

Introduzione al coding

Dialoghi con un modello di linguaggio

Andrea Fazzi

10.02.2024



Indice

1	Introduzione	1
2	Lo script	1
3	Il linguaggio Python	2
4	Descrizione dello script	2
5	Variabili	3
5.1	Esercizio	3
6	Cicli	4
6.1	Esercizio	6

1 Introduzione

Attraverso il dialogo con un LLM (Large Language Model) si introdurranno i concetti fondamentali del coding a partire da un semplice script Python che implementa un cifrario di Cesare. Il modello di linguaggio utilizzato è **phind_v10_model**.

2 Lo script

Io: In che linguaggio è scritto il seguente programma?

```
1 def cesar_cipher(text, key):
2     result = ""
3
4     for i in range(len(text)):
5         char = text[i]
6
7         if char.isupper():
8             result += chr((ord(char) - 65 + key) % 26 + 65)
9         else:
10            result += chr((ord(char) - 97 + key) % 26 + 97)
11
12    return result
13
14 text = input("Inserisci il testo da cifrare: ")
15 key = int(input("Inserisci la chiave: "))
16
```

```
17 print("Testo cifrato: ", cesar_cipher(text, key))
```

llm: Il programma è scritto in Python.

3 Il linguaggio Python

lo: Parlami in maniera succinta del linguaggio di programmazione Python (massimo 100 parole).

llm: Python è un linguaggio di programmazione ad alto livello, interpretato e orientato agli oggetti. Creato da Guido van Rossum nel 1991, è noto per la sua sintassi leggibile e semplice, simile al linguaggio inglese, con influenze matematiche. È dinamicamente tipizzato e gestisce l'allocazione della memoria. Supporta vari stili di programmazione, tra cui procedurale, funzionale e orientato agli oggetti. Python è spesso utilizzato per lo sviluppo rapido di applicazioni, l'automazione, l'elaborazione dei dati e l'apprendimento automatico. Ha una vasta gamma di librerie standard e un'ampia comunità di sviluppatori.

4 Descrizione dello script

lo: Descrivimi lo script. Cosa fa?

llm: Lo script Python che hai fornito è un'implementazione di un cifrario di Cesare, un metodo di crittografia di sostituzione molto semplice. Il cifrario di Cesare sposta ogni lettera dell'alfabeto di un numero fissato di posizioni. Ad esempio, con una chiave di 1, 'A' diventa 'B', 'B' diventa 'C', ecc.

Ecco come funziona lo script:

- La funzione `cesar_cipher` prende due argomenti: `text`, la stringa da cifrare, e `key`, la chiave di cifratura.
- Per ogni carattere nel testo, controlla se è maiuscolo o minuscolo e calcola il nuovo carattere cifrato sommando la chiave alla posizione numerica ASCII del carattere originale, poi modulando per il numero di lettere nell'alfabeto (26).
- Restituisce la stringa cifrata.
- All'utente viene chiesto di inserire il testo da cifrare e la chiave.
- Infine, lo script stampa il testo cifrato.

5 Variabili

Io: Nello script sono utilizzati diversi concetti relativi al coding in generale. Sono presenti, ad esempio, alcune variabili alle quali vengono assegnati dei valori. A partire dallo script, spiegami in maniera succinta

1. Cosa sono le variabili
2. Come si assegna un valore ad una variabile
3. I diversi tipi di variabile utilizzati nello script

llm:

1. Le variabili sono locazioni di memoria con un nome e un valore associato. In Python, le variabili non devono essere dichiarate prima dell'uso; si creano semplicemente assegnando un valore ad un nome con l'operatore `=` [0][1].
2. Per assegnare un valore ad una variabile, si utilizza l'operatore di assegnazione `=`. Ad esempio, `result = ""` assegna una stringa vuota alla variabile `result`. In Python, è possibile assegnare valori di diversi tipi di dati a una variabile e il tipo di dato verrà determinato automaticamente [0][1].
3. Nel tuo script, ci sono vari tipi di variabili:
 - `text`: è una stringa, poiché contiene testo che l'utente inserirà tramite l'input.
 - `key`: è un intero, poiché l'input dell'utente viene convertito in un numero intero con `int()`.
 - `char`: è un carattere, poiché estrai singoli caratteri dalla stringa `text`.
 - `result`: è anch'essa una stringa, poiché accumula i caratteri cifrati [0][2].

5.1 Esercizio

Io: Proponimi un esercizio per mettere in pratica l'utilizzo delle variabili in Python. L'esercizio dev'essere coerente con quanto mi hai spiegato in precedenza. Insieme alla traccia dell'esercizio, genera una soluzione in Python.

llm:

Esercizio:

Scrivi uno script Python che calcoli l'area di un cerchio dato il raggio. Utilizza una variabile per memorizzare il raggio e un'altra per memorizzare l'area. Stampa l'area sul terminale.

Soluzione:

```
1 # Assegnazione del raggio alla variabile 'raggio'  
2
```

```
3 raggio = 5
4
5 # Calcolo dell'area del cerchio utilizzando la formula:
6 # area = pi * raggio^2
7 # Nota: In Python, 'pi' é rappresentato come math.pi
8
9 import math
10 area = math.pi * raggio ** 2
11
12 # Stampa dell'area sul terminale
13 print("L'area del cerchio con raggio", raggio, "é", area)
```

6 Cicli

Io: Nello script è presente un ciclo **for**. Parlami di questo costrutto e di come viene utilizzato nello script.

llm: Nello script Python fornito, il costrutto **for** viene utilizzato per iterare su una sequenza, in particolare una stringa. Questo costrutto è comune in Python e funziona come un iteratore, consentendo di eseguire un blocco di codice per ogni elemento in una sequenza, come una lista, una tupla, un dizionario, un insieme o una stringa [0][1][2][3].

Ecco come viene utilizzato nel tuo script:

- **for i in range(len(text))** : - Il ciclo **for** itera sulla lunghezza della stringa **text** utilizzando la funzione **range()**, che genera una sequenza di numeri. In questo caso, **i** assume il valore di ogni indice nella stringa **text** durante l'iterazione [2][3].
- All'interno del ciclo, **char = text[i]** assegna il carattere corrispondente all'indice **i** alla variabile **char**.
- Successivamente, il codice controlla se il carattere è maiuscolo o minuscolo e calcola il nuovo carattere cifrato basandosi sulla posizione ASCII del carattere originale e sulla chiave fornita dall'utente.
- Il risultato viene accumulato nella stringa **result**, che viene poi stampata alla fine dello script.

Quindi, il ciclo **for** in questo script è fondamentale per applicare la cifratura di Cesare a ciascun carattere del testo fornito dall'utente.

Io: Esistono altri modi per costruire in Python parti di codice iterative? Potresti prima elencarle e poi descrivermele succintamente?

llm: Esistono diversi modi per costruire parti di codice iterative in Python oltre al ciclo **for**. Ecco alcuni dei più comuni:

- Ciclo **while**: Si utilizza quando non si conosce in anticipo quante volte il ciclo deve essere eseguito, ma si sa solo quando deve terminare [1].
- List comprehension: Una forma concisa per creare liste basate su espressioni e cicli for [1].
- Funzioni ricorsive: Chiamano se stesse per risolvere problemi che possono essere scomposti in sottoproblemi più piccoli [1].
- Generatori: Fanno parte delle iterazioni, ma sono più leggeri perché non memorizzano tutti gli elementi in memoria, ma li producono su richiesta [1].

Io: Potresti riscrivere lo script utilizzando un ciclo **while** al posto del **for**?

llm: Per riscrivere lo script utilizzando un ciclo **while** invece del **for**, puoi seguire questi passaggi:

1. Inizializza una variabile, ad esempio **i**, con valore 0.
2. Imposta una condizione per il ciclo **while** che controlla se **i** è minore della lunghezza della stringa **text**.
3. All'interno del ciclo **while**, esegui il codice che era precedentemente all'interno del ciclo **for**, assicurandoti di aggiornare il valore di **i** alla fine di ogni iterazione.
4. Alla fine del ciclo, assicurati che **i** sia incrementato di 1.

Ecco come potrebbe apparire lo script modificato:

```
1 def cesar_cipher(text, key):
2     result = ""
3     i = 0
4
5     while i < len(text):
6         char = text[i]
7
8         if char.isupper():
9             result += chr((ord(char) - 65 + key) % 26 + 65)
10        else:
11            result += chr((ord(char) - 97 + key) % 26 + 97)
12
13        i += 1
14
15    return result
16
17 text = input("Inserisci il testo da cifrare: ")
18 key = int(input("Inserisci la chiave: "))
19
20 print("Testo cifrato: ", cesar_cipher(text, key))
```

In questo modo, il ciclo **while** continua a eseguire il blocco di codice fino a quando **i** è inferiore alla lunghezza del testo, che è equivalente al comportamento del ciclo **for** originale.

6.1 Esercizio

Io: Proponimi un esercizio nel quale sia necessario utilizzare un ciclo **for**. Genera la soluzione dell'esercizio.

llm:

Esercizio:

Scrivi uno script Python che stampi la somma dei primi n numeri naturali, dove n è un numero intero fornito dall'utente.

Traccia dell'esercizio:

1. Chiedi all'utente di inserire un numero intero n .
2. Utilizza un ciclo **for** per sommare i numeri da 1 a n .
3. Stampa la somma ottenuta.

Soluzione in Python:

```
1 # Chiedi all'utente di inserire un numero intero
2 n = int(input("Inserisci un numero intero: "))
3
4 # Inizializza una variabile per tenere traccia della somma
5 somma = 0
6
7 # Utilizza un ciclo for per sommare i numeri da 1 a n
8 for i in range(1, n + 1):
9     somma += i
10
11 # Stampa la somma ottenuta
12 print("La somma dei primi", n, "numeri naturali é", somma)
```

Io: E' possibile scrivere un'implementazione ricorsiva dello stesso esercizio? Potresti eventualmente generarla e spiegarmi passo passo come funziona?

llm:

Esercizio: Scrivi uno script Python che, data un'intera positiva n , calcoli la somma dei primi n numeri naturali.

Soluzione in Python:

```
1 def sum_of_naturals(n):
2     # Caso base per la ricorsione: se n é 0, la somma é 0
3     if n == 0:
4         return 0
5     # Somma del numero corrente più la somma dei numeri successivi
6     else:
```

```
7     return n + sum_of_naturals(n - 1)
8
9     n = int(input("Inserisci un numero intero positivo: "))
10
11    print("La somma dei primi", n, "numeri naturali é", sum_of_naturals(n))
```

In questa soluzione, la funzione `sum_of_naturals` è una funzione ricorsiva che calcola la somma dei primi n numeri naturali. La ricorsione avviene quando la funzione si chiama se stessa con un valore ridotto di n (da n a $n - 1$) fino a raggiungere il caso base in cui n è 0, altrimenti la somma viene calcolata sommando n al risultato della chiamata ricorsiva.