

Guía docente

270086 - CAP - Conceptos Avanzados de Programación

Última modificación: 09/07/2026

Unidad responsable: Facultad de Informática de Barcelona
Unidad que imparte: 723 - CS - Departamento de Ciencias de la Computación.
Titulación: GRADO EN INGENIERÍA INFORMÁTICA (Plan 2010). (Asignatura optativa).
GRADO EN INTELIGENCIA ARTIFICIAL (Plan 2021). (Asignatura optativa).
Curso: 2026 **Créditos ECTS:** 6.0 **Idiomas:** Catalán

PROFESORADO

Profesorado responsable: JORDI DELGADO PIN - GERARD ESCUDERO BAKX

Otros: Primer quadrimestre:
JORDI DELGADO PIN - 11, 12, 13
GERARD ESCUDERO BAKX - 11, 13

CAPACIDADES PREVIAS

Los alumnos deberían tener conocimientos suficientes de estructuras de datos y algoritmos

REQUISITOS

- Pre-requisito IES
- Pre-requisito PROP

COMPETENCIAS DE LA TITULACIÓN A LAS QUE CONTRIBUYE LA ASIGNATURA

Específicas:

CCO2.2. Capacidad para adquirir, obtener, formalizar y representar el conocimiento humano de una forma computable para la resolución de problemas mediante un sistema informático en cualquier ámbito de aplicación, particularmente los relacionados con aspectos de computación, percepción y actuación en ambientes o entornos inteligentes.

CES1.1. Desarrollar mantener y evaluar sistemas y servicios software complejos y/o críticos.

CES1.3. Identificar, evaluar y gestionar los riesgos potenciales asociados a la construcción de software que pudiesen presentarse.

CES1.7. Controlar la calidad y diseñar pruebas en la producción de software.

CT1.2C. Interpretar, seleccionar y valorar conceptos, teorías, usos y desarrollos tecnológicos relacionados con la informática y su aplicación a partir de los fundamentos matemáticos, estadísticos y físicos necesarios. CEFB1: Capacidad para la resolución de los problemas matemáticos que puedan plantarse en la ingeniería. Aptitud para aplicar los conocimientos sobre: álgebra, cálculo diferencial e integral i métodos numéricos; estadística y optimización.

CT4.1. Identificar las soluciones algorítmicas más adecuadas para resolver problemas de dificultad mediana.

CT4.2. Razonar sobre la corrección y la eficiencia de una solución algorítmica.

CT5.1. Escoger, combinar y explotar diferentes paradigmas de programación, en el momento de construir software, atendiendo a criterios como la facilidad de desarrollo, la eficiencia, la portabilidad y la mantenibilidad.

CT5.2. Conocer, diseñar y utilizar de forma eficiente los tipos y las estructuras de datos más adecuados para la resolución de un problema.

CT5.3. Diseñar, escribir, probar, depurar, documentar y mantener código en un lenguaje de alto nivel para resolver problemas de programación aplicando esquemas algorítmicos y usando estructuras de datos.

Genéricas:

G5. TRABAJO EN EQUIPO: Ser capaz de trabajar como miembro de un equipo, ya sea como un miembro más, o realizando tareas de dirección con la finalidad de contribuir a desarrollar proyectos con pragmatismo y sentido de la responsabilidad, asumiendo compromisos teniendo en cuenta los recursos disponibles.

METODOLOGÍAS DOCENTES

La docencia de la asignatura está estructurada en clases de teoría y clases de laboratorio.

En las clases de teoría los profesores presentan los contenidos esenciales de la asignatura. En las clases de laboratorio se practican los contenidos de la asignatura (los presentados en clase y los adquiridos autónomamente) mediante la realización de problemas prácticos. Las clases de laboratorio serán una continuación de las clases teóricas, donde los conceptos nuevos se implementarán a medida que vayan apareciendo.

OBJETIVOS DE APRENDIZAJE DE LA ASIGNATURA

1. Conocer las capacidades que proporciona a un lenguaje de programación disponer de funciones de orden superior
2. Trabajar las técnicas asociadas a funciones de orden superior en el lenguaje de programación Clojure
3. Conocer qué es un cierre (closure) y algunas técnicas asociadas a su utilización.
4. Conocer un lenguaje de programación dinámico y funcional (híbrido) como Clojure.
5. Conocer estructuras de datos inmutables y qué consecuencias tiene disponer de ellas en un lenguaje de programación.
6. Conocer el concepto de evaluación perezosa y las consecuencias de poder elegir entre distintos tipos de evaluación
7. Ser capaz de desarrollar en equipo ejercicios de laboratorio que utilicen parte de lo que se enseña en el curso

HORAS TOTALES DE DEDICACIÓN DEL ESTUDIANTE

Tipo	Horas	Porcentaje
Horas actividades dirigidas	6,0	4.00
Horas grupo grande	30,0	20.00
Horas grupo pequeño	30,0	20.00
Horas aprendizaje autónomo	84,0	56.00

Dedicación total: 150 h

CONTENIDOS

Introducción a Clojure

Descripción:

Expresiones, condicionales, variables locales (let), definición de funciones, secuencias (listas, vectores, strings), recursividad e iteración.

Funcions de primera-clase

Descripción:

Funciones como parámetro, guardar funciones en estructuras de datos. Maneras de iterar sobre colecciones: reduce, map, filtro, foldr, etc. Retornar funciones

Closures: El modelo de entornos

Descripción:

Contexto léxico (lexical scope), Las funciones no son tales: Capturan el contexto léxico, Si tenemos closures no se necesitan objetos: Ejemplos. Explicar el funcionamiento basado en el modelo de entornos.



Programación con funciones de orden superior: Técnicas

Descripción:

Composición, Pipelining, CPS, TCO & Trampolining

Estructuras de datos inmutables

Descripción:

Ventajas e inconvenientes. El caso de Clojure: Separar la gestión del estado y hacerla explícita (inmutabilidad por defecto)

Secuencias perezosas

Descripción:

Estructuras de datos "infinitas": Aplicaciones.

Macros

Descripción:

El concepto de 'Macro' en el mundo de los lenguajes Lisp. Reflexión.

ACTIVIDADES

Introducción a Clojure

Descripción:

El estudiante debería estar atento en clase y trabajar los ejercicios propuestos por el profesor.

Objetivos específicos:

4

Dedicación: 18h

Grupo grande/Teoría: 4h

Grupo pequeño/Laboratorio: 4h

Aprendizaje autónomo: 10h

Funcions de primera-clase y Closures: El modelo de entornos

Descripción:

Es necesario que el estudiante esté atento en clase y realice los ejercicios propuestos.

Objetivos específicos:

1, 2, 3

Dedicación: 34h

Grupo grande/Teoría: 8h

Grupo pequeño/Laboratorio: 8h

Aprendizaje autónomo: 18h

Programación con funciones de orden superior: Técnicas

Descripción:

Es necesario que el estudiante esté atento en clase y realice los ejercicios propuestos.

Objetivos específicos:

1, 2, 3

Dedicación: 34h

Grupo grande/Teoría: 8h

Grupo pequeño/Laboratorio: 8h

Aprendizaje autónomo: 18h

Control teoría

Objetivos específicos:

1, 2, 3, 4

Dedicación: 14h

Actividades dirigidas: 2h

Aprendizaje autónomo: 12h

Estructuras de datos inmutables

Descripción:

El estudiante debería estar atento en clase y trabajar los ejercicios propuestos por el profesor.

Objetivos específicos:

5

Dedicación: 10h

Grupo grande/Teoría: 2h

Grupo pequeño/Laboratorio: 2h

Aprendizaje autónomo: 6h

Secuencias perezosas

Descripción:

El estudiante debería estar atento en clase y trabajar los ejercicios propuestos por el profesor.

Objetivos específicos:

6

Dedicación: 10h

Grupo grande/Teoría: 2h

Grupo pequeño/Laboratorio: 2h

Aprendizaje autónomo: 6h

Macros

Descripción:

El estudiante debería estar atento en clase y trabajar los ejercicios propuestos por el profesor.

Objetivos específicos:

4

Dedicación: 18h

Grupo grande/Teoría: 4h

Grupo pequeño/Laboratorio: 4h

Aprendizaje autónomo: 10h

Práctica

Descripción:

Fecha de entrega: A lo largo del curso

Objetivos específicos:

7

Competencias relacionadas:

G5. TRABAJO EN EQUIPO: Ser capaz de trabajar como miembro de un equipo, ya sea como un miembro más, o realizando tareas de dirección con la finalidad de contribuir a desarrollar proyectos con pragmatismo y sentido de la responsabilidad, asumiendo compromisos teniendo en cuenta los recursos disponibles.

Control final

Objetivos específicos:

1, 2, 3, 4, 5, 6

Dedicación: 12h

Actividades dirigidas: 2h

Aprendizaje autónomo: 10h

SISTEMA DE CALIFICACIÓN

Cualquier acto de fraude académico, plagio o uso o mera tenencia al alcance de medios no autorizados en cualquier actividad de evaluación comportará la calificación de cero (0) en la prueba o entrega afectada. Además, de acuerdo con la normativa de la Universidad, la posible derivación de los hechos para la apertura de un expediente disciplinario implicará que la asignatura quede en el estado provisional de "pendiente de evaluación" hasta la resolución del expediente. La gestión de estas incidencias se lleva a cabo de acuerdo con el [Marco de actuación para la integridad académica en la evaluación de la UPC](#).

El método de evaluación de la asignatura consistirá en dos pruebas de carácter teórico (P y F), una a mediados de curso y otra al final y ejercicios de laboratorio que se entregaran en grupos de dos personas (L). La evaluación del laboratorio se realizará de la siguiente manera: Los estudiantes harán entrega de entre 5 y 7 ejercicios a lo largo del curso en grupos de dos personas, siendo la nota de laboratorio la media de estas notas.

Entonces, el método de evaluación es = Teoría*0.8 + L*0.2

donde:

Teoría: $\text{MAX}(F, (P+F)/2)$

Competencia transversal "Trabajo en equipo":

Se evalúa usando una rúbrica simple en que el tutor de cada grupo puntúa los diferentes aspectos del trabajo en equipo de cada miembro de los grupos.

BIBLIOGRAFÍA

Básica:

- Fogus, Michael; Houser, Chris. The Joy of Clojure. 2nd edition. Shelter Island: Manning, 2014. ISBN 9781638351283.
- Emerick, Chas; Carper, Brian; Grand, Christophe. Clojure Programming: Practical Lisp for the Java World. O'Reilly Media, 2012. ISBN 9781449394707.
- Jones, Colin. Mastering Clojure Macros: Write Cleaner, Faster, Smarter Code. The Pragmatic Programmers, 2014. ISBN 1941222226.

Complementaria:

- Reade, Chris. Elements of functional programming. Wokingham [etc.]: Addison-Wesley, 1989. ISBN 0201129159.
- Bird, Richard. Thinking Functionally with Haskell. Cambridge [etc.]: Cambridge University Press, 2014. ISBN 9781107452640.