PROGRAMA DE ASIGNATURA	
ASIGNATURA: Algoritmos y Programación	**AÑO:** 2026
CARACTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 1° año 1° cuatrimestre
CARRERA: Licenciatura en Matemática Aplicada	
REGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 Horas.

FUNDAMENTACIÓN Y OBJETIVOS

De los diversos modelos de lenguajes de programación existentes en la actualidad, el de la programación imperativa es el más antiguo y aún hoy el más ampliamente utilizado. Según este paradigma y en un sentido amplio del término, un programa se compone de instrucciones para ser ejecutadas por una máquina. La ejecución de una instrucción lleva la máquina de un estado a otro. Las instrucciones que pueden emplearse en la construcción de un programa dependen de la máquina que desea programarse, deben ser instrucciones que ella sea capaz de ejecutar. Esto da lugar a una amplia variedad de lenguajes de programación imperativos, desde lenguajes de propósito general, capaces de programar todo tipo de computadoras a otros destinados a máquinas específicas, físicas o virtuales.

A pesar de la diversidad de lenguajes de programación imperativos de propósito general, existen ciertas características comunes a todos ellos: la posibilidad de acceder y modificar porciones de memoria de una manera amigable (variable y asignación), de representar datos estructurados (definición de tipos y/o clases), de realizar diferentes acciones según el caso (condicionales), de repetir instrucciones (ciclos), de programar en forma estructurada (descomponiendo en bloques, funciones, procedimientos, módulos, etc.), de interactuar con un usuario (entrada/salida), entre otras.

El propósito de este curso es ofrecer un abordaje gradual a estas características generales a través de la presentación de los conceptos y la ejercitación continua, introduciendo simultáneamente técnicas de diseño, buenas prácticas de programación, identificando vicios comunes y proponiendo métodos para razonar sobre los programas.

Entre los objetivos se busca asimilar conceptos fundamentales de programación imperativa mediante la ejercitación, conocer técnicas habituales de programación, incorporar buenas prácticas, reconocer la necesidad de comprobar el buen funcionamiento de los programas y familiarizarse con técnicas para ello, adquirir habilidad en el abordaje computacional de problemas numéricos simples.

CONTENIDO

1. Lenguajes de programación de propósito general.

Lenguajes dinámicos y estáticos, tipado débil vs tipado fuerte. Lenguajes compilados vs interpretados. Paradigmas de programación: imperativo, funcional. Historia y evolución de los lenguajes. Criterios de elección de un lenguaje según el problema. El ecosistema Python: intérprete, notebooks y terminal. Instalación y configuración del entorno. Ejecución de scripts y uso interactivo. Introducción a Jupyter Notebooks. Buenas prácticas de estilo: PEP8, legibilidad y documentación básica del código.

2. Construcciones básicas.

Variables, constantes, literales, expresiones. Asignación, asignación múltiple. Iteración. Ciclo for, índice, rango, cuerpo. Valores numéricos enteros y reales, valores lógicos. Errores de representación de los valores reales. Operadores aritméticos, operadores relacionales y operadores lógicos. Expresiones aritméticas, definición de funciones aritméticas simples. Expresiones lógicas, definición de funciones lógicas simples. Instrucciones condicionales, forma

general. Buenas y malas prácticas de programación con condicionales. Funciones: declaración o definición, invocación o llamada. Parámetros o argumentos. Parámetro formal y parámetro actual.

3. Iteración.
Ciclo while, condición, cuerpo. Ejemplos: algoritmo de división, chequeo de primalidad, factores primos, factorización, raíz entera, mcd, otros ciclos. Buenas y malas prácticas de programación con ciclos. Corrección parcial de un programa: tipos, precondición, estado, invariante, demostración del invariante, poscondición. Ejemplos: algoritmo de división, chequeo de primalidad, factores primos, raíz entera, mcd, factorial.

4. Secuencias, diccionarios y conjuntos.
Programando sobre secuencias (listas o arreglos) y cadenas de caracteres. Operaciones básicas. Recorrida de secuencias, conteo de ocurrencias de elementos en secuencias. Mínimo y máximo de secuencias. Posición del mínimo. Posición del mínimo a partir de un índice dado. Determinar si una secuencia está ordenada. Diccionarios. Recorriendo un diccionario. Métodos en diccionarios. Conversiones de tipos iterables. Conjuntos. Operadores y métodos disponibles en conjuntos. Definiciones por comprensión.

5. Recursión.
Definiciones recursivas. Recursión sobre números naturales. Ejemplos: factorial, Fibonacci, potenciación, división entera, mcd, mcd extendido, número combinatorio. Recursión e inducción. Corrección. Iteración versus recursión. Recursión sobre secuencias. Ejemplos: mínimo, posición del mínimo, búsqueda lineal, búsqueda binaria.

6. Tipos abstractos de datos.
Tipos de datos: valores y operaciones. Tipos básicos. Tipos abstractos. Objetos y clases. Ejemplos: paréntesis balanceados y el tipo contador. Delimitadores balanceados y el tipo pila. Objetos inmutables y objetos mutables. Ocultación de campos de datos. Abstracción y encapsulación de clases. Pensamiento orientado a objetos.

7. Entrada/salida.
Interfaz con el usuario. Lectura y escritura de archivos. Lectura de archivos en la red: el módulo requests. Lectura de archivos CSV, XML y JSON.

8. Stack científico de Python
Introducción a la biblioteca NumPy. Selección de datos en arreglos NumPy. Indexación y asignación en NumPy. Referenciación de arreglos en NumPy. Operaciones aritméticas con arreglos de NumPy. Arreglos booleanos. Máscaras. Lectura y escritura de archivos con NumPy. Introducción a Matplotlib y pyplot. Gráfico de funciones. Histogramas. Gráficos de barras. Representación de datos experimentales. Pandas.

9. Introducción a C
Lenguaje tipado estáticamente. Etapas de compilación: preprocesamiento, compilación, ensamblado y enlazado. Diferencias con Python: tipado explícito, gestión manual de memoria, ausencia de recolector de basura. Sintaxis básica: declaración de variables, tipos primitivos (int, float, char, double). Operadores aritméticos, relacionales y lógicos. Instrucciones condicionales y ciclos (if, for, while). Definición e invocación de funciones. Pasaje de parámetros por valor y por referencia. Punteros: concepto, declaración y uso básico. Arreglos en C: declaración, acceso e iteración. Cadenas de caracteres como arreglos de char. Manejo básico de entrada/salida con printf y scanf. Compilación con gcc desde la terminal.

BIBLIOGRAFÍA

BIBLIOGRAFÍA BÁSICA

- Introduction to programming using Python. Daniel Liang. Armstrong Atlantic State University, 2013.
- Tutorial oficial de Python: Tutorial de Python — documentación de Python - 3.9.1 o posterior
- Python Practice Book, 2019, Anand Chitipothu (creative commons).

BIBLIOGRAFÍA COMPLEMENTARIA

- Aprenda a pensar como un programador con Python. Allen Downey, Jeffrey Elkner, Chris Meyers. Green Tea Press, 2012.
- Pensar en Python, 2015, 2da edición, Allen B. Downey.
- Introduction to computation and programming using Python. John V. Guttag. The MIT Press. Revised and Expanded Edition, 2013.
- Algoritmos de Programación con Python. R. Wachenchauzer, M. Manterola, M. Curia, M. Medrano, N. Paez. uniwebsidad.com
- Python Tutorial. w3schools.com.
- Blockly: A JavaScript library for building visual programming editors. https://developers.google.com/blockly
- Programming Fundamentals: A Modular Structured Approach, 2nd Edition, Kenneth Leroy Busbee and, Dave Braunschweig (creative commons.)
- Structured Programming, O.-J. Dahl, E. W. Dijkstra, C. A. R. Hoare, Academic Press, London,1972.
- Python Practice Book, 2019, Anand Chitipothu (creative commons).

EVALUACIÓN

FORMAS DE EVALUACIÓN

Tres parciales. Luego de rendir ambos parciales quienes hayan aprobado solo uno de ellos podrán recuperar los otros dos. En ningún caso se podrán recuperar los tres parciales.

El examen final contendrá una parte de ejercicios de programación y una parte de preguntas conceptuales.
Para aprobar el examen se deberá tener un 60% de los ejercicios de programación y el 60% de las preguntas conceptuales resueltos correctamente.
Estudiantes libres deberán resolver ejercicios adicionales eliminatorios.

Quienes promocionen obtendrán como nota final de la materia el promedio de las notas de los parciales, redondeado. No se podrá rendir recuperatorio para mejorar la nota de promoción. Quienes promocionen no podrán rendir el examen final para mejorar la calificación.

REGULARIDAD

Para regularizar la materia se debe:

- Estar inscripto/a como estudiante regular.
- Aprobar tres parciales. Si después de rendir los tres parciales un alumno ha aprobado un parcial y ha desaprobado los otros, podrá rendir recuperatorios del los parciales no aprobados.

PROMOCIÓN

Para promocionar la materia se debe:

- Estar inscripto/a como estudiante regular.
- Haber aprobado el Curso de Nivelación.
- Aprobar los tres parciales con una nota no menor a 6 (seis), y obteniendo un promedio no menor a 7 (siete).

No se podrá rendir un recuperatorio para alcanzar la promoción.