COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI ICADE CIHS

GUÍA DOCENTE
2024 - 2025

FICHA TÉCNICA DE LA ASIGNATURA

Datos de la asignatura	
Nombre completo	Programación
Código	DTC-IMAT-112
Título	Grado en Ingeniería Matemática e Inteligencia Artificial
Impartido en	Grado en Ingeniería Matemática e Inteligencia Artificial [Primer Curso]
Nivel	Reglada Grado Europeo
Cuatrimestre	Semestral
Créditos	6,0 ECTS
Carácter	Básico
Departamento / Área	Departamento de Telemática y Computación
Responsable	David Contreras Bárcena
Horario	Consultar horarios de la escuela
Horario de tutorías	Se comunicará el primer día de clase.

Datos del profesorado	
Profesor	
Nombre	David Contreras Bárcena
Departamento / Área	Departamento de Telemática y Computación
Despacho	5ª planta de ICAI
Correo electrónico	davidcb@comillas.edu
Profesor	
Nombre	Lucas Francisco Novales Peleato
Departamento / Área	Departamento de Electrónica, Automática y Comunicaciones
Despacho	D-401
Correo electrónico	lfnovales@comillas.edu

DATOS ESPECÍFICOS DE LA ASIGNATURA

Contextualización de la asignatura

Aportación al perfil profesional de la titulación

La programación representa los cimientos de la Inteligencia Artficial. Todos los modelos desarrollados enfocados a la conducción de vehículos de forma autónoma, dialogar con humanos, generar contenidos, interpretar imágenes, etc. están programados en algún lenguaje. Es por ello, que esta asignatura de primer curso proporciona la base necesaria de programación para establecer la bases de esta área de conocimiento para poder llegar a ser un experto durante el grado y poder desarrollar desde los modelos de análisis más sencillos hasta la IA más compleja.

En esta asignatura se asientan las bases del análisis, diseño y la programación utilizando el lenguaje más utilizado en este campo: Python.

Prerrequisitos

No se exige ningún prerequisito. El alumno no debe tener conocimientos previso de programación.

Competencias - Objetivos

Competencias

GENERALES	
CG04	Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería.
CG05	Conocimiento de la estructura, organización, funcionamiento e interconexión de los sistemas informáticos, los fundamentos de su programación, y su aplicación para la resolución de problemas propios de la ingeniería
ESPECÍFICAS	
CE09	Capacidad para analizar, diseñar y resolver problemas reales a través de técnicas algorítmicas mediante un lenguaje de programación
CE10	Conocimiento de la sintaxis, las estructuras principales y los elementos básicos de un lenguaje de programación en el contexto del análisis de datos y la inteligencia artificial

Resultados de Aprendizaje

RA1	Conocer el funcionamiento de los compiladores e intérpretes de programación.
RA2	Dominar los fundamentos de la programación, como son la definición de variables y trabajar con los tipos de datos nativos.
RA3	Dominar el uso de expresiones, operadores y estructuras de control para el desarrollo de programas mediante un lenguaje imperativo.
RA4	Conocer en profundidad las estructuras básicas de datos que permitan almacenar información estructurada en memoria.
RA5	Ser capaz de definir y realizar llamadas a funciones que apliquen los conceptos de modularidad de código.
RA6	Dominar la escritura y lectura de información en disco de forma persistente.
RA7	Analizar y diseñar algoritmos básicos para la resolución de problemas.

BLOQUES TEMÁTICOS Y CONTENIDOS

Contenidos – Bloques Temáticos

1. Introducción a los lenguajes de programación
    - Introducción a la informática

- Introducción a los computadores

2. Desarrollo de programas y algortimos
    - Ciclo de vida del desarrollo de software
    - Análisis de un problema
    - Diseño de un programa
    - Introducción a la programación imperativa
3. De la programación de bloques a Python
    - Abstracción del lenguaje y sintaxis
    - Lenguajes de programación
    - Conversión de Bloques a Python (Jupyter)
    - Introducción a Python
4. Variables y tipos de datos básicos
    - Numéricos
    - Booleanos
    - Strings (básico)
    - Operadores de asignación
    - Trabajando con variables
5. Cadenas de caracteres (strings)
    - Métodos de la clase str
    - Operaciones sobre cadenas
    - Strings formateados (f-strings)
    - Representación de una cadena de caracteres
    - Estructuras de control. Bucles y condicionales avanzados
6. Programación estructurada
    - Álgebra de Boole
    - Estructuras condicionales
    - Estructuras de control avanzadas
    - Buenas prácticas
7. Listas
    - Operaciones básicas de las listas
    - Manipulación básica de listas
    - Estructuras de control. Bucles y condicionales avanzados
    - Operaciones avanzadas de las listas
    - Matrices
8. Estructuras de datos básicas
    - Tuplas
    - Diccionarios
9. Funciones
    - Tipificación de parámetros
    - Estructura de un programa
    - Creación de módulos
    - Programación modular
    - Documentación de código
10. Excepciones
    - Desarrollo de programas con finalización controlada
11. Objetos como tipos de datos
    - Encapsulación

12. Programación avanzada en Python
    - Gestión de memoria
    - Buenas prácticas de programación
    - Estructuras específicas del lenguaje
    - Ejecución por consola
    - Librerías
13. Mecanismos de Entrada/Salida: Ficheros
    - Operaciones estándar

METODOLOGÍA DOCENTE

Aspectos metodológicos generales de la asignatura

La metodologías docentes a seguir en estas actividades serán:

- Lección magistral
- Aprendizaje práctico
- Aprendizaje colaborativo
- Clase invertida

Metodología Presencial: Actividades

Las actividades formativas se desarrollarán durante las 4 horas de clase a la semana que se distribuirán secuencialmente de la siguiente forma:

- 1h Interactiva + 1h Práctica + 2h Teoría

Estas actividades serán:

- **Clases magistrales expositivas y participativas**:
    - Sesión de 2h de Teoría.
    - El profesor realizará un exposición de los contenidos teóricos, combinando la clase magistral con la programación en directo (livecoding) de pequeños ejemplos ilustrativos.
    - Los códigos generados en el aula estarán a disposición del alumno en el repositorio de la asignatura.
    - Después de esta sesión el profesor liberará:
        - Test de autoevaluación del tema: el alumno identifica si domina los conceptos teóricos.
        - Mini-prácticas: el alumno identifica si sabe traladar a código esos conceptos teóricos mediante programas triviales que no requieran prácticamente algorítmica.
        - Práctica semanal: programa a realizar la semana siguiente orientado a la resolución de problemas con peso algorítmico.
- **Ejercicios prácticos y resolución de problemas**:
    - Sesión de 1h Interactiva.
    - El alumno planteará dudas sobre los conceptos teóricos expuestos en la clase magistral, la mini-práctica y, sobre todo, de la práctica semanal prouesta.
    - Se crearán dinámicas interacticas de resolución de las dudas entre toda la clase con programación de ejemplos por parte del profesor y los alumnos de forma colaborativa.
- **Sesiones prácticas con uso de software**:
    - Sesión de 1h de Prácticas.
    - Los alumnos llegarán a esta sesión con al menos el 80% de la práctica semanal desarrollada, a

CG04, CG05, CE09, CE10

falta de retoques o mejoras por las dudas que pudiera tener.
    - Se dedicará una primera parte de la sesión estas dudas para que los alumnos puedan finalizar su práctica.
    - En la segunda parte de la sesión se plantearán distintos escenarios metodológicos:
        - Mejora y evolución de la propia práctica mediante retos WhatIf. Trabajo individual.
        - Resolución gamificada de retos planteados por el profesor a modo de Challenge: dar solución a un problema, optimizar un código suministrado, eliminar errores de un código.
- **Actividades de evaluación continua del rendimiento:** se realizarán pruebas, desarrollarán prácticas complementarias a las semanales y retos gamificados.
    - Test de autoevaluación del tema: el alumno identifica si domina los conceptos teóricos.
- **Tutoría para resolución de dudas:** esta actividad se realizará de forma implícita durante el resto de actividades descritas.

Metodología No presencial: Actividades	

Las actividades formativas serán:
• **Ejercicios prácticos y resolución de problemas**:
  ○ El alumno dispondrá de problemas concretos enfocados a asimilar los conceptos explicados téoricos en la sesión anterior de teoría para desarrollar de forma no presencial, denominados como **mini-prácticas**. La solución de estos probelmas será subida a la plataforma la semana siguiente, antes de empezar el nuevo tema.
• **Sesiones prácticas con uso de software**:
  ○ Una vez liberada la práctica semanal después de la sesión de teoría correspondiente, el alumno trabajará sobre ella de forma no presencial. El alumno deberá llegar a la sesión presencial de prácticas con un 80% de la práctica desarrollada, en la cual se resolverán las dudas pertinentes. En función de la complejidad de la práctica, esta tarea podrá llevar menos tiempo de la dedicada a la sesión.
• **Estudio personal:** el objetivo principal del trabajo no presencial es llegar a entender y comprender los conceptos teóricos de la asignatura, así como ser capaz de poner en práctica estos conocimientos para resolver los diferentes tipos de problemas. Después de cada explicación teórica el profesor subirá a la web todos los códigos desarrollados y el alumno deberá revisarlos y plantearse cuestiones "Whatif" para asimilar mejor los conceptos teóricos. La forma de estudiar la asignatura será la siguiente:
  ○ Estudiar los conceptos explicados con una papel y un lápiz, sin necesidad de trabajar con el ordenador. El alumno debe saber analizar, diseñar y resolver problemas de un forma abstracta sin el ordenador.
  ○ Analizará el código suministrado por el profesor con el fin de asentar los coneptos teóricos.
  ○ Por último, deberá ser capaz de programar los ejercicios realizados por el profesor desde cero: sin copiar y pegar ni mirar los códigos suministrados.
	CG04, CG05, CE09, CE10

RESUMEN HORAS DE TRABAJO DEL ALUMNO

HORAS PRESENCIALES				
Clases magistrales expositivas y participativas	Tutorías para resolución de dudas	Ejercicios prácticos y resolución de problemas	Actividades de evaluación continua del rendimiento	Sesiones prácticas con uso de software
30.00	5.00	5.00	5.00	20.00
HORAS NO PRESENCIALES				

Sesiones prácticas con uso de software	Estudio personal	Ejercicios prácticos y resolución de problemas
60.00	45.00	10.00
		CRÉDITOS ECTS: 6,0 (180,00 horas)

EVALUACIÓN Y CRITERIOS DE CALIFICACIÓN

Actividades de evaluación	Criterios de evaluación	Peso
Exámenes: • Prueba Intersemestral. • Examen Final.	• **Prueba Intersemestral (20%)**: comprensión de los conceptos fundamentales de la informática y la programación. • **Examen Final (50%)**: se evaluará el pensamiento computacional y abstracto para la resolución de problemas mediante la programación.	70
Sesiones prácticas: • Retos colaborativos • Trabajos no presenciales • Prácticas	La actitud, participación y realización de las **prácticas semanales y** los **retos** planteados en sesiones colaborativas e individuales.	10
Proyecto final	Proyecto final de la asignatura que el alumno entregará al finalizar el curso. Este proyecto consistirá en el desarrollo de un juego donde el alumno podrá elegir entre: • Un juego que diseñe y programe el alumno desde cero con el visto bueno del profesor. • Un juego que proponga de forma genérica el profesor.	20

Calificaciones

La calificación final en **convocatoria ordinaria** y **extraodrinaria** de la asignatura dependerá de la evaluación de las siguientes actividades:

Nota Final = 20% Prueba_Intersemestral + 50% Examen_Final + 10% Prácticas semanales + 20% Proyecto final

Para aprobar la asignatura los alumnos tienen que obtener al menos 5 puntos sobre 10 en el examen final de la asignatura y en la práctica final, tanto en la convocatoria ordinaria como en la extraordinaria.

La inasistencia al 15% o más de las horas presenciales de esta asignatura puede tener como consecuencia la imposibilidad de presentarse a las convocatorias ordinaria y extraordinaria.

BIBLIOGRAFÍA Y RECURSOS

Bibliografía Básica

Libro básico de referencia: Python Basics: A Practical Introduction to Python 3. David Amos , Dan Bader , et ál.

Libro para mejorar el estilo de programación: Beyond the Basic Stuff with Python: Best Practices for Writing Clean Code. Al Sweigart.