PROGRAMACIÓN en PYTHON II (nivel intermedio)

Carácter del curso	Curso de GRADO/POSGRADO
Semestre en que se dicta	PAR – PRIMER HEMISEMESTRE
Número de créditos	6 (grado) 8 (posgrado)
Carga horaria semanal (hs)	Clases teóricas: 16 horas a lo largo del hemisemestre Clases teórico-prácticas: 24 horas a lo largo del hemisemestre Proyecto Integrador de 30 horas solo para estudiantes de posgrado
Previaturas	competencias en Programación básica
Cupo	

Estructura Responsable:
Centro de Bioinformática Estructural (CEBIOINFO), DETEMA

Docentes Responsables:
Margot Paulino
Daniele Toti

Docentes colaboradores:
Pablo García
Andres Camilo Ballesteros
Jorge Cantero

Objetivos:

El curso tiene como objetivo proporcionar a los estudiantes una visión general y completa de la programación en Python, a un nivel de competencia intermedio, tanto desde un punto de vista teórico como práctico, explicado desde la perspectiva de un ingeniero informático familiarizado con las aplicaciones de software biomédico. Durante el curso, también habrá comparaciones con otros lenguajes, en particular Java.
Saber desarrollar programas es una habilidad fundamental, requerida en la mayoría de las actividades técnicas y científicas modernas. En este sentido, el curso tiene un triple objetivo:

- Proporcionar las herramientas metodológicas y tecnológicas que permitan a los estudiantes que nunca han tomado un curso de Programación desarrollar programas con cierto grado de autonomía al final del curso;
- Ayudar a los estudiantes que ya poseen conocimientos básicos de programación a consolidarlos y aplicarlos en el lenguaje Python, al mismo tiempo que les proporciona los principios de modelado y diseño de software;
- Dirigir a los estudiantes hacia cursos más avanzados, como el curso "*Deep Learning aplicado al diseño de compuestos bioactivos*", sentando las bases para temas avanzados que requieren sólidas habilidades de programación.

Al finalizar el curso, los estudiantes: estarán familiarizados con las técnicas fundamentales de programación y serán capaces de desarrollar programas de mediana complejidad en el lenguaje Python; conocerán el paradigma procedimental de la programación; tendrán una comprensión básica del paradigma orientado a objetos y la programación orientada a objetos; y podrán usar algunas bibliotecas adicionales para realizar

operaciones más complejas, especialmente aquellas más utilizadas para abordar problemas biomédicos y de ciencia de datos. Además, también conocerán y comprenderán los principios fundamentales para el diseño de aplicaciones de software.

Contenido:

Unidad 1:
Introducción a la programación y al lenguaje Python. Visión general y recapitulación de la sintaxis básica, asignaciones, variables y tipos primitivos, operadores básicos, expresiones booleanas, sentencias condicionales e instrucciones iterativas/ bucles. Esto también incluirá discusiones sobre los lenguajes de tipado débil y fuerte, las operaciones de bits, la tabla de caracteres ASCII, las palabras clave reservadas de Python, las variables admisibles, el control avanzado de los bucles, la introducción al manejo de excepciones, etc. Introducción a un entorno de desarrollo integrado y a Pycharm.

Unidad 2:
Definición y propósito de las funciones. Se discutirán las funciones void/no-void, las funciones sin-argumentos, las funciones con argumentos variables, las funciones con argumentos opcionales, y las mejores prácticas sobre cómo desarrollar y depurar funciones. Importación y uso de las bibliotecas y funciones incorporadas y personalizadas de Python.

Unidad 3:
Estructuras de datos fundamentales: cadenas y colecciones. Operaciones con cadenas, métodos y funcionalidades básicas. Operaciones sobre colecciones: listas, conjuntos, tuplas y diccionarios, con métodos básicos y características. La discusión incluirá una introducción sobre el concepto de "objetos" en Python y cómo las cadenas y las colecciones se comportan como objetos complejos, incluyendo los conceptos de paso por valor/paso por referencia y las ideas detrás de la gestión de memoria de Python para variables, funciones y objetos.

Unidad 4:
Punto de control y profundización en los conceptos de funciones y estructuras de datos/objetos complejos, incluyendo el concepto de "programación por interfaces", el paradigma divide-et-impera, las ejecuciones y los efectos colaterales, las diferencias entre el desarrollo de scripts y las Interfaces de Programación de Aplicaciones, las modificaciones de los objetos en memoria, el shadowing, las copias profundas y superficiales de los objetos y el manejo avanzado de las excepciones; la gestión de archivos y del sistema de archivos, incluyendo los mecanismos para la E/S, las funciones incorporadas y la biblioteca os.

Unidad 5:
El paradigma orientado a objetos y la programación orientada a objetos. Principios y conceptos básicos, incluyendo clases, objetos, abstracción, encapsulación, herencia, polimorfismo. Discusión sobre métodos, sobrecarga y anulación. Motivación y verticalización de los conceptos OO en el lenguaje Python, con elementos centrales como la definición de clases, atributos de instancia y estáticos, métodos de instancia y estáticos. Visibilidad de atributos y métodos, acceso e invocación, comparaciones con otros lenguajes OO como Java. Métodos especiales como to-string (str), getters y setters, lt/gt, eq/ne, hash, y ejemplos de

implementaciones reales. Implementación de la herencia y el polimorfismo y uso de referencias a superclases.

Unidad 6:
Otros conceptos de OO, incluyendo la igualdad y los hash. Recursión, gestión de fechas y tiempo. Principios fundamentales de modelado, patrones de diseño y arquitectura. La discusión incluirá una introducción sobre el diseño orientado a objetos, el diseño impulsado por la responsabilidad, las responsabilidades de los objetos y el concepto de patrón. Se mencionarán las arquitecturas 2-Tier/3-Tier, MVC, los principios GRASP más comunes y algunos de los patrones de diseño GoF.

Unidad 7:
Introducción a las bibliotecas y estructuras de datos adicionales para la computación científica y la visualización y el procesamiento de datos. Se hablará de: la librería Numpy, la estructura de datos Array de Numpy y las funcionalidades para operar con arrays y matrices; Matplotlib y una variedad de gráficos diferentes para visualizar datos; Pandas, el concepto de datasets, las estructuras de datos de Pandas (Series y Dataframe)

Unidad 8:
Introducción a las bibliotecas específicamente orientadas a la resolución de problemas biológicos/biomédicos y al tratamiento de estructuras bioquímicas, incluyendo RDKit, BioPython, etc. Visión general del software de investigación de libre acceso y productos en el ámbito biomédico, incluyendo LIBRA-WA, DockingApp, Fragment Graph Database, etc.

Material de Referencia:

Diapositivas de las clases y ejercicios prácticos, notas de clase, soluciones de los ejercicios realizados, recursos en línea disponible para los estudiantes.

Libro de texto recomendado: Thinking Python, segunda edición, traducción al español, disponible de forma gratuita en el siguiente e nlace:

https://github.com/espinoza/ThinkPython2-spanish/blob/master/book/thinkpython2-spanish.pdf

Modalidad del Curso:

	Teórico	Practico	Laboratorio	Otros
Asistencia Obligatoria	4 hs/sem	6 hs/sem		PI (posgr)
Modalidad Flexible	100%	100%		

PROGRAMACIÓN en PYTHON II (nivel intermedio)

Régimen de ganancia:

La ganancia del curso se obtiene a través de la realización de dos actividades: i) prácticos semanales ii) entregable final (Proyecto Integrador). Los prácticos semanales y el trabajo de Proyecto (entregable final) consistirán en archivos conteniendo el código con la solución al problema propuesto o al Proyecto desarrollado.
El planteo de un Proyecto Integrador que deberá ser desarrollado a lo largo del curso y expuesto al finalizar mediante explosión oral. Para los estudiantes de grado, el Proyecto Integrador será diseñado para que insuma 15 horas de dedicación y en el caso de los estudiantes de posgrado, 30 horas. La duración de la dedicación a la tarea será nivelada a través mas simples para los estudiantes de grado y mas complejas para los de posgrado.
La segunda actividad, que será igual para estudiantes tanto de grado como de posgrado, será definirá de acuerdo con las herramientas técnicas disponibles (tareas escritas supervisadas/pruebas orales). Podrá incluir problemas a resolver mediante el desarrollo de programas, porciones de código a leer e interpretar y preguntas sobre aspectos teóricos de los temas del curso. Dependiendo de las posibilidades técnicas y logísticas, los ejercicios de programación podrían realizarse directamente en una computadora.

Los porcentajes de cada componente para la nota final es el siguiente: 60% prácticos semanales, y 40% entregable final Proyecto Integrador con un mínimo del 50% en cada uno.

La nota de la asignatura (nota final) corresponde 100% de la nota del curso.

Requisitos para ganar el curso: el estudiante debe alcanzar un 66% en su calificación.

Créditos:

Contenido	Cantidad	Créditos
Teórico	16hs	2.1
Práctico	24hs	2.4
Entregables	1	1
Proyecto Integrador (solo posgrado)	30	2
Total		6 (grado) / 8 (posgrado)

Observación:

El curso no tiene requisitos previos "formales", siendo lo suficientemente autocontenido como para que lo puedan tomar incluso los estudiantes sin conocimientos previos de programación en Python y llegar a una comprensión sólida del tema. Sin embargo, la familiaridad con el uso de una computadora y un conocimiento básico de los temas ilustrados en el curso “Programación para Bioinformática” pueden ser útiles.

Por mayor información visitar la página del curso o consultar directamente con el docente responsable
margot@fq.edu.uy