CARRERA: Tecnicatura Superior en Análisis de Sistemas y Desarrollo de Software

PLAN DE ESTUDIOS: Resolución Ministerial N° 013/23

AÑO: 2024

CAMPO: Campo de la Formación Específica

CURSO: 2° año

DIVISIÓN: 1°

ASIGNATURA: Programación II

DOCENTE RESPONSABLE: Margarita de los Ángeles Ruiz

RÉGIMEN DE LA ASIGNATURA:Cuatrimestral

CANTIDAD DE HORAS-CÁTEDRA: 5

CONDICIONES PARA REGULARIZAR LA ASIGNATURA.

- ✔ **70% de asistencia obligatoria a clases teóricas y prácticas previstas por la docente responsable de la cátedra.** Este porcentaje podrá reducirse al 60% cuando las ausencias obedezcan a razones de salud, de trabajo o de fuerza mayor debidamente justificadas por el Consejo Asesor. Art. 36° - RAM 2484/2013. Al reducirse el porcentaje de asistencias al 60% por las razones mencionadas, el profesor podrá realizar una evaluación integradora que incluya los temas abordados durante el período lectivo correspondiente para regularizar la asignatura en cuestión. En caso de no aprobarse el mencionado examen, el estudiante quedará en condición de libre. RAM 2484/2013 – Art. 37°.
- ✔ **75% de aprobación de trabajos prácticos como mínimo, 3 trabajos prácticos aprobados de un total de 4 trabajos prácticos durante la cursada.**
- ✔ **100% de parciales aprobados. 1 (uno) con nota mínima 6 (seis). Con su instancia recuperatoria correspondiente.**

CONDICIONES PARA RENDIR COMO ALUMNO REGULAR.

El estudiante se deberá presentar ante el tribunal con DNI, libreta y programa de la asignatura.

Se evaluará de acuerdo a su condición:

Para estudiantes Regulares

Modalidad: oral

- ✔ Para la acreditación del espacio y en condición de regular el alumno deberá realizar una defensa frente al Tribunal, de una unidad o tema seleccionado por la docente. Luego de dicha exposición, el Tribunal podrá realizar preguntas sobre el mismo tema y sobre otros temas del programa.
- ✔ Se considera acreditado el espacio si el Tribunal califica al alumno con una nota de al menos 4 (cuatro) puntos sobre 10 (diez) posible.

CONDICIONES PARA RENDIR COMO ALUMNO LIBRE.

Consta de dos instancias:

- ✔ En primera instancia, deberá realizar un examen en computadora, que consistirá en la resolución de un problema computacional, el cual abarcará todas las unidades temáticas planteadas en el presente programa. Para su aprobación deberá obtener una calificación mínima de 4 (cuatro) puntos sobre 10 (diez) posible.
- ✔ En segunda instancia deberá realizar la defensa de una unidad o tema seleccionado por el docente. Luego de dicha defensa, el Tribunal podrá realizar preguntas sobre el mismo tema y sobre otros temas del programa.

Nota: Sólo podrá pasar a la segunda instancia si aprueba la primera instancia y la calificación final, será el promedio de ambas instancias. En caso de desaprobar la primera o segunda instancia, la nota del aplazo será la definitiva.

MARCO TEÓRICO

En los últimos años se ha diferenciado entre usar la tecnología, consumir aplicaciones de forma instrumental y participar del mundo tecnológico a través de la comprensión de la computación para pensar, resolver problemas y crear tecnología. Si bien la alfabetización digital, requiere del saber usar los dispositivos y programas, limitamos a ello en el contexto del sistema educativo supondría negarles a los estudiantes la posibilidad de comprender los modos de producción, funcionamiento y potencialidad de los artefactos computacionales. Para participar críticamente del mundo digital es necesario formar en el uso responsable y profundo de las tecnologías y habilitar a los estudiantes experiencias para que desarrollen nuevas soluciones y herramientas.
La inclusión de las Ciencias de la Computación en el nivel educativo se ha traducido en un leve incremento de los estudiantes que eligen carreras informáticas como formación superior, pero que sigue siendo enormemente desigual e insuficiente desde el punto de vista de los requerimientos de recursos humanos por parte del sector productivo.

¿Por qué es importante la programación de sistemas?

Hoy en día y después de vivir una pandemia, hemos visto como la tecnología ha repuntado de una manera sorprendente, y gran base de es repunte ha sido gracias a la programación, ya que aplicaciones, páginas web, software y todas las herramientas para trabajar en home office o tomar clases desde casa están hechas a base de programación. Y es ahí donde la programación se hace de suma importancia. Pero ¿Qué es la programación? ¿Por qué tiene tanta importancia hoy saber programar?

En la actualidad, la noción de programación se encuentra muy asociada a la creación de aplicaciones de informática y video juegos. En este sentido, es el proceso por el cual una persona desarrolla un programa, valiéndose de un lenguaje de programación que le permita escribir el código, como C++, C#, Java, PHP, Python, Javascript, entre otros. La programación es un idioma más y aprenderlo nos beneficia y ayuda a lograr una comunicación directa con el ambiente tecnológico que nos rodea. Actualmente se dice que aprender el arte de la programación tiene tanta importancia en el campo laboral como aprender inglés, siendo ambos un idioma universal.

¿Cuáles son los beneficios de implementar la programación en el aula?
Implementar la programación en el aula busca que los alumnos no sólo sean consumidores de tecnología sino también productores.

La integración de este campo de conocimiento permite a los estudiantes desarrollar habilidades fundamentales para solucionar diversas problemáticas sociales, crear oportunidades y prepararse para su integración en el mundo del trabajo.

La capacidad de abstracción, de encontrar patrones, de ordenar de manera operativa y de identificar los componentes de un problema son habilidades sobre las que trabaja la programación. No están necesariamente vinculadas con una computadora, y pueden aplicarse a diversas situaciones. Es por eso que la programación resulta una disciplina fundamental en la educación contemporánea.

Algunos de los beneficios son:

- ✔ Desarrolla habilidades lingüísticas y lógico-matemática: Trabajar con la programación desarrolla habilidades lingüísticas y lógico-matemática, dado que necesitamos establecer, un orden, una secuencia, abstraer ideas, reconocer patrones, diseñar alternativas de soluciones y socializarlas.
- ✔ Estimula la creatividad: Estimula la creatividad en el desarrollo de soluciones. Existen muchos caminos a una solución y el dejar que los alumnos propongan una solución, motiva su desarrollo creativo.
- ✔ Trabaja la resolución de problemas: La creatividad y el desarrollo de habilidades lógico-matemática, se potencian al buscar una solución a una problemática. Vemos que no es lo mismo, enseñar cómo solucionar una actividad propuesta, a pedirles que ellos lo solucionen. No les enseñamos el procedimiento de solución. Ellos lo descubren y ese aprendizaje por descubrimiento le da un significado a un nuevo conocimiento.

- ✔ Fomenta el aprendizaje colaborativo: La programación incentiva el espíritu crítico y facilita la interactividad. Si los alumnos trabajan en grupo para resolver problemas veremos que cada integrante es diferente y aporta un conocimiento distinto, desde su realidad subjetiva.

Estos beneficios están relacionados con la teoría de las Inteligencias Múltiples de Howard Garden (1983): La Inteligencia Lingüística, Inteligencia Lógico-Matemática, Inteligencia Espacial, Inteligencia Interpersonal e Inteligencia Intrapersonal.

¿Por qué aprender estructuras de datos: pila (stack), cola (queue), listas (lists)?

- ✔ Aprender estos conceptos es fundamental en programación porque estas estructuras de datos proporcionan formas eficientes de organizar y manipular datos en función de las necesidades específicas de cada problema. En otras palabras, conocer estas estructuras y cómo implementarlas adecuadamente te permitirá escribir programas más eficientes. Por ejemplo, optimizar recursos, resolver problemas y algoritmos, comprender y utilizar lenguajes de programación eficientemente, y tener éxito en entrevistas y pruebas técnicas relacionadas con la informática:

- ✔ Optimización de recursos: El conocimiento de estas estructuras de datos permite optimizar el uso de recursos de la computadora. Por ejemplo, las pilas son útiles para gestionar la memoria en llamadas recursivas y en la resolución de problemas donde se necesita un seguimiento de estados anteriores. Las colas son eficientes para gestionar recursos compartidos entre múltiples tareas, evitando bloqueos y maximizando el rendimiento. Las listas, al ser dinámicas, permiten administrar y aprovechar el espacio de manera más eficiente, ya que pueden crecer o reducirse según la cantidad de datos que contengan.

- ✔ Resolución de problemas y algoritmos: Estas estructuras de datos se utilizan ampliamente en algoritmos y problemas comunes en ciencias de la computación. Por ejemplo, en la búsqueda del camino más corto en un grafo (como el algoritmo de Dijkstra), en la implementación de algoritmos de ordenamiento (como el algoritmo de ordenamiento rápido Quicksort), en la simulación de procesos (como en el algoritmo de simulación de colas) y en el manejo de expresiones matemáticas complejas con paréntesis.

- ✔ Comprensión de lenguajes y bibliotecas de programación: Muchos lenguajes de programación y bibliotecas estándar implementan estas estructuras de datos de manera predeterminada o proporcionan bibliotecas que permiten su fácil implementación. Comprender cómo funcionan internamente las pilas, colas y listas, y cómo usarlas eficientemente, permite sacar el máximo provecho de las características que proporcionan estos lenguajes y bibliotecas, y te permite escribir código más limpio y eficiente.

- ✔ Entrevistas y pruebas técnicas: En el campo de la informática, especialmente en entrevistas de trabajo y pruebas técnicas, es común que se presenten preguntas relacionadas con estas estructuras de datos y su uso en diferentes problemas. Tener un buen dominio de las pilas, colas y listas te ayudará a destacarte como candidato y resolver con éxito este tipo de preguntas.

PROPÓSITOS Y METAS DE COMPRENSIÓN.

PROPÓSITOS DEL DOCENTE

- ✔ Potenciar el pensamiento lógico, matemático y algorítmico.
- ✔ Desarrollar la capacidad de resolución de problemas.
- ✔ Incentivar el aprendizaje autónomo.
- ✔ Fomentar el pensamiento crítico.
- ✔ Propiciar el trabajo colaborativo.

OBJETIVOS

- ✔ Entender el funcionamiento y la utilidad del IDE Python
- ✔ Aprender la sintaxis y semántica del lenguaje de programación Python

- ✔ Adquirir habilidades para la resolución de problemas utilizando tipos de datos y estructuras de control: estructuras repetitivas, estructuras de alternativa.
- ✔ Entender los conceptos de las estructuras dinámicas de datos (pilas, colas y listas enlazadas) y el algoritmo asociado a cada una de ellas.
- ✔ Entender el concepto de sistemas de archivos y los algoritmos asociados para crear, leer y modificar un archivo.

CONTENIDOS: EJES / UNIDADES / MÓDULOS.

Unidad 1: Fundamentos de la programación orientada a objetos y estructuras de datos dinámicos

- ✔ **Introducción a la programación orientada a objetos (POO)**
  - o Conceptos básicos de POO: clases, objetos, encapsulamiento, herencia, polimorfismo.
- ✔ **Tipos abstractos de datos (TAD)**
  - o Definición y características.
- ✔ **Variables dinámicas y concepto de referencia**
  - o Uso de punteros en la gestión de memoria.
- ✔ **Estructuras dinámicas lineales**
  - o Listas, pilas y colas.
  - o Representación en memoria y operaciones básicas.

Unidad 2: Programación avanzada y manipulación de estructuras de datos lineales

- ✔ **Comparación entre estructuras de datos lineales y arreglos**
  - o Ventajas y desventajas.
- ✔ **Array de punteros y punteros a funciones**
  - o Utilidades y aplicaciones.
- ✔ **Extensiones a listas, pilas y colas**
  - o Implementación de funcionalidades adicionales.
- ✔ **Recursividad**
  - o Concepto y características.
  - o Análisis comparativo con soluciones iterativas.

Unidad 3: Estructuras de datos compuestas y manipulación de archivos

- ✔ **Introducción a estructuras de datos compuestas**
  - o Registros como estructuras de datos compuestas.
- ✔ **Archivos en programación**
  - o Definición y modos de apertura.
  - o Operaciones de alta, baja y modificación.
- ✔ **Acceso secuencial y acceso directo a archivos**
  - o Diferencias y aplicaciones.
- ✔ **Manejo de errores en operaciones de archivos**

Unidad 4: Introducción a estructuras dinámicas no lineales y aplicación práctica en Python

- ✔ **Introducción a estructuras dinámicas no lineales**
  - o Árboles y grafos.
- ✔ **Representación en memoria**
  - o Métodos de almacenamiento y acceso.
- ✔ **Operaciones básicas con estructuras dinámicas no lineales**
  - o Implementación en Python.
- ✔ **Aplicación práctica en Python utilizando Visual Studio Code**
  - o Ejemplos y ejercicios para consolidar los conceptos aprendidos.

METODOLOGÍA DE TRABAJO.

La metodología predominante para llevar a cabo la consecución de los objetivos planteados es el Aprendizaje Basado en la Indagación y Aula Invertida.

Para cada eje temático planteado y como primera medida se pondrá en conocimiento de los alumnos cuáles serán los objetivos que tendrán que alcanzar, tanto a nivel teórico como en la práctica, los cuales son en definitiva los objetivos que se proponen para la asignatura. Los que se evaluarán y discutirán a la finalización de cada una de las unidades temáticas.

La evaluación diagnóstica servirá para recabar información sobre los conocimientos previos de los estudiantes, de esa forma poder ajustar el nivel de las clases teóricas.

Para el desarrollo de cada uno de los temas teóricos se utilizará la estrategia de Aula Invertida, donde cada alumno recibirá con anterioridad la teoría que se verá en clases. Esto facilita la enseñanza del docente, ya que el alumno llega al aula con un conocimiento básico del tema. De esta manera el docente puede enfocarse más tiempo en la resolución de ejemplos prácticos, que servirán de base a los alumnos en la resolución de los trabajos prácticos.

Para el desarrollo de los trabajos prácticos se utilizará la estrategia del Aprendizaje Basado en la Indagación, donde el docente a través de la indagación va guiando al alumno en la comprensión del problema y en su resolución. Para dicho fin se utilizará el Visual Studio Code y Python

También se incorporarán otros tipos de recursos de acuerdo a las necesidades del grupo para promover el aprendizaje, por ejemplo, video tutorial, plataforma web de la institución y/o material bibliográfico en formato PDF.

Cabe aclarar que la tarea fundamental del docente se centrará en el trabajo tanto individual como grupal de los alumnos, lo segundo como tarea social entre pares tendiente a compartir conocimientos e ideas.

RECURSOS DIDÁCTICOS

Teniendo en cuenta que existen elementos que conllevan una intencionalidad didáctica y que, en consecuencia, pueden utilizarse en determinadas circunstancias como recurso para facilitar procesos de enseñanza y aprendizaje, se propone implementa los siguientes recursos didácticos, con la intención de facilitar al alumno el acceso al conocimiento, la reflexión y la toma de decisiones.

Materiales:

- ✔ Diapositivas con la presentación de cada tema.
- ✔ Material bibliográfico en formato digital
- ✔ Vídeos.
- ✔ Páginas web de consulta.
- ✔ Software Visual Studio Code, Python y Google Colab
- ✔ Proyector.
- ✔ Computadora/netbook.
- ✔ Pizarra, tizas y borrador

EVALUACIÓN: MODALIDAD, CRITERIOS DE EVALUACIÓN PARA EL CURSADO.

En esta propuesta, se entiende a la evaluación como una herramienta sistemática y continua, parte del proceso de enseñanza y aprendizaje y no como un componente aislado, que tiene como objeto recabar información para mejorar, orientar y reflexionar sobre el proceso llevado a cabo. En otras palabras, sabemos que la evaluación tiene un rol formativo.

Criterios de evaluación

Se evaluará según los siguientes criterios:

- ✔ Manejo de sintaxis y semántica del lenguaje Python
- ✔ Construcción y relación entre conceptos de Pila, Cola, Listas Enlazadas y Archivos.
- ✔ Planteamiento y resolución de problemas a través de la codificación de un programa utilizando el lenguaje Python
- ✔ Actitud personal frente a situaciones problemáticas y las argumentaciones propuestas para la misma.
- ✔ Expresión oral utilizada.
- ✔ Participación y pertinencia de las actuaciones e intervenciones.
- ✔ Predisposición para la tarea y el aprendizaje individual y grupal.

Instrumentos de evaluación

- ✔ Trabajos prácticos evaluativos en computadora.
- ✔ Exposición de soluciones a problemas planteados.
- ✔ Corrección de trabajos prácticos.

CRITERIOS DE EVALUACIÓN PARA RENDIR EXÁMENES FINALES.

Se evaluará según los siguientes criterios:

- ✔ Manejo de sintaxis y semántica del lenguaje Python.
- ✔ Construcción y relación entre conceptos de Pila, Cola, Listas Enlazadas y Archivos.
- ✔ Ejemplos utilizando el l enguaje Python.
- ✔ Expresión oral y lenguaje técnico utilizado.

Instrumentos de evaluación

- ✔ Evaluaciones orales.
- ✔ Exposición de soluciones a problemas planteados.

BIBLIOGRAFÍA GENERAL Y ESPECÍFICA Y/O COMPLEMENTARIA.

Bibliografía General:

McLaughlin, B.D. (2007). *Análisis y Diseño Orientado a Objetos de Cabeza Primero*, Editorial: **O'Reilly Media.**
Weisfeld, M. 5° Ed. (2019). *El Proceso de Pensamiento Orientado a Objetos.* Editorial: **Addison-Wesley Professional.**
Freeman, E. 2° Ed. (2021). *Patrones de diseño de cabeza y en primer lugar.* Editorial: O'Reilly Media.
Phillips, D. 3° Ed. (2018). *Programación orientada a objetos con Python 3*. Editorial: Packt Publishing.

Bibliografía especifica:

Unidad 1: Phillips, D. 3° Ed. (2018). *Programación orientada a objetos con Python 3*. Editorial: Packt Publishing. Unidad 1 a 5.
Unidad 2: Phillips, D. 3° Ed. (2018). *Programación orientada a objetos con Python 3*. Editorial: Packt Publishing. Unidad 6 y 7.
Unidad 3: Phillips, D. 3° Ed. (2018). *Programación orientada a objetos con Python 3*. Editorial: Packt Publishing. Unidad 8 y 9.
Unidad 4: Phillips, D. 3° Ed. (2018). *Programación orientada a objetos con Python 3*. Editorial: Packt Publishing. Unidad 10 a 13.