National University of Engineering (UNI)
School of Computer Science
Syllabus 2026-I

1. COURSE

CS111. Introduction to Programming (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS111. Introduction to Programming
2.2 Semester	:	1^{st} Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	None

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This is the first course in the sequence of introductory courses to Computer Science. This course introduces participants to the fundamental concepts of programming using Python. Topics include data types, control structures, functions, lists, recursion, and the mechanics of execution, testing, and debugging. Programming is one of the pillars of Computer Science; any professional in the area will need to program to materialize their models and proposals.

5. GOALS

- Introduce the fundamental concepts of programming.
- Develop the ability of abstraction using a programming language.
- Develop problem-solving skills through algorithmic thinking.
- Understand basic data structures and their applications.
- Master the Python programming language for problem solving.

6. COMPETENCES

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Familiarity)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Usage)

7. TOPICS

Unit 1: Introduction to Programming and History (4 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Brief history of computing and programming languages. • Overview of Python programming language. • Programming environments and tools. • Basic program structure and execution. • Writing, running, and debugging simple programs.	• Identify important trends in the history of computing. [Familiarizarse] • Discuss the historical context of programming language paradigms. [Familiarizarse] • Compare daily life before and after personal computers and the Internet. [Evaluar] • Write, compile, and run a simple Python program. [Usar]
Readings : [Zel10], [Gut13], [BB19]	

Unit 2: Sistemas de Tipos I: Fundamentos (6 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Un tipo como un conjunto de valores junto con un conjunto de operaciones: – Tipos primitivos (por ejemplo, números, booleanos) – Tipos compuestos construidos a partir de otros tipos (por ejemplo, registros/estructuras, uniones, arreglos, listas, funciones, referencias usando operaciones de conjuntos) • Seguridad de tipos y errores causados por usar valores inconsistentemente dados sus tipos pretendidos • Objetivos y limitaciones del tipado estático y dinámico: detectar y eliminar errores lo antes posible. • Primitive data types in Python (int, float, str, bool). • Type conversion and casting.	• Describir, tanto para un tipo primitivo como para uno compuesto, los valores que tienen ese tipo [Familiarizarse] • Describir ejemplos de errores de programa detectados por un sistema de tipos [Familiarizarse] • Usar tipos y mensajes de error de tipos para escribir y depurar programas [Usar] • Explain Python's dynamic typing system. [Familiarizarse]
Readings : [Zel10], [Gut13]	

Unit 3: Programación Orientada a Objetos I: Fundamentos (14 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Programación imperativa como subconjunto de la programación orientada a objetos. • Definición de clases: campos, métodos y constructores. • Propiedades dinámicas vs estáticas. • Variables, expressions, and assignments. • Input and output operations. • Conditional statements (if, elif, else). • Iteration structures (for loops, while loops). • Nested loops and loop control statements. • Exception handling basics.	• Componer una clase a través del diseño, implementación y prueba para cumplir con los requisitos de comportamiento [Usar] • Predecir y validar el flujo de control en un programa usando despacho dinámico [Evaluar] • Usar mecanismos de encapsulación orientados a objetos como interfaces y miembros privados [Usar] • Implement programs using conditional and iterative structures. [Usar] • Debug programs with syntax and logical errors. [Usar] • Design and implement simple algorithms using loops. [Usar]
Readings : [Zel10], [Gut13], [BB19]	

Unit 4: Functions and Parameter Passing (10 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Function definition and invocation. • Parameter passing mechanisms (by value, by reference). • Return values and multiple returns. • Variable scope and lifetime (local, global, nonlocal). • Recursion fundamentals. • Lambda expressions and higher-order functions. • Functions as first-class objects. • Documentation and testing of functions.	• Define and implement functions with proper parameter passing. [Usar] • Explain the difference between pass-by-value and pass-by-reference. [Familiarizarse] • Implement recursive solutions to simple problems. [Usar] • Use lambda expressions for functional programming constructs. [Usar] • Design functions with appropriate scope and documentation. [Evaluar]
Readings : [Zel10], [Gut13], [HT99]	

Unit 5: Basic Data Structures (10 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Lists and list operations (indexing, slicing, mutability). • Tuples and immutability. • Dictionaries and key-value mapping. • Sets and set operations. • Strings as sequences. • List comprehensions and generator expressions. • Basic searching and sorting algorithms. • Introduction to algorithmic complexity for basic operations.	• Implement programs using lists, dictionaries, and sets. [Usar] • Choose appropriate data structures for given problems. [Evaluar] • Explain the trade-offs between different data structures. [Familiarizarse] • Implement basic search and sort operations on lists. [Usar]
Readings : [Zel10], [Gut13]	

Unit 6: Programación Orientada a Objetos I: Fundamentos (10 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Subclases, herencia (incluyendo herencia múltiple) y sobrescritura de métodos. • Despacho dinámico: definición de llamada a método. • Idiomas orientados a objetos para encapsulación: – Privacidad, ocultación de datos y visibilidad de miembros de clase. – Interfaces que revelan solo firmas de métodos. – Clases base abstractas, rasgos (*traits*) y mixins. • Classes and objects in Python. • Attributes and methods. • Inheritance and polymorphism basics. • Encapsulation and information hiding.	• Construir una jerarquía de clases simple utilizando subclases que permita reutilizar código para subclases distintas [Usar] • Explicar la relación entre la herencia orientada a objetos (compartición de código y sobrescritura) y el subtipado (la idea de que un subtipo sea usable en un contexto que espera el supertipo) [Familiarizarse] • Comparar y contrastar los beneficios y costos/impacto de usar herencia (subclases) y composición (específicamente, cómo basar la composición en funciones de orden superior) [Evaluar] • Design and implement simple classes in Python. [Usar] • Explain the benefits of OOP for code organization. [Familiarizarse]
Readings : [Zel10], [Gut13]	

Unit 7: Estrategias Algorítmicas (6 hours)

Competences Expected: AG-C08,AG-C12

Topics	**Learning Outcomes**
• Paradigmas: – Fuerza Bruta (por ejemplo, búsqueda lineal, ordenamiento por selección, viajante de comercio, mochila) – Disminuir y Vencer: ∗ Por una Constante (por ejemplo, ordenamiento por inserción, ordenamiento topológico) ∗ Por un Factor Constante (por ejemplo, búsqueda binaria) ∗ Por un Tamaño Variable (por ejemplo, Euclides) – Dividir y Vencer (por ejemplo, búsqueda binaria, quicksort, mergesort, Strassen) – Voraz (por ejemplo, Dijkstra, Kruskal, Mochila) – Transformar y Vencer: ∗ Simplificación de instancia (por ejemplo, encontrar duplicados mediante preordenamiento de lista) ∗ Cambio de representación (por ejemplo, heapsort) ∗ Reducción de problema (por ejemplo, mínimo común múltiplo, programación lineal) ∗ Programación dinámica (por ejemplo, Floyd, Marshall, Bellman-Ford) – Compromisos espacio vs tiempo (por ejemplo, hash) • Iteración vs recursión (por ejemplo, factorial, búsqueda en árbol) • Problem decomposition strategies. • Algorithm design techniques. • Debugging and testing strategies. • Tracing program execution.	• Para cada uno de los paradigmas en esta unidad: – Explicar sus características definitorias – Explicar un ejemplo que demuestre el paradigma incluyendo cómo este ejemplo satisface las características del paradigma. [Familiarizarse] • Dar ejemplos de algoritmos iterativos y recursivos que resuelvan el mismo problema, explicar los beneficios y desventajas de cada enfoque [Usar] • Evaluar si un enfoque voraz conduce a una solución óptima [Evaluar] • Apply systematic debugging techniques to locate and fix errors. [Usar] • Analyze algorithm complexity for simple problems. [Familiarizarse]

Readings : [Gut13], [Cor+09]

Unit 8: Modern Development Methods and Tool Usage (4 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Using modern programming environments (IDEs, notebooks). • Code search and reuse best practices. • Using standard libraries and APIs. • Debugging and testing techniques. • Version control basics. • Code documentation and style guidelines. • Virtual environments and package management.	• Use modern development tools effectively. [Usar] • Apply code reuse principles through libraries and APIs. [Usar] • Debug and test programs systematically. [Usar] • Follow coding standards and documentation practices. [Familiarizarse] • Manage Python packages and environments. [Familiarizarse]
Readings : [Zel10], [HT99]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY

[HT99] Andrew Hunt and David Thomas. *The Pragmatic Programmer: From Journeyman to Master*. 1st. Addison-Wesley, 1999.

[Cor+09] Thomas H. Cormen et al. *Introduction to Algorithms*. 3rd. MIT Press, 2009.

[Zel10] John M. Zelle. *Python Programming: An Introduction to Computer Science*. 2nd. Franklin, Beedle & Associates Inc, 2010.

[Gut13] John V. Guttag. *Introduction to Computation and Programming Using Python*. 1st. MIT Press, 2013.

[BB19] J. Glenn Brookshear and Dennis Brylow. *Computer Science: An Overview*. Global Edition. Pearson, 2019.