Introduzione

A chi si rivolge questo libro?

Se la vostra intenzione è quella di imparare a programmare, quella che avete fra le mani è un'ottima risorsa. Python è un linguaggio particolarmente adatto a chi è agli inizi, ed è anche una delle competenze più richieste, oggi, nel mondo del lavoro.

Anche il momento è quello giusto, perché oggi imparare a programmare è più facile che mai. Grazie ad assistenti virtuali come ChatGPT, non vi troverete mai a studiare da soli. In questo libro, vi suggerirò vari modi in cui potete usare questi nuovi strumenti per accelerare l'apprendimento.

Questo libro si rivolge principalmente a chi non ha mai programmato, e anche a chi ha una certa esperienza ma con un altro linguaggio di programmazione. Se però avete già programmato in Python, potreste trovare i primi capitoli un po' troppo lenti.

Una delle difficoltà insite nell'imparare a programmare è che occorre studiare due linguaggi: uno, naturalmente, è il linguaggio di programmazione; l'altro è il vocabolario usato per parlare di programmi. Apprendendo solo il linguaggio di programmazione, probabilmente avreste problemi nell'interpretare un messaggio d'errore, nel leggere la documentazione, nel parlare con altri o nell'usare gli assistenti virtuali. Se avete già fatto un po' di programmazione, ma non avete ancora acquisito anche questo "gergo", credo che questo libro possa esservi utile.

Obiettivi del libro

Scrivendo questo libro, ho cercato di essere il più possibile attento al vocabolario. Definisco ogni termine ogni volta che lo introduco. E poi ho posto un Glossario alla fine di ogni capitolo, che riepiloga i termini introdotti.

Ho anche cercato di essere conciso. Minore è lo sforzo mentale necessario per leggere il libro, maggiori saranno le energie che potrete dedicare alla programmazione.

Ma non potete imparare a programmare solo leggendo un libro: dovete esercitarvi. Per questo motivo, il libro include vari esercizi che troverete alla fine di ogni capitolo, grazie ai quali potrete mettere in pratica ciò che avete appena imparato.

Se leggerete attentamente il testo e svolgerete gli esercizi in modo costante, vi vedrete fare enormi progressi. Ma vi avverto: imparare a programmare non è facile e può essere frustrante anche per i programmatori esperti. Mentre procederemo,

vi suggerirò varie strategie per aiutarvi a scrivere programmi corretti e a correggere quelli errati.

Argomenti trattati

Ogni capitolo tratta argomenti che si basano sui capitoli precedenti, quindi vi consiglio di leggere i capitoli nell'ordine in cui si presentano, e dedicare del tempo alla risoluzione degli esercizi, prima di proseguire.

I Capitoli dall'1 al 6 introducono elementi di base, come l'aritmetica, le istruzioni condizionali e i cicli. Presentano anche il concetto più importante della programmazione, le funzioni, e un modo particolarmente potente per usarle: la ricorsione.

I Capitoli 7 e 8 introducono le stringhe, elementi che possono rappresentare lettere, parole e frasi, più gli algoritmi per manipolarle.

I Capitoli dal 9 al 12 introducono le strutture di dati principali di Python, ovvero le liste, i dizionari e le tuple, strumenti potenti per scrivere programmi efficienti. Il Capitolo 12 presenta gli algoritmi per l'analisi del testo e per generare nuovi testi. Algoritmi come questi sono al centro dei modelli LLM (*Large Language Model*), quindi questo capitolo vi darà un'idea di come funzionano strumenti come ChatGPT.

Il Capitolo 13 riguarda i modi per archiviare i dati in archivi persistenti: file e database. Come esercizio, scriverete un programma che cerca in un file system e trova i file duplicati.

I Capitoli dal 14 al 17 introducono la programmazione a oggetti (OOP – *Object-Oriented Programming*), un modo per organizzare sia i programmi sia i dati che essi elaborano. Molte librerie Python sono state scritte proprio in uno stile a oggetti, quindi questi capitoli vi aiuteranno a comprenderne la struttura e anche a definire nuovi oggetti.

L'obiettivo di questo libro non è quello di trattare esaustivamente l'intero linguaggio Python. Piuttosto, mi sono concentrato su un sottoinsieme ottimizzato del linguaggio, per fornirvi il maggior numero possibile di competenze con il minor numero possibile di concetti. Tuttavia, Python offre molte funzionalità che potrete usare per risolvere molti problemi in modo particolarmente efficiente. Il Capitolo 18 presenta alcune di queste.

Infine, nel Capitolo 19 vi offro i miei pensieri finali e i miei suggerimenti per proseguire il vostro viaggio nel mondo della programmazione.

Le novità di questa edizione

I cambiamenti più significativi di questa edizione sono stati introdotti da due nuove tecnologie: i notebook Jupyter e gli assistenti virtuali.

Ogni capitolo di questo libro è un notebook Jupyter, ovvero un documento che contiene sia testo sia codice. Per me, questo ha facilitato la scrittura del codice, il collaudo dei programmi e la cura della coerenza. Per voi, significa che potrete eseguire il codice, modificarlo e lavorare sugli esercizi, il tutto in un unico posto.

L'altro grande cambiamento è che ho aggiunto suggerimenti riguardanti il modo in cui lavorare con assistenti virtuali come ChatGPT e per usarli per accelerare l'apprendimento. Quando è stata pubblicata la precedente edizione in lingua inglese di questo libro, nel 2015, i predecessori di questi strumenti erano molto più rudimentali, e la maggior parte delle persone non ne era a conoscenza. Ora essi rappresentano uno strumento standard per l'ingegneria del software e sono convinto che saranno uno strumento trasformativo anche per imparare a programmare (e anche per imparare molte altre cose).

Gli altri cambiamenti sono stati motivati da alcuni miei rimpianti per la precedente edizione. Il primo è che non ho parlato a sufficienza dei test del software. Era un'omissione deplorevole già nel 2015, ma con l'avvento degli assistenti virtuali, il test automatizzato è diventato ancora più importante. Quindi, questa edizione presenta gli strumenti di test più utilizzati in Python, `doctest` e `unittest`, e include diversi esercizi grazie ai quali potrete esercitarvi a utilizzarli.

Un altro mio rammarico era che gli esercizi presenti nella precedente edizione erano irregolari: alcuni erano più interessanti di altri e alcuni erano troppo difficili. Il passaggio ai notebook Jupyter mi ha aiutato a sviluppare e collaudare una sequenza di esercizi più coinvolgente ed efficace.

In questa nuova edizione, la sequenza degli argomenti è pressoché la stessa, ma ho riorganizzato alcuni capitoli e compresso due brevi capitoli, raggruppandoli in uno solo. Inoltre, ho ampliato la trattazione delle stringhe, per includere le espressioni regolari.

Alcuni capitoli usano la grafica "a tartaruga". Nelle edizioni precedenti, ho usato il modulo `turtle` di Python che, sfortunatamente, non funziona nei notebook Jupyter. Quindi l'ho sostituito con un nuovo modulo, che dovrebbe essere anche più facile da usare.

Infine, ho riscritto una parte sostanziale del testo, chiarendo meglio alcuni punti e tagliando quelli in cui non ero stato abbastanza sintetico.

Sono molto orgoglioso di questa nuova edizione e spero che piaccia anche a voi!

Per iniziare

Per la maggior parte dei linguaggi di programmazione, incluso Python, esistono molti strumenti utili per scrivere ed eseguire i programmi. Questi strumenti sono chiamati *ambienti di sviluppo integrati* (IDE – *Integrated Development Environment*). In generale, ci sono due tipi di IDE.

- Alcuni sono adatti ai file di codice, quindi forniscono strumenti appositi per modificare ed eseguire questi tipi di file.
- Altri lavorano principalmente con i notebook, che sono documenti che contengono sia testo sia codice.

Per i principianti, consiglio di iniziare con un ambiente di sviluppo a notebook, come Jupyter. I notebook per questo libro sono disponibili in un repository online su *https://allendowney.github.io/ThinkPython*. Ci sono due modi per usarli.

- Potete scaricare i notebook ed eseguirli sul vostro computer. In tal caso, dovrete installare Python e Jupyter, il che non è difficile, ma se la vostra intenzione è quella di imparare a programmare in Python, può essere frustrante il fatto di dedicare così tanto tempo all'installazione del software.
- Un'alternativa è eseguire i notebook su Colab, un ambiente Jupyter che funziona in un browser web e che non richiede di installare nulla. Colab è gestito da Google ed è gratuito.

Se siete agli inizi, vi consiglio vivamente di iniziare con Colab.

Risorse per i docenti

Se state utilizzando questo libro come testo didattico, ecco alcune risorse che potrebbero esservi utili.

- Potete trovare i notebook con le soluzioni degli esercizi e i link verso le risorse aggiuntive elencate su *https://allendowney.github.io/ThinkPython*.
- Nella versione *O'Reilly Learning Platform* di questo libro sono disponibili i quiz per ogni capitolo, più un quiz riepilogativo per l'intero libro.
- *Teaching and Learning with Jupyter* è un libro online con molti suggerimenti per usare Jupyter in modo efficace in aula. Potete trovare questo libro su *https://jupyter4edu.github.io/jupyter-edu-book*.
- Uno dei modi migliori per usare i notebook è il *live coding*: il docente scrive il codice e gli studenti lo seguono nei loro notebook. Per saperne di più sul live coding e per altri ottimi consigli sull'insegnamento della programmazione, vi consiglio la formazione per docenti fornita da *The Carpentries*, su *https://carpentries.github.io/instructor-training*.

Convenzioni utilizzate

In questo libro vengono utilizzate alcune convenzioni tipografiche.

Corsivo
Indica i nuovi termini, gli URL, gli indirizzi e-mail, i nomi e le estensioni dei file.

Grassetto
Indica la prima occorrenza di un nuovo termine tecnico, cui corrisponde sempre anche una voce nel glossario.

```
Font monospaziato
```

Utilizzato nei listati e anche all'interno dei paragrafi per fare riferimento a determinati elementi del programma, quali nomi di variabili o funzioni, database, tipi di dati, variabili d'ambiente, istruzioni e parole chiave.

Uso degli esempi di codice

Il materiale supplementare (esempi di codice, esercizi e così via) è disponibile per il download su *https://allendowney.github.io/ThinkPython*.

Se avete domande tecniche o problemi nell'utilizzo degli esempi di codice, inviate un'e-mail a *support@oreilly.com*.

Questo libro ha lo scopo di aiutarvi a svolgere il vostro lavoro. In generale, se in questo libro trovate un esempio, potete tranquillamente usarlo nei vostri programmi e nella vostra documentazione. Non avete bisogno di chiedere alcun permesso per ottenere l'autorizzazione, a meno che non stiate riproducendo una parte significativa del codice. Per esempio, se dovete scrivere un programma che utilizza più blocchi di codice di questo libro non dovete chiedere alcuna autorizzazione. Al contrario, vendere o distribuire esempi tratti dai libri O'Reilly richiede un'autorizzazione. Rispondere a una domanda citando questo libro e il codice di esempio non richiede alcuna autorizzazione. Incorporare nella documentazione di un vostro prodotto una quantità significativa di codice tratto da questo libro, invece, richiede un'autorizzazione.

Apprezziamo, ma in genere non richiediamo, l'attribuzione. Un'attribuzione, di solito, include il titolo, l'autore, l'editore e l'ISBN. Per esempio: "*Think Python* di Allen B. Downey (O'Reilly). Copyright 2024 Allen B. Downey, 978-1-098-15543-8".

Se ritenete che il vostro uso degli esempi di codice non sia conforme alle indicazioni sopra riportate, contattaci pure all'indirizzo *permissions@oreilly.com*.

Pensare in Python e i notebook Jupyter

Questa è una piccola introduzione ai notebook Jupyter, rivolta in modo specifico agli utilizzatori di questo libro. Un notebook Jupyter è un documento contenente del testo, del codice e i risultati prodotti dall'esecuzione del codice.

Potete leggere un notebook come un libro, ma potete anche utilizzarlo per eseguire il codice, per modificarlo e per sviluppare nuovi programmi.

I notebook Jupyter vengono eseguiti in un browser web, quindi potete eseguirne il codice senza installare alcun nuovo software. Ma dovranno potersi connettere a un server Jupyter.

Potete anche installare ed eseguire un vostro server, ma per iniziare è più facile utilizzare un servizio come *Colab* (*https://colab.research.google.com*), gestito da Google.

Nella home page del libro (*https://allendowney.github.io/ThinkPython*) troverete un link per i notebook di ogni capitolo. Facendo clic su uno di questi link, aprirete un notebook su Colab.

Che cos'è un notebook

Un notebook Jupyter è composto da celle, ognuna delle quali contiene testo o codice. Una cella contenente codice ha il seguente aspetto:

```
[ ] print('Hello')
```

In questo caso, fate clic sulla cella per selezionarla. Fra le parentesi quadre comparirà un pulsante "Play". Facendo clic su questo pulsante, Jupyter eseguirà il codice presente nella cella e visualizzerà il risultato.

Quando in un notebook eseguite del codice per la prima volta, potrebbero essere necessari alcuni secondi per avviarlo. E se si tratta di un notebook scritto da altri, potreste ricevere un messaggio *Avviso*. Se l'origine del notebook è fidata, come per esempio nel caso dei notebook di questo libro, potete eseguire tranquillamente il codice facendo clic su *Esegui comunque*.

Invece di fare clic sul pulsante "Play", potete eseguire il codice presente in una cella con la combinazione di tasti Maiusc-Invio.

Se state eseguendo questo notebook su Colab, dovreste vedere in alto a sinistra i pulsanti "+ Codice" e "+ Testo". Il primo aggiunge una cella di codice e il secondo aggiunge una cella di testo. Per provarli, selezionate una cella con un clic, quindi fate clic sul pulsante "+ Testo". A questo punto dovrebbe apparire una nuova cella sotto quella selezionata.

Aggiungete del testo a piacere alla cella. Potete usare alcuni pulsanti di formattazione, oppure potete contrassegnare il testo usando vari elementi di formattazione (*https://www.markdownguide.org/basic-syntax*). Quando avrete finito, premete Maiusc-Invio per formattare il testo appena digitato e passare alla cella successiva.

In qualsiasi momento, Jupyter può trovarsi in due modalità.

- In **modalità comando** potete eseguire le operazioni che interessano le celle, come aggiungere e rimuovere intere celle.

- In **modalità editing** potete modificare il contenuto di una cella.

Con le celle di testo, è evidente in quale modalità vi trovate. In modalità di editing, la cella è divisa verticalmente: il testo che state modificando è a sinistra, il testo formattato è a destra e gli strumenti di editing del testo sono in alto. In modalità comando, vedrete solo il testo formattato.

Con le celle di codice, la differenza è meno evidente, ma se nella cella c'è un cursore, siete in modalità editing.

Per passare dalla modalità editing alla modalità comando, premete Esc. Per passare dalla modalità comando alla modalità editing, premete Invio.

Quando avrete finito di lavorare su un notebook, potete chiudere la finestra, ma tutte le modifiche apportate scompariranno. Se apportate modifiche che volete conservare, aprite il menu *File* in alto a sinistra. Troverete diversi modi per salvare il notebook.

- Se avete un account Google, potete salvare il notebook nel vostro Drive.
- Se avete un account GitHub, potete salvarlo su GitHub.
- Oppure se volete salvare il notebook sul vostro computer, selezionate *Download* e poi *Download .ipynb*. Il suffisso *.ipynb* indica che si tratta di un file notebook, mentre il suffisso *.py* indica che il file contiene solo codice Python.

Codice per Pensare in Python

All'inizio di ogni notebook, troverete una cella con codice come questo:

```
from os.path import basename, exists

def download(url):
    filename = basename(url)
    if not exist(filename):
        from urllib.request import urlretrieve

        local, _ = urlretrieve(url, filename)
        print("Downloaded " + str(local))
    return filename

download('https://raw.githubusercontent.com/AllenDowney/
ThinkPython/v3/thinkpython.py')

import thinkpython
```

Al momento non c'è bisogno di sapere come funziona questo codice, ma quando arriverete alla fine del libro avrà molto più senso. Come potete forse immaginare, questo codice scarica un file, in particolare *thinkpython.py*, che contiene il codi-

ce Python creato in modo specifico per questo libro. L'ultima riga "importa" questo codice, il che significa che permette di usare il codice nel notebook.

In altri capitoli, troverete codice che scarica *diagram.py*, utilizzato per generare i diagrammi del libro, e *jupyturtle.py*, che viene utilizzato in diversi capitoli per creare la grafica "a tartaruga".

In alcuni punti vedrete una cella come questa, che inizia con `%%expect`.

```
[ ] %%expect SyntaxError
    abs 42
```

Il comando `%%expect` non fa parte del linguaggio Python: è un "comando magico" di Jupyter che indica che ci aspettiamo che la cella produca un errore. Quando vedete questo comando, significa che l'errore è intenzionale, solitamente per avvisarvi di una trappola comune.

Per maggiori informazioni sull'esecuzione di notebook Jupyter su Colab, consultate *https://colab.research.google.com/notebooks/basic_features_overview.ipynb*. Se vi sentite pronti a iniziare, passate a leggere il Capitolo 1.

O'Reilly Online Learning

Da oltre quarant'anni, O'Reilly Media fornisce un servizio di formazione, studio e approfondimento in ambito tecnologico e aziendale per aiutare le aziende a conseguire il pieno successo.

La nostra rete di esperti e innovatori condivide le proprie conoscenze e competenze tramite libri, articoli e la nostra piattaforma di apprendimento online, che vi offre un accesso *on-demand* a corsi di formazione dal vivo, percorsi di apprendimento, ambienti di programmazione interattivi e una vasta raccolta di testi e video, sia di O'Reilly sia di oltre 200 altri editori. Per maggiori informazioni, visitate *https://oreilly.com*.

Come contattarci

Per commenti e domande riguardanti questo libro, potete contattare l'editore:

O'Reilly Media, Inc.
1005 Gravenstein Highway Nord
Sebastopol, CA 95472
800-889-8969 (Stati Uniti o Canada)
707-827-7019 (internazionale o locale)
707-829-0104 (fax)
support@oreilly.com
https://www.oreilly.com/about/contact.html

Abbiamo una pagina web dedicata a questo libro, dove elenchiamo gli errata, altri esempi e ogni informazione aggiuntiva. Potete accedere a questa pagina su *https://oreil.ly/think-python-3e*.

Per novità e informazioni sui nostri libri e corsi, visitate *https://oreilly.com*.

Potete trovarci su LinkedIn: *https://linkedin.com/company/oreilly-media*.

Potete anche vederci su YouTube: *https://youtube.com/oreillymedia*.

Ringraziamenti

Grazie infinite a Jeff Elkner, che ha tradotto in Python il mio libro su Java, cosa che ha dato il via a questo progetto e mi ha fatto conoscere quello che poi si è rivelato essere il mio linguaggio preferito. Grazie anche a Chris Meyers, che ha contribuito con vari paragrafi di *How to Think Like a Computer Scientist* (Green Tea Press).

Ringrazio la Free Software Foundation per aver sviluppato la *GNU Free Documentation License*, che ha reso possibile la mia collaborazione con Jeff e Chris, e ringrazio Creative Commons per la licenza che sto utilizzando ora.

Ringrazio gli sviluppatori e i curatori del linguaggio Python e delle librerie che ho utilizzato, tra cui il modulo grafico della tartaruga; gli strumenti che ho utilizzato per sviluppare il libro, tra cui Jupyter e JupyterBook; e i servizi che ho utilizzato, tra cui ChatGPT, Copilot, Colab e GitHub.

Grazie ai redattori di Lulu, che hanno lavorato su *How to Think Like a Computer Scientist*, e agli editor di O'Reilly Media, che hanno lavorato su *Think Python*.

Un ringraziamento speciale va ai revisori tecnici della precedente edizione, Melissa Lewis e Luciano Ramalho, e della presente edizione, Sam Lau e Luciano Ramalho (di nuovo). Sono grato a Luciano per aver sviluppato il modulo grafico della "tartaruga" che uso in diversi capitoli, chiamato `jupyturtle`.

Grazie a tutti gli studenti che hanno utilizzato le versioni precedenti di questo libro e a tutti i collaboratori che hanno inviato correzioni e suggerimenti. Sono più di cento i lettori attenti e coscienziosi che hanno inviato suggerimenti e correzioni negli ultimi anni. I loro contributi e l'entusiasmo per questo progetto sono stati di grande aiuto.

Se avete un suggerimento o una correzione, scriveteci a *feedback@thinkpython.com*. Se includete almeno una parte della frase in cui compare l'errore, sarà per me più facile trovarlo. Anche i numeri di pagina e di paragrafo vanno bene, ma non sono altrettanto facili da usare. Grazie!