Programación Python I (nivel inicial)
(Programación, Estructuras de datos y Algoritmos)

Carácter del curso	CURSO DE GRADO/POSGRADO
Semestre en que se dicta	IMPAR
Número de créditos	10 (grado) y 12 (posgrado)
Carga horaria semanal (hs)	4hs téoricas + 4hs prácticas + Proyecto Integrador 30 hs (a lo largo del semestre solo para posgrado)
Previaturas	No tiene previaturas. Es el primer nivel en programación que se imparte
Cupo	30

Estructura Responsable:

Centro de Bioinformatica – Área Bioinformática – DETEMA – Facultad de Química - UdelaR

Docentes Responsables:

Pablo García
Margot Paulino

Docentes Colaboradores:

Andrés Camilo Ballesteros
Jorge Cantero

Objetivos:

Dada la enorme diferencia de tiempos y costos que implica el trabajo in-silico sobre el trabajo tradicional de laboratorio cada vez más los químicos, biólogos y bioquímicos necesitan programar y manipular datos en volúmenes constantemente crecientes para integrar herramientas y automatizar procesos informáticos. Estas herramientas informáticas evolucionan constantemente y un profesional de esta era de abundante disponibilidad de recursos informáticos, necesita poder adaptar, complementar e integrar diversas herramientas publicadas por colegas o por la comunidad y explotar fuentes de datos diversas y de gran volumen. Este curso es el primer paso para ganar autonomía en dicha tarea.

El curso se enfoca en introducir al alumno en un nivel de competencia inicial a la programación imperativa usando el lenguaje Python. Se introducen los conceptos sin asumir conocimiento previo de los alumnos en programación, integrando progresivamente las herramientas que le permitan abordar un problema y resolverlo implementando un algoritmo, llegando a un nivel de manejo de las abstracciones, estructuras de datos y construcciones del lenguaje que permitan el adecuado desenvolvimiento del alumno en la resolución

de problemas generales de programación y obteniendo la formación necesaria para abordar el aprendizaje de algoritmos bioinformáticos de complejidad mayor.

Dado el desafío de adquirir en un tiempo acotado conocimientos que permitan el uso completo de un lenguaje de programación, con abstracciones que se construyen sobre otras abstracciones que llevan su tiempo de asimilación el curso incluye entregas semanales de trabajos de validación de que los alumnos dominan los conceptos necesarios para pasar al aprendizaje del siguiente concepto, el cual en general es una evolución o generalización del anterior.

En cuanto a los problemas biológicos abordados, si bien el curso se puede considerar un curso general de introducción a la programación, los prácticos y ejemplos en clase se enfocan en problemas de la biología, así, en casos que pueden considerarse generales, como leer y grabar un archivo, se realizan los ejemplos con herramientas asociadas a la biología, en este caso, grabando y leyendo un archivo con formato FASTA, de forma de que el alumno quede con conocimiento de componentes y bibliotecas que usará frecuentemente en el resto de la maestría en bioinformática.

Contenido:

1. Conceptos de lenguajes dinámicos y compilados.
   Introducción al concepto de lenguajes de programación, tipos de lenguajes y su uso específico, conocimiento del entorno del lenguaje Python y sus componentes principales. Preparación del entorno de trabajo.

2. Variables y tipos de datos.
   introducción al manejo de memoria, concepto de tipo de datos, conversión de tipos y gestión de la memoria. Tipos de datos numéricos, lógicos y de cadena de caracteres.

3. Comparaciones y lógica booleana.
   Introducción a las operaciones lógicas, sentencia de bloque IF, operadores lógicos AND, OR y NOT, precedencia de operadores y evaluación perezosa de condicionales.

4. Manipulación de cadenas de caracteres.
   Operaciones con cadenas de caracteres, funciones del lenguaje de control de cadenas de caracteres, manejo de indexadores y concepto de rebanadas, métodos del tipo de datos string.

5. Manejo de entrada y salida estándar
   Operación de la entrada-salida, redireccionamiento desde el sistema operativo, formateo de de resultados, construcción de un programa completo con entrada de datos, procesamiento con toma de decisiones y salida de resultados. Ciclo de vida de un programa, depuración del código para resolver problemas.

6. Instrucciones de repetición (for, while)
   Introducción a las secuencias (listas, tuplas, diccionarios y grupos). Generalización de indexadores y segmentadores. Introducción a los algoritmos básicos de ordenamiento, búsqueda, inserción, actualización y borrado usando for y while, concepto de orden de ejecución de los algoritmos.

7. Definiciones de listas por comprensión.
   Listas y diccionarios definidos por comprensión, condicionales aplicados en línea.

8. Funciones.
   Definición de funciones, argumentos nominales y efectivos, por omisión, posición y clave. Refactorización del código con funciones. Recursividad, revisión de algoritmos con uso de recursividad. Documentación de funciones.

9. Gestión de archivos.
   Lectura y escritura, serialización y re-hidratación de variables, introducción a archivos csv y fasta, escritura con diccionarios y manejo del filesystem en forma independiente del sistema operativo.

10. Gestión de excepciones.
    Captura, manejo, análisis y generación de excepciones. Revisión de los tipos más comunes de excepciones, finalización segura de bloques de código, manejo estructurado de excepciones y gestores de contexto.

11. Módulos y paquetes.
    Conceptos de módulos y paquetes, espacios de nombres, inicialización de módulos, creación de componentes reutilizables entre proyectos. Uso de bibliotecas especializadas en bioinformática, incorporar bibliotecas desde el pipy, instalar con pip y con distribuciones como conda, trabajo con entornos virtuales. Trabajo con biopython.

12. Expresiones regulares.
    Definición de expresiones regulares, creación y so de las mismas para validar entradas de texto y para recuperar información de archivos.

13. Orientación a objetos.
    Conceptos y definición de clases, herencia, constructores y destructores. Propiedades y métodos, de instancia y de clase. Métodos con especial funcionamiento en Python (convenciones del lenguaje)

14. Funciones de alto desempeño:
    Uso de Map, Reduce y Filter. Operaciones avanzadas con diccionarios, definición y uso de funciones lambda.

15. Trabajo con datos.
    Uso de bibliotecas para recuperar datos, introducción a bibliotecas elementales de manipulación de datos (sqlite3, datetime, requests y json). Manejo avanzado de datos con pandas. Carga, transformación, filtrado, selección y agregación de datos con Pandas. Visualización de datos con matplotlib (gráficos básicos de tipo barplot, boxplot, histogram y scatterplot), visualización de distribuciones y correlaciones con seaborn.

16. Performance.
    Introducción a la biblioteca Numpy, arreglos y matrices numéricas, trabajo en paralelo con ipyparallel y su uso desde notebooks de Jupyter.

Créditos:

Contenido	**Cantidad**	**Créditos**
Teórico	40hs	5
Práctico	40hs	4
Entregables	1	1
Proyecto Integrador (solo posgrado)	30	2
	TOTAL	**10 (grado) 12 (posgrado)**

Bibliografía:

Sitio web del lenguaje python: http://www.python.org/

Tutorial oficial del lenguaje Python: https://docs.python.org/3/tutorial/index.html

Bibliotecas - Describe todas las funciones y métodos de la librería estándar de Python: http://docs.python.org/library/

Referencia del Lenguaje - Describe con precisión la sintaxis y semántica del lenguaje Python: http://docs.python.org/reference/

Modalidad del Curso:

	Teórico	Practico	Laboratorio	Otros (*)
Asistencia Obligatoria	4 hs/sem	2 hs/sem		PI (posgr)
Modalidad Flexible	**100%**	**100%**		

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 exposición 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 más simples para los estudiantes de grado y más 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.