Bazy danych i Dashbordy

Paweł Gliwny

Czym jest baza danych?

- Ustrukturyzowany sposób przechowywania i dostępu do danych.
- Popularne typy: relacyjne (SQLite, PostgreSQL), NoSQL (MongoDB).
- Python umożliwia łączenie się z bazami i wykonywanie zapytań.

SQLite – lekka baza danych SQL

- Wbudowany silnik bazy danych bez potrzeby serwera.
- Dane przechowywane w jednym pliku `.db`.
- Idealna do małych aplikacji, testów i dashboardów.

Python i SQLite

- Połączenie z bazą
- Utworzenie kursora
- Wykonanie zapytań SQL
- Zatwierdzenie zmian i zamknięcie połączenia

```
import sqlite3
conn = sqlite3.connect("sales.db")
cursor = conn.cursor()
cursor.execute("SELECT * FROM sales")
conn.close()
```

Zobacz: bazy danych i dashboard

Odczyt danych z SQLite za pomocą `fetchall()`

- Połącz się z bazą danych
- Wykonaj zapytanie SQL
- Użyj **`fetchall()`** aby pobrać wszystkie rekordy
- Przejdź po wynikach w pętli i je wyświetł

Co zwraca **`fetchall()`**?

- Listę krotek (tuples), gdzie każda krotka to jeden rekord z tabeli.
- Umożliwia dalsze przetwarzanie danych w Pythonie.

```
import sqlite3

# Połączenie z istniejącą bazą danych
conn = sqlite3.connect("sales.db")
cursor = conn.cursor()

# Wykonanie zapytania
cursor.execute("SELECT * FROM sales")

# Pobranie i wyświetlenie wyników
rows = cursor.fetchall()
for row in rows:
    print(row)

conn.close()
```

Pandas i SQL – analiza danych

- `pandas.read_sql()` umożliwia integrację SQL z DataFrame'ami.
- Ułatwia analizę danych i tworzenie wykresów.

```
import pandas as pd
df = pd.read_sql("SELECT * FROM sales", conn)
```

Dobre praktyki pracy z bazami danych

- Zawsze zamykaj połączenia po zakończeniu pracy.
- Waliduj dane wejściowe przy dodawaniu do bazy.
- Do większych projektów używaj `SQLAlchemy` (ORM).
- Oddzielaj logikę (zapytania) od prezentacji (dashboardy).

Dashboard

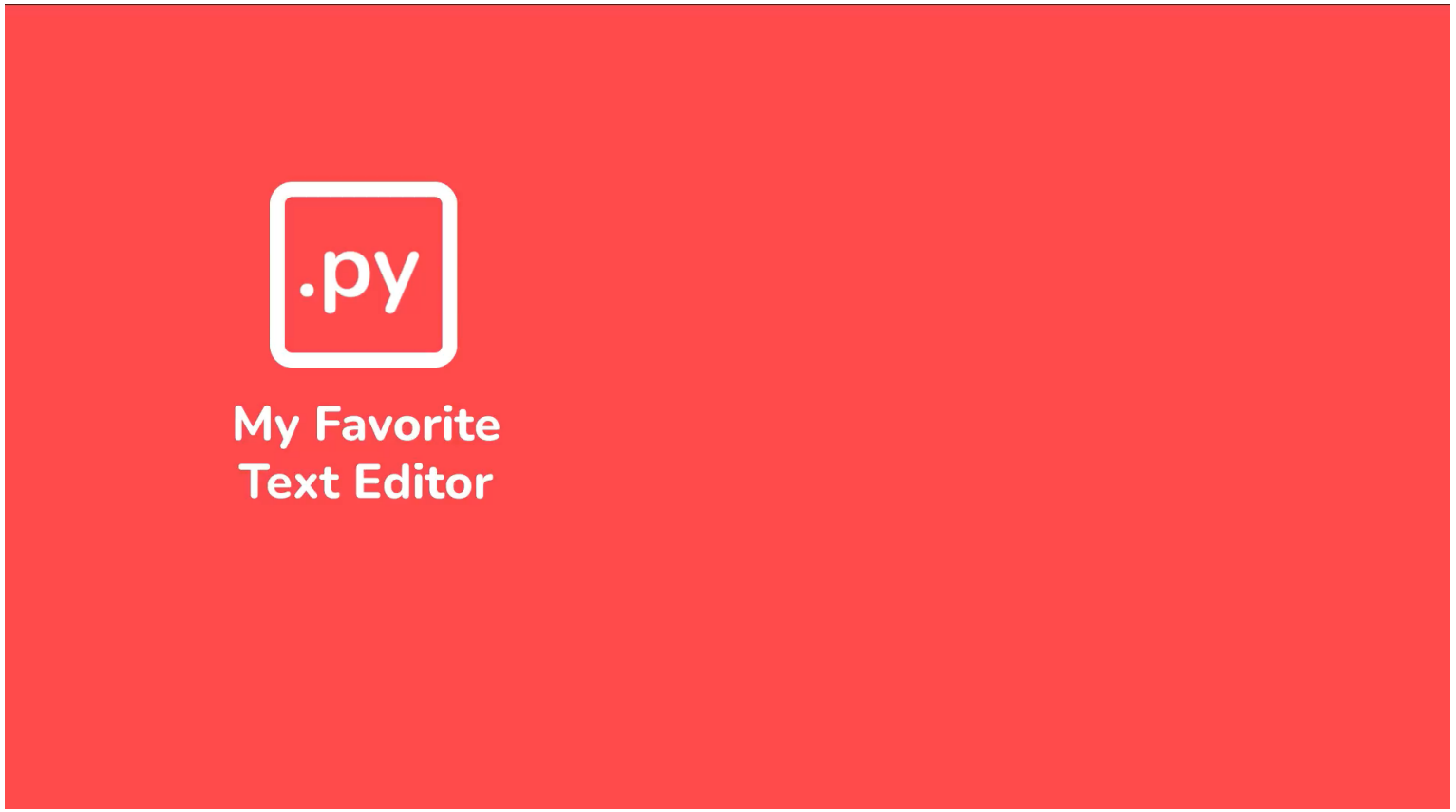
- **Dashboard** to interaktywna aplikacja lub strona internetowa, która wizualizuje dane w formie wykresów, tabel, wskaźników KPI, …
- Umożliwienie **szybkiego wglądu** w najważniejsze informacje, metryki lub wyniki analizy danych.
- Ułatwiają **monitorowanie** stanu systemów, modeli ML, …
- Łączą świat **analityki** i **biznesu** – przystępna forma prezentacji danych dla osób nietechnicznych.

Dashboardy i bazy danych?

- Dashboardy mogą pobierać dane **w czasie rzeczywistym** lub w określonych interwałach z różnych źródeł:
  - **Relacyjne bazy danych**: MySQL, PostgreSQL, SQLite.
  - **NoSQL**: MongoDB, Redis.
  - **APIs**: REST, GraphQL – pobieranie danych z zewnętrznych usług.
- W Pythonie często używa się bibliotek:
  - **SQLAlchemy**, **psycopg2**, **pymysql** – do komunikacji z bazami SQL.
  - **Pandas** – do obróbki danych po ich pobraniu.
- **Streamlit** i **Dash** mają wbudowane mechanizmy do pobierania i aktualizowania danych z baz w tle.

Streamlit

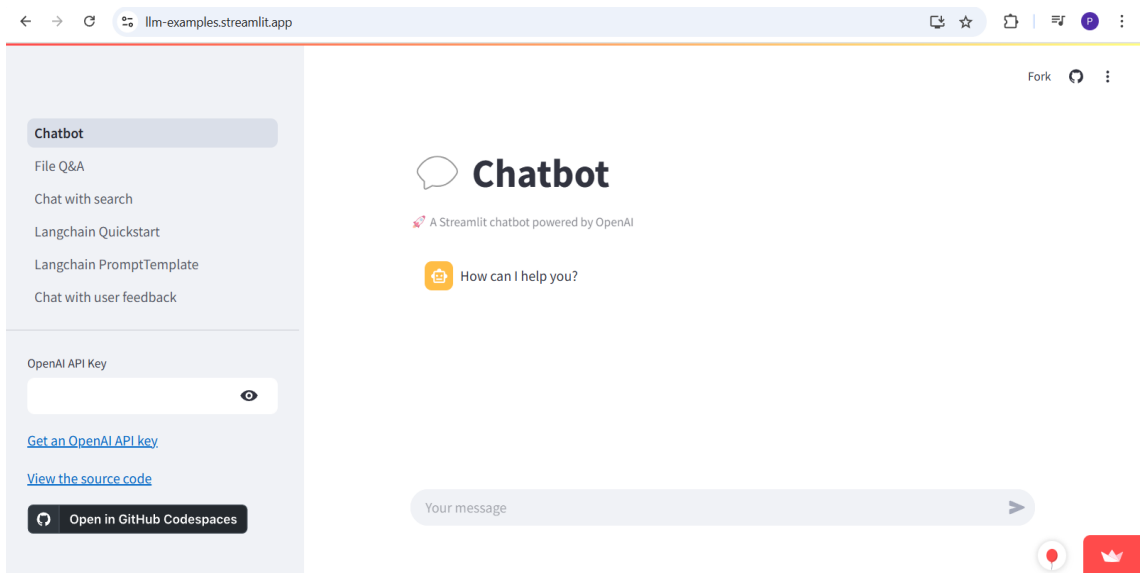
- **Streamlit** to framework open-source do budowania interaktywnych aplikacji webowych.
- Został zaprojektowany głównie dla **data scientistów**, **analityków danych** oraz **machine learning engineerów**

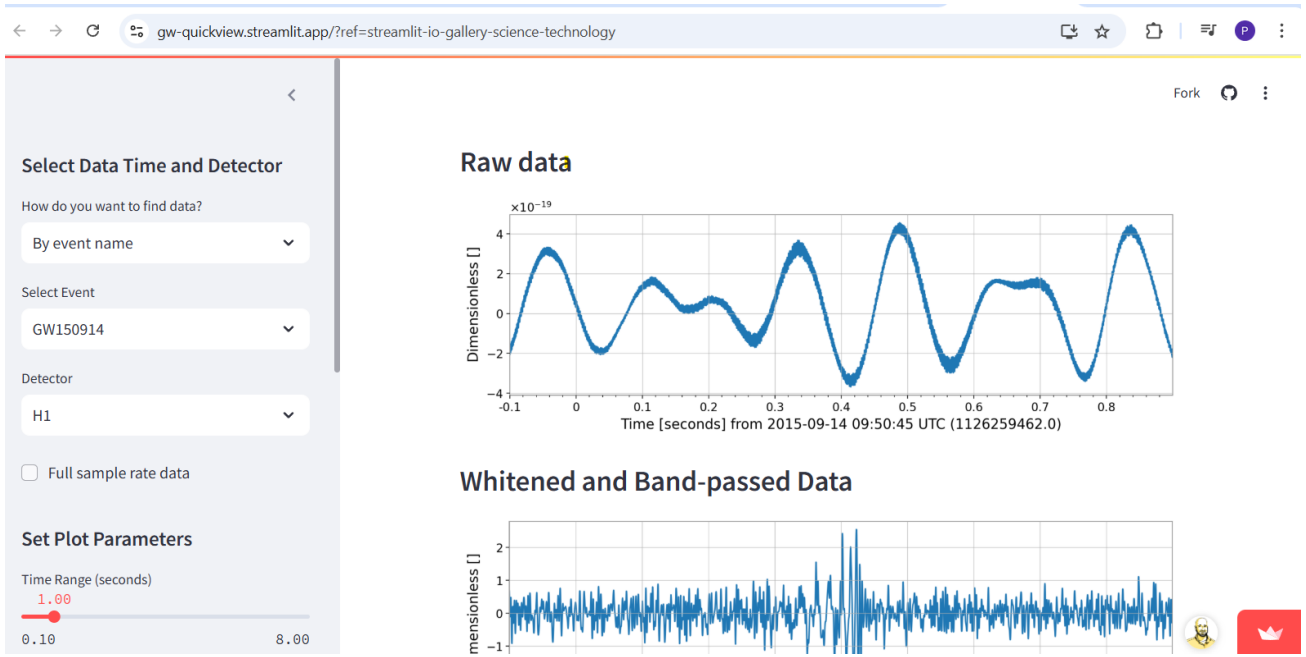


Przykłady

https://llm-examples.streamlit.app/

https://gw-quickview.streamlit.app/





Alternatywy dla **Streamlit**

- **Dash** (od Plotly)
  - Bardziej rozbudowany, wymaga większej znajomości frontendu (HTML, CSS).
  - Lepszy do **bardziej zaawansowanych**, skalowalnych aplikacji.
- **Gradio**
  - Podobny do Streamlit, ale bardziej skupiony na **interfejsach dla modeli ML**.
  - Umożliwia łatwe testowanie modeli machine learning.
- **Voila**
  - Przekształca Jupyter Notebook w interaktywną aplikację webową.
  - Działa na bazie widgetów z **ipywidgets**.
- **Panel** (od HoloViz)
  - Wspiera wiele bibliotek wizualizacyjnych (**Bokeh, Matplotlib, Plotly**).
  - Bardziej elastyczny niż Streamlit, ale mniej intuicyjny dla początkujących.

Dlaczego warto używać Streamlit?

- **Brak potrzeby znajomości front-endu** (HTML, CSS, JavaScript)
- **Szybkie wdrażanie aplikacji** z modelami ML i analizą danych
- **Intuicyjny i nowoczesny interfejs użytkownika**
- Tworzenie aplikacji w **minuty, nie tygodnie**
- Obsługa **większości popularnych bibliotek** Pythona:

  `pandas`, `matplotlib`, `seaborn`, `plotly`, `Keras`, `PyTorch`, `SymPy` i inne

Streamlit – plusy i zalety

- Idealne dla osób bez doświadczenia w tworzeniu stron www
- Umożliwia szybkie publikowanie:
  - modeli uczenia maszynowego,
  - wizualizacji danych,
  - interaktywnych narzędzi analitycznych
- **Mniej kodu → więcej efektu**
- Wbudowany **mechanizm cache'owania** przyspiesza działanie aplikacji
- Open-source i stale rozwijany przez społeczność

Jak zacząć

```
$ pip install streamlit
$ streamlit hello
```

Dokumentacja

Forum

Wyświetlanie tekstu w Streamlit

Funkcje do wyświetlania tekstu:

- `st.title()` – tytuł aplikacji
- `st.header()` – nagłówek sekcji
- `st.markdown()` – tekst formatowany w stylu Markdown
- `st.latex()` – wzory matematyczne (LaTeX)
- `st.write()` – uniwersalna funkcja: tekst, dane, wykresy, modele, itd.
- ...

Wyświetlanie multimediów w Streamlit

- `Streamlit` umożliwia bardzo łatwe osadzanie obrazów, dźwięków i filmów w aplikacji – bez potrzeby używania HTML ani dodatkowych bibliotek front-endowych.

Funkcje do wyświetlania multimediów:

- `st.image()` – wyświetla obraz (np. JPG, PNG)
- `st.audio()` – odtwarza plik audio (np. MP3, WAV)
- `st.video()` – odtwarza plik wideo (np. MP4)

Interaktywne widżety wejściowe w Streamlit

- Widżety (widgets) to kluczowe elementy interfejsu użytkownika. Umożliwiają tworzenie **interaktywnych aplikacji** bez pisania kodu front-endowego.

Najczęściej używane widżety:

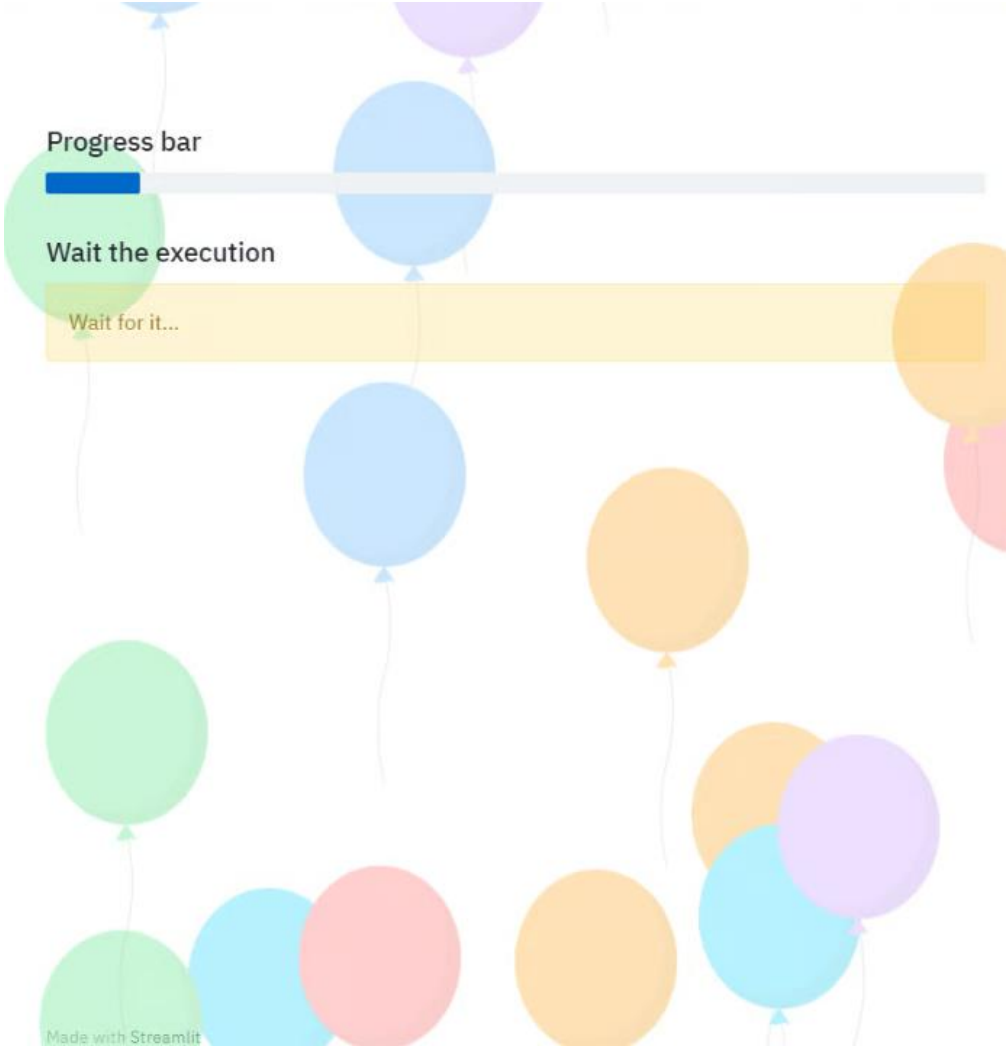
- `st.checkbox()` – zwraca wartość `True`/`False` w zależności od zaznaczenia
- `st.button()` – wyświetla przycisk
- `st.radio()` – przyciski jednokrotnego wyboru
- `st.slider()` – klasyczny suwak liczbow
- …

Wyświetlanie postępu i komunikatów statusu

- Streamlit pozwala w prosty sposób informować użytkownika o postępie działania aplikacji, jej stanie lub zakończeniu operacji.

Funkcje do wyświetlania postępu i statusu:

- `st.progress()` – wyświetla pasek postępu (od 0 do 100%)
- `st.spinner()` – pokazuje komunikat „czekaj...” podczas wykonywania operacji
- `st.balloons()` – animowane baloniki (np. po zakończeniu zadania jako forma gratulacji 🎈)



Wizualizacja danych w Streamlit

Dlaczego wizualizacja danych jest ważna?

- Ułatwia zrozumienie dużych zbiorów danych – zamiast surowych tabel, widzimy **wzorce, trendy i wyjątki**.
- Dobra wizualizacja **opowiada historię** – nie tylko "ładnie wygląda", ale **konkretnie informuje**.
- Skuteczna wizualizacja to **balans między formą a treścią** – powinna być czytelna i celna.
  - Nikt nie przeanalizuje miliona rekordów w tabeli okiem — wykresy są do tego niezbędne.

Wbudowane wykresy i mapy w Streamlit

- Streamlit oferuje szybkie funkcje do tworzenia prostych wykresów i map — bez potrzeby korzystania z zewnętrznych bibliotek wizualizacji.

Typy wykresów:

- `st.line_chart(data)` → Wykres liniowy – idealny do analizy trendów w czasie
- `st.bar_chart(data)` → Wykres słupkowy – do porównania wartości między kategoriami
- `st.area_chart(data)` → Wykres powierzchniowy – podobny do liniowego, ale z wypełnieniem

🌍 **Mapa:**

- `st.map(data)` → Wyświetla punkty na mapie świata (potrzebne kolumny `latitude` i `longitude`)

Przykładowa aplikacja

- Zobacz: github