## DENOMINAZIONE INSEGNAMENTO / ATTIVITÀ FORMATIVA

# LABORATORIO DI PROGRAMMAZIONE

*CODING LAB*

## INFORMAZIONI INSEGNAMENTO / ATTIVITÀ FORMATIVA

| | | | |
|---|---|---|---|
| **A.A.** | 2026-2027 | **CdS** | BIOINFORMATICA |
| **Codice** | 8067792 | **Canale** | Unico |
| **CFU** | 2 | **Lingua** | Italiano |

## DOCENTE RESPONSABILE DELL'INSEGNAMENTO / ATTIVITÀ FORMATIVA

**Gabriele Ausiello**

## CODOCENTE

**-**

### OBIETTIVI FORMATIVI E RISULTATI DI APPRENDIMENTO ATTESI

| | |
|---|---|
| **Italiano** | **OBIETTIVI FORMATIVI:**<br>Gli obiettivi formativi e i risultati di apprendimento attesi sono definiti secondo i Descrittori di Dublino.<br>L'insegnamento si propone di fornire alle studentesse e agli studenti le basi della programmazione mediante il linguaggio Ruby e di applicarle alla soluzione di semplici problemi bioinformatici, con particolare riferimento alla manipolazione e all'allineamento di sequenze biologiche. Al termine del corso, le studentesse e gli studenti dovranno essere in grado di leggere, comprendere, modificare e sviluppare semplici programmi, traducendo un problema in una sequenza di operazioni formalizzate in codice.<br><br>**CONOSCENZA E CAPACITÀ DI COMPRENSIONE:**<br>Le studentesse e gli studenti acquisiranno conoscenze relative alla sintassi e ai principali costrutti del linguaggio Ruby: variabili, tipi di dato elementari, stringhe, operatori, espressioni condizionali, cicli, array, metodi, funzioni e gestione di file. Comprenderanno inoltre la logica con cui un problema viene scomposto in passaggi algoritmici e i principi di base necessari per sviluppare codice destinato alla semplice analisi e all'allineamento di sequenze biologiche.<br><br>**CAPACITÀ DI APPLICARE CONOSCENZA E COMPRENSIONE:**<br>Le studentesse e gli studenti saranno in grado di tradurre semplici problemi informatici e bioinformatici in procedure algoritmiche, scegliere strutture di controllo e strutture dati adeguate, scrivere ed eseguire programmi, individuare e correggere errori, leggere e processare dati di sequenza e sviluppare un semplice programma per l'allineamento di sequenze biologiche.<br><br>**AUTONOMIA DI GIUDIZIO:**<br>Le studentesse e gli studenti svilupperanno la capacità di valutare la correttezza di un procedimento algoritmico e dei risultati prodotti dal programma, riconoscere casi limite ed |

| | |
|---|---|
| | errori, confrontare soluzioni alternative e scegliere una strategia adeguata alla natura del problema e ai dati disponibili.<br><br>**ABILITÀ COMUNICATIVE:**<br>Le studentesse e gli studenti impareranno a scrivere codice chiaro, leggibile, organizzato e opportunamente commentato, nonché a descrivere in modo sintetico il problema affrontato, il procedimento seguito e le principali scelte implementative, utilizzando una terminologia informatica e bioinformatica appropriata.<br><br>**CAPACITÀ DI APPRENDIMENTO:**<br>Le studentesse e gli studenti acquisiranno gli strumenti necessari per consultare documentazione tecnica ed esempi di codice, approfondire autonomamente nuove funzionalità del linguaggio Ruby e trasferire i principi appresi allo studio di altri linguaggi di programmazione e alla soluzione di problemi bioinformatici affini. |
| *English* | **LEARNING OBJECTIVES:**<br>The learning objectives and expected learning outcomes are defined according to the Dublin Descriptors.<br>The course aims to provide students with the fundamentals of programming through the Ruby language and to apply them to simple bioinformatics problems, with particular emphasis on the manipulation and alignment of biological sequences. By the end of the course, students should be able to read, understand, modify, and develop simple programs, translating a problem into a sequence of operations expressed as code.<br><br>**KNOWLEDGE AND UNDERSTANDING:**<br>Students will acquire knowledge of the syntax and main constructs of the Ruby language, including variables, basic data types, strings, operators, conditional statements, loops, arrays, methods, functions, and file handling. They will also understand how a problem can be decomposed into algorithmic steps and the basic principles required to develop code for simple biological sequence analysis and alignment.<br><br>**APPLYING KNOWLEDGE AND UNDERSTANDING:**<br>Students will be able to translate simple computing and bioinformatics problems into algorithmic procedures, select suitable control structures and data structures, write and run programs, identify and correct errors, read and process sequence data, and develop a simple program for biological sequence alignment.<br><br>**MAKING JUDGEMENTS:**<br>Students will develop the ability to assess the correctness of an algorithmic procedure and of the results produced by a program, identify errors and edge cases, compare alternative solutions, and select an approach suited to the problem and the available data.<br><br>**COMMUNICATION SKILLS:**<br>Students will learn to write clear, readable, well-organized, and appropriately commented code. They will also be able to describe concisely the problem addressed, the procedure followed, and the main implementation choices, using appropriate computing and bioinformatics terminology.<br><br>**LEARNING SKILLS:**<br>Students will acquire the skills required to consult technical documentation and code examples, independently explore additional Ruby features, and transfer the principles learned to other programming languages and related bioinformatics problems. |

## PREREQUISITI

| | |
|---|---|
| **Italiano** | Conoscenza di base dell'uso di un computer e di un sistema operativo, nonché dei concetti fondamentali di bioinformatica e di sequenza biologica. Non sono richieste conoscenze avanzate di programmazione. |
| *English* | Basic knowledge of computer and operating system use, together with a basic understanding of bioinformatics and biological sequences. Advanced programming knowledge is not required. |

## PROGRAMMA E CRONOPROGRAMMA

| | |
|---|---|
| **Italiano** | Il corso si articola in due parti, sviluppate in successione e integrate da esercitazioni pratiche di difficoltà progressiva.<br><br>**PRIMA PARTE - Fondamenti di programmazione in Ruby**<br>Introduzione alla programmazione e all'esecuzione di script. Variabili, tipi di dato elementari, stringhe, operatori ed espressioni. Istruzioni condizionali. Cicli. Array. Metodi e funzioni. Lettura e scrittura di file. Scomposizione di un problema in passaggi elementari. Tecniche di base per il test, l'individuazione degli errori e il miglioramento della leggibilità del codice.<br><br>**SECONDA PARTE - Applicazioni alla bioinformatica**<br>Rappresentazione e manipolazione di sequenze biologiche mediante stringhe e array. Lettura di dati di sequenza da file. Confronto tra sequenze e sviluppo guidato di semplice codice per il loro allineamento. Verifica del programma su casi di prova e interpretazione dei risultati prodotti.<br><br>**CRONOPROGRAMMA**<br>Nella fase iniziale vengono introdotti i costrutti fondamentali del linguaggio Ruby e applicati attraverso esercizi brevi. Nella fase successiva gli stessi costrutti vengono combinati per affrontare problemi bioinformatici progressivamente più articolati. La parte conclusiva è dedicata all'integrazione delle conoscenze acquisite nello sviluppo di un programma completo per la semplice analisi e l'allineamento di sequenze. |
| *English* | The course is divided into two consecutive parts, both integrated with practical exercises of increasing difficulty.<br><br>**PART 1 - Fundamentals of programming in Ruby**<br>Introduction to programming and script execution. Variables, basic data types, strings, operators, and expressions. Conditional statements. Loops. Arrays. Methods and functions. Reading from and writing to files. Decomposition of a problem into elementary steps. Basic techniques for testing, debugging, and improving code readability.<br><br>**PART 2 - Bioinformatics applications**<br>Representation and manipulation of biological sequences using strings and arrays. Reading sequence data from files. Sequence comparison and guided development of simple code for sequence alignment. Testing the program on representative cases and interpreting the results.<br><br>**COURSE SCHEDULE**<br>The initial part introduces the fundamental Ruby constructs and applies them through short exercises. In the following part, the same constructs are combined to address progressively more structured bioinformatics problems. The final part is devoted to integrating the |

| | |
|---|---|
| | acquired knowledge into the development of a complete program for simple biological sequence analysis and alignment. |

## TESTI ADOTTATI E BIBLIOGRAFIA DI RIFERIMENTO

| | |
|---|---|
| **Italiano** | **Materiale didattico:**<br>Codice, esercizi e soluzioni sviluppati durante le lezioni e messi a disposizione dal docente. Il materiale può essere richiesto tramite e-mail all'indirizzo gabriele.ausiello@uniroma2.it.<br><br>**Bibliografia di riferimento:**<br>Documentazione ufficiale del linguaggio Ruby e ulteriori risorse tecniche indicate dal docente durante il corso. |
| *English* | **Teaching material:**<br>Code, exercises, and solutions developed during the classes and made available by the instructor. The material may be requested by e-mail at gabriele.ausiello@uniroma2.it.<br><br>**Reference material:**<br>Official Ruby language documentation and additional technical resources indicated by the instructor during the course. |

## DESCRIZIONE DELLE MODALITÀ DI SVOLGIMENTO E METODI DIDATTICI ADOTTATI

☒ In presenza ☐ A distanza

| | |
|---|---|
| **Italiano** | Le attività si svolgono in presenza e alternano brevi spiegazioni teoriche, dimostrazioni ed esercitazioni pratiche. Il docente presenta problemi di programmazione alla lavagna o sullo schermo e le studentesse e gli studenti sviluppano sul proprio computer il codice necessario a risolverli, individualmente o in piccoli gruppi. Gli esercizi possono essere affrontati autonomamente o con il supporto del docente e dei compagni; le soluzioni vengono quindi mostrate, confrontate e discusse collegialmente. Il metodo didattico privilegia l'apprendimento attraverso la pratica, la progressione graduale della difficoltà e il feedback immediato sul codice prodotto. |
| *English* | Teaching is delivered in person and combines short theoretical explanations, demonstrations, and practical exercises. The instructor presents programming problems on the board or screen, and students develop the code required to solve them on their own computers, either individually or in small groups. Exercises may be completed independently or with support from the instructor and classmates; solutions are then presented, compared, and discussed with the class. The teaching method emphasizes learning by doing, a gradual increase in difficulty, and immediate feedback on the code produced. |

## MODALITÀ DI FREQUENZA

☐ frequenza obbligatoria ☒ frequenza facoltativa

## DESCRIZIONE DELLE MODALITÀ DI FREQUENZA

| | |
|---|---|
| **Italiano** | La frequenza è facoltativa, ma è fortemente consigliata per la natura laboratoriale e progressiva delle attività. La partecipazione alle lezioni consente di svolgere gli esercizi con feedback immediato e di chiarire le difficoltà incontrate durante lo sviluppo del codice. Le studentesse e gli studenti che non possono frequentare possono comunque accedere alla valutazione finale e non ricevono alcuna penalizzazione per la mancata frequenza. |

| | |
|---|---|
| *English* | Attendance is optional but strongly recommended because of the practical and progressive nature of the course. Participation allows students to complete exercises with immediate feedback and to clarify difficulties encountered while developing code. Students who are unable to attend may still take the final assessment and will not be penalized for non-attendance. |

## MODALITÀ DI VALUTAZIONE

☐ Prova scritta ☐ Prova orale ☐ Prova di laboratorio ☐ Prova pratica

☐ Valutazione in itinere ☒ Valutazione di progetto ☐ Valutazione di tirocinio

## COMPOSIZIONE DELLA COMMISSIONE VALUTATRICE

Gabriele Ausiello, Gerardo Pepe

## DESCRIZIONE DELLE MODALITÀ E DEI CRITERI DI VERIFICA DELL'APPRENDIMENTO

| | |
|---|---|
| **Italiano** | La valutazione finale consiste in un progetto pratico individuale da svolgere autonomamente a casa. Le studentesse e gli studenti devono sviluppare e consegnare un programma in Ruby per la soluzione di un semplice problema di analisi di sequenze biologiche, con particolare riferimento all'allineamento di sequenze, applicando i contenuti affrontati durante il corso. La consegna deve comprendere il codice sorgente e le indicazioni necessarie per eseguirlo.<br><br>**CRITERI DI VALUTAZIONE**<br>Il progetto è valutato sulla base della correttezza funzionale e della rispondenza alla consegna, dell'uso appropriato dei costrutti di programmazione trattati nel corso, dell'organizzazione e leggibilità del codice, della capacità di testare il programma e gestire semplici casi limite, nonché del grado di autonomia e rielaborazione dimostrato nella soluzione proposta.<br><br>**ESITO DELLA VALUTAZIONE**<br>L'esame prevede un giudizio di idoneità e non l'attribuzione di un voto numerico.<br><br>**IDONEO:**<br>Il progetto è completo e coerente con la consegna; il programma viene eseguito correttamente e produce risultati adeguati su casi di prova rappresentativi. Il codice mostra una sufficiente padronanza dei concetti fondamentali della programmazione, utilizza in modo appropriato le principali strutture di controllo e strutture dati, ed è complessivamente chiaro, organizzato e comprensibile.<br><br>**NON IDONEO:**<br>Il progetto è incompleto, non coerente con la consegna o non funzionante, oppure presenta errori sostanziali nella logica e nell'uso dei costrutti fondamentali. Il programma non produce risultati corretti neppure su casi semplici, il codice risulta scarsamente comprensibile o disorganizzato, oppure non emerge una capacità sufficiente di applicare autonomamente le conoscenze acquisite. |
| *English* | The final assessment consists of an individual practical project to be completed independently at home. Students are required to develop and submit a Ruby program addressing a simple biological sequence analysis problem, with particular reference to |

| | |
|---|---|
| | sequence alignment, by applying the topics covered during the course. The submission must include the source code and the instructions required to run it.<br><br>**ASSESSMENT CRITERIA**<br>The project is assessed on the basis of functional correctness and compliance with the assignment, appropriate use of the programming constructs covered in the course, code organization and readability, the ability to test the program and handle simple edge cases, and the degree of autonomy and independent elaboration demonstrated in the proposed solution.<br><br>**ASSESSMENT OUTCOME**<br>The examination is assessed on a pass/fail basis and does not receive a numerical grade.<br><br>**PASS:**<br>The project is complete and consistent with the assignment; the program runs correctly and produces appropriate results on representative test cases. The code demonstrates an adequate command of fundamental programming concepts, uses the main control structures and data structures appropriately, and is generally clear, organized, and understandable.<br><br>**FAIL:**<br>The project is incomplete, inconsistent with the assignment, or non-functional, or it contains major errors in its logic and in the use of fundamental programming constructs. The program does not produce correct results even on simple cases, the code is poorly organized or difficult to understand, or the work does not demonstrate a sufficient ability to apply the acquired knowledge independently. |