[L110] [Introducción al Pensamiento Computacional] - [Primavera] [2022]

[Roberto Bunge, Claudio Pose, Ernesto Mislej, Ignacio Mas, Rodolfo Sumoza]

[Patricio Witingslow, Emilio Basualdo, Gabriel Torre, Tadeo Casiraghi,Guillermo Marzik, Axel Lacapmesure, Franco Jofre]

1. Objetivos de aprendizaje

Este curso tiene como objetivo central introducir a los estudiantes a la programación como un eje fundamental del pensamiento computacional. Se introduce a los estudiantes a los elementos de la programación imperativa en Python. Esto involucra la introducción al pensamiento lógico y algorítmico, que incluye el modelado de problemas, su representación abstracta, y su resolución a través del paradigma divide y vencerás. Se introduce el concepto de variables, expresiones, tipos datos básicos, funciones, tipos de datos estructurados, estructuras de control (condicionales y ciclos), testing, debugging, patrones de programación, entre otros. Los estudiantes también aprenderán las nociones elementales del uso de herramientas básicas para el análisis de datos incluyendo el uso de bibliotecas para cálculos matemáticos, para la visualización de datos, y para la gestión de datos representados en formatos o estructuras más eficientes. A medida que los estudiantes aprendan a utilizar las herramientas básicas de la programación, paralelamente adquirirán las habilidades involucradas al pensamiento computacional, lógico y algorítmico. Esto incluye la descomposición de problemas en unidades computables, abstracción y modularización de procesos computacionales en funciones de entrada-salida, representación de datos y relaciones mediante estructuras de datos. Se introducirá a los estudiantes a los conceptos de lógica y algoritmia, y se proveerán nociones básicas para entender la complejidad algorítmica como medio para comparar sus soluciones. Al terminar el curso, los estudiantes podrán resolver de manera independiente problemas concretos de mediana envergadura.

2. Contenidos

2.1. **Introducción a la programación y computación.** La computadora es una herramienta para resolver problemas. Un programa son instrucciones que debe seguir la computadora para resolver dicho problema. Descripción conceptual de una computadora como una máquina que puede tomar

entradas, hacer cálculos, guardarlos y mostrar salidas (esquema sencillo: memoria, ALU, contador). Concepto de cálculos primitivos y máquina de Turing. Motivación a través de ejemplos de problemas resolubles mediante la computadora. El objetivo es aprender a programar la computadora para que resuelva problemas de interés.

2.2. **Tipos de datos, operadores, expresiones y variables.** Números enteros, flotantes, booleanos, secuencias: listas, tuplas, cadenas, diccionarios. Operadores aritméticos: suma, resta, multiplicación, división, resto o módulo. Inmutabilidad. Índices. Reglas de precedencia de operadores. Operadores lógicos: and, or, not. Construcción de expresiones combinando expresiones y operadores. Asignación de valor a una variable. Motivación de la utilización de variables. Operadores de comparación: is greater than or equal, is less than or equal, is equal, not equal.

2.3. **Estructuras de control.** If, elseif, else. Concepto de bloques de código. Iteradores: while loops, for loops. While loops vs. for loops. Relación entre while loops con contador y for loops. Break. Continue.

2.4. **Funciones.** Motivación de utilización de funciones mediante conceptos de descomposición y abstracción. Definición de funciones: nombre de función, parámetros y argumentos, docstring, cuerpo (código de la función) y return (argumento de salida). Alcance o ámbitos de las variables: variables locales y globales. Motivación adicional de funciones: modularización (paquetes/librerías), replicación, mantenimiento y debuggeo de código.

2.5. **Uso de estructuras de datos (secuencias).** Mutabilidad, slicing, iteradores sobre secuencias, operadores para agregar, remover, ordenar elementos, aliasing (muchas variables apuntan al mismo elemento), cloning (copias), secuencia dentro de secuencias. Indices vs “key-values”.

2.6. **Testeo y debugging.** Conceptos y técnicas utilizadas para testear y debuggear código. Utilización de print() en proceso de debuggeo, aislación del bug, búsqueda por bisección, analogía con trabajo de detective (¿cómo ocurrió el error?).

2.7. Nociones básicas de complejidad. Medida de eficiencia de un programa, recursos, tiempo de ejecución, espacio en memoria, complejidad, interpretación práctica y ejemplos.

2.8. Uso de herramientas para el análisis de datos: Uso de bibliotecas para cálculos matemáticos. Arreglos. Uso de estructuras de datos eficientes. Dataframe. Herramientas para la visualización de datos. Graficos.

3. Modalidad de trabajo:

El curso incluye dos clases teóricas por semana y una clase para la resolución de problemas, y una clase práctica o tutoría por semana. Durante las clases teóricas haremos una introducción teórico-conceptual a los temas del programa. Las sesiones para la resolución de problemas, se expondrán ejercicios con diferente nivel de complejidad, y se mostrarán diferentes formas de resolverlos. Las sesiones prácticas estarán centradas en la resolución de problemas prácticos con mayor participación de los alumnos, donde deberán aplicar los contenidos aprendidos y ejercitarlos. Los alumnos trabajarán de forma individual o grupal, dependiendo de la actividad.

4. Mecanismo de Evaluación:

4.1. Trabajos Prácticos

Se tendrán dos trabajos prácticos, el primero incluirá técnicas básicas de programación: tipos de datos, condicionales y ciclos, el segundo trabajo práctico será integrador, incluirá todo el contenido visto en la materia.

Método de Entrega: Los trabajos prácticos deberán ser entregados digitalmente vía el Campus Virtual del curso. En ningún caso se aceptarán trabajos prácticos entregados por otros medios o vías. En el Campus Virtual del curso habrá un link donde podrán subir los archivos correspondientes a cada trabajo práctico.

Formato de Entrega: La entrega de cada trabajo práctico consistirá de un archivo zip que contendrá **un archivo formato .pdf**, donde estará un informe técnico que incluirá las respuestas y justificaciones a cada uno de los problemas, y **los archivos .py** (Python) que contengan el código necesario para resolver cada problema planteado. **El código correspondiente a cada problema debe poder ser ejecutado sin errores, si éste no fuera el caso se descontará 50% a la nota de dicho problema. El archivo zip debe nombrarse del siguiente modo: tpx_apellido.zip,** donde X es reemplazado por el número de trabajo práctico, y en forma similar el pdf: *tpx_apellido.pdf* contenido dentro del zip. Por ejemplo, la entrega del trabajo práctico número 3 de Aike Perez, debiera consistir en el archivo: *tp3_perez.zip*, y contiene, además del código fuente, el documento *tp3_perez.pdf*.

Fechas de Entrega: cada trabajo práctico tendrá una *"fecha de entrega límite"*. **Las entregas de trabajos prácticos posteriores a la *"fecha de entrega límite"* tendrán una penalización de hasta el 100% de su valoración dependiendo del tiempo de retraso.**

El primer trabajo práctico tendrá un peso del 15% de la nota final, y el segundo trabajo práctico tendrá un peso del 25% de la nota final.

4.3. **Exámenes Parciales:** tendremos un exámen parcial, y dos exámenes cortos, denominados parcialitos. La nota del parcial y los parcialitos se tomará a través del promedio ponderado entre estás, donde el parcial tendrá un peso del 50% y cada parcialito 25% cada uno, a esto se le denominará "grupo de parciales". Este grupo de parciales tendrá un peso del 30% de la nota final. El parcial se rendirá a mitad del período lectivo, durante las semanas denominadas de parciales. El primer parcialito se rendira en la primera mitad del período y el segundo, en la segunda mitad. Cada instancia de parciales se realizará de forma individual.

4.4. **Examen Final:** el Examen Final se realizará de forma individual, y se rendirá en el período comprendido como la semana de finales. El examen final tendrá un peso del 30% de la nota final.

4.5. **Nota de cursada:** la nota de cursada de la materia surge del promedio ponderado del grupo de parciales (parciales y parcialitos), el examen final y los dos trabajos prácticos, cuyos pesos están definidos en el punto 4.9.

4.6. **Condiciones necesarias para aprobar la materia:** Para aprobar la materia las condiciones necesarias, pero no suficientes, son:

- Tener una nota de cursada mayor a 4 puntos.
- Tener una nota mayor a 4 puntos en el grupo de parciales:
    - Esta nota se obtiene a través del promedio ponderado entre el parcial, con peso del 50%, y los dos parcialitos, 25% cada uno. Esos porcentajes están dentro del grupo de parciales, que tendrán una sola nota como grupo de parciales.
- Tener una nota mayor a 4 puntos en el examen final.

4.8 **Instancia de Recuperatorio:** Para tener acceso al examen único recuperatorio, deberá haber aprobado el grupo de parciales (parcial y dos parcialitos) o el examen final. Al menos uno de ellos. El recuperatorio sustituirá la nota desaprobada, la del grupo de parciales o el examen final. Sí los dos están desaprobados, no se tendrá acceso al recuperatorio.

4.9. **Nota final del curso:** La nota final de la materia será un promedio ponderado con el siguiente peso relativo:

- 15% Trabajo práctico 1.
- 25% Trabajo práctico 2.
- 30% Grupo de parciales.
    - Esta calificación se obtiene considerando 50% del parcial, y 25% cada uno de los dos parcialitos.
- 30% Examen final.

5. Mecanismos de Ejercitación y Monitoreo:

5.1. Guías de ejercicios: para cada tema se publicará una guía de ejercicios que

contienen problemas vinculados a cada contenido, tanto de abordaje teórico como práctico. La resolución de las mismas no es obligatoria, pero es recomendable hacer todos los ejercicios y realizar consultas sobre los mismos a fin de afianzar los conocimientos y como preparación para los exámenes parciales.

6. Mecanismos de Comunicación: con el fin de mantener un orden y de maximizar las posibilidades de atender a consultas adecuadamente, se requiere que la comunicación entre profesores y estudiantes sea en los siguientes medios, en orden descendente de prioridad.

6.1. Campus Virtual: se utilizará como medio de comunicación para avisos, consultas, comentarios,

6.2. Consultas en Clase: *se espera que la mayoría de las consultas en persona sean evacuadas durante las clases magistrales, de problemas o tutoriales.* Los profesores intentarán dejar una porción al final de cada clase para contestar preguntas relacionadas al curso.

6.3. Horarios de Consulta: cada docente definirá horarios de consulta regulares, fuera del horario de clase, en los cuales estará disponible para contestar preguntas.

7. Calendario.

El curso se guiará por el calendario de la UdeSA para el segundo cuatrimestre 2022, el cual estará publicado en el campus virtual del curso.

8. Plagio y Deshonestidad Intelectual

La Universidad de San Andrés exige un estricto apego a los cánones de honestidad intelectual. La existencia de plagio constituye un grave deshonor, impropio de la vida universitaria. Su configuración no sólo se produce con la existencia de copia literal en los exámenes, sino toda vez que se advierta un aprovechamiento abusivo del esfuerzo intelectual ajeno. El código de Ética considera conducta punible la apropiación de la labor intelectual ajena, por lo que se recomienda apegarse a los formatos académicos generalmente aceptados (MLA, APA, Chicago, etc) para las citas y referencias bibliográficas (incluyendo los formatos on-line). En caso de duda recomendamos consultar los sitios:

http://www.udesa.edu.ar/Unidades-Academicas/departamentos-y-escuelas/Hu manidades/Prevencion-del-plagio/Que-es-el-plagio

https://udesa.edu.ar/sites/default/files/codigo_de_etica.pdf

La violación de estas normas dará lugar a sanciones académicas y disciplinarias que van desde el apercibimiento hasta la expulsión de la Universidad.

9. Bibliografía:

El curso contará el libro de referencia listado abajo. Además, durante la cursada se podrán agregar otras fuentes.

- Guttag, J. V., *Introduction to Computation and Programming Using Python: With Application to Understanding Data*, MIT Press, 3rd Edition (2021).
- Downey, A., *Think Python: How to Think like a Computer Scientist*, Green Tea Press, 2nd Edition (version 2.4.0) (2015)

10. Vigencia y Modificación del Programa:

Los profesores se reservan el derecho a modificar el contenido del programa durante el semestre de clase si la evolución del curso lo encontrase apropiado.