LICEO B. RUSSELL

Via IV Novembre 35 – 38023 Cles

PROGRAMMAZIONE DIDATTICA

Informatica

Classe **2C**

Liceo Scientifico – Opzione Scienze Applicate

DOCENTE
Prof. Mattia Marini

ANNO SCOLASTICO
2026/27

ORE ANNUE
66

ORE A CONSUNTIVO
/

COMPETENZE DA RAGGIUNGERE AL TERMINE DELL'ANNO SCOLASTICO	• Saper analizzare un problema e formulare una soluzione algoritmica, scomponendolo in istruzioni elementari e strutture di controllo. • Saper utilizzare un foglio di calcolo e le sue funzionalità avanzate. • Saper sfruttare efficientemente gli strumenti offerti dagli ambienti di sviluppo integrati (IDE), in particolare Visual Studio Code. • Avere confidenza con le strutture dati di base: liste, stringhe e matrici. • Saper utilizzare e riconoscere le strutture dati più avanzate: set e dizionari. • Saper utilizzare le funzioni di base della libreria standard di Python. • Saper strutturare un progetto Python in funzioni e moduli, con particolare attenzione alla chiarezza e alla modularità del codice.
METODOLOGIE DIDATTICHE	• Lezione frontale e dialogata, con coinvolgimento attivo della classe. • Esercitazioni in laboratorio, individuali o a coppie. • Problem solving guidato, con analisi collettiva degli errori. • Lavori di gruppo basati su peer learning. • Assegnazione e sviluppo di progetti da svolgere a casa. • Simulazioni delle prove di verifica. • Attività di recupero e consolidamento per gli studenti che incontrano difficoltà.
STRUMENTI E CRITERI DI VALUTAZIONE	• Verifiche scritte, pratiche e orali. • Sviluppo di progetti individuali o di gruppo. • Partecipazione attiva e qualità degli interventi. • Progressi rispetto ai livelli di partenza. • Capacità di analisi e di risoluzione dei problemi. • Correttezza, chiarezza e completezza del codice prodotto.
TESTI, MATERIALI E STRUMENTI ADOTTATI	• Ambiente di sviluppo integrato (IDE): Visual Studio Code. • LIM (Lavagna Interattiva Multimediale). • Presentazioni multimediali interattive, esercizi e appunti a cura del docente. • Sito web del docente. • Laboratorio di informatica con PC e connessione a Internet.

Modulo 1 – Ripasso dei concetti di programmazione in Python

Ore previste: 10

CONOSCENZE E/O CONTENUTI	• Concetto di variabile e di tipo di dato. • Istruzioni di input e output: `input()` e `print()`. • Costrutti di controllo condizionali: `if`, `elif`, `else`. • Istruzioni di iterazione: `while` e `for` (iterazione su strutture dati). • Strutture dati di base: liste e stringhe. • Funzioni base della libreria standard di Python: `len()`, `max()`, `min()`, `sum()`, `sorted()`, `reversed()`, `enumerate()`. • Algoritmi e pattern noti.
ABILITÀ E COMPETENZE	• Saper scrivere semplici programmi in Python, utilizzando correttamente variabili, tipi di dato, operatori e strutture condizionali.

	• Avere confidenza con i concetti di iterazione e con le strutture dati di base, sapendo utilizzare tali elementi in semplici programmi Python. • Avere un buon vocabolario di algoritmi e pattern noti, sapendo riconoscerli e riadattarli a problemi analoghi.

Modulo 2 – Concetti più avanzati di programmazione in Python

Ore previste: 15

CONOSCENZE E/O CONTENUTI	• Concetto di funzione, parametri e valori di ritorno; passaggio di parametri per valore e per riferimento. • Annotazioni di tipo e docstring. • Organizzazione del codice in moduli e buone pratiche. • Utilizzo di costrutti sintattici compatti (syntactic sugar), come list comprehension e f-string. • Utilizzo delle funzioni della libreria standard di Python, in particolare quelle per la manipolazione di stringhe e liste.
ABILITÀ E COMPETENZE	• Saper definire e utilizzare opportunamente le funzioni e i moduli. • Sviluppare codice Python in modo chiaro e modulare, suddividendo il programma in funzioni e moduli. • Sviluppare in autonomia programmi completi, che vadano oltre singoli brevi esercizi.

Modulo 3 – Strutture dati: set, dizionari e matrici

Ore previste: 10

CONOSCENZE E/O CONTENUTI	• set: casi di utilizzo, operazioni di base e algoritmi tipici. • dizionari: casi di utilizzo, operazioni di base e algoritmi tipici. • matrici: casi di utilizzo, operazioni di base e algoritmi tipici.
ABILITÀ E COMPETENZE	• Comprendere le caratteristiche di ciascuna struttura dati e i rispettivi casi di utilizzo. • Utilizzare e comprendere le operazioni di base e gli algoritmi tipici per ciascuna struttura dati. Risolvere problemi di programmazione utilizzando la struttura dati più appropriata.

Modulo 4 – Algoritmi

Ore previste: 15

CONOSCENZE E/O CONTENUTI	• Stima del costo di un algoritmo in termini di memoria e tempo di esecuzione. Classi di complessità computazionale e notazione Big O. • Risoluzione di problemi tramite algoritmi ricorsivi: concetto di ricorsione e di stack di esecuzione. Individuazione di caso base e passo ricorsivo. **ALGORITMI DI ORDINAMENTO** • `bubble sort` • `insertion sort` • `selection sort` • `merge sort`

	ALGORITMI RICORSIVI • `ricerca binaria` • `fibonacci` • `fattoriale` • `torre di Hanoi`
ABILITÀ E COMPETENZE	• Saper ordinare un vettore. • Saper stimare il numero di operazioni elementari necessarie per eseguire un algoritmo, comprendendo l'impatto della complessità computazionale sul tempo di esecuzione. • Saper sviluppare algoritmi ricorsivi, individuando correttamente il caso base e il passo ricorsivo. • Saper approcciare la risoluzione di problemi complessi con consapevolezza della complessità computazionale e delle strutture dati più appropriate da utilizzare.