

Planificación de la Asignatura: Programación Avanzada - Bioingeniería

Fecha: 23/10/2024 13:02

Código: B0810

Carrera: Bioingeniería

Departamento Académico: Informática

Docente a cargo: Javier Eduardo Díaz Zamboni

Correo del docente a cargo: javier.diaz@uner.edu.ar

Régimen de Dictado: Cuatrimestral doble oferta

Carga Horaria Semanal: 8 horas semanales

Carga Horaria Total: 112 horas

Contenidos Mínimos:

Diseño y programación orientada a objetos. Diseño de algoritmos. Encapsulamiento y abstracción.

Introducción al análisis algoritmo básico. Clases contenedoras y protocolos de iteradores. Recursividad.

Organización de archivos. Software gráfico; uso de API's para graficación; coordenadas homogéneas.

Construcción de un sistema de tamaño medio, en equipo, teniendo en consideración la eficiencia de los algoritmos. Algoritmos básicos de cálculo numérico.

Competencias Genéricas:

TECNOLÓGICAS

CT 1: Identificación, formulación y resolución de problemas de Bioingeniería. Nivel de Dominio 1.

CT 2: Concepción, diseño y desarrollo de proyectos de Bioingeniería. Nivel de Dominio 1.

CT 4: Utilización de técnicas y herramientas de aplicación en Bioingeniería. Nivel de Dominio 1.

CT 5: Contribución a la generación de desarrollos tecnológicos y/o innovaciones tecnológicas. Nivel de Dominio 1.

SOCIALES, POLÍTICAS Y ACTITUDINALES

CS 1: Fundamentos para el desempeño en equipos de trabajo. Nivel de Dominio 1.

CS 2: Fundamentos para una comunicación efectiva. Nivel de Dominio 1.

CS 5: Fundamentos para el aprendizaje continuo y autónomo. Nivel de Dominio 1.

Competencias Específicas:

CE 1.1: Diseñar, calcular y proyectar instalaciones, equipamientos e instrumental de tecnología biomédica, procesamiento de señales biomédicas y sistemas derivados de biomateriales utilizados en el área de la salud. Nivel de Dominio 1.

CE 1.2: Procesar señales e imágenes biológicas. Nivel de Dominio 1.

Argumentación de aportes marcados en la matriz de competencias:

La asignatura de Programación Avanzada proporciona una base sólida para el desarrollo fundamental de competencias clave de los estudiantes. Aunque el aporte es inicial, nivel 1 de 3, sienta las bases para su desarrollo posterior así como para otras competencias de egreso más avanzadas. Al enseñar el paradigma de diseño y programación orientada a objetos con una carga teórico-práctica significativa en relación a otros contenidos afines y el marco de un trabajo metodológico que implica diseño, desarrollo, depuración y testeo, los estudiantes desarrollan habilidades para abordar problemas de complejidad media en el contexto de la ingeniería integrando el recurso computacional como herramienta de trabajo. Esta base es fundamental como aporte al conjunto de competencias genéricas indicadas como la resolución de problemas y el trabajo en equipo, así como a las competencias específicas a las cuales se aporta como lo son el diseño y construcción de software. Además, al estar ubicada en el ciclo básico de la carrera, esta asignatura prepara a los estudiantes para espacios curriculares posteriores que seguirán fortaleciendo estas competencias y desarrollándolas hasta su nivel de egreso.

Correlativas Regulares para cursar:

Fundamentos de Programación

Cálculo en una variable

Álgebra Lineal y Geometría Analítica

Correlativas Aprobadas para cursar:**Correlativas Aprobadas para promocionar o rendir el examen final:**

Fundamentos de Programación

Insercion de la Asignatura en el plan de Estudios:

La asignatura Programación Avanzada, con un seguimiento docente muy cercano a los estudiantes, les aporta al desarrollo de la capacidad crítica y creativa, al desarrollo de habilidades para el eficiente uso de los recursos informáticos, tecnológicos y de la programación, habilidades para el aprendizaje continuo y autónomo, habilidades de comunicación efectiva y disposición al trabajo en equipo para la resolución de problemas de la ciencia y la ingeniería.

Programación Avanzada, junto con Fundamentos de Programación y Algoritmos y Estructuras de Datos, son asignaturas que brindan a la formación de los futuros profesionales conocimientos y herramientas tecnológicas y metodológicas básicas de las ciencias de la computación y la ingeniería del software. En el trayecto de los estudiantes por estos cursos desarrollan habilidades para el manejo de las herramientas computacionales, fomentando la tarea creativa como una actividad metódica y sistemática, desarrollando habilidades iniciales para el reconocimiento, planteo de problemas, aplicación de métodos de resolución, diseño e implementación de la solución e interpretación de los resultados.

El curso de Programación Avanzada aporta al desarrollo inicial de competencias generales de los futuros profesionales y, también, al desarrollo de competencias específicas relacionadas al diseño, análisis y procesamiento de datos y la resolución de problemas de la ciencia y la ingeniería empleando computadoras y lenguajes de programación de alto nivel como herramientas de trabajo. Por lo tanto, se espera que el aporte en ambos tipos de competencia permita la continuidad de su desarrollo en los cursos curriculares posteriores de la carrera.

Objetivo General:

Lograr que el alumno integre el recurso informático a su proceso de formación básica, científica y técnica, favoreciendo el desarrollo de habilidades en el uso de la programación y los recursos informáticos, el pensamiento lógico y crítico dentro del contexto de trabajo grupal con el propósito de desarrollar habilidades y actitudes para el trabajo en equipos a fin de aplicarlos de manera efectiva en la actividad profesional.

Objetivos Particulares:

Que las y los estudiantes puedan:

- Continuar su desarrollo en el diseño de programas iniciada en Fundamentos de Programación, con especial atención al estilo y la expresión, en la depuración y pruebas, sobre todo tratándose de programas de mayor extensión y razonamiento lógico para fomentar el aprendizaje continuo y autónomo.
- Aplicar análisis y diseño orientado a objetos para resolver problemas de ingeniería.
- Desarrollar habilidades de documentación de parte del proceso de desarrollo de software como una forma de comunicación efectiva.
- Diseñar y probar clases que representan datos y comportamientos acorde a cada problema particular para fomentar competencias de diseño y desarrollo.
- Realizar un análisis crítico de los resultados arrojados por la computación determinando su validez para fomentar la capacidad de análisis y la formación de criterios.
- Utilizar entornos de desarrollo para incrementar la eficiencia personal y profesional.
- Comprender los fundamentos de la computación científica para su aplicación correcta en problemas de ingeniería de baja complejidad.
- Adquirir capacidades de incorporación de bibliotecas realizadas por terceros a programas propios para potenciar la capacidad de resolución de problemas de ingeniería.
- Aprender a manejar recursos bibliográficos y otras fuentes de información para la resolución de problemas técnicos y para desarrollar habilidades de aprendizaje en forma continua y autónoma.

Programa Analítico:

Unidad 1 – Programación orientada a objetos.

Clases y objetos. Definición de una clase. Miembros de clase e instancia: atributos y métodos. Métodos especializados: constructores y destructores. Especificación de acceso. Instanciación de objetos.

Contención. Herencia. Clases abstractas. Polimorfismo. Depuración de errores. Unidades de testeo.

Unidad 2 – Análisis y diseño orientado a objetos.

Elementos fundamentales y secundarios del modelo de objetos. Introducción al diseño orientado a objetos.

Diseño dirigido por responsabilidades. Pruebas del diseño. Herramientas de documentación del análisis y diseño: UML.

Unidad 3 – Biblioteca estándar.

Introducción a la biblioteca estándar del lenguaje. Tipos integrados. Principales algoritmos y estructuras de datos incorporadas. Iteradores. Persistencia de datos. Interacción con el sistema operativo.

Unidad 4 – Manejo de excepciones.

Introducción al manejo de excepciones: definición de unidad trabajo (función), precondiciones y postcondiciones. Lanzamiento y procesamiento de excepciones. Excepciones de la Biblioteca Estándar. Excepciones definidas por el usuario.

Unidad 5 – Graficación.

Fundamentos matemáticos para la graficación: primitivas geométricas, transformaciones geométricas y coordenadas homogéneas. Manejo de color en gráficas. APIs para visualización de información y datos científicos.

Unidad 6 – Interfaces de usuario.

Introducción a interfaces de usuario. Interfaz de consola. Interfaz gráfica de usuario. Manejo de eventos. Componentes gráficos.

Unidad 7 – Algoritmos computacionales y numéricos.

Algoritmos recursivos: Búsqueda binaria, ordenamiento rápido. Algoritmos numéricos: bisección, diferencia finita e integración numérica.

Metodología Didáctica:

El dictado de la asignatura será de carácter teórico y práctico, estableciéndose en primer lugar los conceptos teóricos seguido de una aplicación inmediata por medio de estudio dirigido y trabajos prácticos sobre la computadora.

Las instancias para el desarrollo de la asignatura serán clases teóricas, prácticas y de coloquio. Para el desarrollo de cada una de las partes en que se divide la asignatura, se partirá de elementos simples sobre los que se estructurará el conocimiento. El docente actuará como orientador y estimulador en la resolución de problemas.

La materia cuenta con una carga horaria semanal de 8 horas, las que se distribuirán de la siguiente forma: Una clase semanal de teoría o coloquio de 3 hs. que estará a cargo del docente titular y/o adjunto, en colaboración con los ayudantes. Las prácticas serán de 5 hs a cargo de Jefes de Trabajos Prácticos en colaboración con Auxiliares Docentes, con una distribución de usos de laboratorios de computación acordada con la coordinación académica.

Formación Práctica:

En las prácticas se trabajará sobre un conjunto de Trabajos Prácticos siendo uno de ellos de carácter integrador, mediana complejidad y características más reales, cuyo desarrollo abarca la mayor parte del cuatrimestre. Esta complejidad requiere que los alumnos trabajen en forma grupal con división de tareas, lo que conlleva a que los integrantes del grupo realicen regularmente puestas en común sobre lo que están estudiando y desarrollando. La presentación y defensa de los TPs para obtener la regularidad se realiza regularmente y en forma grupal.

Listado de Actividades de Formación Práctica:

- Guía de Trabajo Práctico N° 1: Introducción a la asignatura.
- Guía de Trabajo Práctico N° 2: Diseño y programación orientada a objetos.
- Guía de Trabajo Práctico N° 3: Proyecto de desarrollo integrador.
 - Análisis y diseño.
 - Desarrollo, pruebas y depuración.
 - Entrega del producto final: software y documentación.

Intensidad de la formación práctica

Detalle de la carga horaria total prevista para cada una de las siguientes actividades:

Actividades prácticas que aportan a las competencias específicas en el Nivel de dominio 1: 90 horas

Actividades prácticas que aportan a las competencias específicas en el Nivel de dominio 2: 0 horas

Actividades prácticas que aportan a las competencias específicas en el Nivel de dominio 3: 0 horas

Horas totales de actividades de formación práctica: 90 horas

Metodología de Evaluación Durante el cursado:

La evaluación se realiza en dos partes, las cuales en conjunto permiten indagar las capacidades adquiridas en resolución de problemas, habilidades de programación y manejo de conceptos de la asignatura. Para lograr esto, por un lado, los alumnos deben desarrollar, entregar y defender Trabajos Prácticos, y por otro, completar una evaluación de conceptos.

Regularidad:

Evaluación de Trabajos Prácticos

Los alumnos deberán resolver, entregar, defender en forma teórico-práctica de manera grupal tres trabajos prácticos con la guía y supervisión del JTP. Los trabajos prácticos se realizan en grupos de dos personas. La defensa se realizará en los horarios de las clases de práctica, debiendo estar presentes los integrantes del grupo. La aprobación de los trabajos prácticos es grupal, por lo que cada miembro del grupo debe responder correctamente a las preguntas formuladas por los docentes. En caso de que un grupo no apruebe algún trabajo práctico puede volver a defender en la siguiente instancia de entrega y defensa.

Respecto al trabajo práctico integrador se realizarán entregas parciales por etapas organizadas por el docente a cargo de la práctica.

Evaluación de conceptos

Esta evaluación está dividida en dos exámenes parciales basados en cuestionarios de carácter teórico-práctico de resolución individual, los cuales deben ser aprobados con una nota mayor o igual al 60%. En caso de no aprobar, podrán recuperarse en instancias evaluativas semejantes al finalizar el período de cursado.

Promoción:

La promoción se alcanza realizando la defensa teórico-práctica del diseño y desarrollo realizado en el trabajo práctico integrador a lo largo del curso. Durante la defensa el alumno deberá exponer individualmente de forma oral el trabajo práctico, justificando su diseño, implementación y pruebas, respondiendo correctamente a las preguntas que formulen los docentes basándose en la documentación presentada.

Se tendrá en cuenta durante la defensa:

- La correcta explicación del problema global que está resolviendo mientras muestra el funcionamiento del programa. El alumno debe exponer al nivel de un usuario final, evitando los detalles del diseño y desarrollo.
- Un adecuado nivel de cumplimiento de requisitos funcionales y robustez del programa.
- Un adecuado dominio del diseño y desarrollo del programa haciendo uso de la documentación de diseño del código del proyecto.

Para el acceso a la defensa del examen de promoción se requerirá al alumno:

- Haber cumplido los requisitos de regularidad.
- Entregar y presentar la documentación del diseño de las soluciones de los problemas y del código.
- Entregar y presentar el código fuente de los programas solicitados, los cuales deben ejecutarse sin errores.

Metodología de Evaluación en Exámenes Finales:

Regulares

El examen final consta de las siguientes etapas eliminatorias:

- 1 - Aprobar un cuestionario de evaluación de carácter teórico-práctica con un puntaje mayor o igual al 60%.
- 2 - Presentar la documentación del diseño de las soluciones de los problemas y del código.
- 3 - Presentar el código fuente de los programas solicitados, los cuales deben ejecutarse sin errores.
- 4 - Aprobar la defensa teórico-práctica del trabajo práctico integrador correspondiente a su cursado con un puntaje mayor o igual al 60%.

Durante la defensa el alumno deberá exponer individualmente de forma oral el trabajo práctico, justificando su diseño, implementación y pruebas, respondiendo correctamente a las preguntas que formulen los docentes basándose en el material presentado. Se tendrá en cuenta durante la defensa:

- La correcta explicación del problema global que está resolviendo mientras muestra el funcionamiento del programa. El alumno debe exponer al nivel de un usuario final, evitando los detalles del diseño y desarrollo.
- Un adecuado nivel de cumplimiento de requisitos funcionales y robustez del programa.
- Un adecuado dominio del diseño y desarrollo del programa haciendo uso de la documentación de diseño del código del proyecto.

Libres

El examen final consta de las siguientes etapas eliminatorias:

- 1 - Aprobar la defensa de la entrega de los trabajos prácticos de la presente planificación.
- 2 - Continúa con etapas eliminatorias de examen final para regulares.

Los alumnos que rindan libres deberán presentar y defender los Trabajos Prácticos implementados en el cuatrimestre inmediato anterior. En este caso, los alumnos deben coordinar con antelación con el responsable de la asignatura, o el profesor que este disponga, la entrega de los TPs.

La defensa de la entrega de los trabajos prácticos consiste en dar cuenta del correcto planteo de las soluciones a los programas de los trabajos prácticos y de su correcta ejecución.

Condiciones de Regularidad :

Serán reconocidos como alumnos regulares aquellos que hayan aprobado los trabajos prácticos, y los cuestionarios de evaluación con nota mayor o igual a 60%.

Serán reconocidos como alumnos promocionados aquellos que hayan regularizado y aprobado con nota mayor o igual al 60% la defensa del trabajo práctico integrador.

Todas las instancias de evaluación para alcanzar la regularidad contarán con su correspondiente recuperatorio.

Cronograma de parciales durante el primer Cuatrimestre:

Primer Examen Parcial: 22 de Abril de 2024

Segundo Examen Parcial: 03 de Junio de 2024

Recuperatorio 01: 17 de Junio de 2024

Recuperatorio 02: 20 de Junio de 2024

Cronograma de parciales durante el segundo Cuatrimestre:

Primer Examen Parcial: 16 de Septiembre de 2024

Segundo Examen Parcial: 28 de Octubre de 2024

Recuperatorio 01: 11 de Noviembre de 2024

Recuperatorio 02: 14 de Noviembre de 2024

Bibliografía Principal:

Booch, Grady, Juan Manuel Cueva Lovelle, et al. Análisis y diseño orientado a objetos con aplicaciones. 2.a ed., Addison Wesley Longman: Pearson Educación, 1996.

Booch, Grady, James Rumbaugh, et al. El lenguaje unificado de modelado. 2.a ed., Addison-Wesley, 2006.

Chapra, Steven C., et al. Métodos numéricos para ingenieros. 7.a ed., McGraw-Hill, 2015.

El tutorial de Python — documentación de Python. <https://docs.python.org/es/3/tutorial/>.

La Biblioteca Estándar de Python — documentación de Python - 3.10.1.
<https://docs.python.org/es/3/library/index.html#library-index>.

Miller, Brad, y David Ranum. Solución de problemas con algoritmos y estructuras de datos usando Python. Traducido por Mauricio Orozco-Alzate, 2.a ed., 2013.

PEP 8 – Style Guide for Python Code | [peps.python.org](https://peps.python.org/pep-0008/). <https://peps.python.org/pep-0008/>.

PEP 3119 – Introducing Abstract Base Classes | [peps.python.org](https://peps.python.org/pep-3119/#abcs-vs-alternatives).
<https://peps.python.org/pep-3119/#abcs-vs-alternatives>.

Programación Avanzada. Introducción a las Pruebas Unitarias. Apunte de cátedra. 2022.

Referencia del Lenguaje Python — documentación de Python - 3.10.1.
<https://docs.python.org/es/3/reference/index.html#reference-index>.

Rougier, Nicolas. Scientific Visualization: Python + Matplotlib. 2021,
<https://github.com/rougier/scientific-visualization-book/raw/master/pdf/book.pdf>.

Bibliografía Complementaria:

Börstler, J. (2004). Object-oriented analysis and design through scenario role-play. Citeseer.
<https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.63.7570&rep=rep1&type=pdf>

Bourque, P., y Fairley, R. E. (Eds.). (2014). SWEBOK: Guide to the software engineering body of knowledge (Version 3.0). IEEE Computer Society.

Budd, T. (1994). Introducción a la programación orientada a objetos. Addison-Wesley Iberoamericana.

Deitel, P. J., & Deitel, H. M. (2020). Intro to Python for the computer and data sciences: Learning to program with AI, big data and the cloud (1.a ed.). Pearson Education, Inc.

EtnasSoft. (s. f.-a). OpenLibra | Algoritmos y Programación con lenguaje Python. OpenLibra. Recuperado 27 de diciembre de 2021, de <https://openlibra.com/es/book/algoritmos-y-programacion-con-lenguaje-python>

EtnasSoft. (s. f.-b). OpenLibra | Introducción al Lenguaje de Modelado Unificado. OpenLibra. Recuperado 27 de diciembre de 2021, de <https://openlibra.com/es/book/introduccion-al-lenguaje-de-modelado-unificado>

Khorikov, V. (2020). Unit Testing Principles, Practices, and Patterns (1st edition). Manning Publications.

Lutz, M. (2013). Learning python (5.a ed.). O'Reilly Media, Inc.

Nakamura, S., & Palmas Velasco, Ó. A. (1992). Métodos numéricos aplicados con software (1.a ed.).

Prentice Hall Hispanoamericana.

Severance, C. R. (2020). Python para todos: Explorando la información con Python 3 (1.a ed.).

<https://www.py4e.com/book.php>

Wirfs-Brock, R., Wilkerson, B., & Wiener, L. (s. f.). Traducción del libro Designing Object Oriented Software (A. Sigura, D. Milone, & C. Milone, Trads.).

Wirfs-Brock, R., Wilkerson, B., & Wiener, L. (1990). Designing object-oriented software. Prentice Hall.

Zelle, J. M. (2017). Python programming: An introduction to computer science (Third edition). Franklin, Beedle & Associates Inc.

Equipo de Cátedra:

Profesor Titular: Diaz Zamboni Javier Eduardo

Profesor Adjunto: Jordán Francisco Insfrán

Jefe de Trabajos Prácticos: Diana Vertiz del Valle

Jefes de Trabajos Prácticos: Jonathan José Carlos Nicolet

Auxiliar de Primera: Bruno Breggia

Auxiliar Alumna: Giovanna Viñoli Miotti

Auxiliar Alumno: Santiago Salinas Sosa

Actividades de Investigación Gestión y Extensión:

-

Requisitos de admisión para alumnos oyentes:

Tener conocimientos de programación en un lenguaje estructurado moderno.

Infraestructura, equipamiento y recursos necesarios:

Para instancias de clases prácticas se precisa de laboratorios de computación.

Para todas las instancias de clases se requieren proyectores (cañones) y marcadores de pizarra.

Otros: