Universitat Autònoma de Barcelona

2022/2023

Programación Orientada a los Objetos

Código: 104389
Créditos ECTS: 6

Titulación	Tipo	Curso	Semestre
2503740 Matemática Computacional y Analítica de Datos	FB	1	2

Contacto

Nombre: Joan Serrat Gual
Correo electrónico: joan.serrat@uab.cat

Uso de idiomas

Lengua vehicular mayoritaria: catalán (cat)
Algún grupo íntegramente en inglés: No
Algún grupo íntegramente en catalán: Sí
Algún grupo íntegramente en español: No

Otras observaciones sobre los idiomas

Los materiales de estudio (transparencias, enunciados etc) estan en inglés.

Equipo docente externo a la UAB

Abel Serrat Castella

Prerequisitos

No hay prerrequisitos oficiales, pero es necesario saber programar en Python.

Objetivos y contextualización

Una de las metodologías más extendidas de diseño y programación es la orientada a objetos, según la cual un software se organiza en clases que contienen métodos (procedimientos) y atributos (datos). Aparte del concepto de clase, tres otros elementos fundamentales son la herencia, la composición y la separación de las partes pública y privada de las clases. Estos elementos son pues necesarios para la programación OO, pero no suficientes: hay que saber cómo utilizarlos para llegar a soluciones que sean fácilmente extensibles en el futuro, ya que una ley del software es que los cambios son inevitables. Existen una serie de principios de diseño o heurísticas que nos dicen cómo emplear estos elementos con el fin de conseguirlo. Y sobre ellos todavía, se construyen los patrones de diseño, que son soluciones a nivel de diseño OO a problemas recurrentes, que aparecen a menudo en resolver problemas aparentemente independientes. En esta asignatura se introducen y se practican los conceptos OO así como los principios y patrones de diseño.

El aprendizaje gira en torno a un proyecto --- la práctica --- en que es necesario aplicar todos los temas explicados para diseñar e implementar una aplicación de software. Por el contexto de la asignatura, el proyecto esta relacionado con el aprendizaje computacional (machine learning). Los lenguajes de programación utilizados en la asignatura serán Python principalmente por que es en el que se hará el proyecto, pero también Java que utilizaremos en algunos ejercicios y soporta mejor los conceptos de orientación a objeto. Así, podremos comparar cómo uno y otro los implementa y en qué grado.

Competencias

- Aplicar conocimientos básicos sobre la estructura, el uso y la programación de ordenadores, sistemas operativos y programas informáticos para solucionar problemas de distintos ámbitos.
- Diseñar, desarrollar y evaluar soluciones algorítmicas eficientes para problemas computacionales de acuerdo con los requisitos establecidos.
- Evaluar de manera crítica y con criterios de calidad el trabajo realizado.
- Que los estudiantes hayan demostrado poseer y comprender conocimientos en un área de estudio que parte de la base de la educación secundaria general, y se suele encontrar a un nivel que, si bien se apoya en libros de texto avanzados, incluye también algunos aspectos que implican conocimientos procedentes de la vanguardia de su campo de estudio.
- Que los estudiantes puedan transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.
- Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.
- Trabajar cooperativamente en un contexto multidisciplinar asumiendo y respetando el rol de los diferentes miembros del equipo.
- Utilizar eficazmente bibliografía y recursos electrónicos para obtener información.

Resultados de aprendizaje

1. Comprender los principios básicos de la lógica de los computadores.
2. Conocer los conceptos básicos de la estructura y la programación de los computadores.
3. Describir el funcionamiento básico de los sistemas de cómputo.
4. Evaluar de manera crítica y con criterios de calidad el trabajo realizado.
5. Evaluar y analizar la complejidad computacional de las soluciones algorítmicas para poder desarrollar e implementar aquella que garantice el mejor rendimiento.
6. Que los estudiantes hayan demostrado poseer y comprender conocimientos en un área de estudio que parte de la base de la educación secundaria general, y se suele encontrar a un nivel que, si bien se apoya en libros de texto avanzados, incluye también algunos aspectos que implican conocimientos procedentes de la vanguardia de su campo de estudio.
7. Que los estudiantes puedan transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.
8. Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.
9. Reconocer e identificar los métodos, sistemas y tecnologías propios de la computación.
10. Seleccionar y utilizar las estructuras algorítmicas y de representación de los datos apropiadas para la resolución de un problema.
11. Trabajar cooperativamente en un contexto multidisciplinar asumiendo y respetando el rol de los diferentes miembros del equipo.
12. Utilizar eficazmente bibliografía y recursos electrónicos para obtener información.
13. Utilizar los sistemas operativos y programas de uso común en distintos campos.
14. Verificar y asegurar el funcionamiento correcto de una solución algorítmica de acuerdo con los requisitos del problema a resolver.

Contenido

1. Conceptos de orientación a objeto: encapsulamiento, herencia, composición, interfaz. Diagramas de clase UML
2. Programación orientada a objeto en Python y Java
3. Asignación de responsabilidades: patrones GRASP
4. Principios generales de diseño orientado a objeto
5. Patrones de diseño: creacionales, estructurales, de comportamiento
6. Estilo de codificación, *logging*, comentarios y documentación

Metodología

- Clases de teoría. En ellas el profesor introduce los contenidos teóricos y da referencias útiles (libros, artículos, páginas web) para que el estudiante después pueda continuar su aprendizaje, así como desarrollar el proyecto propuesto. Las transparencias utilizadas en estas clases pues no deben tomarse como única fuente de estudio. Hay que estudiar los libros recomendados, y los artículos, capítulos de libros de acceso abierto y otros, disponibles en la página web de la asignatura. En estas clases también resolveremos pequeños problemas, de diseño y / o programación OO en Java y Python.

- Sesiones de prácticas. Los estudiantes deberán formado a principio de curso grupos pequeños. En estas sesiones los estudiantes vienen a 1) revisar con el profesor del trabajo hecho cada semana del proyecto y resolver dudas, y 2) a ser evaluados de manera continua, de acuerdo con el baremo que figura en el enunciado de la práctica. Por lo tanto, la práctica *no* se hace sólo durante estas sesiones sino que pide un trabajo previo.

Todos los materiales de la asignatura junto con su planificación detallada se encontrarán en el campus virtual o bien a una página web que ya se anunciará. La mayoría de estos materiales son en inglés.

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			
Clases de teoria	75	3	5, 2, 3, 9, 10
clases de prácticas	75	3	7, 11, 13

Evaluación

Una parte de la evaluación es en grupo y otra individual. La nota de la primeraa (PR) corresponde a la práctica. La segunda (EX) al promedio de dos pruebas escritas durante el trimestre (EX1, EX2), o bien del examen de recuperación (EXR).

El algoritmo para el cálculo de la nota final (FM) es el siguiente (todas las notas son sobre 10) :

EX = max(0.5 * (EX1 + EX2), EXR)
if (EX >= 4.0) and (PR >= 4.0) :
FM = 0.5 * (EX + PR)
else :
FM = min(EX, PR)

La práctica

La realización de la práctica és esencial para asimilar los conceptos de teoria. La primera cosa pues que queremos asegurar es que la pràctica haya sido realmente hecha por el grupo que la presenta, y que cada miembro del grupo haya contribuido en la misma medida que los demás. La segunda es que la práctica se haga de manera progresiva durante el curso, en paralelo a la teoria. Por eso haremos los siguiente:

- La nota de la práctica se obtiene combinando la nota de les partes en las que ésta se dividide. Cada parte será evaluada in situ en una de les sesiones de prácticas que anunciaremos a principio de curs ("hitos"). Los puntos que se pueden obtener de cada parte s'especificaran en el enunciado. Estas avaluaciones no tienen recuperación después.

- Los dias de avaluació de la práctica (hitos), si de resultas de las respuestas a les preguntas que haga el profesor durante la evaluación, tenemos dudas sobre la autoria o la contribución de alguno de los miembros del grupo, entonces la nota de práctica pasará de ser de grupo a individual y en función de cuanta i qué parte de la práctica haya hecho cada miembro.

Los ejercicios

Propondremos algunos ejercicios que son opcionales. Estos ejercicios se pueden hacer indivualmente o en pequeño grup, el mismo o diferente que el de práctiques. Los ejercicios pueden sumar punts a la nota de teoria EX, hasta un máximo de 10. Decimos 'pueden' porque la obtenciób de estos puntos extra estará condicionada a una prueba de validación para comprobar que el estudiant ha realitzado efectivamente exercicio que ha entregado. Estas pruebas de validación las haremos sólo una vez, en los exámenes EX1 y EX2, y no tienen recuperación después.

Otros

No convalidamos prácticas, exercicios ni exámenes de cursos anteriores. Los estudiantes repetidores no reciben ningún tratamiento diferenciado.

Si no se entrega un examen o una parte de la práctica, la nota de esta parte es cero. La nota final será 'no evaluable' si no se ha entregado nada evaluable, ya sea examen, práctica o ejercicio.

Los hitos de práctiques son evaluaciones equivalentes a examenes escritos. Si un estudiant no se presenta, la nota és cero.

Las notas MH las concederemos de manera discrecional a los estudiantes con nota final superior a 9.0 y teniendo en cuenta todo tipo de trabajos hechos (exámenes, práctica y ejercicios).

Las fechas de evaluación continuada y entrega de trabajos se publicaran en el campus virtual de la assignatura y pueden estar sujectas a cambios en respuesta a posibles incidéncies. Siempre se informará mediante el campus virtual, que será el mecanismo de comunicación entre docentes y estudiantes.

Para las actividades de evaluación de tipo examen escrito, se indicará un lugar, fecha y hora de revisión en la que el estudiantepodrá revisar la activitat con el professor. Si el estudiante no se presenta a esta revisión, no se revisará posteriorment esta actividad.

Sin perjuicio de otras medidas disciplinarias que se estimen oportunas, y de acuerdo con la normativa académica vigente, se calificarán con un cero las irregularidades cometidas por el estudiante que puedan conducir a una variación de la calificación de un acto de evaluación. Por lo tanto, plagiar, copiar o dejar copiar la práctica o cualquier otra actividad de evaluación implicará suspender con un cero y no se podrá recuperar en el mismo curso académico. Si esta actividad tiene una nota mínima asociada, entonces la asignatura quedará suspendida.

Actividades de evaluación

Título	Peso	Horas	ECTS	Resultados de aprendizaje
Exámenes	50%	0	0	5, 2, 3, 6, 7, 8, 9, 10
Práctica	50%	0	0	4, 1, 2, 3, 7, 9, 11, 12, 13, 14

Bibliografía

Básica

Las clases de teoría están basadas en un juego de transparencias, que son un resumen, una síntesis de conceptos que se encuentran más bien explicados en estos libros, que vale la pena leer o consultar para comprenderlos mejor y para ampliar conocimientos.

Patrones de diseño. E. Gama, R. Helm, R. Johnson, J. Vlissides. Pearson Educación 2003.
Es un clásico del patrones de diseño orientado a objeto, por los autores que propusieron este concepto. La versión original en inglés es de 1995 y por ello sus ejemplos y diagramas son ya antiguos. Pero explica muy bien los conceptos.

Design patterns explained simply. Alexander Shvets. Libro on-line, https://sourcemaking.com/design-patterns-ebook. Tambien del mismo autor otro libro online en https://refactoring.guru/design-patterns
Ben explicado e ilustrado con ejemplos, que además encontramos programados en Java, Python, C ++ y más.

Head first design patterns. E. Freeman, E. Freeman, K. Sierra, B. Bates. O'Reilly, 2004.
Un libro muy visual sobre patrones de diseño orientado a objeto. Muy didáctico, los explica paso a paso y con ejemplos en Java. Recientemente ha aparecido una segunda edición (2020).

Hands-on machine learning with Scikit-Learn, Kera, and TensorFlow: conceptos, tools, and techniques to build inteligente systems. Aurélien Gerona. O'Reilly, 2.019.
La práctica consiste en el diseño y programación orientada a objeto de un método de clasificación y regresión conocido como random forest. Tres capítulos de este libro explican muy bien los conceptos alrededor de este método y el algoritmo mismo que hay que implementar.

Complementaria

Introduction to Programming Using Java, David J. Eck.2020.
Libro online, http://math.hws.edu/javanotes/. Algunos ejercicios son en lenguaje Java y este es un libro online y actualizado que explica el lenguaje desde cero. No entra en los detalles más avanzados pero es más que suficiente por lo que necesitamos.

Python: master the art of design patterns. D. Phillips, C. Giridhar, S. Kasampalis. Packt Publishing, 2016.
Un libro muy extenso (700pp) sobre patrones de diseño en Python. Un aspecto muy positivo es que muestra cómo programarlos en Python en detalle, uno a uno.

Mastering object-oriented Python. Steven F.Lott. Packt publishing, 2014.
Python avanzado, o más bien, entrando en los detalles del lenguaje. "Grasp the intricacies of object-oriented programming in Python in order to efficiently build powerful real-world applications". De interés el capítulo de logging y algún otro.

Software

- Python en la versión Anaconda, que ya incluye las librerias numpy, matplotlib.
- Pycharm (en lugar de Spyder) en la versión "community edition", más el plugin PlantUML.
- IntelliJ IDEA también "community edition"