Pensamiento Computacional
2do Semestre 2021

PROFESOR A CARGO
Juan I. Giribet

DOCENTES DE TUTORIALES
Santiago Bassani
Patricio Moreno

1. Objetivos de aprendizaje:

Este curso tiene dos objetivos centrales: introducir a los estudiantes a la programación y exponerlos a los conceptos fundamentales del pensamiento computacional. En cuanto a programación, se introduce a los estudiantes a los elementos de la programación imperativa en Python. Esto incluye objetos, expresiones, tipos y estructuras de datos, variables, funciones, argumentos de entrada y salida, variables locales y globales, estructuras de control (if/then/else, for y while loops), patrones de programación, entre otros.

Los estudiantes también aprenderán las nociones elementales de la programación orientada a objetos, de forma tal de poder interactuar de manera práctica con librerías que utilicen clases y métodos. A medida que los estudiantes aprendan a utilizar las herramientas básicas de la programación, también serán expuestos y entrenados en los fundamentos del pensamiento computacional. 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 y el pensamiento algorítmico. En cuanto a esto último, se introducirá a los estudiantes a los conceptos de algoritmo y complejidad algorítmica mediante el estudio de algoritmos clásicos de búsqueda, ordenamiento, inserción, etc.

Al terminar el curso, los estudiantes podrán resolver de manera independiente problemas concretos de mediana complejidad mediante el diseño e implementación de programas computacionales.

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 programa. 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. Diagramas de flujo.

2.2. **Tipos de objetos básicos, operadores, expresiones y variables.** Números enteros, flotantes, booleanos y None. Type casting. Operadores aritméticos: suma, resta, multiplicación, división, resto. Reglas de precedencia de operadores. Operadores lógicos: and, or, not. Construcción de expresiones combinando objetos y operadores. Asignación de valor a una variable. Motivación de la utilización de variables. Strings. Operadores de comparación y lógicos: 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 (indentation). Iteradores: while loops, for loops. While loops vs. for loops. Relación entre while loops con contador y for loops. Break.

2.4. **Funciones.** Motivación de utilización de funciones mediante conceptos de descomposición y abstracción (ocultar detalles). Definición de funciones: nombre de función, parámetros (argumentos), docstring, cuerpo (código de la función) y return (argumento de salida). Scope de variables: variables locales, globales e instancia ciación en memoria. Funciones como argumentos de otras funciones. Concepto de interfaces. Motivación adicional de funciones: modularización (paquetes/librerías), replicación, mantenimiento y debuggeo de código.

2.5. **Estructuras de datos (compuestas).** Tuplas: definición, inmutabilidad, índices y slicing. Listas: definición, mutabilidad, iteradores sobre listas, operadores sobre listas (add, remove, sort, to strings, etc.), aliasing (muchas variables apuntan al mismo objeto lista), cloning (copias), lista de listas. Diccionarios: motivación de "key-values" vs. listas, definición, operadores. Diccionarios vs. listas.

2.6. **Testeo y debugging.** Conceptos y técnicas utilizadas para testear y debuggear código. Noción de programación defensiva. Tipos de tests: unidad, regresión e integrados. Tipos de mensajes de error: IndexError, TypeError, SyntaxError, etc. Errores "silenciosos", los más difíciles de encontrar y arreglar. Utilización de print() en proceso de debuggeo, aislación del bug, búsqueda por bisección, similar a trabajo de detective (¿cómo ocurrió el error?).

2.7. **Introducción a programación orientada a objetos (POO).** Objetos: tipo, propiedades/atributos y métodos. Motivación y ventajas de POO. Definición de clase, herencia, sub/superclase. Relación entre métodos y funciones. Creator, destructor, getter and setter methods. Overloading de

operadores básicos (suma, resta, len, etc.). Overloading the métodos en subclases. Algunas librerías útiles y manejo de archivos.

2.8. Algoritmos clásicos y complejidad. Medida de un programa, recursos, tiempo de ejecución, espacio en memoria, complejidad, funciones de referencia, órdenes de complejidad, interpretación práctica y ejemplos, propiedades y relaciones de orden, relaciones de recurrencia, clases de problemas P, NP, NP-completos.

3. Modalidad de trabajo:

El curso incluye dos clases magistrales por semana y una sesión tutorial por semana. Durante las clases magistrales haremos una introducción teórico-práctica a los temas del programa. Las sesiones tutoriales estarán centradas en aspectos prácticos y de aplicación de los contenidos. Durante el curso responderemos y resolveremos un buen número de preguntas y problemas, y frecuentemente haremos uso de casos reales que se relacionen con los temas bajo estudio.

Los alumnos trabajarán de forma individual o grupal, dependiendo de la actividad.

4. Mecanismo de Evaluación:

4.1. Componentes de la nota final del curso: La nota final de la materia se calculará de acuerdo a la siguiente rúbrica:

- 10% participación en clase
- 20% trabajos prácticos
- 10% cuestionarios cortos en clase
- 20% examen parcial
- 40% trabajo final

4.2. Trabajos Prácticos

Método de Entrega: Los trabajos prácticos deberán ser entregados digitalmente vía el Campus Virtual del curso. En ningún caso se aceptarán trabajos prácticos entregados vía otro medio. 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 dos archivos: **un archivo formato .pdf**, donde estarán las respuestas y justificaciones a cada uno de los problemas, y **un archivo formato .py** (Python) que contendrá 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. Los archivos deben llamarse, TPX APELLIDO NOMBRE.pdf y TPX APELLIDO NOMBRE codigo.py**, donde X es reemplazado por el número de trabajo práctico. Por ejemplo, la entrega del trabajo práctico número 1 de Juan Perez, debiera consistir en dos

archivos: TP1_PEREZ_JUAN.pdf y TP1_PEREZ_JUAN_codigo.py.

Fechas de Entrega: cada trabajo práctico tendrá una *"fecha de entrega límite"*. **En ningún caso se aceptarán entregas de trabajos prácticos posteriores a la "fecha de entrega límite".**

4.3. **Recuperatorios de Examen Parcial y Final:** El examen parcial no tendrá recuperatorio. El examen final podrá ser recuperado una sola vez, y solamente en el caso que el promedio de la nota de concepto, trabajos prácticos, cuestionarios cortos y examen parcial, ponderados según lo mencionado anteriormente, sea mayor a 4.

4.4. **Condiciones necesarias para aprobar la materia:** Para aprobar la materia es condición necesaria, pero no suficiente, tener una nota mayor a 4 en el examen final (ya sea en el final o su recuperatorio).

5. Clases Virtuales:

5.1. Horarios: Las clases magistrales y tutoriales comenzarán en el horario oficial de manera estricta, a menos que el profesor así lo disponga con previo aviso. El dictado de contenidos finalizará en el horario oficial de la clase, pudiendo extenderse sólo en casos excepcionales. En caso de tener clases virtuales, **se espera que los profesores y estudiantes se conecten 5 minutos antes del horario oficial de la clase,** a fin de asegurar que los sistemas están funcionando correctamente, etc.

5.2. Grabaciones: las grabaciones de las clases magistrales y tutoriales virtuales serán **puestas a disposición de los estudiantes en el Campus Virtual al día siguiente de dictada dicha clase**, y podrán ser editados según lo consideren necesario los profesores.

5.3. Asistencia: se espera la asistencia de los estudiantes a todas las clases magistrales y tutoriales. La asistencia será considerada para la elaboración de la nota de concepto. Los profesores cuentan con los logs digitales para revisar dicha asistencia virtual.

5.4. Participación: se espera la participación activa de todos los estudiantes durante las clases magistrales y tutoriales, según lo permitan los medios virtuales. Si el profesor hace una pregunta directa a un estudiante que figura como conectado sin la cámara prendida, pero este no contesta, se considerará que el estudiante realmente no está atendiendo la clase, y contará negativamente para la nota de concepto.

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. Avisos en Campus Virtual: toda la comunicación general de los profesores a los estudiantes se hará a través de "Avisos" en Campus Virtual. Esto asegura que todos los estudiantes registrados reciban los mensajes, y además los

Avisos quedarán registrados para futuro acceso. Por favor revisar los "spam detectors", para que los emails no sean ocultados.

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

7. Calendario.

El curso se guiará por calendario de UDESA para el primer cuatrimestre, 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/Humanidades/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).

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.