PROGRAMMA DEL CORSO DI PROGRAMMAZIONE

SETTORE SCIENTIFICO

INF/01

CFU

12

OBIETTIVI

L’obiettivo del corso è insegnare agli studenti le basi della programmazione attraverso un linguaggio versatile e di facile apprendimento, ossia Python. Si focalizza sullo sviluppo di competenze logiche e algoritmiche, fornendo agli studenti gli strumenti per risolvere problemi complessi in modo efficiente. Inoltre, il corso mira a familiarizzare gli studenti con le buone pratiche di programmazione, come la gestione di dati, la creazione di script modulari e la comprensione della struttura del software.

Obiettivi formativi specifici sono i seguenti:

Acquisire competenze di base nella programmazione: imparare la sintassi di Python, le strutture dati (liste, tuple, dizionari, set), le strutture di controllo come cicli (for, while) e condizionali (if-else) per creare programmi dinamici, e i concetti fondamentali come variabili, tipi di dato e operatori. Sviluppare capacità logiche e algoritmiche: risolvere problemi con approcci algoritmici, sviluppando soluzioni efficienti tramite il linguaggio Python e adottando buone pratiche di programmazione. Gestire la modularità del codice: utilizzare funzioni e moduli per organizzare il codice in modo chiaro e riutilizzabile. Lavorare con file e input/output: acquisire competenze nell'interazione con il sistema tramite lettura e scrittura di file, gestione dell'input dell'utente e manipolazione di dati esterni. Gestire errori e eccezioni: sviluppare la capacità di identificare e gestire gli errori nel codice attraverso il controllo delle eccezioni. Comprendere concetti avanzati: introdurre concetti più avanzati come la programmazione orientata agli oggetti (OOP), l'uso di librerie standard e la gestione di pacchetti.

RISULTATI DI APPRENDIMENTO ATTESI

Conoscenza e capacità di comprensione

Il corso intende fornire le conoscenze utili per una solida comprensione della sintassi di Python e dei concetti fondamentali di programmazione, incluse le strutture dati, i controlli di flusso e le funzioni. Saranno in grado di comprendere come applicare queste conoscenze per risolvere problemi pratici e sviluppare soluzioni efficienti. Inoltre, avranno una buona comprensione delle librerie Python più comuni e delle tecniche di programmazione orientata agli oggetti.

Capacità di applicare conoscenza e comprensione

Il corso intende fornire agli studenti le capacità necessarie per scrivere programmi utilizzando Python per risolvere problemi complessi, implementare algoritmi efficienti e manipolare dati. Gli studenti saranno capaci di utilizzare le librerie Python per l'analisi dei dati e la visualizzazione e saranno capaci di progettare e implementare soluzioni che rispettano i principi della programmazione strutturata e della modularità.

Autonomia di giudizio

Il corso intende fornire agli studenti le capacità di analizzare criticamente diverse soluzioni per un problema e di scegliere quella più adatta in base ai vincoli di tempo, risorse e performance. Saranno in grado di valutare la qualità del codice, identificare errori e inefficienze, e prendere decisioni informate riguardo alla scelta delle librerie e degli approcci più appropriati per il loro progetto.

Abilità comunicative

Il corso intende fornire agli studenti la capacità di spiegare in modo chiaro e conciso le soluzioni programmatiche adottate, sia in forma scritta, attraverso la documentazione del codice che verbale. Saranno in grado di comunicare efficacemente con altri sviluppatori, spiegando il funzionamento del codice, giustificando le scelte progettuali e presentando i risultati ottenuti.

Capacità di apprendimento

Gli studenti saranno in grado di apprendere in modo autonomo e continuo nuove tecniche, librerie e strumenti Python, adattandosi rapidamente alle evoluzioni del linguaggio e delle tecnologie. Sapranno sfruttare risorse online, documentazioni ufficiali e comunità di sviluppatori per risolvere problemi e approfondire tematiche avanzate legate alla programmazione Python.

RISORSE

/**/
Testo consigliato (per approfondimento volontario):

Gaddis, T. (2022). Introduzione a Python -5/ED. Ediz. mylab. Pearson spa.

VERIFICA

L’esame può essere sostenuto sia in forma scritta che in forma orale.

Gli appelli orali sono previsti nella sola sede centrale. L’esame orale consiste in un colloquio con la Commissione sui contenuti del corso. L’esame scritto consiste nello svolgimento di un test con 30 domande. Per ogni domanda lo studente deve scegliere una di 4 possibili risposte. Solo una risposta è corretta.

Sia le domande orali che le domande scritte sono formulate per valutare il grado di comprensione delle nozioni teoriche e la capacità di ragionare utilizzando tali nozioni. Le domande sulle nozioni teoriche consentiranno di valutare il livello di comprensione. Le domande che richiedono l’elaborazione di un ragionamento consentiranno di valutare il livello di competenza e l’autonomia di giudizio maturati dallo studente.

Le abilità di comunicazione e le capacità di apprendimento saranno valutate ance attraverso le interazioni dirette tra docente e studente che avranno luogo durante la fruizione del corso (videoconferenze ed elaborati proposti dal

docente).

Obbligatoria online.

Ai corsisti viene richiesto di visionare almeno l'80% delle videolezioni presenti in piattaforma.

DESCRIZIONE

/**/
Programma del corso: elenco dei moduli

MODULO 1: Computer e Programmazione

- Lezione 1: Computer e Programmazione
- Lezione 2: Memorizzazione dei dati
- Lezione 3: Funzionamento dei programmi
- Lezione 4: Approfondimento sui linguaggi di programmazione
- Lezione 5: Introduzione a Python

MODULO 2: Progettazione di un programma

- Lezione 6: Progettazione di un programma
- Lezione 7: Ingegneria del Software per la progettazione di programmi
- Lezione 8: Problemi, algoritmi, ed esecutori
- Lezione 9: La descrizione degli algoritmi
- Lezione 10: Diagrammi di flusso
- Lezione 11: Diagrammi di flusso: i Blocchi di iterazione

MODULO 3: Input, elaborazione ed output

- Lezione 12: Output nei programmi Python
- Lezione 13: Variabili e tipi di dati
- Lezione 14: Input, output testuale e commenti nei programmi Python
- Lezione 15: Operatori matematici e costanti con nome in Python
- Lezione 16: Output formattato con le f-string

MODULO 4: Strutture decisionali e logica booleana

- Lezione 17: Strutture decisionali ad alternativa singola
- Lezione 18: Strutture decisionali ad alternativa doppia e nidificate
- Lezione 19: Operatori logici e variabili booleane in Python
- Lezione 20: Esercitazione strutture decisionali

MODULO 5: Strutture iterative

Lezione 21: Strutture iterative: cicli while e for
Lezione 22: Accumulatori, sentinelle, cicli di convalida e nidificati
Lezione 23: Esercitazione strutture iterative semplici
Lezione 24: Esercitazione avanzata di strutture iterative
Lezione 25: Esercitazione strutture iterative e calcoli orari

MODULO 6: Funzioni

Lezione 26: Funzioni void
Lezione 27: Variabili locali e passaggio di argomenti in funzioni
Lezione 28: Funzioni produttive, variabili e costanti globali
Lezione 29: Librerie e moduli
Lezione 30: Memorizzazione delle funzioni nei moduli
Lezione 31: Esercitazione sulle funzioni

MODULO 7: File ed Eccezioni

Lezione 32: Input e output dei file
Lezione 33: Elaborazione dei file e dei record
Lezione 34: Gestione delle eccezioni

MODULO 8: Liste e Tuple

Lezione 35: Liste
Lezione 36: Slicing, ricerca e metodi delle liste
Lezione 37: Copia ed elaborazione di liste
Lezione 38: Comprensioni di liste, liste bidimensionali e tuple
Lezione 39: Plottaggio dati di liste
Lezione 40: Esercitazione sulle liste per calcoli di tempo
Lezione 41: Esercitazione avanzata sulle liste per calcoli di tempo
Lezione 42: Esercitazione sulle operazioni di inserimento ed elaborazione di liste
Lezione 43: Esercitazione sulle matrici e sulla ricerca binaria nelle liste
Lezione 44: Esercitazione avanzata sulle matrici

MODULO 9: Stringhe

Lezione 45: Operazioni di base sulle stringhe
Lezione 46: Test, ricerca e manipolazione di stringhe
Lezione 47: Esercitazione sulle stringhe

MODULO 10: Dizionari e set

Lezione 48: Dizionari
Lezione 49: Set
Lezione 50: Comprensioni e serializzazione di oggetti

MODULO 11: Classi e programmazione ad oggetti

- Lezione 51: Programmazione a oggetti e classi
- Lezione 52: Accesso, modifica e passaggio a funzione di oggetti
- Lezione 53: Tecniche per la progettazione di classi
- Lezione 54: Esercitazione su classi e oggetti
- Lezione 55: Esercitazione avanzata su classi

MODULO 12: Ereditarietà

- Lezione 56: Introduzione all'ereditarietà
- Lezione 57: Polimorfismo
- Lezione 58: Esercitazione su ereditarietà e polimorfismo

MODULO 13: Ricorsione

- Lezione 59: Introduzione alla ricorsione
- Lezione 60: Algoritmi ricorsivi