate_synonym_df(self):

    """Generate dataframe of all words in dataset and their corresponding synonyms"""

    # Get thresholds for synonymisation mased on word counts
    cnts_wordlist['threshold'] = self.get_thresholds(cnts_wordlist['count'])

    # Threshold for product names is set to 1 so that these terms aren't changed to any synonyms
    prod_names = [x for x in list(self.prod_set) if x in cnts_wordlist.index.tolist()]
    cnts_wordlist.loc[prod_names, 'threshold'] = 1

    # Sort words in descending order based on threshold
    cnts_wordlist = cnts_wordlist.sort_values(by = ['threshold','count'], ascending = False)

    # Initiate columns
    cnts_wordlist['most_similar'] = cnts_wordlist.index
    cnts_wordlist['remove'] = False
    cnts_wordlist['parent'] = ''

```

Python

```
    # Iterate over wordlist
    for i, word in enumerate(tqdm(cnts_wordlist.index.tolist())):
        current_synonym = cnts_wordlist.loc[word, 'most_similar'] # The current synonym is the synonym of the current word

        # Get 10 most similar words and their similarity (value between 0 and 1) according to wv-model
        word_similarities = {item[0]:item[1] for item in self.wv_model.wv.most_similar(word, topn=10)}

        # Iterate over most similar words (related words) to check which of them will get the current synonym as their synonym
        for sim in word_similarities.keys():
            # Check if the related word is a more frequent word than the original word
            if self.wv_counts.loc[sim, 'count'] < self.wv_counts.loc[word, 'count']:
                if sim not in self.memo_wordlist: # If the related word doesn't appear in the memo dataset, it is not relevant
                    continue
                elif sim != cnts_wordlist.loc[sim, 'most_similar']: # If the related word has already been synonymized, continue
                    continue

                # Check if the similarity of the related word to the word is higher than the related word's threshold
                elif word_similarities[sim] > cnts_wordlist.loc[sim, 'threshold']:
                    cnts_wordlist.loc[sim, 'most_similar'] = current_synonym # Set current synonym as the related word's synonym
                    cnts_wordlist.loc[sim, 'parent'] = word # The parent of the related word is the current word

        # Check if current synonym is a stopword, then synonymized related words should be removed
        if current_synonym in self.data.remove_words_after_stemming:
            cnts_wordlist.loc[cnts_wordlist.most_similar == current_synonym, 'remove'] = True

    return cnts_wordlist
```

Rechtliche Hinweise

Werbung: Bei vorliegender Publikation handelt es sich um Werbung für ein Finanzinstrument. Sie richtet sich ausschliesslich an Kunden mit entsprechenden Kenntnissen und Erfahrungen mit Wohnsitz bzw. Sitz in der Schweiz. Die Informationen, Produkte und Dienstleistungen sind nicht für Personen bestimmt, die aufgrund von Nationalität, Wohnsitz bzw. Sitz oder sonstiger Gründe einer Rechtsordnung unterliegen, die es ausländischen Finanzdienstleistern verbietet, dort geschäftlich tätig zu sein oder den ihr unterliegenden juristischen oder natürlichen Personen den Zugang zu Informationen, Produkten oder Dienstleistungen ausländischer Finanzdienstleister verbietet oder einschränkt. Personen, die solchen lokalen Beschränkungen unterstehen, ist die Nutzung oder Weitergabe dieser Informationen, Produkte und Dienstleistungen untersagt. Der Verkaufsprospekt mit integriertem Fondsvertrag und das Basisinformationsblatt sind an einem Standort der Berner Kantonalbank AG («BEKB») sowie unter **www.fundinfo.com** kostenlos erhältlich.

Kein Angebot und keine Beratung: Die obigen Informationen dienen ausschliesslich Informationszwecken. Sie stellen insbesondere keine Aufforderung, kein Angebot und keine Empfehlung zum Kauf oder Verkauf von Produkten, zur Ausführung von Transaktionen oder zum Abschluss irgendeines Rechtsgeschäfts dar. Im Weiteren stellen die publizierten Informationen keine Beratung weder in rechtlicher noch in steuerlicher, wirtschaftlicher oder sonstiger Hinsicht dar. Die Informationen haben einzig beschreibenden Charakter und ersetzen keinesfalls eine persönliche Beratung durch eine qualifizierte Fachperson.

Haftungsausschluss: Die im vorliegenden Dokument enthaltenen Daten, Analysen und Beurteilungen ("Angaben") enthalten Informationen von Datenlieferanten und deren Zulieferer ("Drittlieferanten"). Die BEKB und die Drittlieferanten, schliessen ausdrücklich die Gewährleistung für die Aktualität, Richtigkeit, Genauigkeit, Vollständigkeit oder Marktfähigkeit der Angaben aus. Weder die BEKB noch die Drittlieferanten haften für Anlageentscheidungen, Schäden oder Verluste, die mit den Angaben oder den Berechnungen von möglicherweise angewendeten Indices im Zusammenhang stehen oder aus deren Nutzung resultieren. Im Weiteren haften die BEKB und die Drittlieferanten in keinem Fall für unmittelbare oder mittelbare Schäden. Die publizierten Informationen gelten als vorläufig und unverbindlich. Ein bestimmtes Abschneiden in der Vergangenheit ist keine Gewähr für künftige Ergebnisse. Der Wert der Anlage und die Einkünfte aus einer Anlage können sinken und steigen. Die BEKB ist nicht verpflichtet, nicht mehr aktuelle Informationen zu entfernen oder diese ausdrücklich als solche zu kennzeichnen. Kein Teil des vorliegenden Dokuments darf ohne vorherige ausdrückliche Zustimmung der BEKB kopiert, vervielfältigt oder weitergeleitet werden.

Weitere Disclaimer von Datenlieferanten: **https://www.bekb.ch/de/die-bekb/rechtliche-informationen#datenquellen**


BEKB | BCBE

Jan Werth

AI & our Brain $\rightarrow$ not quite there yet

Next: Florian Bruhin

Florian Bruhin

fstring.help

Next: Florian Bruhin

fstring.help

Or: How I bought yet another domain

Florian Bruhin

BRUHIN
SOFTWARE

Swiss Python Summit 2024
October 18th

PyFormat

Using `%` and `.format()` for great good!

Contribute on GitHub

Python has had awesome string formatters for many years but the documentation on them is far too theoretic and technical. With this site we try to show you the most common use-cases covered by the old and new style string formatting API with practical examples.

All examples on this page work out of the box with with Python 2.7, 3.2, 3.3, 3.4, and 3.5 without requiring any additional libraries.

Further details about these two formatting methods can be found in the official Python documentation:

- old style
- new style

If you want to contribute more examples, feel free to create a pull-request on Github!

Basic formatting

Simple positional formatting is probably the most common use-case. Use it if the order of your arguments is not likely to change and you only have very few elements you want to concatenate.

Since the elements are not represented by something as descriptive as a name this simple style should only be used to format a relatively small number of elements.

Old

```
'%s %s' % ('one', 'two')
```

New

```
'{} {}'.format('one', 'two')
```

Output

```
o n e  t w o
```

PEP 0498: f-Strings #24

Open **hut8** opened this issue on Oct 4, 2015 · 11 comments

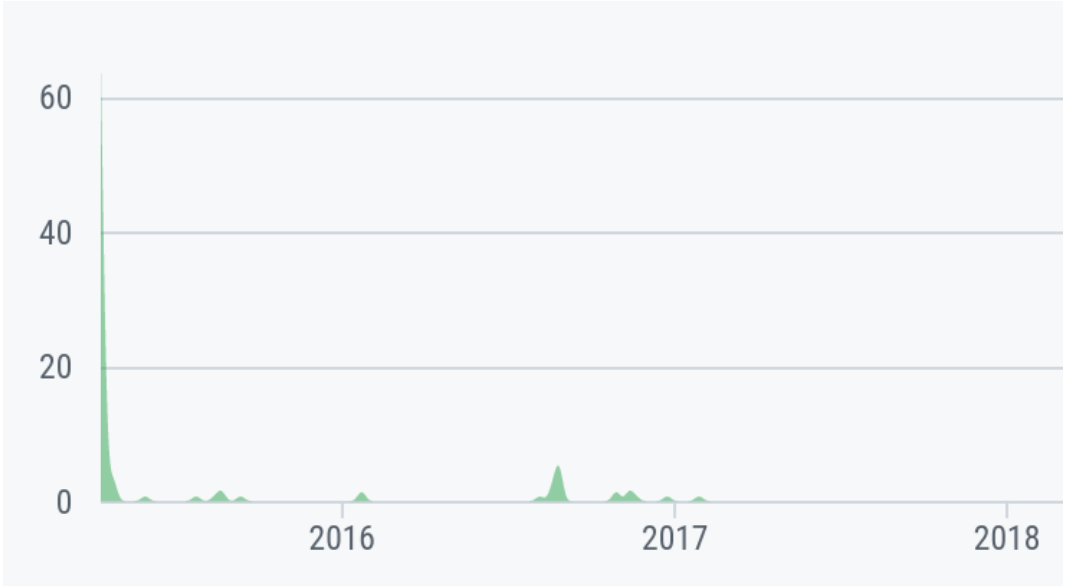
PEP 498 -- Literal String Interpolation #50

Open **danielniccoli** opened this issue on Dec 13, 2017 · 7 comments

Incorporating f-Strings (PEP 498) #56

Open **udincer** opened this issue on Apr 25, 2019 · 1 comment

Contributions to master, excluding merge commits and bot accounts


60
40
20
0
2016
2017
2018

master

Commits on Feb 4, 2017

Mention v2 in README

 **ulope** committed on Feb 4, 2017

Commits on Dec 27, 2016

Slightly reword uneven center align padding note

v2

Commits on Jan 16, 2018

Remove Python 3.3 from testing

 **ulope** committed on Jan 16, 2018 ✗

Fix circleci config

 **ulope** committed on Jan 16, 2018 ✗

$\approx$ 2019: fstring.dev?

At PyConDE 2022:
OMG let's do it!
Lightning talks!!!1!

fstring.dev

DEV

Domain not available Whois

Domain Name: fstring.dev

...

Updated Date: 2022-11-01T17:46:43Z

Creation Date: **2019-09-17**T17:46:43Z

...

Registrant Name: REDACTED FOR PRIVACY

Registrant Organization: **Contact Privacy Inc. Customer 7151571251**

Unable to load page

Error while opening http://fstring.dev/
ERR_NAME_NOT_RESOLVED

Try again

fstri.ng

Nigeria

99.00 €

Verlängerung: 99.00 €

fstring.cat

CAT

PROMO 5.00 €

~~**13.00 €**~~

Verlängerung: 25.00 €

Restrictions [edit]

The .cat domain is not territorial, but applies to the whole Catalan-speaking community, whether or not a site is based in Catalonia. In order to be granted a .cat domain, one needs to belong to the Catalan linguistic and cultural community on the Internet. A person, organization or company is considered to belong if they either:[5]

[A] 1: NYAN.CAT!

Pick a language!

NON-STOP NYAN CAT

CREDITS STATS

CHECK OUT NYAN CAT ON FACEBOOK!

GET YOUR YOUTOOZ COLLECTIBLE TODAY!

ADOPT AN OFFICIAL NYAN CAT NFT

https://www.nyan.cat/index.php?cat=original[top][<][1/1]

fstring.help

HELP

32.80 €

Verlängerung: 32.80 €

Python f-strings

All currently supported Python versions (3.6+) support string-formatting via f-strings. While PEP 498 (Literal String Interpolation) as well as the Python documention (tutorial, syntax reference) have some information on their usage, I was missing a reference which is terse, but still verbose enough to explain the syntax.

Thus, fstring.help was born, made with ❤️ and jupyter (for now, as a quick hack at PyConDE 2022).

Created by BRUHIN SOFTWARE
Consulting, coaching and development for pytest, Qt and Python.

Some content is copied verbatim from pyformat.info (Copyright 2015 Ulrich Petri, Horst Gutmann). Thanks!

Repository on Github, contributions welcome! If you prefer an interactive version, launch binder.

Basic formatting

f-strings are strings with an `f` in front of them: `f"..."`. Inside the f-string, curly braces can be used to format values into it:

In [1]:
```
one = 1
two = 2
```

In [2]:
```
f"{one} {two}"
```

Out[2]:
```
'1 2'
```

Arbitrary code

You can put any Python code into f-strings:

Python f-string cheat sheets

See fstring.help for more examples and for a more detailed discussion of this syntax see this string formatting article.

All numbers

The below examples assume the following variables:

```
>>> number = 4125.6
>>> percent = 0.3738
```

Example Output	Replacement Field	Fill	Width	Grouping	Precision	Type
'4125.60'	{number:.2f}				.2	f
'4,125.60'	{number:,.2f}			,	.2	f
'04125.60'	{number:08.2f}	0	8		.2	f
' 4125.60'	{number: 8.2f}		8		.2	f
'37%'	{percent:.0%}				.0	%

These format specifications only work on all numbers (both `int` and `float`).

Recent news

Python 3.7 is EOL: All Python versions support self-documenting expressions:

```
>>> user = 'eric_idle'
>>> member_since = date(1975, 7, 31)
>>> f'{user=} {member_since=}'
"user='eric_idle' member_since=datetime.date(1975, 7, 31)"
```

Recent news

Python 3.7 is EOL: All Python versions support self-documenting expressions:

```
>>> user = 'eric_idle'
>>> member_since = date(1975, 7, 31)
>>> f'{user=} {member_since=}'
"user='eric_idle' member_since=datetime.date(1975, 7, 31)"
```

Python 3.12: PEP 701 – Syntactic formalization of f-strings:
Nested quotes and other valid Python expressions now work inside f-strings:

```
>>> f"This is the playlist: {", ".join([
...     'Take me back to Eden',  # My, my, those eyes like fire
...     'Alkaline',              # Not acid nor alkaline
...     'Ascensionism'           # Take to the broken skies at last
... ])}"
'This is the playlist: Take me back to Eden, Alkaline, Ascensionism'
```

https://**fstring.help**

https://github.com/
The-Compiler/fstring.help

https://twitter.com/**the_compiler**
florian@**bruhin.software**

Florian Bruhin

A .py/.tex/.pdf quine

genslides.py

```
1 import subprocess
2 from pathlib import Path
3 from jinja2 import Environment
```

genslides.py

```
6  TALKS: list[tuple[str, str, str | None]] = [
7      ("Dominik Gresch", "Better Autocomplete with Type Hints", "better_autoc
8      ("Simon Niederberger", r"Jupyter \& Git", "presentation.pdf"),
9      ("Vita Midori", "ML at Demokratis.ch", "Vita Midori - ML at Demokratis
10     ("Tim Head", "Zero Code Change Acceleration", 'Lightning - Zero Code Ch
11     # wait list (except last)
12     ("Josua Schmid", "Funny Slides for Data Scientist", "2024-10-18 Funny S
13     ("Ricardo Pereira", "Python Superset", "PythonSuperset.pdf"),
14     ("Stefan Keller", "Pythons and Ducks!", "Lightning_Talk_Stefan_Keller_I
15     ("Tamara Reuveni Mazig", "Topic Modelling for Customer experience", "Li
16     ("Jan Werth", r"AI \& our Brain $\rightarrow$ not quite there yet", Non
17     ("Florian Bruhin", "fstring.help", "fstring-help.pdf"),
18     ("Florian Bruhin", "A .py/.tex/.pdf quine", None),
19 ]
```

genslides.py

```
22  def main() -> None:
23      talks: list[tuple[str, str, str | None, str]] = [
24          (speaker, title, pdf, next_speaker)
25          for (speaker, title, pdf), (next_speaker, _, _)
26          in zip(TALKS, TALKS[1:] + [("", "", None)])
27      ]
```

genslides.py

```
29      env = Environment(
30          variable_start_string="((",
31          variable_end_string="))",
32          block_start_string="((*",
33          block_end_string="*))",
34          comment_start_string="((=",
35          comment_end_string="=))",
36          autoescape=False,
37      )
38      j2_src = Path("genslides.tex.j2").read_text()
39      template = env.from_string(j2_src)
40      tex_src = template.render(entries=talks)
```

genslides.py

```
42      tex_path = Path("talks.tex")
43      tex_path.write_text(tex_src)
44
45      subprocess.run(
46          ["latexmk", "-pdf", "-shell-escape", tex_path],
47          check=True,
48      )
49
50
51  if __name__ == "__main__":
52      main()
```

genslides.tex.j2

```
\documentclass[aspectratio=169]{beamer}
\usepackage{pdfpages}
\usepackage{cmbright}
\usepackage{minted}
\usepackage{hyperref}
\setbeamertemplate{navigation symbols}{}

\begin{document}
```

genslides.tex.j2

```
((* for speaker, title, pdf, next_speaker in entries *))
    \begin{frame}{Lightning Talk (( loop.index )) of
            (( entries | length ))}
        \begin{center}
            {\Huge (( speaker ))} \\[1em]
            {\LARGE (( title ))}
            ((* if next_speaker *))
                \vspace{1cm}
                {\Large \emph{Next: (( next_speaker ))}}
            ((* endif *))
        \end{center}
    \end{frame}
    ((* if pdf *))
        \includepdf[pages=-]{slides/(( pdf ))}
    ((* endif *))
((* endfor *))
```

genslides.tex.j2

```
26 \newcommand{\quine}[4][python]{%
27   \begin{frame}{#2}
28   \inputminted[firstline=#3,lastline=#4,linenos]{#1}{#2}
29   \end{frame}
30 }
31
32 \quine{genslides.py}{1}{3}     % imports
33 \quine{genslides.py}{6}{19}    % TALKS
34 \quine{genslides.py}{22}{27}   % setting talks=
35 \quine{genslides.py}{29}{40}   % jinja
36 \quine{genslides.py}{42}{52}   % build / main
37
38 \quine[tex]{genslides.tex.j2}{1}{8}     % preamble
39 \quine[tex]{genslides.tex.j2}{9}{24}    % slide
40 \quine[tex]{genslides.tex.j2}{26}{40}   % quine
```

CCC Hackerethik

Man kann mit einem Computer Kunst und Schönheit schaffen.
You can create art and beauty on a computer.

https://www.ccc.de/de/hackerethics / https://www.ccc.de/en/hackerethics

```
42 \begin{frame}{CCC Hackerethik}
43 Man kann mit einem Computer Kunst und
44 Schönheit schaffen. \\
45 You can create art and beauty on a computer.
46
47 {\scriptsize \url{https://www.ccc.de/de/hackerethics} /
48 \url{https://www.ccc.de/en/hackerethics}} \\[.2em]
49
50 \inputminted[firstline=42,linenos]{tex}{genslides.tex.j2}
51 \end{frame}
52 \end{document}
```