SVILUPPO APPLICAZIONI E ANALISI DATI CON PYTHON

Jupyter Notebook

Sviluppo Python Interattivo e Data Science

Docente: Marino Speciale

Indice della Lezione

- Cos'è Jupyter Notebook?
- Installazione e Setup
- Interfaccia e Celle
- Code Cells e Markdown Cells
- Librerie Data Science (NumPy, Pandas, Matplotlib)
- Visualizzazioni Grafiche
- Best Practices e Sharing

Cos'è Jupyter Notebook?

Jupyter è un ambiente web interattivo che permette di:

- Eseguire codice Python **cella per cella**
- Visualizzare output, grafici e tabelle direttamente
- Alternare codice e documentazione (Markdown)
- Prototipare e esplorare dati dinamicamente
- Salvare come file (JSON)

Vantaggi e Use Cases

Data Science

Analisi esplorativa, visualizzazioni, statistiche

Machine Learning

Prototipazione modelli, training, evaluation

Didattica

Esercitazioni interattive, spiegazioni paso-passo

Documentazione

Analisi con reportistica integrata e condivisibile

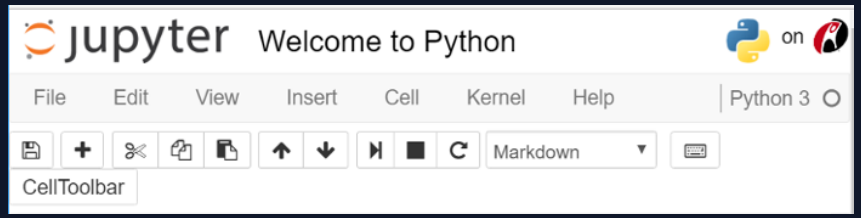
Installazione: Setup base

```
# Metodo 1: Tramite pip (consigliato)
pip install jupyter

# Metodo 2: Anaconda (include tutto)
# Scarica da: https://www.anaconda.com/

# Avviare Jupyter Notebook
jupyter notebook

# Avviare Jupyter Lab (versione moderna)
pip install jupyterlab
jupyter lab

# In alternativa (dev mode)
python -m jupyter notebook
```

Interfaccia Principale

Elementi principali:

- **Menu Bar:** File, Edit, View, Insert, Cell, Kernel, Help
- **Toolbar:** Pulsanti rapidi (Save, Add Cell, Run, etc.)
- **Celle:** Unità di esecuzione e documentazione
- **Output:** Risultati visibili sotto ogni cella
- **Kernel:** Processo Python che esegue il codice



Code Cell: Esecuzione codice

```
# In una Code Cell puoi scrivere Python normalmente
import numpy as np

# Calcoli
x = np.array([1, 2, 3, 4, 5])
media = x.mean()
print(f"Media: {media}")  # Output: Media: 3.0

# L'output è visibile sotto
# Puoi modificare e rieseguire la cella (Ctrl+Enter)
# Variabili rimangono in memoria per celle successive
```

Esegui con o

Markdown Cell: Documentazione

```
# Markdown Cell (seleziona "Markdown" dal menu)
## Titolo Sezione

Testo normale con **grassetto** e *corsivo*.

### Sottosezione
- Punto 1
- Punto 2

[Link a Google](https://google.com)

$$\frac{1}{n}\sum_{i=1}^{n} x_i$$  # Formula LaTeX
```

Perfetto per spiegazioni, formule, struttura del notebook

Scorciatoie Tastiera essenziali

- Esegui cella attuale
- Esegui e vai a prossima cella
- Esegui e inserisci cella sotto
- Inserisci cella sopra (in edit mode)
- Inserisci cella sotto
- Elimina cella
- Converti a Markdown
- Converti a Code

Magic Commands: Comandi speciali

```
# Magic commands (solo in Jupyter)

%timeit sum(range(1000))  # Cronometra esecuzione
# Output: 100000 loops, best of 5: 12.3 µs per loop

%run script.py  # Esegue un file esterno

%matplotlib inline  # Mostra grafici sotto le celle

%whos  # Lista tutte le variabili in memoria

!pip install pandas  # Esegui comando shell (!)

%%time  # Cronometra intera cella
```

NumPy: Array numerici

```
import numpy as np

# Creare array
arr = np.array([1, 2, 3, 4, 5])
matrix = np.zeros((3, 3))      # Matrice di zeri
range_arr = np.arange(0, 10, 2) # [0, 2, 4, 6, 8]

# Operazioni
print(arr.mean())     # Media
print(arr.std())      # Deviazione standard
print(arr.sum())      # Somma
print(arr.max())      # Massimo

# Broadcasting
arr2 = arr * 2        # Moltiplica ogni elemento per 2
```

Pandas: Gestione dati tabulari

```
import pandas as pd

# Creare DataFrame
data = {
    "Nome": ["Alice", "Bob", "Charlie"],
    "Età": [30, 25, 35],
    "Città": ["Roma", "Milano", "Napoli"]
}
df = pd.DataFrame(data)

# Accesso ai dati
print(df["Nome"])       # Colonna
print(df.iloc[0])       # Prima riga
print(df[df["Età"] > 25]) # Filtro

# Statistiche
print(df.describe())    # Riepilogo statistico
print(df.groupby("Città").size())  # Raggruppamento
```

Matplotlib: Visualizzazioni (Part 1)

```
import matplotlib.pyplot as plt

%matplotlib inline  # Mostra grafici inline

# Grafico a linea
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
plt.plot(x, y)
plt.title("Grafico Lineare")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()

# Grafico a barre
categorie = ["A", "B", "C"]
valori = [10, 25, 15]
plt.bar(categorie, valori)
plt.show()
```

Matplotlib: Visualizzazioni (Part 2)

```
# Scatter plot (dispersione)
import numpy as np
x = np.random.rand(50)
y = np.random.rand(50)
plt.scatter(x, y)
plt.show()

# Istogramma
data = [1, 1, 2, 2, 2, 3, 3, 3, 3, 4, 5, 5]
plt.hist(data, bins=5, edgecolor="black")
plt.show()

# Più grafici nella stessa figura
fig, (ax1, ax2) = plt.subplots(1, 2)
ax1.plot([1, 2, 3], [1, 4, 9])
ax2.bar(["A", "B"], [10, 20])
plt.show()
```

Progetto: Analisi Dataset (Setup)

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Carica dataset (CSV da file o URL)
df = pd.read_csv("vendite.csv")

# Esplorare il dataset
print(f"Shape: {df.shape}")  # Righe e colonne
print(df.head())             # Prime 5 righe
print(df.info())             # Tipi di dato
print(df.describe())         # Statistiche

# Controllare valori mancanti
print(df.isnull().sum())

# Pulire dati (dropna, fillna, etc.)
df = df.dropna()
```

Progetto: Analisi Dataset (Analisi)

```
# Analisi esplicative

# Vendite totali per categoria
vendite_cat = df.groupby("Categoria")["Vendite"].sum()
print(vendite_cat)

# Media vendite per prodotto
media_vendite = df.groupby("Prodotto")["Vendite"].mean()

# Correlazione tra variabili numeriche
print(df.corr())

# Filtri e aggregazioni
df_filtro = df[df["Vendite"] > 1000]
print(f"Vendite > 1000: {len(df_filtro)} record")
```

Progetto: Visualizzazioni

```
# Grafici per la presentazione

fig, axes = plt.subplots(2, 2, figsize=(12, 8))

# 1. Vendite per categoria (barre)
vendite_cat.plot(kind="bar", ax=axes[0, 0])
axes[0, 0].set_title("Vendite per Categoria")

# 2. Distribuzione prezzi (istogramma)
df["Prezzo"].hist(ax=axes[0, 1], bins=20)
axes[0, 1].set_title("Distribuzione Prezzi")

# 3. Tendenza temporale (linea)
df.groupby("Data")["Vendite"].sum().plot(ax=axes[1, 0])
axes[1, 0].set_title("Vendite nel Tempo")

# 4. Scatter (Prezzo vs Quantità)
axes[1, 1].scatter(df["Prezzo"], df["Quantità"])
axes[1, 1].set_title("Prezzo vs Quantità")

plt.tight_layout()
plt.show()
```

Condivisione e Esportazione

```
# Esportare il notebook

# Come HTML (condivisibile)
# Menu → File → Download as → HTML

# Come PDF (per report)
# Menu → File → Download as → PDF

# Come Python Script (.py)
# Menu → File → Download as → Python

# Come Markdown
# Menu → File → Download as → Markdown

# Salvare come immagine (PNG)
plt.savefig("grafico.png", dpi=300, bbox_inches="tight")
```

Kernel e Ambienti virtuali

```
# Jupyter può usare diversi kernel (non solo Python)

# Selezionare un environment specifico
# 1. Crea virtual environment
python -m venv data_env

# 2. Attiva e installa jupyter
source data_env/bin/activate
pip install jupyter numpy pandas

# 3. Esegui jupyter da dentro
jupyter notebook

# Restart Kernel durante sviluppo
# Menu → Kernel → Restart Kernel (per liberare memoria)

# Interrupt Kernel se una cella è bloccata
# Menu → Kernel → Interrupt
```

Debugging e Troubleshooting

```
# Debug in Jupyter

# %debug dopo un errore
try:
    x = 1 / 0
except:
    %debug  # Accedi al prompt interattivo

# Stampe intermediate (print debugging)
for i in range(10):
    print(f"Iterazione {i}")

# Variabili locali di una funzione
def my_func(x):
    y = x * 2
    print(locals())  # Vedi variabili locali

# Misura performance
%timeit [i**2 for i in range(1000)]
```

Best practices con Jupyter

- Usa **Markdown cells** per spiegare la logica
- Mantieni celle **piccole e focalizzate** (una idea per cella)
- Salva spesso con
- Esegui tutto top-down: non dipendere dall'ordine non sequenziale
- Pulisci il kernel periodicamente ()
- Documenta input e output attesi
- Usa version control (Git) per file

Estensioni e Tool utili

```
# Estensioni per Jupyter Notebook

# Jupyter Nbextensions
pip install jupyter_contrib_nbextensions

# Spellchecker, TOC, Variable Inspector
# Menu → Nbextensions per attivarli

# Jupyter Lab Extensions
pip install jupyterlab-toc

# Strumenti di visualizzazione avanzata
pip install plotly seaborn bokeh

# Per condivisione online
pip install nbconvert
pip install voila  # Trasforma notebook in app web
```

Riepilogo Lezione

✓ Jupyter: ambiente interattivo browser-based

✓ Code + Markdown cells per codice e documentazione

✓ NumPy, Pandas, Matplotlib: stack data science

✓ Perfetto per prototipazione, analisi, didattica

Domande?

Approfondimenti:

→ Machine Learning con Scikit-Learn

→ Deep Learning (TensorFlow, PyTorch)

→ Web Scraping e API REST