Prefazione

Questo è un testo introduttivo di programmazione dei calcolatori, con l'utilizzo del linguaggio Python, che focalizza l'attenzione sugli elementi essenziali e sull'efficacia dell'apprendimento. Progettato per essere d'aiuto a un'ampia gamma di studenti, con diversi interessi e differenti abilità, è un volume adatto a un primo corso di programmazione per studenti di informatica, ingegneria e altre discipline. Non è richiesta alcuna precedente esperienza nell'ambito della programmazione e anche le conoscenze di algebra che si rivelano necessarie sono modeste, normalmente coperte dai programmi svolti nelle scuole superiori. Il libro usa Python 3, che è più adatto alla didattica rispetto a Python 2.

Vediamo alcune caratteristiche essenziali del testo:

- **Presentare prima gli aspetti fondamentali.**
  Il libro segue un percorso didattico tradizionale, ponendo prima l'accento sulle strutture di controllo, le funzioni, la scomposizione procedurale e le strutture dati predefinite. Gli oggetti vengono utilizzati, tutte le volte che questo si rivela utile, anche nei primi capitoli, ma gli studenti iniziano a progettare e a realizzare classi soltanto nel Capitolo 9.
- **Assistere gli studenti con guide mirate ed esempi completi.**
  Spesso i programmatori principianti chiedono: "Come posso cominciare? E ora cosa devo fare?" Un'attività complessa come la programmazione non può, ovviamente, ridursi a un ricettario di istruzioni, però guide mirate e dettagliate possono essere estremamente utili per acquisire confidenza con la materia, costituendo uno schema da tenere sotto mano durante la soluzione dei problemi assegnati. I paragrafi che si intitolano "Soluzione di problemi: …" mettono in forte evidenza l'importanza della progettazione e della pianificazione. Il libro contiene, poi, un'ampia rassegna di guide, denominate "Consigli pratici" e focalizzate su problemi frequenti, seguite

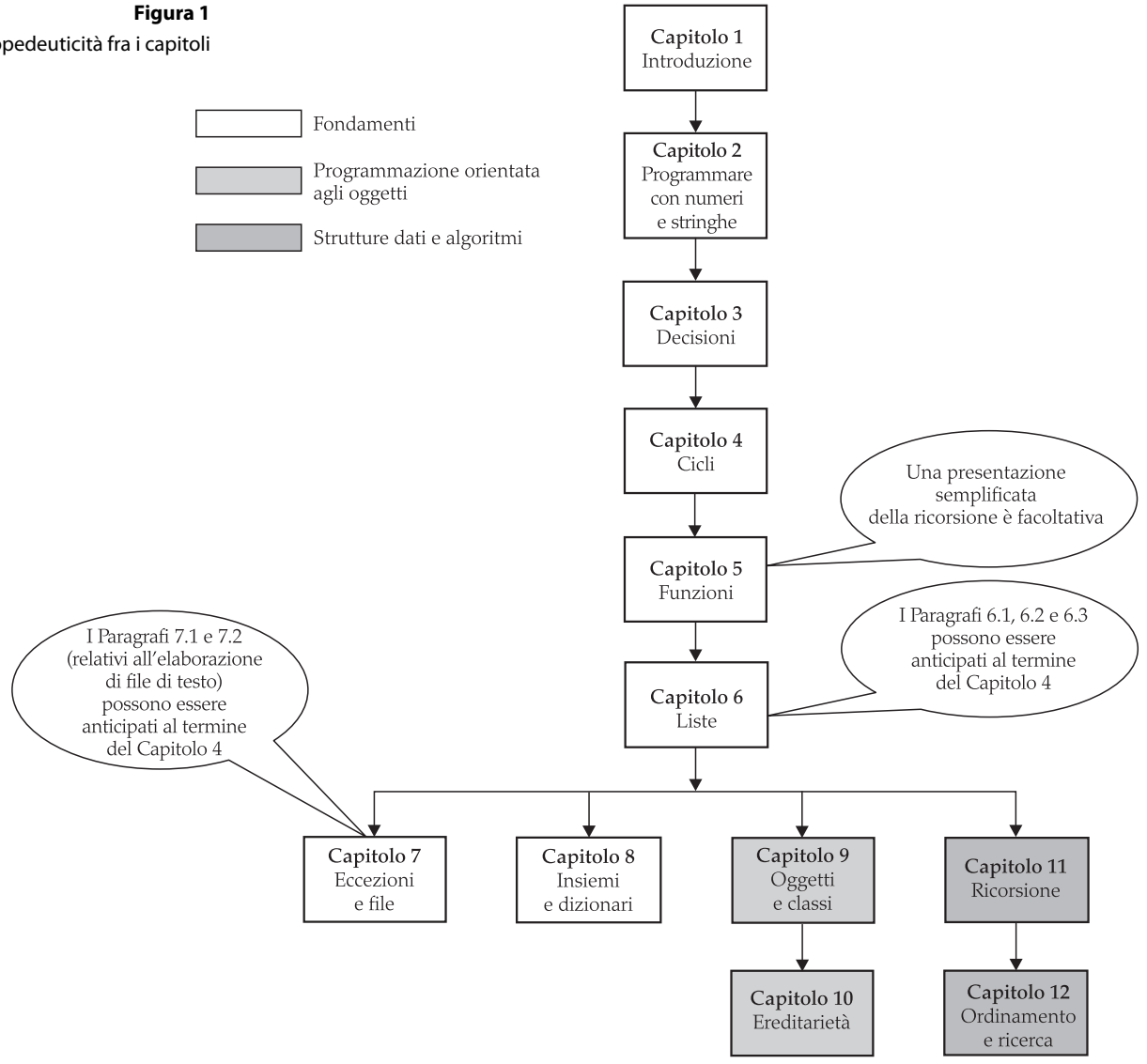
dalla presentazione di ulteriori "Esempi completi", che mostrano come applicare a problemi interessanti i concetti appresi nel capitolo.

- **Le strategie di risoluzione dei problemi sono rese esplicite.**
  L'esposizione passo dopo passo di tecniche veramente pratiche aiuta gli studenti a scoprire e valutare soluzioni ai problemi di programmazione. Queste strategie, presentate nei punti del percorso in cui si ritiene siano più efficaci, consentono a molti studenti di superare le barriere che li separano dal successo:

  – Progettazione di algoritmi (con pseudocodice)
  – Prima lo si fa a mano
  – Diagrammi di flusso
  – Casi di prova
  – Tenere traccia dell'esecuzione
  – Storyboard
  – Iniziare con un caso semplice
  – Funzioni riutilizzabili
  – Miglioramenti successivi
  – Adattamento di algoritmi
  – Scoprire algoritmi facendo esperimenti concreti
  – Tenere traccia di oggetti
  – Schemi ricorrenti per oggetti
  – Pensare ricorsivamente
  – Stimare il tempo di esecuzione di un algoritmo

- **La pratica rende migliori.**
  Al termine dei loro studi, gli studenti di un corso di programmazione devono ovviamente essere in grado di realizzare programmi non banali, ma, inizialmente, devono prendere confidenza con tutto ciò che accade. In questo libro, ciascun capitolo contiene molti esercizi che chiedono agli studenti di portare a termine progetti progressivamente più impegnativi: analizzare l'esecuzione di codice passo dopo passo e comprenderne gli effetti, progettare spezzoni di programmi a partire da esercizi già predisposti e, infine, scrivere completamente alcuni semplici programmi, tutto suddiviso in esercizi di ripasso ed esercizi di programmazione.
- **Un approccio visivo motiva il lettore e agevola lo studio.**
  Molte figure illustrano, passo dopo passo, il funzionamento dei programmi più complessi. Riquadri di richiami sintattici e tabelle che riportano esempi sono soltanto alcuni degli ausili grafici di cui è disseminato il testo. Questi elementi agevolano la comprensione del materiale didattico ancor prima di una lettura approfondita del testo.
- **Concentrarsi sugli elementi essenziali, pur rimanendo tecnicamente corretti.**
  Una trattazione enciclopedica non è di alcun aiuto per un programmatore principiante, ma non lo è nemmeno un approccio opposto, che riduca il materiale a un elenco di punti semplificati, che danno soltanto l'illusione di aver compreso. In questo libro, gli aspetti essenziali di ciascun argomento sono presentati con dimensioni digeribili, aggiungendo note che consentono di approfondire le buone pratiche di programmazione o le caratteristiche del linguaggio, una volta che il lettore sia pronto per tali informazioni aggiuntive.

Figura 1
Propedeuticità fra i capitoli



Una panoramica del libro

La Figura 1 mostra le propedeuticità fra i capitoli e l'organizzazione degli argomenti. Gli aspetti fondamentali della materia trattata nel libro sono coperti dai primi otto capitoli:

Capitolo 1. Introduzione
Capitolo 2. Programmare con numeri e stringhe
Capitolo 3. Decisioni
Capitolo 4. Cicli
Capitolo 5. Funzioni

Capitolo 6. Liste
Capitolo 7. Eccezioni e file
Capitolo 8. Insiemi e dizionari

I due capitoli seguenti si occupano della programmazione orientata agli oggetti:

Capitolo 9. Oggetti e classi
Capitolo 10. Ereditarietà

Infine, gli ultimi due capitoli consentono a un corso di approfondire l'analisi e la progettazione di algoritmi:

Capitolo 11. Ricorsione
Capitolo 12. Ordinamento e ricerca

Le appendici, disponibili online all'indirizzo `www.apogeoeducation.com`, contengono utile materiale di riferimento per gli studenti:

Appendice A. Linguaggio Python: operatori
Appendice B. Linguaggio Python: parole riservate
Appendice C. Linguaggio Python: libreria standard
Appendice D. I sottoinsiemi Basic Latin e Latin-1 di Unicode
Appendice E. Numeri binari e operazioni tra bit
Appendice F. Compendio di HTML
Appendice G. Compendio di Python

Novità di questa edizione

Un aspetto innovativo: la scienza dei dati (*data science*)

I metodi della *data science* stanno diventando così importanti che studenti di molte discipline, anche lontane dall'informatica, sono impazienti di apprendere le basi della programmazione. Python è divenuto il linguaggio abilitante per gli scienziati principianti, grazie alla sua struttura logica, alle librerie dedicate alla programmazione interattiva, che invitano all'esplorazione, e a una grande disponibilità di librerie per la manipolazione dei dati.

Questo libro mette in campo una metodologia per l'insegnamento della programmazione che è ben consolidata e non specificatamente limitata ai corsi di laurea di informatica. In questa edizione abbiamo inserito ulteriori esempi ed esercizi nell'ambito specifico della scienza dei dati.

Tanto Python quanto serve

Scrivendo questo libro, il nostro obiettivo è stato quello di fornire un buon insegnamento sia di programmazione sia di informatica generale, usando Python come strumento piuttosto che come fine.

Seguendo i preziosi suggerimenti degli utilizzatori delle edizioni precedenti, in questa edizione abbiamo anticipato la trattazione delle operazioni più importanti per operare con stringhe e liste.

Sempre più strumenti operativi

Le sezioni di "Strumenti" presentano alcuni pacchetti assai utili tra quelli che popolano il meraviglioso ecosistema delle librerie di Python. Gli studenti imparano a lavorare con la statistica, a disegnare grafici e tabelle, a inviare messaggi di posta elettronica, a elaborare fogli di calcolo elettronici e ad analizzare pagine web. Le librerie vengono inquadrate nell'ambito dei principi della scienza dell'informazione e gli studenti capiscono come quelle linee teoriche si applicano per risolvere problemi del mondo reale. Ciascuna sezione "Strumenti" è accompagnata da molti nuovi esercizi di ripasso e di programmazione, alla fine del capitolo.

Una libreria per la grafica e l'elaborazione d'immagini

La scrittura di programmi che creano disegni o elaborano immagini può aiutare gli studenti a visualizzare in modo efficace argomenti complessi. Nel Capitolo 2 presentiamo una libreria grafica *open source*, EzGraphics, mostrando come la si possa utilizzare per i primi semplici disegni. La libreria è decisamente semplificata rispetto alla libreria standard di Python per la grafica, Tkinter, e consente anche alcune semplici elaborazioni d'immagini. Il Capitolo 5 presenta una sezione di "Strumenti" dedicata alla *turtle graphics* e in vari capitoli sono presenti sezioni di "Esempi completi" relative alla grafica.

Esercizi

Gli esercizi che chiudono ogni capitolo contengono un'ampia rassegna di domande di ripasso e di progetti di programmazione, con gruppi di domande, facoltativi e ben segnalati, che riguardano la grafica, le scienze e l'economia. Progettati per attrarre e coinvolgere gli studenti, gli esercizi, accuratamente graduati (come evidenziato dagli asterischi, da uno a tre, che ne qualificano la difficoltà), rendono evidente il valore della programmazione nei campi applicativi.

Risorse web

All'indirizzo `http://www.apogeoeducation.com` si trova la pagina web dedicata a questo libro, dove è possibile scaricare il codice sorgente degli esempi utilizzati nel testo.

Una panoramica degli ausili didattici

I differenti elementi pedagogici di questo libro cooperano per focalizzare l'attenzione degli studenti e per rendere più evidenti i concetti chiave e i principi fondamentali

della programmazione, fornendo suggerimenti e dettagli che costituiscono interessanti approfondimenti. Oltre alle caratteristiche tradizionali, come gli obiettivi didattici e un buon numero di esercizi, ogni capitolo contiene svariati elementi che possono attirare l'attenzione e motivare gli studenti, tra i quali citiamo:

- Le **note a margine**, che evidenziano i punti in cui vengono introdotti nuovi concetti: costituiscono uno schema delle idee chiave e sono poi riassunte alla fine di ogni capitolo, suddivise per obiettivi di apprendimento, così da poter essere utilizzate per un rapido ripasso.
- I **riquadri di sintassi** offrono una rapida e sintetica panoramica dei nuovi costrutti del linguaggio, nell'ordine in cui vengono appresi. Al loro interno, molte **annotazioni grafiche** illustrano gli elementi sintattici richiesti e forniscono ulteriori informazioni sugli errori più frequenti e sulle buone pratiche di programmazione relativamente alla sintassi.
- I paragrafi dedicati alla **soluzione di problemi (*problem solving*)** insegnano tecniche che aiutano a individuare soluzioni e a valutare quelle effettivamente interessanti, spesso usando carta e penna o altri oggetti di uso comune. Questi paragrafi intendono mettere in evidenza il fatto che la maggior parte delle azioni di pianificazione e di risoluzione dei problemi che consentono a uno studente di avere successo nella preparazione dell'esame avvengono senza l'utilizzo del calcolatore.
- Molte **tabelle esemplificative** agevolano il compito dei principianti, fornendo esempi concreti, per un veloce ripasso delle caratteristiche salienti del linguaggio e dei relativi errori frequenti.
- Un aiuto analogo viene dalle **figure** che illustrano, in fasi successive, il flusso di esecuzione di parti salienti di un programma, affiancandosi al codice sorgente.
- La **presentazione del codice sorgente** è stata molto curata e fornisce uno strumento di confronto con buone pratiche di programmazione.
- Numerosi **esempi di programmazione grafica**, facoltativa, agevolano la comprensione dei fondamenti della programmazione mediante un approccio visuale.
- Infine, gli **esercizi che hanno per argomento le scienze, l'economia e la grafica** spingono gli studenti a risolvere problemi più attinenti alla realtà.

Vi sono, inoltre, sette diverse tipologie di sezioni speciali inserite nel testo, evidenziate in modo particolare per non interrompere il flusso principale degli argomenti:

- I **Consigli pratici**, ispirati dalle guide "How To" di Linux, hanno lo scopo di rispondere a domande frequenti degli studenti, del tipo "Che cosa devo fare ora?", fornendo istruzioni sull'esecuzione dei compiti più comuni, passo dopo passo, concentrandosi in particolar modo sulla pianificazione dell'attività di programmazione e sul relativo collaudo.
- Gli **Esempi completi** applicano i *Consigli pratici* a un caso diverso, mostrando come quei consigli siano effettivamente assai utili per pianificare, realizzare e collaudare la soluzione di un altro problema di programmazione.
- Gli **Errori comuni** descrivono i tipi di errori compiuti spesso dagli studenti, con una spiegazione del motivo dell'errore e di come rimediarvi.
- I **Suggerimenti per la programmazione** illustrano buone abitudini di programmazione e insegnano agli studenti come usare in modo più efficace gli strumenti e il tempo che hanno a disposizione, spingendoli a essere più produttivi.

- Le sezioni denominate **Computer e società** forniscono informazioni storiche e sociali sull'informatica.
- Gli **Argomenti avanzati** trattano materiale facoltativo o più complesso.
- Gli **Strumenti** insegnano agli studenti come si usano alcune specifiche librerie di Python per risolvere problemi del mondo reale.

Ringraziamenti

I nostri ringraziamenti vanno a Joanna Dingle, Crystal Franks, Graig Domini e Michael MacDougald di John Wiley & Sons, oltre a Vickie Piercey del gruppo Publishing Services, per l'aiuto che hanno dato a questo progetto. Un ringraziamento speciale e profondo va a Cindy Johnson per l'incredibile cura con cui ha lavorato, per i suoi preziosi consigli e per la grande attenzione ai dettagli.

Siamo grati a Ben Stephenson, University of Calgary, per il suo eccellente lavoro nella preparazione e revisione del materiale di supporto.

Dobbiamo, poi, ringraziare le numerosissime persone che hanno rivisto il manoscritto, dando utili consigli e portando alla nostra attenzione errori e omissioni. Tra tutti, citiamo:

William Bulko, *University of Texas, Austin*
John Conery, *University of Oregon*
Lee D. Cornell, *Minnesota State University, Mankato*
Mike Domaratzki, *University of Manitoba*
Rich Enbody, *Michigan State University*
Jackie Horton, *University of Vermont*
Winona Istre, *University of Louisiana, Lafayette*
Swami Iyer, *University of Massachusetts, Boston*
ToniAnn Marini, *North Carolina State University*
Melinda McDaniel, *Georgia Institute of Technology*
Shyamal Mitra, *University of Texas, Austin*
Ben Stephenson, *University of Calgary*
Mehmet Ulema, *Manhattan College*
David Wilkins, *University of Oregon*

Infine, ogni nuova edizione si costruisce sui consigli dei revisori e lettori delle edizioni precedenti, per cui siamo particolarmente grati a: Claude Anderson, *Rose-Hulman Institute of Technology*; Jim Carrier, *Guilford Technical Community College*; Gokeen Cilingir, *Washington State University*; Akshaye Dhawan, *Ursinus College*; Dirk Grunwald, *University of Colorado Boulder*; Andrew Harrington, *Loyola University Chicago*; Byron Hoy, *Stockton University*; Debbie Keen, *University of Kentucky*; Nicholas A. Kraft, *University of Alabama*; Aaron Langille, *Laurentian University*; Maria Laurent-Rice, *Orange Coast College*; John McManus, *Randolph-Macon College*; Chandan R. Rupakheti, *Rose-Hulman Institute of Technology*; John Schneider, *Washington State University*; Amit Singhal, *University od Rochester*; Amanda Stouder, *Rose-Hulman Institute of Technology*; Dave Sullivan, *Boston University*; Jay Summet, *Georgia Institute of Technology*; James Tam, *University od Calgary*; Krishnaprasad Thirunarayan, *Wright State University*; Leon Tietz, *Minnesota State University, Mankato*; Peter Tucker, *Whitworth University*; Frances VanScoy, *West Virginia University*; Dean Zeller, *University of Northern Colorado*.