Universitat Autònoma de Barcelona

2023/2024

Fundamentos de Programación I

Código: 106553
Créditos ECTS: 9

Titulación	Tipo	Curso	Semestre
2504392 Inteligencia Artificial	FB	1	1

Contacto

Nombre: Alicia Fornes Bisquerra

Correo electrónico: alicia.fornes@uab.cat

Idiomas de los grupos

Puede consutarlo a través de este enlace. Para consultar el idioma necesitará introducir el CÓDIGO de la asignatura. Tenga en cuenta que la información es provisional hasta el 30 de noviembre del 2023.

Prerrequisitos

Ninguno

Objetivos y contextualización

Esta asignatura pretende proporcionar una introducción general a la programación. Los objetivos generales son:

- Comprender el ciclo de vida del software: análisis del problema, diseño, implementación y prueba.
- Diseño de algoritmos para la resolución de problemas, con una rigurosa metodología de programación estructurada.
- Introducir un lenguaje de programación real, percibiendo la diferencia entre una notación pseudoalgorítmica y la sintaxis de un lenguaje de programación real. Incluye la comprensión de los aspectos léxicos (palabras válidas del idioma), sintácticos (reglas para combinarlas) y semánticos (significado).
- Comprender y aplicar los conceptos básicos de programación: estructuras de control, estructuras de datos y subprogramas.
- Comprender y aplicar los principios básicos de la programación orientada a objetos: concepto de clase, objeto, encapsulación de datos, herencia.
- Introducir el concepto de recursividad y su aplicación.
- Familiarizarse con los entornos de desarrollo, siguiendo unas normas de estilo y reglas para una mejor comprensión del código: comentarios, sangrado, etc.

Competencias

- Analizar y resolver problemas de forma efectiva, generando propuestas innovadoras y creativas para alcanzar los objetivos.
- Conocer y utilizar de forma eficiente las técnicas y herramientas de representación, manipulación, análisis y gestión de datos a gran escala.
- Diseñar, implementar, analizar y validar soluciones algorítmicas eficientes y robustas a problemas computacionales derivados del diseño de sistemas inteligentes.
- Trabajar cooperativamente para la consecución de objetivos comunes, asumiendo la propia responsabilidad y respetando el rol de los diferentes miembros del equipo.

Resultados de aprendizaje

1. Analizar y resolver problemas de forma efectiva, generando propuestas innovadoras y creativas para alcanzar los objetivos.
2. Aplicar estrategias de depuración, prueba y corrección para verificar el funcionamiento correcto de una solución algorítmica de acuerdo con los requisitos del problema a resolver.
3. Aplicar los principios de la programación orientada a objetos en el desarrollo de programas.
4. Comprender las estructuras de datos comunes y los algoritmos que las utilizan y manipulan.
5. Conocimiento de las limitaciones y ventajas de diferentes representaciones de datos y capacidad para elegir las estructuras más eficientes que faciliten el posterior análisis y explotación de los datos.
6. Desarrollar programas con un buen estilo de programación y bien documentados.
7. Seleccionar y utilizar las estructuras algorítmicas y de representación de los datos apropiadas para la resolución de un problema.
8. Trabajar cooperativamente para la consecución de objetivos comunes, asumiendo la propia responsabilidad y respetando el rol de los diferentes miembros del equipo.

Contenido

- Resolución de problemas: Introducción a la algorítmica y programación.
- Conceptos básicos y estructuras de control
- Estructuras de datos.
- Subprogramas
- Ficheros
- Prevención y detección de errores
- Introducción a la programación orientada a objetos
- Introducción a la recursividad

Metodología

Se utilizará la plataforma Caronte (http://caronte.uab.cat/) para compartir los materiales, entregar los trabajos/proyectos, consultar las notas de la asignatura, comunicarse con el profesorado, etc. Para matricularse en el curso, hay que matricularse (con nombre, NIU y foto) y matricularse de asignatura (el código de la asignatura se facilita el primer día de clase).

La metodología docente se centrará principalmente en el trabajo práctico. Se organizarán sesiones presenciales para la discusión de los contenidos teóricos de la asignatura, seguidas de ejercicios y problemas de programación. Concretamente, los diferentes tipos de actividades docentes son los siguientes:

- Exposición de contenidos. Las actividades de teoría están dirigidas a consolidar los aspectos más teóricos de la materia, desde una perspectiva muy práctica con ejemplos. Algunos de estos contenidos teóricos deben haber sido preparados antes de la clase: lectura de textos, visionado de vídeos, búsqueda de información, etc.
- Clases expositivas participativas: Resolución conjunta del conjunto de problemas propuestos para consolidar los contenidos teóricos. Todos los problemas estarán disponibles en la plataforma de

Caronte, y serán autoevaluables. Estas actividades permiten al alumno profundizar en la comprensión y obtener un conocimiento personalizado. Son autoevaluables para permitir ajustar el ritmo de consolidación y reflexionar sobre nuestro propio aprendizaje.
- Proyectos: Programación de proyectos prácticos breves para profundizar en los conceptos teóricos aplicados. Estos proyectos se resolverán en pequeños grupos, donde cada miembro deberá hacer una parte del trabajo y ponerlo en común con el resto del grupo para tener la solución final.
- Las tutorías estarán disponibles de forma gratuita para que los alumnos resuelvan dudas relativas a la asignatura.

Dentro del horario marcado por el centro o titulación, se reservarán 15 minutos de una clase para que los alumnos evalúen a sus profesores y sus cursos o módulos a través de cuestionarios.

Nota: se reservarán 15 minutos de una clase dentro del calendario establecido por el centro o por la titulación para que el alumnado rellene las encuestas de evaluación de la actuación del profesorado y de evaluación de la asignatura o módulo.

Actividades

Título	Horas	ECTS	Resultados de aprendizaje
Tipo: Dirigidas			
Seminarios de resolución de problemas	50	2	1, 3, 2, 6, 7
Sesiones teóricas	25	1	4, 5
Tipo: Supervisadas			
Proyectos de programación	50	2	1, 3, 2, 4, 5, 6, 7, 8
Tipo: Autónomas			
Resolución de problemas	60	2,4	1, 3, 4, 5, 6, 7
Trabajo individual	35	1,4	1, 3, 2, 4, 5, 7

Evaluación

La evaluación será continua, por tanto no se ofrece la posibilidad de evaluación única.

- Exámenes (Par): Se realizarán dos exámenes teórico-prácticos escritos individuales. El primero (Par1) se realizará aproximadamente a la mitad del semestre. El segundo (Par2) se realizará al final del semestre y evaluará los conceptos teóricos no evaluados en el Par1.
- Examen de recuperación: En caso de que la Nota de Teoría no alcance el aprobado (5), el alumno podrá realizar un examen de recuperación sobre la parte suspendida.
- Problemas (NProb): La mayoría de las semanas habrá la posibilidad de entregar actividades trabajadas durante la semana. La entrega es a través de Caronte.
- Proyectos (Proj): La nota de los proyectos se obtendrá de la media ponderada de las notas obtenidas en los dos proyectos. Cada nota será la media aritmética de la nota de grupo y la nota individual. Cada una de las entregas debe aprobarse por separado. En caso de suspender alguna entrega, existirá la opción de recuperar la parte suspendida. La nota de la recuperación será un máximo de 5.

INDICADORES DE LA NOTA:

La nota final se calcula de la siguiente manera:

- Examen Parcial 1 = Par1
- Examen Parcial 2 = Par2
- Nota Teoría = 0'5*Par1 +0'5*Par2
- Nota de proyecto = 0'5*Proy1 + 0'5*Proy2
- NOTA FINAL DE CURSO = 0'5 * Nota de Teoría + 0'4 * Nota de Proyecto + 0'1 * NProb

CRITERIOS DE EVALUACIÓN

- Para la consideración de la Nota de Teoría es necesario obtener una nota mínima de 4 en cada parcial para realizar la media.
- Para considerar la Nota Proyecto, es necesario obtener una nota mínima de 4 en cada entrega para poder hacer la media.
- Si el alumno suspende alguna de las entregas del proyecto, la puntuación máxima que puede obtener en la recuperación es de 5.
- Para aprobar cada parte (teoría y proyecto), se debe obtener un mínimo de 5.
- La asignatura será aprobada si la PUNTUACIÓN FINAL DE LA CURSO es mayor o igual a 5.
- En caso de no alcanzar el mínimo exigido en alguna de las actividades de evaluación, la nota numérica del expediente será el valor inferior entre 4,5 y la media ponderada de las notas.

NO EVALUABLE: Si el alumno no entrega ninguna actividad de evaluación.

ESTUDIANTES REPETIDORES: Ninguna parte aprobada por separado (teoría, proyecto) se conserva/convalida de un curso académico a otro.

MATRÍCULA DE HONOR (MH): La concesión de matrícula de honor es decisión del profesorado responsable de la asignatura. La normativa de la UAB indica que sólo se puede conceder MH a los alumnos que hayan obtenido una nota final igual o superior a 9,00. Se puede otorgar hasta un 5% de MH del número total de estudiantes matriculados.

CALENDARIO DE EVALUACIÓN:

- Exámenes parciales: según el calendario académico de la Escuela de Ingeniería.
- Examen de Recuperación: según el calendario académico de la Escuela de Ingeniería.
- Entregas de actividades: fecha y hora fijada con antelación en Caronte.
- Entregas del proyecto: fecha y hora fijada con antelación en Caronte.

Las fechas de entregas en Caronte pueden estar sujetas a cambios de programación por motivos de adaptación a posibles incidencias. Estos cambios siempre serán comunicados en Caronte, ya que es el mecanismo habitual de intercambio de información entre profesores y alumnos.

Para cada actividad de evaluación se indicará un lugar, fecha y hora de revisión en la que el alumno podrá revisar la actividad con el profesor. En este contexto, se podrán realizar reclamaciones sobre la calificación de la actividad, que será evaluada por los profesores responsables de la asignatura. Si el alumno no acude a esta revisión, la actividad no se revisará más adelante.

COPIAS Y PLAGIO

Sin perjuicio de otras que se estimen oportunas y de acuerdo con la legislación académica vigente en materia de disciplina, las irregularidades cometidas por un alumno que puedan dar lugar a una variación de la calificación se calificarán con una nota cero (0). Las actividades de evaluación clasificadas de esta forma y por este procedimiento no serán recuperables. Si es necesario aprobar alguna de estas actividades de evaluación para aprobar la asignatura, se suspenderá directamente esta asignatura, sin posibilidad de recuperarla en el mismo curso. Estas irregularidades incluyen, entre otras:

- la copia total o parcial de una práctica, informe o cualquier otra actividad de evaluación;
- dejar que otros copien el examen/trabajo;
- presentar un trabajo en grupo que no haya sido realizado en su totalidad por los miembros del grupo;
- presentar como propios aquellos materiales producidos por un tercero, así sean traducciones o adaptaciones, y en general obras con elementos no originales y exclusivos del estudiante;

- disponer de dispositivos de comunicación (como teléfonos móviles, relojes inteligentes, etc.) accesibles durante las pruebas de evaluación teórica - prácticas individuales (exámenes).
- En caso de que el alumno haya cometido irregularidades en alguna parte de la evaluación (y por tanto no se podrá aprobar vía compensación), la nota numérica de la asignaturaserá el valor inferior entre 3,5 y la media ponderada de las notas. En resumen: copiar, dejar que otros copien tu trabajo o plagiar en cualquiera de las actividades de evaluación equivale a un suspenso con una nota inferior a 3,5.

Actividades de evaluación continuada

Título	Peso	Horas	ECTS	Resultados de aprendizaje
1er examen parcial	25	2	0,08	1, 3, 4, 5, 6, 7
2º examen parcial	25	2	0,08	1, 3, 2, 4, 5, 6, 7
Entrega de problemes	10	0	0	1, 3, 2, 4, 5, 6, 7
Proyectos de programación	40	1	0,04	1, 3, 2, 4, 5, 6, 7, 8

Bibliografía

- J. Guttag. Introduction to Computation and Programming Using Python: With Application to Understanding Data. Second Edition. MIT Press. ISBN-10: 9780262529624.
- S. Chazallet Python 3. Los fundamentos del lenguaje. Eni, ISBN-10: 2409006140.
- E. Matthes. Python Crash Course: A Hands-On, Project-Based Introduction to Programming. No Starch Press ISBN-10: 1593276036.
- M. Myers. A Smarter Way to Learn Python: Learn it faster. Remember it longer. Createspace Independent Pub ISBN-10: 1974431479.
- D. Phillips, C. Giridhar, S. Kasampalis. Python: master the art of design patterns. Packt Publishing, 2016.
- Steven F. Lott. Mastering object-oriented Python. Packt publishing, 2014.
- Clean code: a handbook of agile software craftmanship. R.C. Martin. Prentice Hall, 2008.

Software

Anaconda, que incluye Python y Spyder (https://www.anaconda.com/download)