Miniguida Python per Studenti

Nome dell'Insegnante

August 2, 2024

Contents

1 Introduzione

Python è un linguaggio di programmazione versatile e potente, ideale per iniziare a programmare e per l'analisi dati. In questa guida, esploreremo la sintassi base, l'ambiente di sviluppo Google Colab, e alcuni elementi chiave di Python, NumPy e Matplotlib.

2 Sintassi Base e Ambiente di Sviluppo Google Colab

Google Colab è un ambiente di sviluppo integrato (IDE) basato su Jupyter Notebook, che permette di eseguire codice Python direttamente nel browser.

2.1 Accedere a Google Colab

1. Apri il browser e vai su `https://colab.research.google.com/`. 2. Accedi con il tuo account Google. 3. Crea un nuovo notebook cliccando su *New Notebook*.

2.2 Scrivere ed Eseguire Codice

Nel notebook, puoi scrivere ed eseguire codice Python in celle. Per eseguire una cella, premi *Shift+Enter*.

Esempio:

```
print("Hello, World!")
```

3 Elementi del Linguaggio Python

3.1 Variabili e Oggetti

Le variabili in Python non richiedono dichiarazioni esplicite del tipo. Esempio:

```
x = 10
y = "Ciao"
z = 3.14
```

3.2 Stringhe

Le stringhe sono delimitate da virgolette singole o doppie. Esempio:

```
stringa = "Questa è una stringa"
```

3.2.1 F-String

Le f-string sono una funzionalità di Python che permette di inserire variabili all'interno delle stringhe in modo semplice e leggibile. Si utilizzano anteponendo una 'f' alla stringa e racchiudendo le variabili tra parentesi graffe ".

Esempio:

```
nome = "Alice"
eta = 25
stringa = f"Il␣nome␣è␣{nome}␣e␣l'età␣è␣{eta}␣anni."
print(stringa)
```

In questo esempio, le variabili 'nome' e 'eta' sono incluse direttamente all'interno della stringa senza la necessità di concatenare manualmente o utilizzare metodi di formattazione complessi.

Esempio con espressioni:

```
a = 10
b = 5
stringa = f"La␣somma␣di␣{a}␣e␣{b}␣è␣{a␣+␣b}."
print(stringa)
```

In questo caso, anche le espressioni possono essere valutate direttamente all'interno delle parentesi graffe.

3.3 Costrutto di Selezione

Il costrutto di selezione in Python usa le parole chiave *if*, *elif* e *else*. Esempio:

```
x = 5
if x > 0:
    print("x␣è␣positivo")
elif x == 0:
    print("x␣è␣zero")
else:
    print("x␣è␣negativo")
```

3.4 Controllo di Flusso

I controlli di flusso includono strutture come cicli e dichiarazioni condizionali.

3.4.1 Indentazione

Python utilizza l'indentazione per definire blocchi di codice. Ogni blocco deve essere indentato con lo stesso numero di spazi o tabulazioni. Esempio:

```
x = 10
if x > 0:
    print("x␣è␣positivo")
    if x > 5:
        print("x␣è␣maggiore␣di␣5")
```

3.4.2 Cicli *for*

Il ciclo *for* in Python è estremamente versatile e può essere utilizzato per iterare su una varietà di sequenze e strutture dati.

Iterare su una Lista Il ciclo *for* può iterare attraverso ogni elemento di una lista. Esempio:

```
frutti = ['mela', 'banana', 'ciliegia']
for frutto in frutti:
    print(frutto)
```

Iterare su un Range di Numeri Il ciclo *for* può iterare attraverso un range di numeri generato dalla funzione *range*. Esempio:

```
for i in range(5):
    print(i)
```

Iterare su un Insieme Gli insiemi (*sets*) sono collezioni non ordinate di elementi unici. Esempio:

```
insieme = {1, 2, 3, 4}
for numero in insieme:
    print(numero)
```

Iterare su un Dizionario I dizionari (*dict*) sono collezioni di coppie chiave-valore. Esempio di iterazione su chiavi e valori:

```
dizionario = {'nome': 'Alice', 'età': 25, 'città': 'Roma'}

# Iterare sulle chiavi
for chiave in dizionario:
    print(chiave, dizionario[chiave])

# Iterare su chiavi e valori
for chiave, valore in dizionario.items():
    print(chiave, valore)
```

Enumerare gli Elementi di una Sequenza La funzione *enumerate* restituisce una coppia (indice, valore) per ogni elemento di una sequenza. Esempio:

```
frutti = ['mela', 'banana', 'ciliegia']
for indice, frutto in enumerate(frutti):
    print(indice, frutto)
```

Uso di *zip* per Iterare su Più Sequenze La funzione *zip* può essere utilizzata per iterare su più sequenze contemporaneamente. Esempio:

```
nomi = ['Alice', 'Bob', 'Charlie']
eta = [25, 30, 35]

for nome, eta_persona in zip(nomi, eta):
    print(nome, eta_persona)
```

3.4.3 Cicli *while*

Il ciclo *while* itera finché una condizione è vera. È utile quando non si conosce in anticipo il numero di iterazioni. Ecco alcuni esempi:

Esempio Base Un ciclo ‘while‘ di base per contare da 0 a 4:

```
count = 0
while count < 5:
    print(count)
    count += 1
```

Esempio con Lista In questo esempio, gli elementi della lista vengono rimossi uno alla volta fino a che la lista è vuota:

```
lista = [1, 2, 3, 4, 5]
while lista:
    elemento = lista.pop(0)
    print(f"Eliminato: {elemento}, Lista rimanente: {lista}")
```

4 Elementi di NumPy

4.1 Array

NumPy è una libreria fondamentale per il calcolo scientifico in Python. Gli array NumPy sono simili alle liste, ma permettono operazioni matematiche vettorizate. Esempio:

```
import numpy as np
arr = np.array([1, 2, 3, 4])
print(arr)
```

4.2 Generazione di Array

NumPy fornisce diverse funzioni per creare array. Esempio:

```
zeros = np.zeros(5)
ones = np.ones(5)
range_arr = np.arange(10)
```

4.3 Indicizzazione degli Array

Gli elementi di un array possono essere accessi e modificati utilizzando gli indici. Esempio:

```
arr = np.array([1, 2, 3, 4, 5])
print(arr[0])  # Primo elemento
print(arr[-1])  # Ultimo elemento
arr[0] = 10  # Modifica il primo elemento
```

4.4 Matematica con gli Array

NumPy permette di eseguire operazioni matematiche sugli array in modo semplice. Esempio:

```
arr = np.array([1, 2, 3, 4])
print(arr + 2)
print(arr * 2)
print(np.sqrt(arr))
```

4.5 Principali Funzioni Matematiche di NumPy

NumPy offre molte funzioni matematiche utili per operare sugli array. Ecco alcune delle principali:

4.5.1 Funzioni Trigonometriche

```
import numpy as np
angles = np.array([0, np.pi/2, np.pi])
print(np.sin(angles))  # Calcola il seno
print(np.cos(angles))  # Calcola il coseno
print(np.tan(angles))  # Calcola la tangente
```

4.5.2 Funzioni Esponenziali e Logaritmiche

```
values = np.array([1, 2, 3])
print(np.exp(values))  # Calcola l'esponenziale
print(np.log(values))  # Calcola il logaritmo naturale
print(np.log10(values))  # Calcola il logaritmo in base 10
```

4.5.3 Statistiche di Base

```
data = np.array([1, 2, 3, 4, 5])
print(np.mean(data))  # Media
print(np.median(data))  # Mediana
print(np.std(data))  # Deviazione standard
```

4.5.4 Altre Funzioni Utili

```
values = np.array([-1, 2, -3])
print(np.abs(values))  # Valore assoluto
print(np.sqrt(values + 4))  # Radice quadrata (aggiungiamo 4 per
    evitare valori negativi)
print(np.sum(values))  # Somma degli elementi
print(np.prod(values))  # Prodotto degli elementi
```

5 Elementi di Matplotlib

5.1 Grafici a Dispersione

Matplotlib è una libreria per la creazione di grafici in Python. Esempio di grafico a dispersione:

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

x = np.random.rand(50)
y = np.random.rand(50)

plt.scatter(x, y)
plt.title("Grafico␣a␣Dispersione")
plt.xlabel("X")
plt.ylabel("Y")
plt.savefig('grafico_dispersione.png')
plt.show()
```

Includi il grafico nel documento:

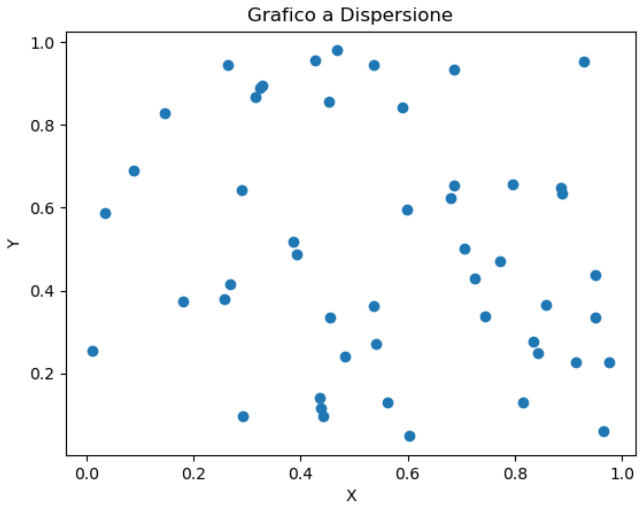

Figure 1: Grafico a Dispersione

5.2 Grafici con Barre di Errore

I grafici con barre di errore mostrano l'incertezza nei dati. Esempio:

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

x = np.linspace(0, 10, 10)
y = np.sin(x)
yerr = 0.2

plt.errorbar(x, y, yerr=yerr, fmt='-o')
plt.title("Grafico con Barre di Errore")
plt.xlabel("X")
plt.ylabel("Y")
plt.savefig('grafico_barre_errore.png')
plt.show()
```

Includi il grafico nel documento:

5.3 Grafico Spazio-Tempo e Velocità-Tempo per Moto Uniforme a Tratti

Consideriamo un moto uniforme a tratti, dove un oggetto si muove a tre diverse velocità in tre intervalli di tempo distinti. Creiamo due grafici: uno per il moto spaziale-temporale e uno per la velocità-tempo.

5.3.1 Grafico Spazio-Tempo

Nel grafico spazio-tempo, mostriamo il percorso dell'oggetto in funzione del tempo. Supponiamo che l'oggetto abbia velocità costanti di 2 m/s, 4 m/s e 6 m/s nei rispettivi intervalli di tempo.

Esempio di codice:

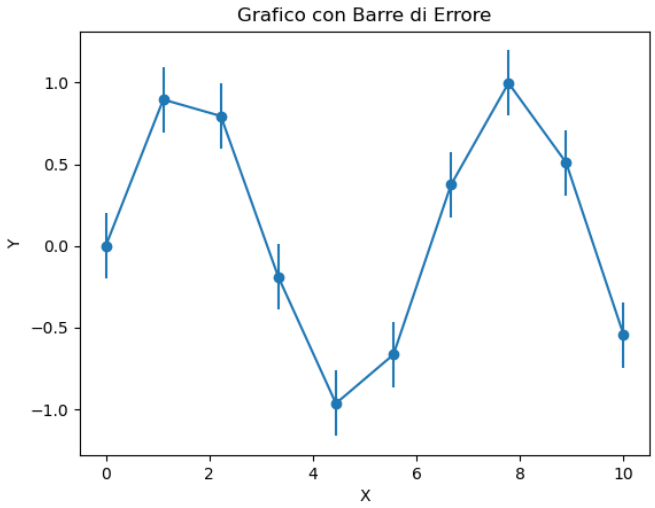

Figure 2: Grafico con Barre di Errore

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

# Dati
tempi = [0, 2, 5, 8, 10]
posizioni = [0, 4, 16, 28, 40]

# Creazione del grafico
plt.figure(figsize=(12, 6))
plt.plot(tempi, posizioni, marker='o')
plt.title("Grafico␣Spazio-Tempo␣per␣Moto␣Uniforme␣a␣Tratti")
plt.xlabel("Tempo␣(s)")
plt.ylabel("Posizione␣(m)")
plt.grid(True)
plt.savefig('grafico_spazio_temporale.png')
plt.show()
```

5.3.2 Grafico Velocità-Tempo

Il grafico velocità-tempo mostra la variazione della velocità in funzione del tempo. Ogni intervallo di tempo corrisponde a una velocità costante.

Esempio di codice:

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

# Dati
tempi = [0, 2, 5, 8, 10]
velocita = [2, 2, 4, 4, 6]

# Creazione del grafico
plt.figure(figsize=(12, 6))
plt.step(tempi, velocita, where='post', marker='o')
plt.title("Grafico␣Velocità-Tempo␣per␣Moto␣Uniforme␣a␣Tratti")
```

```
plt.xlabel("Tempo␣(s)")
plt.ylabel("Velocità␣(m/s)")
plt.grid(True)
plt.savefig('grafico_velocita_temporale.png')
plt.show()
```

5.3.3 Spiegazione dei Grafici

- **Grafico Spazio-Tempo:** Questo grafico mostra come la posizione dell'oggetto cambia nel tempo. È possibile osservare che il grafico è composto da segmenti lineari, ognuno con una pendenza diversa, che rappresenta le diverse velocità. L'area sotto la curva corrisponde alla distanza percorsa.
- **Grafico Velocità-Tempo:** Questo grafico mostra come la velocità cambia nel tempo. Utilizzando un grafico a passo, ogni intervallo di tempo mostra una velocità costante. Il grafico indica chiaramente le transizioni tra le diverse velocità.

Includi i grafici nel documento:

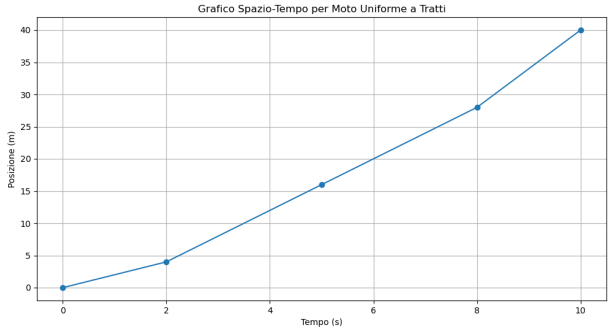

Figure 3: Grafico Spazio-Tempo per Moto Uniforme a Tratti

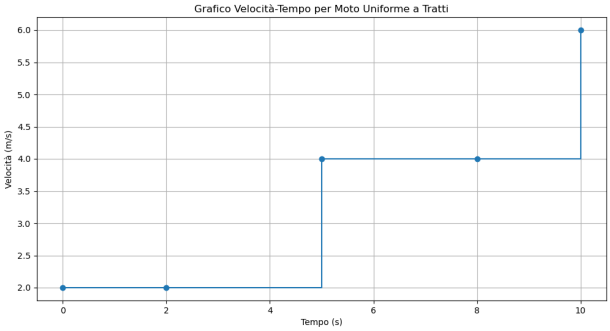

Figure 4: Grafico Velocità-Tempo per Moto Uniforme a Tratti

5.4 Grafico Altezza-Diametro di un Cilindro con Volume Fisso

Infine, consideriamo un grafico che mostra come l'altezza di un cilindro varia in funzione del diametro della base, mantenendo fisso il volume.

Esempio di codice:

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

# Costanti
volume = 1000  # Volume costante del cilindro in m³

# Diametro della base
diametro = np.linspace(1, 10, 100)  # Diametro varia da 1 a 10 metri

# Calcolo dell'altezza
altezza = volume / (np.pi * (diametro / 2)**2)

# Creazione del grafico
plt.figure(figsize=(12, 6))
plt.plot(diametro, altezza, label='Altezza = Volume / (π × (Diametro / 2)²)', color='green')
plt.title("Grafico dell'Altezza in Funzione del Diametro di Base (Volume Fisso)")
plt.xlabel("Diametro della Base (m)")
plt.ylabel("Altezza (m)")
plt.grid(True)
plt.legend()
plt.savefig('grafico_altezza_diametro.png')
plt.show()
```

Includi il grafico nel documento:

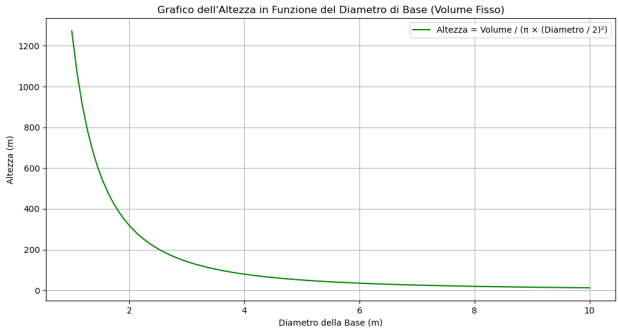

Figure 5: Grafico dell'Altezza in Funzione del Diametro di Base (Volume Fisso)