Laboratorio di Programmazione I

Con il linguaggio Python

ARTIFICIAL INTELLIGENCE
& DATA ANALYTICS

UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

Laura Nenzi, Giulio Caravagna, Stefano Alberto Russo

Corso di Laurea Triennale in Intelligenza Artificiale e Data Analytics (AIDA)

Dipartimento di Matematica, Informatica e Geoscienze
Università di Trieste

Dispense trascritte dalle lezioni grazie ai contributi cumulativi di: Gianluca Guglielmo (stesura iniziale), Michele Rispoli, Pietro Morichetti, Nicolas Solomita, Andrea Mecchina, Elena Buscaroli, Federico Pigozzi, Lucrezia Valeriani e Valentina Blasone.

Versione 1.0.1 (23 febbraio 2026)

[1] La licenza dice grossomodo che, a condizione di citare l'autore e di utilizzare la stessa tipologia di licenza, questi materiali possono essere liberamente copiati, modificati, usati per creare opere derivate e ridistribuiti.

Indice

Preambolo. Questa dispensa costituisce il materiale didattico per il corso di *Laboratorio di Programmazione Modulo 1*, relativo all'anno accademico di riferimento. Il corso è un modulo del corso *Introduzione alla Programmazione e Laboratorio*.

A seconda delle annualità e del corso di laurea, il corso prevede 9, 12 o 15 CFU e si articola in due o tre moduli: uno dedicato alla programmazione (oggetto della dispensa del prof. Caravagna) e uno (o due) al laboratorio (oggetto della presente dispensa).

Questo modulo di laboratorio si concentra interamente su `Python`. Il primo modulo è propedeutico a questo quindi viene assunto che lo studente sia già familiare con i costrutti e gli operatori di base di un linguaggio di programmazione. Verranno inoltre affrontati i fondamenti della programmazione a oggetti e alcuni concetti di programmazione funzionale, entrambi trattati in `Python`.

Gli obiettivi formativi e i materiali didattici possono variare di anno in anno: non sempre l'intera dispensa sarà utilizzata, e talvolta potranno essere forniti contenuti integrativi. In linea generale, il modulo di programmazione mira a introdurre i concetti essenziali del *computational thinking*, con particolare attenzione ai principi di ragionamento e progettazione che restano validi indipendentemente dal linguaggio utilizzato, mentre il laboratorio approfondisce l'uso di `Python`, affrontando tematiche complementari e fornendo competenze pratiche che si affiancano alla solida base concettuale costruita nel modulo di programmazione, utile per affrontare con agilità diversi linguaggi.

L'intelligenza artificiale (IA) scrive codice, ma non capisce il vostro problema. L'IA produce testo plausibile anche se non sapete formalizzare il problema, ed il codice plausibile è pericoloso: compila, ma fa la cosa sbagliata. Senza esercizio non sviluppate il "debug mentale", cioè la capacità di leggere un algoritmo e chiedervi: ha senso? copre i casi limite? cosa succede se l'input è sporco? Programmare non è scrivere righe, è pensare in modo strutturato. Bisogna decidere cosa fare prima di decidere come farlo. Gli esercizi allenano la decomposizione del problema, potete vederla come una ginnastica cognitiva. La verità è che oggi programmare è ancora più importante, non meno. Perché ora dovete saper valutare, correggere, integrare codice generato automaticamente. È un ruolo più maturo: da scrittori a ingegneri del ragionamento. Se vogliamo dirla in modo quasi filosofico: l'IA è un acceleratore di pensiero. Ma se non c'è pensiero, accelera il vuoto. E il vuoto, quando lo acceleri, resta vuoto. Per cui vi consiglio vivamente di provare a fare gli esercizi usando l'IA in maniera limitata.

Libri di testo. Durante la prima lezione e nella pagina del corso sono forniti dei riferimenti testuali aggiornati ogni anno. Gli studenti che desiderino consultare testi di riferimento più strutturati possono rivolgersi al docente per ricevere indicazioni sui titoli più appropriati.

Guida alla dispensa. Questa dispensa si "evolve" di anno in anno, e pertanto a volte potrebbe contenere errori. Dubbi, chiarimenti ed eventuali errori possono essere segnalati via mail a `lnenzi@units.it`. È prevista una forma di ricompensa per la segnalazione di errori (non banali).

Nel testo le spiegazioni sono colloquiali, sacrificando il rigore per la chiarezza espositiva per fornire un accesso semplice e diretto ai concetti base. Esempi di codice sono colorati e discussi. In generale una variabile indicata come x, y, etc. denota una quantità matematica, mentre `x` o `y` una variabile scritta nel programma, quindi codice. Spesso le due coincidono e la notazione non è necessariamente consistente tra le varie sezioni.

PART

I

Strumenti di lavoro

Per affrontare il *Laboratorio di Programmazione in Python* è necessario usare alcuni strumenti, che vediamo brevemente.

File manager Il *File Manager* è un software che consente di gestire e organizzare i file e le cartelle presenti in un sistema operativo. In pratica, è quello strumento che permette all'utente di vedere, modificare, copiare ed eliminare file e cartelle. Esempi sono Windows Explorer (Esplora file) su Windows, il Finder su MacOS e Nautilus o Dolphin sui sistemi Linux.

Shell La *shell* permette di eseguire programmi e navigare il file system attraverso una linea di comando, *Command-Line Interface (CLI)*, ovvero di interagire direttamente con la macchina. Nei sistemi Windows le shell a disposizione sono il *Prompt dei Comandi (CMD)* e la *PowerShell*, mentre nei sistemi Unix si ha *BASH* (in genere accessibile tramite l'applicazione *Terminale*).

Editor del codice L'*Editor del Codice* può essere un qualsiasi programma per la scrittura del testo *semplice*, ovvero che non aggiunga della formattazione come il grassetto, corsivo, paragrafi etc. È possibile usare un editor che interpreti il linguaggio utilizzato per colorare le diverse sintassi e facilitarne la lettura. Per il corso è necessario impostare l'editor in modo che utilizzi 4 spazi al posto dei tab, che serviranno per indentare il codice. Python segue la convenzione PEP 8 (le linee guida ufficiali di stile) che raccomanda 4 spazi per livello di indentazione, un tab può essere interpretato in modo diverso da sistemi diversi.

IDE L'*Integrated Development Environment* è un editor che supporta lo sviluppatore attraverso l'integrazione con sistemi di debugging, File Mangaer, Shell, talvolta Git e svariate altre agevolazioni, le quali rendono la programmazione più scorrevole. Gli IDE più diffusi supportano diversi linguaggi e possono adattare i propri suggerimenti a seconda della sintassi utilizzata. Tra i più utilizzati troviamo Sublime e Code::Blocks.

Visual Studio Code *Visual Studio Code* (VS Code) è un editor di codice estensibile sviluppato da Microsoft. Non è propriamente un IDE tradizionale, bensì un editor leggero che può essere arricchito tramite estensioni per supportare numerosi linguaggi di programmazione. Integra un terminale,

strumenti di debugging e il supporto al versionamento tramite *Git*. La sua architettura modulare consente di aggiungere funzionalità come completamento automatico, analisi statica del codice e gestione di ambienti virtuali. Grazie al sistema di estensioni e al *Language Server Protocol*, VS Code può assumere il comportamento di un vero e proprio ambiente di sviluppo integrato.

Git Il software per il versionamento *Git* è uno strumento per tenere traccia delle modifiche fatte al codice, può essere sincronizzato con dei collaboratori ed è decentralizzato. Le *Repository* (repo) sono le cartelle dei progetti e possono contenere sia codice che file di altro tipo. L'operazione base è il *commit*, che crea un "salvataggio" del codice in Git permettendo di avere dei checkpoint, univocamente identificati da un cosiddetto *hash*. È possibile trovare qui una guida su Git scritta da Michele Rispoli.

GitHub *GitHub* è una piattaforma online che ospita repository *Git* e facilita la collaborazione tra sviluppatori. Permette di condividere progetti, gestire modifiche tramite *pull request*, discutere problemi attraverso le *issue* e integrare sistemi di automazione per test e distribuzione del software. Oltre all'hosting del codice, offre strumenti per la documentazione, la revisione collaborativa e la gestione di progetti.

GitHub Codespaces *GitHub Codespaces* è un servizio cloud che consente di creare ambienti di sviluppo remoti direttamente collegati a una repository GitHub. L'ambiente viene eseguito su una macchina virtuale nel cloud e può essere utilizzato tramite browser oppure collegandosi con *Visual Studio Code*. In questo modo lo sviluppo non dipende dalle risorse locali del computer, ma da un ambiente configurato e riproducibile, accessibile da qualsiasi dispositivo.

Ambiente virtuale Un *ambiente virtuale* è uno spazio isolato che contiene una specifica versione di Python e le librerie necessarie a un determinato progetto. Ogni progetto può avere il proprio ambiente, indipendente dagli altri e dal sistema operativo. Questo isolamento è fondamentale per evitare conflitti tra versioni diverse delle stesse librerie o tra progetti che richiedono configurazioni differenti.

Nel corso utilizzeremo *Miniconda* per creare e gestire ambienti virtuali. In particolare, verrà creato un ambiente dedicato al laboratorio, all'interno del quale verranno installate solo le librerie necessarie. Quando un ambiente è attivo, il terminale mostra il suo nome tra parentesi all'inizio della riga di comando: ciò indica che i comandi Python e le installazioni di librerie verranno eseguiti all'interno di quell'ambiente.

È buona pratica non lavorare mai nell'ambiente `base`, ma creare un ambiente specifico per ciascun progetto.

Pseudocodice Un ultimo strumento concettuale è costituito dal cosiddetto *pseudocodice*. Prima di scrivere il codice vero e proprio, in qualsiasi linguaggio di programmazione, è utile scriverne una bozza, anche a carta e penna, focalizzandosi non sul come, ma sul *cosa* fare. Non c'è uno standard con cui scrivere questa bozza: lo pseducodice è del tutto inventato, usando parole anche in linguaggio naturale (italiane o inglesi) che permettano di trasporre il problema che si cerca di risolvere in logica di programmazione imperativa ma senza, appunto, preoccuparsi di dettagli come la sintassi, i tipi dati etc. per non interrompere il flusso di ragionamento logico.