Il Ciclo di Vita dei Dati

Dal Quiz alla Visualizzazione con Streamlit

💻 III Liceo Scientifico Biella - Scienze Applicate

Python Biella Group

📊 Oggi vedremo

1. Il ciclo di vita completo di un dato
2. Cos'è Streamlit e perché è utile
3. Comandi base di Streamlit
4. La nostra prima dashboard

🔄 Il Ciclo di Vita di un Dato

```
APPLICAZIONE → RACCOLTA → ELABORAZIONE → ANALISI → VISUALIZZAZIONE
Esempio:
Quiz           CSV        Polars         Insights  Streamlit
```

I nostri dati seguono questo percorso:

1. 📝 **Applicazione**: Il quiz legge le domande e genera le risposte
2. 💾 **Raccolta**: Salviamo le risposte in CSV (file di testo strutturato)
3. 🔧 **Elaborazione**: Polars legge e manipola i dati (domande e risposte)
4. 🔍 **Analisi**: Cerchiamo pattern e informazioni
5. 📊 **Visualizzazione**: Streamlit mostra i risultati

Esempio Pratico: Il Nostro Quiz

1. APPLICAZIONE (Quiz)

```
# Il quiz raccoglie dati mentre gli studenti rispondono
nome_utente = "alice"
id_domanda = 1
risposta_fornita = 2
tempo_risposta = 1200  # millisecondi
```

2. RACCOLTA (CSV)

```
nome_utente,id_domanda,numero_risposta_fornita,tempo_risposta
alice,1,2,1200
paola,1,1,1800
marco,1,2,1500
```

🔧 Dal CSV agli Insights

3. ELABORAZIONE (Analisi dei Dati)

Cosa significa elaborare i dati?

- 📖 **Leggere i file CSV**: Importare i dati nel programma
- 🔗 **Combinare informazioni**: Unire dati da file diversi (es. risposte + soluzioni)
- 🔢 **Calcolare metriche**: Contare, sommare, fare medie, percentuali
- 🎯 **Filtrare**: Selezionare solo i dati che ci interessano
- 📊 **Raggruppare**: Organizzare i dati per categorie (es. per studente, per domanda)

> *Nel prossimo incontro vedremo come fare tutto questo con **Polars**!*

🔍 Analisi dei Dati

4. ANALISI (Cercare Insights)

Domande che possiamo farci sui dati del quiz:

- 👥 **Quanti** studenti hanno partecipato?
- ✅ **Qual è** la percentuale di risposte corrette?
- 🏆 **Chi** ha fatto il punteggio migliore?
- 📉 **Quali** domande sono risultate più difficili?
- ⏱️ **Quanto tempo** hanno impiegato in media?
- 🎯 **Ci sono** pattern interessanti? (es. chi va veloce sbaglia di più?)

Insight Esempio: "Il 65% delle risposte è corretto → la classe ha capito bene!"

📊 Visualizzazione: Il Problema

Senza visualizzazione:

```
alice ha risposto correttamente a 15 domande su 20
marco ha risposto correttamente a 12 domande su 20
paola ha risposto correttamente a 18 domande su 20
diana ha risposto correttamente a 8 domande su 20
emma ha risposto correttamente a 16 domande su 20
...
```

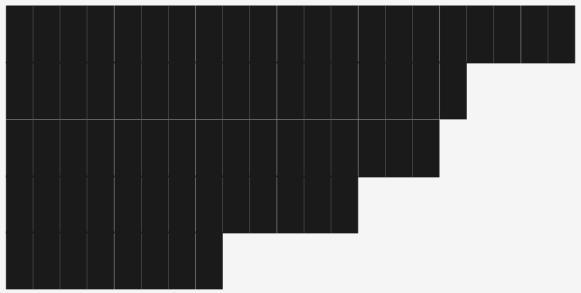
❌ Problemi:

- Difficile capire chi è il migliore
- Non si vedono i pattern
- Poco coinvolgente

Con Streamlit: Visualizzazione Immediata

Stesso dato, presentato meglio:

🏆 CLASSIFICA STUDENTI



Paola (18/20)
Emma (16/20)
Alice (15/20)
Marco (12/20)
Diana (8/20)

✅ Vantaggi:

- Immediato capire il ranking, si vedono le differenze
- Più coinvolgente e professionale

🚀 Cos'è Streamlit?

Definizione

Streamlit è una libreria Python che trasforma i tuoi script Python in applicazioni web interattive **senza scrivere HTML/CSS/JavaScript**.

Perché è utile?

- ✅ **Facile**: Solo Python, niente web development
- ✅ **Veloce**: Poche righe di codice → App completa
- ✅ **Interattiva**: Aggiornamenti automatici
- ✅ **Professionale**: Risultati che sembrano app vere

🌐 Classificazione di Streamlit

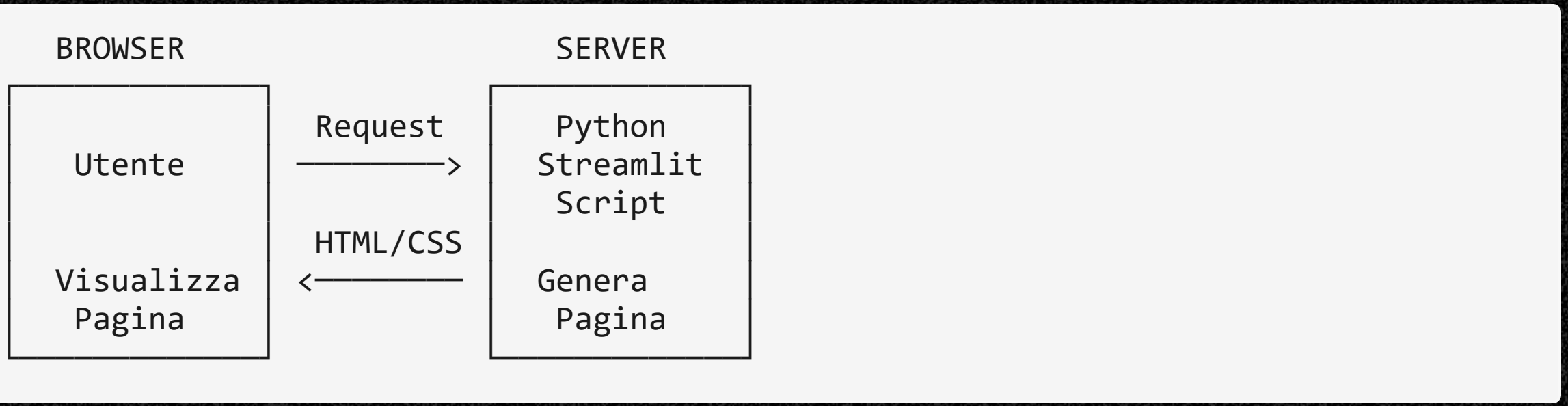
Ricordiamo: Pagine Web

Tipo	Caratteristiche	Esempi
Pagina Statica	HTML + CSS fisso, stesso contenuto per tutti	Sito vetrina, Blog
Applicazione Client-Side	JavaScript manipola il DOM nel browser	React, Vue, Angular
Applicazione Server-Side	Il server genera HTML dinamico	PHP, Django, Flask

🤔 Dove si colloca Streamlit?

🎯 Streamlit: app Web Server-Side

Come funziona Streamlit:



🎯 Streamlit: app Web Server-Side

Caratteristiche:

- ✅ **Server-Side**: Python gira sul server, non nel browser
- ✅ **Dinamico**: Ogni interazione → riesecuzione dello script → nuova pagina
- ✅ **Zero JavaScript**: Streamlit genera tutto (HTML/CSS/JS) automaticamente
- ✅ **Real-time**: Connessione WebSocket per aggiornamenti istantanei

🎨 Streamlit vs Script Normale

Script Python Tradizionale

```python
import polars as pl

df = pl.read_csv("dati.csv")
print(df)
print(f"Media: {df['voto'].mean()}")

#Output
```

nome	voto
Alice	8
Bob	7

```
Media: 7.5
```

🎨 Streamlit vs Script Normale (2)

Con Streamlit

```python
import streamlit as st
import polars as pl

st.title("📊 Analisi Voti")
df = pl.read_csv("dati.csv")

st.dataframe(df)  # Tabella interattiva!
st.metric("Media Classe", f"{df['voto'].mean():.1f}")
st.bar_chart(df, x="nome", y="voto")
```

Output: Una vera web app con tabelle interattive, grafici e metriche! 🎉

💻 Installazione

Nel terminale:

```
pip install streamlit
```

Verificare l'installazione:

```
streamlit --version
```

Eseguire un'app Streamlit:

```
streamlit run nome_file.py
# oppure
python -m streamlit run nome_file.py
```

📚 I Comandi Base di Streamlit

1. `st.title()` - Titolo Principale

```
import streamlit as st

st.title("🎮 Dashboard Quiz Python")
```

Quando usarlo: All'inizio dell'app per il titolo principale

📚 I Comandi Base (2)

2. `st.header()` - Intestazione Sezione

```
st.header("📊 Statistiche Generali")
```

Quando usarlo: Per dividere l'app in sezioni logiche

3. `st.write()` - Mostra Qualsiasi Cosa

```
st.write("Benvenuti nella dashboard!")
st.write(df)  # Mostra un dataframe
st.write(42)  # Mostra un numero
```

Quando usarlo: Versatile! Testo, dataframe, numeri, etc.

📚 I Comandi Base (3)

4. `st.metric()` - Numero Importante

```
st.metric(
    label="👥 Studenti Partecipanti",
    value=15
)
```

Output: Un riquadro grande con il numero in evidenza

Quando usarlo: Per KPI (Key Performance Indicators) - metriche chiave

📚 I Comandi Base (4)

5. `st.dataframe()` - Tabella Interattiva

```
st.dataframe(
    df,
    use_container_width=True,
    hide_index=True
)
```

Caratteristiche:

- ✅ Scrollabile, Ordinabile (click su colonna), Ridimensionabile

Quando usarlo: Per mostrare dati tabulari in modo professionale

📚 I Comandi Base (5)

6. `st.bar_chart()` - Grafico a Barre

```
st.bar_chart(
    df,
    x="nome_utente",
    y="risposte_corrette"
)
```

Quando usarlo: Per confronti visivi tra categorie

📚 I Comandi Base (6)

7. `st.columns()` - Dividere in Colonne

```
col1, col2, col3 = st.columns(3)

with col1:
    st.metric("Studenti", 15)
with col2:
    st.metric("Domande", 20)
with col3:
    st.metric("Risposte", 300)
```

Output: Tre metriche affiancate invece che una sotto l'altra

Quando usarlo: Per layout più compatti e professionali

🎯 Esempio Completo Minimo

File: `prima_app.py`

```
import streamlit as st
import polars as pl
st.title("🎮 La Mia Prima App")
st.write("Questa è la mia prima applicazione Streamlit!")
df = pl.DataFrame({
    "nome": ["Alice", "Bob", "Charlie"],
    "voto": [8, 7, 9]
})
st.dataframe(df)
st.bar_chart(df, x="nome", y="voto")
```

🎯 Esempio Completo Minimo (2)

Eseguire l'app:

```
streamlit run prima_app.py
```

Cosa succede:

1. Si apre automaticamente il browser
2. L'app è visibile all'indirizzo `http://localhost:8501`
3. Modifiche al codice → Refresh automatico!

🔄 Come Funziona Streamlit

Il Flusso di Esecuzione

```python
import streamlit as st

st.title("Contatore")
numero = 42
st.write(f"Il numero è: {numero}")
```

Caratteristica Importante:

Lo script viene rieseguito dall'inizio ogni volta che:

- Modifichi il codice, l'utente interagisce con l'app

⚠️ **Attenzione**: Non c'è "memoria" tra le esecuzioni (a differenza di un programma normale)

🎨 Best Practices per Layout

❌ Layout Brutto (Tutto in verticale)

```
st.metric("Studenti", 15)
st.metric("Domande", 20)
st.metric("Risposte", 300)
```

✅ Layout Professionale (Colonne)

```
col1, col2, col3 = st.columns(3)
with col1:
    st.metric("Studenti", 15)
with col2:
    st.metric("Domande", 20)
with col3:
    st.metric("Risposte", 300)
```

💡 Tips & Tricks

1. Usare Emoji per Rendere l'App Più Bella

```
st.title("🎮 Dashboard Quiz")
st.metric("👥 Studenti", 15)
st.header("📊 Statistiche")
```

2. Usare `width='stretch'`

```
st.dataframe(df, width='stretch')  # ✅ Si adatta allo schermo
```

3. Nascondere l'Indice nelle Tabelle

```
st.dataframe(df, hide_index=True)  # ✅ Più pulito
```

🔍 Dal Dato all'insight: Esempio Pratico

Scenario: Vogliamo sapere qual è la domanda più difficile

1. DATI (CSV)

```
nome_utente,id_domanda,numero_risposta_fornita
alice,1,2
paola,1,1
marco,1,2
```

Dal Dato all'insight: Esempio Pratico

2. ELABORAZIONE (Analisi)

Cosa dobbiamo fare per rispondere alla domanda:

1. 📖 Leggere i dati delle risposte date dagli studenti e le corrette
2. 🔗 Confrontare le risposte date con quelle corrette
3. 📊 Raggruppare per domanda
4. 🔢 Calcolare per ogni domanda: quanti hanno risposto corretto?
5. 📈 Calcolare la percentuale di successo
6. 📉 Ordinare dalla più difficile (% più bassa) alla più facile

> *Con **Polars** tutto questo si fa in poche righe! Lo vedremo prossimamente.*

🎯 Struttura di una Dashboard Tipica

```python
import streamlit as st
import polars as pl
# 1. CONFIGURAZIONE
st.set_page_config(page_title="Quiz Dashboard", page_icon="🎮")
# 2. TITOLO
st.title("🎮 Dashboard Risultati Quiz")
# 3. CARICAMENTO DATI
df = pl.read_csv("risposte_tutti.csv")
# 4. SEZIONI CON HEADER
st.header("📊 Statistiche Generali")
# ... metriche ...
st.header("🏆 Classifica")
# ... tabella e grafico ...
st.header("📈 Analisi Difficoltà")
# ... grafico domande difficili ...
```

📝 Ricapitolando

>>>>>> Il ciclo completo dei dati:

```
Quiz → CSV → Polars → Insights → Streamlit
```

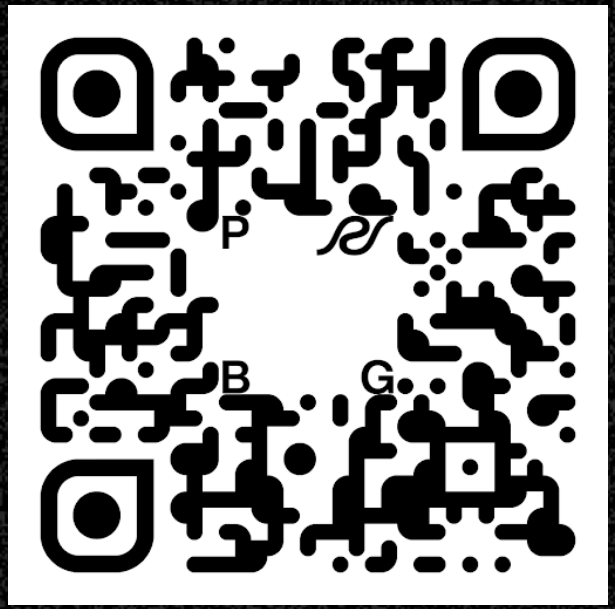
Comandi Streamlit Base:

1. `st.title()` - Titolo
2. `st.header()` - Sezione
3. `st.write()` - Mostra qualsiasi cosa
4. `st.metric()` - Numero importante
5. `st.dataframe()` - Tabella
6. `st.bar_chart()` - Grafico
7. `st.columns()` - Layout a colonne

📚 Risorse Utili

- **Documentazione ufficiale**: https://docs.streamlit.io
- **Gallery (esempi)**: https://streamlit.io/gallery
- **Cheat Sheet**: https://cheat-sheet.streamlit.app

Grazie per l'attenzione...



> *"C'è sempre qualcosa da imparare per migliorarci e crescere...**insieme!**"*