



CURRICOLO DI INFORMATICA

LICEO SCIENTIFICO - OPZIONE DELLE SCIENZE APPLICATE SECONDO BIENNIO

Elenco dei moduli

1. Funzioni in Python	1
2. Gestione di stringhe e file	2
3. Tipi di dato composti	3
4. Ricerca, ordinamento e ricorsione	4
5. Il web e i suoi linguaggi	5
6. Architettura di applicazioni e interfacce grafiche	6
7. Introduzione alla programmazione orientata agli oggetti	7
8. Strutture dati astratte	8
9. Database e modello concettuale entità-relazione	9
10. Progettazione logica e modello relazionale	10
11. SQL e algebra relazionale	10
12. Sviluppo di applicazioni	11
13. Gestione dell'ambiente scolastico digitale (comune a tutti gli anni)	12

1. Funzioni in Python

Obiettivi didattici

In questo primo modulo del secondo biennio, gli studenti imparano a organizzare il codice in modo ordinato ed efficace attraverso l'uso delle funzioni. L'obiettivo è comprendere come un problema complesso possa essere scomposto in parti più semplici e gestibili, sviluppando quindi il pensiero logico e progettuale e avvicinando gli studenti a un approccio modulare alla soluzione dei problemi.

Parallelamente, si introduce il Test-Driven Development (TDD), cioè la pratica di scrivere piccoli test prima del codice, allo scopo di favorire un approccio preventivo agli errori e una maggiore consapevolezza della qualità del codice, stimolando il ragionamento critico e l'autovalutazione.



Competenze attese

Scomporre un programma in funzioni

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Struttura e sintassi delle funzioni in Python • Parametri, argomenti, valori di ritorno • Variabili locali e globali • Il ruolo del call stack 	<ul style="list-style-type: none"> • Definire e richiamare funzioni • Usare parametri e valori di ritorno • Scrivere codice privo di conflitti di nomi e con comportamenti prevedibili 	<ul style="list-style-type: none"> • Abitudine a pianificare prima di scrivere codice • Propensione a cercare soluzioni modulari e riutilizzabili

Applicare i principi base del Test-Driven Development (TDD)

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Il paradigma legato al TDD • Ruolo dei test automatici nello sviluppo del software 	<ul style="list-style-type: none"> • Scrivere semplici test prima del codice • Verificare automaticamente il corretto funzionamento delle funzioni 	<ul style="list-style-type: none"> • Mentalità preventiva e proattiva nella risoluzione dei problemi • Capacità di accettare e correggere gli errori come parte del processo di apprendimento • Interesse per la qualità e l'affidabilità del lavoro svolto

Riferimenti

<i>Indicazioni nazionali</i>	<i>DigiComp 2.2</i>	<i>Competenze chiave</i>
12.1, 12.2	3.d	C, D

2. Gestione di stringhe e file

Obiettivi didattici

In questo modulo gli studenti proseguono il percorso di crescita nella programmazione, sviluppando la capacità di ragionare in modo analitico e di tradurre problemi complessi in sequenze di istruzioni.

L'analisi e la trasformazione di stringhe è un'occasione per affinare la precisione logica e la creatività nell'individuare le soluzioni ai problemi. La gestione dei file e l'analisi automatica di testi introducono lo studente alla dimensione del dato come risorsa da cui estrapolare informazioni. Quest'ultimo argomento sarà anche utilizzato per introdurre la gestione degli errori.

Competenze attese

Analizzare e trasformare stringhe in Python

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Struttura e caratteristiche delle stringhe • Operatori e metodi principali 	<ul style="list-style-type: none"> • Estrarre e riorganizzare parti di testo • Applicare metodi predefiniti per analizzare contenuti testuali 	<ul style="list-style-type: none"> • Precisione nell'elaborazione di dati testuali • Curiosità nel riconoscere e sfruttare schemi ricorrenti



Gestire file di testo

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Operazione di base sui file • Concetto di encoding 	<ul style="list-style-type: none"> • Aprire, leggere, elaborare e salvare dati testuali • Interpretare correttamente la struttura di un file 	<ul style="list-style-type: none"> • Cura nell'integrità e correttezza dei dati • Consapevolezza del valore dei dati come risorsa da gestire

Gestire gli errori nella programmazione

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Tipologie di errori di programmazione • Strategie di gestione degli errori 	<ul style="list-style-type: none"> • Scrivere codice robusto che gestisca errori in fase di esecuzione 	<ul style="list-style-type: none"> • Visione della gestione dell'errore come parte integrante del processo di programmazione • Spirito critico nel valutare possibili cause e conseguenze di un malfunzionamento

Riferimenti

<i>Indicazioni nazionali</i>	<i>DigiComp 2.2</i>	<i>Competenze chiave</i>
12.1, 12.2	3.d	C, D

3. Tipi di dato composti

Obiettivi didattici

In questo modulo, gli studenti approfondiscono la conoscenza dei tipi di dato in Python, distinguendo tra quelli semplici e quelli composti e imparando ad utilizzare le strutture dati per rappresentare informazioni in modo efficace e come strumento per risolvere problemi sempre più complessi.

Competenze attese

Utilizzare i tipi di dato composti

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Differenza tra tipi di dato semplici e composti • Caratteristiche di ciascun tipo di dato composto • Operazioni fondamentali sui tipi di dato composti 	<ul style="list-style-type: none"> • Creare e gestire collezioni di dati con diverse strutture • Effettuare operazioni di trasformazione e filtraggio 	<ul style="list-style-type: none"> • Sviluppare un approccio riflessivo alle scelte progettuali

Integrare conoscenze e abilità nella risoluzione di problemi reali

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
	<ul style="list-style-type: none"> • Identificare il tipo di dato più adatto per risolvere un problema • Combinare tipi di dati per risolvere problemi concreti 	<ul style="list-style-type: none"> • Autonomia nella progettazione delle soluzioni • Responsabilità nelle scelte progettuali



Riferimenti

<i>Indicazioni nazionali</i>	<i>DigiComp 2.2</i>	<i>Competenze chiave</i>
12.1, 12.2	3.d	C, D

4. Ricerca, ordinamento e ricorsione

Obiettivi didattici

Questo modulo accompagna gli studenti a un livello più avanzato di programmazione, in cui non ci si limita a scrivere codice funzionante, ma si inizia a riflettere sulla struttura e l'efficienza degli algoritmi.

Lo studio degli algoritmi di ricerca consente di comprendere come individuare informazioni all'interno di insiemi di dati, l'analisi degli algoritmi di ordinamento mette in evidenza il ruolo dell'organizzazione dei dati stimolando la capacità di valutare soluzioni diverse, la comprensione della ricorsione rappresenta un esercizio di pensiero astratto in cui la soluzione passa per la ripetizione di uno stesso schema logico.

Competenze attese

Comprendere e applicare algoritmi di ricerca

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> Principi della ricerca lineare e dicotomica Concetto di complessità computazionale in termini qualitativi 	<ul style="list-style-type: none"> Implementare semplici algoritmi di ricerca Confrontare soluzioni diverse in termini di efficienza 	<ul style="list-style-type: none"> Apertura alla sperimentazione di più metodi

Comprendere e applicare algoritmi di ordinamento

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> Funzionamento degli algoritmi di ordinamento La complessità computazionale degli algoritmi di ordinamento 	<ul style="list-style-type: none"> Implementare algoritmi di ordinamento Analizzare in modo qualitativo la complessità e le prestazioni 	<ul style="list-style-type: none"> Apertura alla sperimentazione di più metodi Curiosità verso l'analisi comparativa delle soluzioni

Comprendere e utilizzare la ricorsione

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> Definizione e struttura di una funzione ricorsiva Concetto di caso base e passo ricorsivo Confronto tra approccio ricorsivo e iterativo 	<ul style="list-style-type: none"> Risolvere problemi semplici utilizzando la ricorsione Convertire un algoritmo ricorsivo in iterativo e viceversa Valutare vantaggi e limiti della ricorsione 	<ul style="list-style-type: none"> Capacità di vedere problemi da più punti di vista Flessibilità nell'adottare strategie diverse in base al contesto

Riferimenti

<i>Indicazioni nazionali</i>	<i>DigiComp 2.2</i>	<i>Competenze chiave</i>
12.1, 12.2	3.d	C, D



5. Il web e i suoi linguaggi

Obiettivi didattici

In questo modulo gli studenti sono guidati alla comprensione del concetto di documento digitale nelle sue diverse dimensioni, imparando a distinguere tra contenuto, metadati, struttura, modalità di presentazione ed eventualmente interazione. L'introduzione graduale ai linguaggi di marcatura favorisce lo sviluppo di una scrittura strutturata e ordinata, leggibile sia da persone sia da macchine. L'uso dei fogli di stile CSS permette di cogliere la separazione tra contenuto e presentazione, stimolando la capacità di pensare in termini più astratti e modulari.

Parallelamente, la storia di Internet e del World Wide Web consente di collocare le tecnologie digitali in un percorso evolutivo, offrendo una prospettiva critica sul loro impatto sociale, culturale ed economico. La creazione di semplici pagine web statiche rappresenta un'occasione per mettere in pratica le nozioni teoriche apprese, incoraggiando al tempo stesso creatività, rigore e capacità di progettazione.

Competenze attese

Comprendere e analizzare la natura di un documento digitale

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Concetto di documento, supporto e formato • Distinzione tra dati e metadati • Caratteristiche generali dei linguaggi di marcatura • Sintassi di base del linguaggio Markdown 	<ul style="list-style-type: none"> • Riconoscere e descrivere i diversi elementi di un documento digitale • Strutturare un documento in modo chiaro e coerente • Utilizzare il linguaggio Markdown 	<ul style="list-style-type: none"> • Propensione all'ordine e alla chiarezza comunicativa • Sensibilità verso l'accessibilità e l'inclusione • Cura nell'organizzazione delle informazioni

Comprendere la natura del World Wide Web

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Principali tappe storiche dello sviluppo di Internet e del Web • Linguaggi di base del web: HTML, CSS, JavaScript • Differenza tra siti statici e dinamici 	<ul style="list-style-type: none"> • Collocare le tecnologie di Internet e del Web in un contesto storico e sociale • Scrivere pagine web statiche • Organizzare i file sorgente di un sito in un progetto coerente 	<ul style="list-style-type: none"> • Consapevolezza critica del ruolo delle tecnologie digitali • Propensione al lavoro ordinato e modulare

Riferimenti

<i>Indicazioni nazionali</i>	<i>DigiComp 2.2</i>	<i>Competenze chiave</i>
5.3, 6, 7, 10	1.c, 3.a, 5.b	C, D, F



6. Architettura di applicazioni e interfacce grafiche

Obiettivi didattici

In questo modulo gli studenti ampliano la propria prospettiva sullo sviluppo software, passando dalla realizzazione di semplici programmi o funzioni isolate alla progettazione di applicazioni più strutturate e complete. Viene introdotto il concetto di architettura di un'applicazione, che comprende l'organizzazione delle diverse componenti, i loro ruoli specifici e le modalità con cui interagiscono. Questo passaggio aiuta a comprendere come i sistemi informatici non siano costituiti soltanto da codice, ma anche da un insieme di scelte progettuali che incidono direttamente sulla qualità e sull'efficacia delle soluzioni sviluppate.

Parallelamente, l'attenzione alle interfacce grafiche permette di riflettere sul ruolo dell'usabilità, dell'accessibilità e della chiarezza visiva come elementi centrali nella comunicazione uomo-macchina, mentre la panoramica sulle applicazioni web permette di applicare i concetti architetturali ad una delle categorie di software più diffuse.

Complessivamente, il modulo offre un'occasione per coniugare logica e creatività, stimolando lo studente a ragionare come progettista e a considerare lo sviluppo del software come un processo complesso che integra competenze tecniche, capacità organizzative e attenzione al destinatario finale.

Competenze attese

Comprendere l'architettura di un'applicazione

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Concetto di architettura software • Differenza tra applicazioni locali e client-server • Flussi di dati e interazione tra componenti 	<ul style="list-style-type: none"> • Rappresentare in forma schematica l'architettura di un'applicazione • Suddividere un progetto in moduli coerenti 	<ul style="list-style-type: none"> • Propensione a ragionare in termini sistemici • Apertura al confronto tra soluzioni diverse

Conoscere le interfacce grafiche per le applicazioni

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Principi base di progettazione delle GUI • Concetti di usabilità e accessibilità 	<ul style="list-style-type: none"> • Realizzare finestre ed elementi grafici • Collegare componenti grafici alla logica del programma 	<ul style="list-style-type: none"> • Attenzione all'esperienza dell'utente • Cura nella presentazione visiva delle informazioni

Conoscere le applicazioni web

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Struttura base di un'applicazione web • Esempi di tecnologie di base per un'applicazione web 	<ul style="list-style-type: none"> • Comprendere il flusso di dati tra client e server 	<ul style="list-style-type: none"> • Curiosità verso i sistemi distribuiti • Consapevolezza delle implicazioni di sicurezza e privacy



Pensare e lavorare come un progettista

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Metodologie di progettazione del software 	<ul style="list-style-type: none"> • Formalizzare requisiti e vincoli • Organizzare il lavoro in fasi e monitorarne l'avanzamento 	<ul style="list-style-type: none"> • Approccio sistematico alla risoluzione dei problemi

Riferimenti

<i>Indicazioni nazionali</i>	<i>DigiComp 2.2</i>	<i>Competenze chiave</i>
10, 12.1, 12.2	3.d	C, D

7. Introduzione alla programmazione orientata agli oggetti

Obiettivi didattici

Questo modulo introduce il paradigma di sviluppo della programmazione orientata agli oggetti (OOP, Object-Oriented Programming). Dopo aver lavorato secondo la logica procedurale, gli studenti imparano a organizzare il pensiero e il codice in modo diverso, riconoscendo negli oggetti e nelle classi gli elementi fondamentali con cui descrivere la realtà. Questo passaggio non rappresenta soltanto un cambiamento tecnico, ma anche un esercizio di formalizzazione concettuale che li porta a individuare gli aspetti essenziali di un problema e a modellare fenomeni complessi attraverso entità astratte e relazioni tra di esse.

L'introduzione dei diagrammi UML (Unified Modeling Language) offre inoltre la possibilità di tradurre le loro idee in rappresentazioni visive condivisibili, sviluppando la capacità di comunicare modelli e soluzioni. In questo modo si rafforza la competenza nel trasformare intuizioni e progetti in schemi chiari, comprensibili e formalizzati. L'esperienza favorisce lo sviluppo del pensiero critico e sistemico e consolida la consapevolezza che la programmazione è al tempo stesso logica, creatività e metodo.

Competenze attese

Progettare sistemi secondo i principi della programmazione a oggetti

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Concetti di oggetto, classe e messaggio • Principi fondamentali della OOP • Sintassi e semantica della OOP in Python • Associazione tra classi 	<ul style="list-style-type: none"> • Scrivere programmi Python definendo classi e istanziando oggetti • Applicare i principi fondamentali della OOP • Progettare modelli che riflettono relazioni tra entità complesse 	<ul style="list-style-type: none"> • Capacità di analizzare un problema da prospettive diverse • Apertura a nuovi modelli di progettazione • Attitudine a integrare logica e creatività nel processo di sviluppo

Rappresentare classi e oggetti con i diagrammi UML

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Introduzione al linguaggio UML 	<ul style="list-style-type: none"> • Disegnare diagrammi UML di classi e oggetti con relazioni appropriate 	<ul style="list-style-type: none"> • Cura della chiarezza e leggibilità delle rappresentazioni grafiche



<ul style="list-style-type: none"> • Diagramma delle classi e degli oggetti • Rappresentazione delle associazioni 	<ul style="list-style-type: none"> • Tradurre un modello UML in codice coerente in Python 	<ul style="list-style-type: none"> • Consapevolezza del valore della documentazione come parte integrante della progettazione • Attitudine a comunicare soluzioni tecniche anche in forma visiva
---	--	--

Riferimenti

<i>Indicazioni nazionali</i>	<i>DigiComp 2.2</i>	<i>Competenze chiave</i>
12.1, 12.2, 12.3	3.d	C, D

8. Strutture dati astratte

Obiettivi didattici

In questo modulo gli studenti consolidano le conoscenze acquisite con la programmazione orientata agli oggetti e vengono introdotti al concetto di struttura dati astratta, imparando a distinguere tra interfaccia e implementazione e comprendendo che una stessa idea di struttura può essere realizzata in modi diversi a seconda del contesto e delle esigenze.

Il percorso rafforza la capacità di astrazione e formalizzazione, favorendo un approccio al problema che non si limita alla codifica immediata, ma parte da una progettazione concettuale. Lo studente impara a scegliere e modellare strutture dati adatte a situazioni specifiche, sviluppando pensiero critico nella valutazione delle soluzioni più efficienti e appropriate.

Competenze attese

Progettare e formalizzare tipi di dati astratti

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Definizione di tipo di dato astratto • Distinzione tra interfaccia e implementazione 	<ul style="list-style-type: none"> • Definire interfacce chiare e documentate • Tradurre un modello concettuale in una classe con attributi e metodi appropriati • Scegliere l'implementazione più adeguata in base al contesto 	<ul style="list-style-type: none"> • Precisione nella definizione dei contratti d'uso • Attenzione alla chiarezza nella progettazione • Capacità di ragionare in modo astratto e sistemico

Implementare tipi di dati astratti

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Struttura e funzionamento di alcune tipologie di strutture dati astratte • Possibili varianti di implementazione e relative proprietà 	<ul style="list-style-type: none"> • Implementare in Python alcune strutture dati astratte • Analizzare vantaggi e limiti di ciascuna implementazione • Stimare in modo qualitativo la complessità delle operazioni 	<ul style="list-style-type: none"> • Disponibilità a sperimentare soluzioni alternative • Attitudine a scomporre problemi complessi in parti elementari • Curiosità verso l'analisi delle prestazioni e la ricerca di efficienza



Riferimenti

<i>Indicazioni nazionali</i>	<i>DigiComp 2.2</i>	<i>Competenze chiave</i>
12.1, 12.2, 12.3	3.d	C, D

9. Database e modello concettuale entità-relazione

Obiettivi didattici

Questo modulo accompagna gli studenti nella comprensione dell'importanza dei dati e della necessità di gestirli in modo corretto e strutturato. Partendo dall'analisi di archivi semplici, come i file CSV, si analizzano le difficoltà che emergono quando la complessità cresce e si introduce la necessità di strumenti più avanzati come i database.

Il cuore del modulo è rappresentato dal modello concettuale entità-relazione (ER), che permette di descrivere la realtà attraverso concetti astratti e formalizzati. Gli studenti apprendono a individuare entità, attributi e relazioni, a rappresentarli graficamente e a costruire gerarchie di entità basate sulla generalizzazione e la specializzazione. In questo modo esercitano la capacità di astrazione, sviluppano precisione nel descrivere sistemi complessi e acquisiscono un metodo di lavoro che integra logica, chiarezza comunicativa e attenzione alla coerenza.

Competenze attese

Conoscere i database come strumento di gestione dei dati

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> Definizione e funzioni di un database Principali tipologie di database e loro caratteristiche 	<ul style="list-style-type: none"> Identificare la tipologia di database più adatta a diversi contesti 	<ul style="list-style-type: none"> Spirito critico nella valutazione delle tecnologie Consapevolezza del valore dei dati come risorsa strategica

Applicare il modello concettuale Entità-Associazione

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> Concetti di entità, attributo, associazione e identificatore Notazioni di Chen e Crow's Foot Generalizzazione e specializzazione Ereditarietà degli attributi 	<ul style="list-style-type: none"> Modellare un sistema informativo individuando entità e relazioni Rappresentare graficamente il modello utilizzando diverse notazioni Costruire gerarchie di entità coerenti e funzionali 	<ul style="list-style-type: none"> Attitudine a ragionare in termini astratti Cura nella rappresentazione grafica Capacità di sintesi e di formalizzazione di fenomeni complessi

Riferimenti

<i>Indicazioni nazionali</i>	<i>DigiComp 2.2</i>	<i>Competenze chiave</i>
11.1	1.c, 5.b	C, D



10. Progettazione logica e modello relazionale

Obiettivi didattici

Gli studenti proseguono il percorso iniziato con lo studio del modello entità–relazione e vengono guidati alla traduzione di un modello concettuale in uno schema logico relazionale. L'esperienza li conduce a comprendere come una realtà complessa possa essere rappresentata in modo rigoroso attraverso tabelle, vincoli e relazioni.

La progettazione logica richiede loro di analizzare con attenzione le dipendenze tra attributi, di valutare diverse soluzioni possibili e di applicare il processo di normalizzazione. Il modulo contribuisce quindi al consolidamento di abilità trasversali come la precisione, l'ordine metodologico e la consapevolezza del valore della qualità dei dati. Gli studenti imparano che la progettazione non è un'attività astratta, ma un passaggio necessario per garantire coerenza, completezza e affidabilità alle informazioni.

Competenze attese

Progettare uno schema logico relazionale

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> Principi del modello logico relazionale Principali costrutti e vincoli di integrità 	<ul style="list-style-type: none"> Definire la struttura di un database coerente con le specifiche Individuare e applicare correttamente i vincoli Organizzare i dati in modo ordinato e rigoroso 	<ul style="list-style-type: none"> Precisione nel rispetto delle regole formali Attenzione alla coerenza e completezza delle informazioni Consapevolezza dell'importanza della qualità dei dati

Tradurre uno schema concettuale ER in schema logico relazionale

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> Regole di traduzione dal modello entità–relazione al modello logico relazionale Rappresentazione dei principali costrutti 	<ul style="list-style-type: none"> Applicare le regole di traduzione in maniera sistematica Rielaborare lo schema per garantire correttezza e chiarezza 	<ul style="list-style-type: none"> Predisposizione a lavorare con ordine e metodo

Riferimenti

<i>Indicazioni nazionali</i>	<i>DigiComp 2.2</i>	<i>Competenze chiave</i>
11.1	-	C, D

11. SQL e algebra relazionale

Obiettivi didattici

Questo modulo introduce il linguaggio SQL (Structured Query Language), strumento fondamentale per la gestione e l'interrogazione dei database relazionali. Gli studenti imparano a distinguere tra i diversi



insiemi di comandi che costituiscono il linguaggio, comprendendo le funzioni della definizione delle strutture, della manipolazione dei dati e delle interrogazioni. Accanto alla dimensione operativa, il modulo prevede dei cenni all'algebra relazionale come strumento teorico di formalizzazione delle operazioni sui dati.

Competenze attese

Utilizzare SQL per gestire un database

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Principali comandi di SQL e loro funzioni 	<ul style="list-style-type: none"> • Scrivere interrogazioni • Creare e modificare strutture di dati • Verificare la correttezza delle operazioni eseguite 	<ul style="list-style-type: none"> • Attenzione alle conseguenze delle modifiche • Perseveranza nel perfezionare una query fino a raggiungere l'obiettivo

Confrontare e tradurre interrogazioni tra algebra relazionale e SQL

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Operatori fondamentali dell'algebra relazionale • Corrispondenze concettuali con i comandi SQL 	<ul style="list-style-type: none"> • Scrivere interrogazioni sia in algebra relazionale sia in SQL • Tradurre da un linguaggio all'altro • Verificare la coerenza dei risultati ottenuti 	<ul style="list-style-type: none"> • Flessibilità nel passare da rappresentazioni astratte a implementazioni concrete • Capacità di riconoscere l'equivalenza tra soluzioni diverse

Riferimenti

<i>Indicazioni nazionali</i>	<i>DigiComp 2.2</i>	<i>Competenze chiave</i>
11.1, 11.2	3.d	C, D

12. Sviluppo di applicazioni

Obiettivi didattici

Con questo modulo gli studenti giungono a una fase di sintesi del percorso svolto, in cui le conoscenze e le abilità acquisite nei moduli precedenti vengono integrate e applicate alla realizzazione di un progetto concreto. Partendo da un'idea o da un bisogno reale, saranno chiamati ad analizzare i requisiti, progettare l'architettura del sistema, sviluppare le varie componenti e verificarne il funzionamento. L'esperienza non si limita all'aspetto tecnico della programmazione, ma mette in evidenza la necessità di seguire un metodo strutturato, di lavorare con ordine e di saper gestire tempi e risorse.

Durante il percorso si trovano a prendere decisioni, a confrontare soluzioni alternative, a rispettare vincoli e priorità, fino a trasformare un'idea in un prodotto funzionante.

Il modulo ha quindi la finalità di consolidare non solo le competenze di programmazione, ma anche quelle trasversali legate al problem solving, all'organizzazione del lavoro e alla comunicazione dei risultati, offrendo un'occasione significativa di crescita personale.



Competenze attese

Questo modulo mira al consolidamento delle competenze dei moduli precedenti cui si aggiungono le seguenti.

Analisi e definizione dei requisiti

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Tipologie di requisiti • Tecniche di raccolta e specifica dei requisiti • Concetto di stakeholder e priorità progettuali 	<ul style="list-style-type: none"> • Individuare e formulare requisiti chiari e non ambigui • Riconoscere vincoli e opportunità legati al contesto • Predisporre documentazione iniziale coerente 	<ul style="list-style-type: none"> • Curiosità nell'esplorare il problema prima di proporre soluzioni • Disponibilità a rivedere le ipotesi di lavoro • Attenzione all'ascolto e al confronto

Pianificazione e gestione del progetto

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Principi di project management applicati al software • Concetti di fasi di lavoro • Scadenze e priorità 	<ul style="list-style-type: none"> • Suddividere il lavoro in fasi e attività gestibili • Stimare correttamente tempi e risorse necessarie • Monitorare l'avanzamento • Ricalibrare il piano in base 	<ul style="list-style-type: none"> • Disciplina nel rispetto degli impegni • Capacità di mantenere un ritmo di lavoro sostenibile • Attitudine alla responsabilità condivisa

Validazione, presentazione e riflessione critica

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> • Tecniche di collaudo • Principi di presentazione e comunicazione dei risultati 	<ul style="list-style-type: none"> • Testare e validare l'applicazione prodotta • Presentare il lavoro in modo chiaro ed efficace • Individuare punti di forza e aree di miglioramento 	<ul style="list-style-type: none"> • Orgoglio per il lavoro svolto • Apertura al feedback come occasione di crescita • Consapevolezza del proprio percorso di apprendimento

Riferimenti

<i>Indicazioni nazionali</i>	<i>DigiComp 2.2</i>	<i>Competenze chiave</i>
Dipende dal progetto scelto	3.a, 3.d, 5.a, 5.b, 5.c	C, D, E, G

13. Gestione dell'ambiente scolastico digitale (comune a tutti gli anni)

Obiettivi didattici

La gestione dell'ambiente digitale scolastico costituisce una competenza trasversale che accompagna gli studenti lungo l'intero percorso liceale. L'obiettivo principale è favorire un uso sicuro, responsabile ed efficace degli strumenti informatici e delle piattaforme digitali messe a disposizione dalla scuola, sviluppando autonomia organizzativa e consapevolezza delle regole che ne disciplinano l'utilizzo.



Gli studenti imparano a gestire in modo ordinato il proprio spazio digitale, a utilizzare correttamente le piattaforme di collaborazione, archiviazione e comunicazione e a rispettare i principi di cittadinanza digitale. Questo percorso si propone anche di potenziare la capacità di comunicare in contesti formali e informali attraverso strumenti digitali, adottando comportamenti corretti e responsabili.

Competenze attese

Utilizzare in modo consapevole e sicuro l'ambiente digitale scolastico

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> Struttura e regole del laboratorio di informatica Principali piattaforme digitali scolastiche (registro elettronico, Teams, Microsoft 365, OneDrive) Diritti e doveri legati all'uso delle piattaforme 	<ul style="list-style-type: none"> Accedere e utilizzare correttamente gli strumenti Gestire credenziali e file in autonomia Archiviare e condividere documenti in cloud 	<ul style="list-style-type: none"> Responsabilità nell'uso delle risorse digitali Cura e ordine nella gestione del proprio ambiente digitale Attenzione alla sicurezza e alla protezione dei dati

Comunicare nei contesti digitali scolastici

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> Regole di comunicazione formale e informale Canali di comunicazione ufficiali della scuola Netiquette 	<ul style="list-style-type: none"> Scrivere messaggi chiari, pertinenti e corretti Rispettare tempi e modalità di comunicazione Collaborare in ambienti digitali 	<ul style="list-style-type: none"> Rispetto e cortesia nelle interazioni online Consapevolezza del ruolo personale nelle dinamiche comunicative Attenzione alla chiarezza e alla pertinenza

Organizzare il proprio lavoro digitale scolastico

<i>Conoscenze</i>	<i>Abilità</i>	<i>Atteggiamenti</i>
<ul style="list-style-type: none"> Strumenti per la gestione delle attività scolastiche Modalità di consegna e restituzione dei compiti digitali 	<ul style="list-style-type: none"> Organizzare ordinatamente i materiali digitali Rispettare scadenze e modalità di consegna Utilizzare strumenti digitali per gestire il lavoro scolastico 	<ul style="list-style-type: none"> Autonomia nella gestione dei lavori e dei materiali digitali Precisione e puntualità nelle consegne Impegno costante nell'organizzazione personale

Riferimenti

<i>Indicazioni nazionali</i>	<i>DigiComp 2.2</i>	<i>Competenze chiave</i>
-	2.a, 2.b, 2.c, 2.d, 2.e, 2.f, 4.b, 5.a, 5.b	C, D, E, F



RIFERIMENTI

Indicazioni nazionali riguardanti gli obiettivi specifici di apprendimento

Da "Schema di regolamento recante «Indicazioni nazionali riguardanti gli obiettivi specifici di apprendimento concernenti le attività e gli insegnamenti compresi nei piani degli studi previsti per i percorsi liceali di cui all'articolo 10, comma 3, del decreto del Presidente della Repubblica 15 marzo 2010, n. 89, in relazione all'articolo 2, commi 1 e 3, del medesimo regolamento.»", Allegato B (D.M. 7 ottobre 2010 n. 211)¹.

Are tematiche

AC - Architettura dei computer

SO - Sistemi operativi

AL - Algoritmi e linguaggi di programmazione

DE - Elaborazione digitale dei documenti

RC - Reti di computer

IS - Struttura di Internet e servizi

CS - Computazione, calcolo numerico e simulazione

BD - Basi di dati

Obiettivi specifici di apprendimento

Primo biennio

Nel primo biennio sono usati gli strumenti di lavoro più comuni del computer insieme ai concetti di base ad essi connessi.

Lo studente è introdotto alle caratteristiche architetture di un computer: i concetti di hardware e software [1.1], una introduzione alla codifica binaria [1.2] presenta i codici ASCII e Unicode [1.3], gli elementi funzionali della macchina di Von Neumann: CPU, memoria, dischi, bus e le principali periferiche [1.4]. (AC)

Conosce il concetto di sistema operativo [2.1], le sue funzionalità di base [2.2] e le caratteristiche dei sistemi operativi più comuni [2.3]; il concetto di processo come programma in esecuzione [2.4], il meccanismo base della gestione della memoria [2.5] e le principali funzionalità dei file system [2.6]. (SO)

Lo studente conosce gli elementi costitutivi di un documento elettronico [3.1] e i principali strumenti di produzione [3.2]. Occorre partire da quanto gli studenti hanno già acquisito nella scuola di base per far loro raggiungere la padronanza di tali strumenti, con particolare attenzione al foglio elettronico [3.3]. (DE)

¹ <https://www.istruzione.it/alternanza/allegati/NORMATIVA%20ASL/INDICAZIONI%20NAZIONALI%20PER%20I%20LICEI.pdf>



Apprende la struttura e i servizi di Internet [4.1]. Insieme alle altre discipline si condurranno gli studenti a un uso efficace della comunicazione [4.2] e della ricerca di informazioni [4.3], e alla consapevolezza delle problematiche [4.4] e delle regole di tale uso [4.5]. (IS)

Lo studente è introdotto ai principi alla base dei linguaggi di programmazione [5.1] e gli sono illustrate le principali tipologie di linguaggi [5.2] e il concetto di algoritmo [5.3]. Sviluppa la capacità di implementare un algoritmo in pseudo-codice [5.4] o in un particolare linguaggio di programmazione, di cui si introdurrà la sintassi [5.5]. (AL)

Secondo biennio

Nel secondo biennio si procede ad un allargamento della padronanza di alcuni strumenti e un approfondimento dei loro fondamenti concettuali. La scelta dei temi dipende dal contesto e dai rapporti che si stabiliscono fra l'informatica e le altre discipline. Sarà possibile disegnare un percorso all'interno delle seguenti tematiche: strumenti avanzati di produzione dei documenti elettronici [6], linguaggi di markup (XML etc) [7], formati non testuali (bitmap, vettoriale, formati di compressione) [8], font tipografici [9], progettazione web [10] (DE); introduzione al modello relazionale dei dati [11.1], ai linguaggi di interrogazione e manipolazione dei dati [11.2] (BS); implementazione di un linguaggio di programmazione [12.1], metodologie di programmazione [12.2], sintassi di un linguaggio orientato agli oggetti [12.3] (AL).

Quinto Anno

È opportuno che l'insegnante - che valuterà di volta in volta il percorso didattico più adeguato alla singola classe - realizzi percorsi di approfondimento, auspicabilmente in raccordo con le altre discipline.

Sono studiati i principali algoritmi del calcolo numerico [13] (CS), introdotti i principi teorici della computazione [14] (CS) e affrontate le tematiche relative alle reti di computer, ai protocolli di rete, alla struttura di internet e dei servizi di rete [15] (RC) (IS). Con l'ausilio degli strumenti acquisiti nel corso dei bienni precedenti, sono inoltre sviluppate semplici simulazioni come supporto alla ricerca scientifica (studio quantitativo di una teoria, confronto di un modello con i dati...) in alcuni esempi, possibilmente connessi agli argomenti studiati in fisica o in scienze [16] (CS).

DigComp 2.2: Il Quadro delle Competenze Digitali per i Cittadini

Da "DigComp 2.2: Il Quadro delle Competenze Digitali per i Cittadini"².

1. Alfabetizzazione su informazioni e dati
 - a. Navigare, ricercare e filtrare dati, informazioni e contenuti digitali
 - b. Valutare dati, informazioni e contenuti digitali
 - c. Gestire dati, informazioni e contenuti digitali
2. Comunicazione e collaborazione
 - a. Interagire con gli altri attraverso le tecnologie digitali
 - b. Condividere informazioni attraverso le tecnologie digitali
 - c. Esercitare la cittadinanza attraverso le tecnologie digitali
 - d. Collaborare attraverso le tecnologie digitali

² https://www.agid.gov.it/sites/agid/files/2024-05/digcomp_2.2_italiano.pdf



- e. Netiquette
- f. Gestire l'identità digitale
- 3. Creazione di contenuti digitali
 - a. Sviluppare contenuti digitali
 - b. Integrare e rielaborare contenuti digitali
 - c. Copyright e licenze
 - d. Programmazione
- 4. Sicurezza
 - a. Proteggere i dispositivi
 - b. Proteggere i dati personali e la privacy
 - c. Proteggere la salute e il benessere
 - d. Proteggere l'ambiente
- 5. Risolvere problemi
 - a. Risolvere problemi tecnici
 - b. Individuare fabbisogni e risposte tecnologiche
 - c. Utilizzare in modo creativo le tecnologie digitali
 - d. Individuare divari di competenze digitali

Competenze chiave per l'apprendimento permanente

Dalla " *Raccomandazione del consiglio del 22 maggio 2018 relativa alle competenze chiave per l'apprendimento permanente*" (2018/C 189/01)³.

- A. Competenza alfabetica funzionale
- B. Competenza multilinguistica
- C. Competenza matematica e competenza in scienze, tecnologie e ingegneria
- D. Competenza digitale
- E. Competenza personale, sociale e capacità di imparare a imparare
- F. Competenza in materia di cittadinanza
- G. Competenza imprenditoriale
- H. Competenza in materia di consapevolezza ed espressione culturali

³ [https://eur-lex.europa.eu/legal-content/IT/TXT/PDF/?uri=CELEX:32018H0604\(01\)](https://eur-lex.europa.eu/legal-content/IT/TXT/PDF/?uri=CELEX:32018H0604(01))