

Python

Manuale per le ripetizioni di Informatica

Marco Lampis

7 dicembre 2022

Indice

0	Informazioni generali	1
0.1	Come svolgo le lezioni	1
1	Introduzione	3
2	Tipi	5
3	Variabili	7
4	print	9
5	input	11
6	Commenti	13
7	for	15
8	while	17
9	list	19
10	Tuple	21
11	Main	23

0 Informazioni generali

Ciao! Sono **Marco**, sono uno studente magistrale in *Software Engineering* al Politecnico di Torino e mi sono laureato in Ingegneria Informatica all'università di Pisa. Nella vita sono un programmatore, uno smanettone e amante di videogiochi! Amo quello che studio, e per questo motivo fornisco ripetizioni di informatica con particolare attenzione a:

- programmazione (Java, C++, C, Python, C#, Javascript, PHP)
- algoritmi e strutture dati
- basi di Dati
- più o meno tutto quello che riguarda l'informatica!

Sia per studenti delle scuole superiori che per l'università.

0.1 Come svolgo le lezioni

La mia metodologia è innovativa e interattiva: utilizzo una piattaforma apposita per lavorare contemporaneamente sullo stesso file con gli studenti, come se fossimo accanto. In questo modo, le lezioni sono più coinvolgenti e divertenti, e gli studenti possono imparare in modo facile ed efficiente. Inoltre, le lezioni includono sia esercitazioni che approfondimenti teorici, aiuto compiti e revisione.

Per aiutare gli studenti a prepararsi al meglio e a esercitarsi anche a casa, ho creato un sito web dedicato con risorse utili come teoria, esercizi e slide. Su questo sito, gli studenti possono trovare tutto il materiale di cui hanno bisogno, e il materiale è sempre disponibile gratuitamente per i miei studenti. Inoltre, uso un iPad per prendere appunti durante le lezioni e poi condividerli con gli studenti, in modo che possano rivedere tutto ciò che è stato trattato.

Se volete migliorare le vostre conoscenze in informatica e avere un supporto personalizzato e professionale, non esitate a contattarmi. Sarò felice di fornirvi maggiori informazioni e di fissare una lezione con voi.

1 Introduzione

Python è un linguaggio di programmazione e ad alto livello, il che significa che è facile da leggere, da scrivere e ha una sintassi semplice da comprendere. Questo lo rende ideale per chiunque, dai principianti agli esperti, che vogliano imparare a programmare.

Una delle cose che rende Python così popolare è la sua versatilità. Può essere utilizzato per sviluppare molti tipi diversi di software, come applicazioni desktop, applicazioni web, giochi e perfino sistemi operativi. Inoltre, ci sono una vasta gamma di librerie e framework disponibili per Python, che possono essere utilizzati per aggiungere funzionalità extra al tuo codice.

Per iniziare a lavorare con Python, tutto ciò di cui hai bisogno è un semplice editor di testo e una conoscenza di base della sintassi del linguaggio. Ecco un esempio di codice Python che stampa il testo “Ciao, mondo!” sulla finestra del terminale:

```
1 # Questa é una riga di commento, che viene ignorata dall'interprete
  Python
2 # Quando viene eseguito il codice. I commenti sono utili per spiegare
3 # il codice e renderlo più facile da comprendere.
4
5 # Stampa il testo "Ciao, mondo!" sulla finestra del terminale
6 print("Ciao, mondo!")
```

La prima linea è un commento, che viene ignorato dall’interprete Python. La seconda linea utilizza la funzione `print()` per stampare il testo `Ciao, mondo!` sulla finestra del terminale.

In Python, le righe di codice vengono eseguite in ordine, dall’alto verso il basso. Ciò significa che nel nostro esempio, la riga che contiene la funzione `print()` viene eseguita dopo la riga di commento.

2 Tipi

In Python, i tipi di dato sono le categorie in cui vengono suddivisi i valori che possono essere memorizzati e manipolati in un programma. Ogni tipo di dato ha un insieme specifico di proprietà e metodi che possono essere utilizzati per lavorare con i valori di quel tipo.

In Python, ci sono diversi tipi di dati built-in, cioè tipi di dati che vengono forniti con il linguaggio e possono essere utilizzati senza dover importare alcuna libreria aggiuntiva. Ecco alcuni esempi di tipi di dato built-in in Python:

stringa: una sequenza di caratteri, come “Ciao, mondo!” o “12345”. Le stringhe possono essere create utilizzando apici singoli o doppi, ad esempio ‘Ciao, mondo!’ o “12345”. intero: un numero intero, come -5, 0 o 10. Gli interi possono essere positivi, negativi o zero. decimale: un numero con una parte decimale, come 3.14 o -2.718. I decimali possono essere positivi, negativi o zero. booleano: un valore logico che può essere vero o falso. In Python, i valori veri sono rappresentati dalla parola chiave True e i valori falsi dalla parola chiave False. Ecco alcuni esempi di come puoi creare variabili di questi tipi di dato in Python:

```
1 # Dichiarare una variabile chiamata "nome" e assegnare il valore "Mario"
  (stringa)
2 nome = "Mario"
3
4 # Dichiarare una variabile chiamata "anni" e assegnare il valore 25 (
  intero)
5 anni = 25
6
7 # Dichiarare una variabile chiamata "pi" e assegnare il valore 3.14 (
  decimale)
8 pi = 3.14
9
10 # Dichiarare una variabile chiamata "vero" e assegnare il valore True (
    booleano)
11 vero = True
```

Per creare una variabile di un determinato tipo di dato, basta assegnarle un valore di quel tipo. Ad esempio, per creare una variabile di tipo stringa, basta assegnarle una stringa tra apici singoli o doppi. Per creare una variabile di tipo intero o decimale, basta assegnarle un numero intero o decimale. E per creare una variabile di tipo booleano, basta assegnarle il valore True o False.

Una volta che hai creato una variabile, puoi utilizzare il suo valore nel tuo codice per eseguire operazioni o stamparlo sulla finestra del terminale, come vedremo successivamente

Inoltre, puoi utilizzare le variabili per eseguire operazioni matematiche. Ad esempio, puoi utilizzare la variabile `pi` per calcolare il perimetro di un cerchio di raggio 5 come segue:

```
1 # Dichiaro una variabile chiamata "pi" e assegno il valore 3.14 (
    decimale)
2 pi = 3.14
3
4 # Dichiaro una variabile chiamata "raggio" e assegno il valore 5 (
    intero)
5 raggio = 5
6
7 # Dichiaro una variabile chiamata "perimetro" e assegno il risultato
    del calcolo del perimetro del cerchio
8 perimetro = 2 * pi * raggio
```

In questo esempio, la variabile `perimetro` viene assegnata il valore 31.4, che è il risultato del calcolo del perimetro di un cerchio di raggio 5.

In sintesi: i tipi di dato in Python sono le categorie in cui vengono suddivisi i valori che possono essere utilizzati in un programma. Ogni tipo di dato ha un insieme specifico di proprietà e metodi che possono essere utilizzati per lavorare con i valori di quel tipo.

3 Variabili

In Python, le variabili sono utilizzate per memorizzare i valori che possono essere utilizzati in un programma. Una variabile può contenere una stringa, un numero intero o decimale, una booleana (cioè un valore vero o falso) o perfino un oggetto.

Per creare una variabile in Python, devi assegnare un valore a un nome. Ad esempio, puoi creare una variabile chiamata `nome` e assegnarle il valore “Mario” come segue:

```
1 # Dichiaro una variabile chiamata "nome" e assegno il valore "Mario"
2 nome = "Mario"
```

Una volta che hai creato una variabile, puoi utilizzarla nel tuo codice per accedere al valore che contiene. Ad esempio, puoi utilizzare la variabile `nome` per stampare il suo valore sulla finestra del terminale utilizzando la funzione `print()` come segue:

```
1 # Dichiaro una variabile chiamata "nome" e assegno il valore "Mario"
2 nome = "Mario"
3
4 # Stampa il valore della variabile "nome" sulla finestra del terminale
5 print(nome)
```

In questo esempio, la funzione `print()` stampa il valore della variabile `nome`, che è “Mario”.

Inoltre, puoi utilizzare le variabili per eseguire operazioni matematiche. Ad esempio, puoi utilizzare la variabile `nome` per eseguire un’operazione aritmetica come segue:

```
1 # Dichiaro due variabili chiamate "a" e "b" e assegno i valori 10 e 5
  rispettivamente
2 a = 10
3 b = 5
4
5 # Dichiaro una variabile chiamata "risultato" e assegno il risultato
  della somma di "a" e "b"
6 risultato = a + b
7
8 # Stampa il valore della variabile "risultato" sulla finestra del
  terminale
9 print(risultato)
```

In questo esempio, la variabile risultato viene assegnata il valore 15, che è il risultato della somma di a e b. Quindi, la funzione print() stampa il valore della variabile risultato, che è 15.

In sintesi: le variabili sono una parte essenziale della programmazione in Python. Ti **consentono di memorizzare i valori** che puoi utilizzare e modificare nel tuo codice, rendendo più facile e flessibile scrivere programmi complessi.

4 print

In Python, la funzione `print()` viene utilizzata per stampare una stringa (cioè una sequenza di caratteri) sulla finestra del terminale. Ad esempio, è possibile utilizzare la funzione `print()` per stampare il testo `Ciao, mondo!` come segue:

```
1 print("Ciao, mondo!")
```

La funzione `print()` può anche essere utilizzata per stampare il valore di una variabile. Ad esempio, se si ha una variabile chiamata `nome` che contiene un nome, è possibile utilizzare la funzione per stamparlo sulla finestra del terminale come segue:

```
1 # Dichiarare una variabile chiamata "nome" e assegnare il tuo nome
2 nome = "Mario"
3
4 # Stampa il valore della variabile "nome" sulla finestra del terminale
5 print(nome)
```

In questo esempio, la funzione `print()` stampa il valore della variabile `nome`, che è `Mario`.

Può anche essere utilizzata per stampare il risultato di un'espressione. Ad esempio, è possibile utilizzare stampare il risultato di un'operazione aritmetica come segue:

```
1 # Dichiarare due variabili chiamate "a" e "b" e assegnare i valori 10 e 5
  # rispettivamente
2 a = 10
3 b = 5
4
5 # Stampa il risultato della somma di "a" e "b" sulla finestra del
  # terminale
6 print(a + b)
```

In questo esempio stampa il risultato della somma di `a` e `b`, che è 15.

5 input

La funzione `input` in Python è una funzione che consente agli utenti di immettere dati dalla tastiera. La sintassi generale per utilizzare la funzione `input` è la seguente:

```
1 input([prompt])
```

Dove `prompt` è una stringa che viene visualizzata per invitare l'utente a immettere un input. Ad esempio, potresti utilizzare la funzione `input` come segue:

```
1 nome = input("Inserisci il tuo nome: ")
2 print("Ciao, " + nome + "!")
```

In questo esempio, verrebbe visualizzata la stringa `Inserisci il tuo nome:` e l'utente potrebbe immettere un valore come `Marioo Lucia`. Quando l'utente preme il tasto Invio, il valore immesso viene assegnato alla variabile `nome`, che viene quindi utilizzata dalla funzione `print` per stampare un messaggio di benvenuto.

La funzione `input` può anche essere utilizzata per chiedere all'utente di immettere un numero, ad esempio un intero o un numero in virgola mobile. In questo caso, il valore immesso dall'utente verrà automaticamente convertito in un valore numerico, che potrà essere utilizzato in operazioni matematiche o in altre parti del programma. Ad esempio:

```
1 anni = input("Inserisci la tua età: ")
2 anni = int(anni) # converte il valore immesso in un intero
3 print("Tra 10 anni avrai " + str(età + 10) + " anni.")
```

In questo esempio, viene chiesto all'utente di immettere la propria età come un numero intero. Il valore immesso viene assegnato alla variabile `anni` e quindi convertito in un intero con la funzione `int()`. Infine, viene utilizzata la funzione `print` per stampare un messaggio che indica quanti anni avrà l'utente tra 10 anni.

In sintesi: la funzione `input` in Python è una funzione molto utile per consentire agli utenti di immettere dati dalla tastiera e utilizzarli all'interno del programma.

6 Commenti

In Python, i commenti sono linee di codice che vengono ignorate dall'interprete Python. I commenti vengono utilizzati per inserire note o spiegazioni sul codice, in modo che sia più facile per altre persone comprenderlo.

Per inserire un commento in Python, si utilizza il carattere di commento # all'inizio della linea. Ad esempio:

```
1 # Questa é una linea di commento
```


7 for

In Python, il costrutto `for` è utilizzato per eseguire un blocco di codice ripetutamente per ogni elemento di una sequenza di dati. Ad esempio, si può utilizzare il costrutto `for` per stampare tutti gli elementi di una lista di numeri:

```
1 # Crea una lista di numeri
2 numeri = [1, 2, 3, 4, 5]
3
4 # Per ogni numero nella lista, stampa il numero
5 for numero in numeri:
6     print(numero)
```


8 while

In Python, il costrutto `while` è utilizzato per eseguire un blocco di codice ripetutamente finché una determinata condizione viene soddisfatta. Ad esempio, si può utilizzare il costrutto `while` per stampare una sequenza di numeri fino a quando l'utente non inserisce il numero 0:

```
1 # Inizializza una variabile per memorizzare il numero inserito dall'
  utente
2 numero = 1
3
4 # Continua a chiedere all'utente di inserire un numero finché non
  inserisce 0
5 while numero != 0:
6     # Chiede all'utente di inserire un numero
7     numero = int(input("Inserisci un numero (0 per uscire): "))
8
9     # Se il numero inserito non é 0, stampa il numero
10    if numero != 0:
11        print("Hai inserito:", numero)
```


9 list

In Python, una lista è una raccolta ordinata di elementi. Si può pensare ad una lista come ad una sequenza di oggetti che possono essere di tipi diversi. Ad esempio, una lista potrebbe contenere un intero, una stringa e un'altra lista come elementi.

Per creare una lista in Python, si utilizza la sintassi seguente:

```
1 lista = [elemento1, elemento2, elemento3]
```

Ad esempio, per creare una lista di numeri interi, potremmo scrivere:

```
1 numeri = [1, 2, 3]
```

Per accedere a un elemento della lista, si utilizza l'indice dell'elemento. L'indice di un elemento è il suo numero di posizione nella lista, iniziando da 0. Ad esempio, per accedere al primo elemento della lista numeri creata sopra, scriveremo:

```
1 primo_elemento = numeri[0]
```


10 Tuple

In Python, una tupla è simile ad una lista, ma è immutabile, cioè una volta creata non può essere modificata. Una tupla viene utilizzata quando si vuole rappresentare un insieme di valori che non dovranno mai cambiare.

Per creare una tupla in Python, si utilizza la sintassi seguente:

```
1 tupla = (elemento1, elemento2, elemento3)
```

Ad esempio, per creare una tupla di stringhe, potremmo scrivere:

```
1 colori = ("rosso", "verde", "blu")
```

Come per le liste, è possibile accedere agli elementi della tupla utilizzando gli indici. Ad esempio, per accedere alla prima stringa nella tupla colori, scriveremo:

```
1 primo_colore = colori[0]
```


11 Main

In Python, il termine “main” si riferisce alla funzione principale del programma, ovvero la funzione che viene eseguita quando il programma viene avviato. In molti linguaggi di programmazione, la funzione principale viene solitamente chiamata `main` o `main()`.

La sintassi generale per definire la funzione principale in Python è la seguente:

```
1 if __name__ == "__main__":  
2     # codice da eseguire
```

In questo codice, la prima riga utilizza una costante predefinita di Python chiamata **name** per verificare se il programma viene eseguito direttamente o importato da un altro programma. Se il programma viene eseguito direttamente, cioè se viene avviato dalla riga di comando o dal prompt dei comandi, il valore della costante **name** sarà “main”. In questo caso, il codice all’interno del blocco `if` verrà eseguito.

Ad esempio, il seguente programma Python definisce una funzione principale che stampa il messaggio “Ciao, mondo!” e quindi chiama la funzione `saluta()` che stampa il messaggio “Ciao, utente!”:

```
1 def saluta():  
2     print("Ciao, utente!")  
3  
4 if __name__ == "__main__":  
5     print("Ciao, mondo!")  
6     saluta()
```

Se questo programma viene eseguito direttamente, verrà visualizzato il seguente output:

```
1 Ciao, mondo!  
2 Ciao, utente!
```

In sintesi: la funzione principale in Python è una funzione che viene eseguita quando il programma viene avviato direttamente e non viene importato da un altro programma. La funzione principale può essere utilizzata per eseguire il codice principale del programma, come la definizione di variabili globali, la chiamata di altre funzioni e la gestione degli errori.

