UNIVERSIDAD NACIONALDE SANTIAGO DEL ESTERO

FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍAS

PLANIFICACIÓN ANUAL 2024

ASIGNATURA:

FUNDAMENTOS DE LA PROGRAMACIÓN

LICENCIATURA EN SISTEMAS DE INFORMACIÓN

Plan de Estudio: 2011
Innovación Curricular 2022

Equipo cátedra

Profesores adjuntos: **Ing. Carmen Silva (Docente responsable)**
Lic. Irene Salazar

JTP: **Ing. Beatriz Fernández Reuter**

Auxiliares Docentes de Primera:
Ing. Aldo Roldan, Lic. Víctor Diaz, Prof. Walter Toledo

Ayudantes Estudiantiles:
Martin Atia, Martin Palavecino

PLANIFICACIÓN DE LA ASIGNATURA

1- IDENTIFICACIÓN:

1.1- Nombre de Asignatura: Fundamentos de la Programación

1.2- Carrera: Licenciatura en Sistemas de Información

1.3- Plan de Estudios: 2011 – Innovación Curricular 2022

1.4- Año académico: 2024

1.5- Carácter: Obligatoria

1.6- Ubicación de la Asignatura en el Plan de Estudios

1.6.1- Módulo – Año: anual – 1º Año

1.6.2- Trayecto al que pertenece la Asignatura/Obligación Curricular

TRAYECTO	CARGA HORARIA PRESENCIAL
Ciencias Básicas y Específicas	
Algoritmos y Lenguajes	180 hs.
Arquitectura, Sistemas Operativos y Redes	
Ingeniería de Software, Bases de Datos y Sistemas de Información	
Aspectos Sociales y Profesionales	
Otros contenidos	
CARGA HORARIA TOTAL DE LA ACTIVIDAD CURRICULAR	180 hs.

Tabla 1: Carga horaria por trayecto

1.6.3-Correlativas

1.6.3.1 Anteriores: no tiene

1.6.3.2. Posteriores:

Débiles: Estructuras de datos y Programación I
Organización del Computador
Teoría de Sistemas y Organizaciones

Fuertes: Base de Datos I
Investigación Operativa

1.7- Carga horaria:

1.7.1. Carga horaria semanal total: 6 horas.

1.7.2. Carga horaria semanal destinada a la formación práctica: 4 horas.

1.7.3. Carga horaria total dedicada a las distintas actividades de formación práctica: 120 hs.

1.8. Ámbitos donde se desarrollan las actividades de formación práctica a las que se hace referencia en el punto anterior: aulas y laboratorios de computadoras del Dpto. de Informática - FCEyT.

1.9. Indique la cantidad de comisiones en las que se dicta la asignatura: 3 comisiones

2- PRESENTACIÓN

2.1. Ubicación de la Asignatura como tramo de conocimiento de una disciplina

Esta asignatura forma parte del Trayecto Algoritmos y Lenguajes. Su contenido está principalmente orientado a la resolución de problemas computacionales a través del uso de un lenguaje de programación, adoptando el enfoque del paradigma imperativo. Se exploran estrategias de solución con criterios de calidad, eficiencia y corrección, para finalmente llegar a la implementación acorde a las especificaciones planteadas. En el desarrollo de contenidos se abordan estructuras de datos, algoritmos fundamentales y técnicas de verificación que permiten al alumno, aplicarlos en la resolución de problemas computacionales como propósito fundamental de la asignatura.

2.2. Conocimientos y habilidades previas que permiten encarar el aprendizaje de la Asignatura.

Los conocimientos y habilidades previas requeridas para cursar esta asignatura son los construidos y evaluados en los espacios de matemática e Informática del curso de ingreso dictado por la Facultad de Ciencias Exactas y Tecnologías.

2.3. Aspectos del Perfil Profesional del Egresado a los que contribuye la asignatura

Esta asignatura contribuye a que el egresado posea:

- Los conocimientos básicos (lógico-matemáticos y computacionales) para una adecuada fundamentación teórica de su quehacer profesional específico.

En esta asignatura se proporciona al alumno la capacitación básica necesaria para:

- Identificar, analizar y resolver problemas computacionales, mediante la selección de las estructuras de datos adecuadas para el diseño de algoritmos fundamentados.

2.4. Integración horizontal y vertical con otras asignaturas.

El diagrama de la figura 1 muestra la articulación horizontal y vertical de la asignatura dentro del Plan de Estudios.

- La articulación vertical previa:

Fundamentos de la Programación por ser una asignatura de primer año no tiene establecida una asignatura previa en el Plan de Estudios, sin embargo, la articulación vertical previa se realiza con los contenidos de matemática e informática construidos y evaluados en el curso de ingreso.

- La articulación horizontal se concreta a través de:

Algebra I: donde los estudiantes desarrollan competencias para emplear vectores y matrices como recursos matemáticos para la interpretación y resolución de problemas.

Lógica I: donde los estudiantes desarrollan competencias para interpretar los conceptos, técnicas y procedimientos referidos al Cálculo Proposicional y el Cálculo de Predicados.

- en forma vertical y horizontal, según se muestra en la siguiente figura.

Asignaturas posteriores en las cuales se requiere que los estudiantes hayan desarrollado las competencias definidas en el presente programa para poder alcanzar nuevas competencias:

Estructura de datos y Programación I: donde los estudiantes desarrollan competencias para seleccionar y procesar la información necesaria para la resolución de problemas.

Organización del computador: donde los estudiantes desarrollan competencias para diseñar funciones y procedimientos que resuelvan problemas dados.

Teoría de Sistemas y Organizaciones: donde los estudiantes desarrollan competencias para la resolución de problemas computacionales.

Investigación Operativa: donde los estudiantes desarrollan competencias para aplicar métodos adecuados para la resolución óptima de un determinado problema e interpretar correctamente los resultados obtenidos.

Base de Datos I: donde los estudiantes desarrollan competencias para la resolución de problemas computacionales.

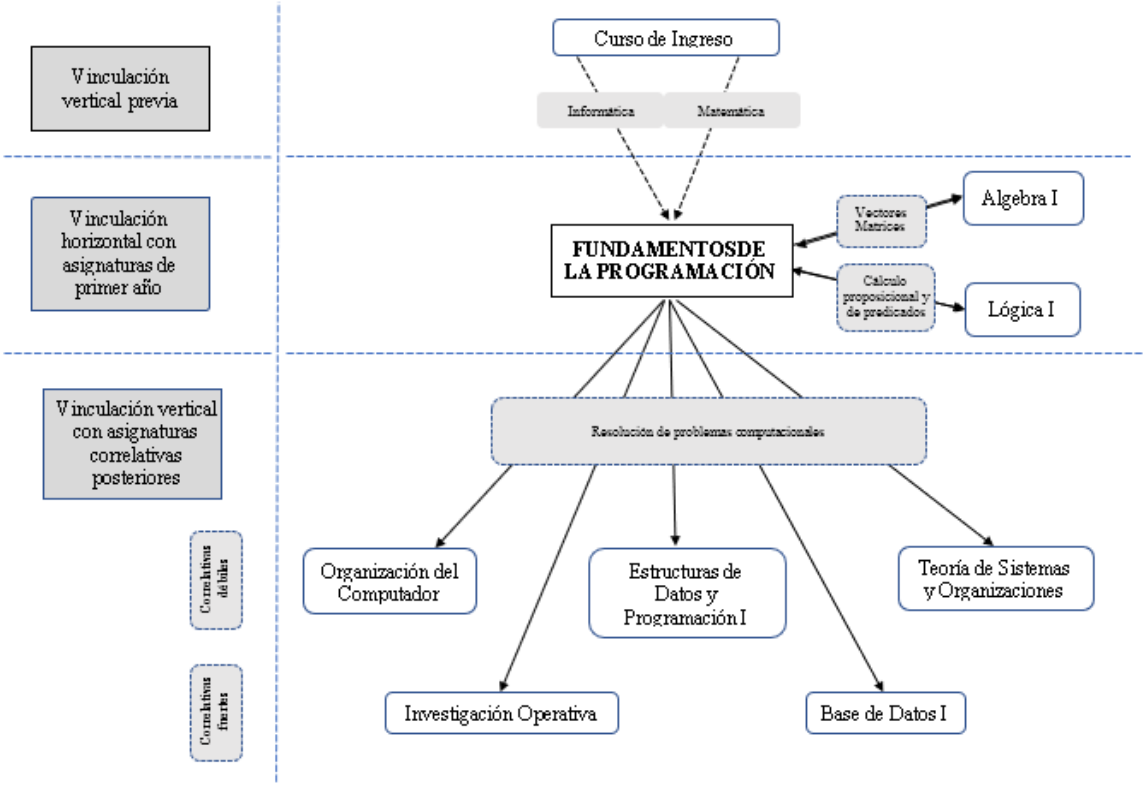

Figura 1. Integración horizontal y vertical de Fundamentos de la Programación en el Plan de Estudios de la LSI

3- OBJETIVOS

Se expresan los objetivos de la asignatura en términos de competencias

3.1. Objetivos Generales

- Desarrollar soluciones algorítmicas de problemas computacionales, aplicando los conceptos fundamentales de la programación imperativa para el logro de soluciones correctas, eficientes y posibles de implementar en un lenguaje de programación.

3.2. Objetivos Específicos

- Que el alumno desarrolle las siguientes competencias básicas:
  - Selecciona y procesa la información necesaria para la resolución de problemas básicos con computadoras yverifica las soluciones encontradas.
- Que el alumno desarrolle las siguientes competencias específicas:
  - Desarrolla habilidades de pensamiento algorítmico adquiriendo la capacidad de pensar y razonar de forma lógica y sistemática para resolver problemas.
  - Identifica los principios transversales de la programación como la abstracción, la modularidad, la reutilización de código, entre otros.
  - Construye aplicaciones de software en el lenguaje de programación Python aplicando, de manera correcta, los conceptos básicos de la programación procedimental.
  - Resuelve, de manera adecuada, situaciones problemáticas aplicando los algoritmos fundamentales de recorrido, búsqueda, ordenamiento, y actualización.
  - Identifica áreas de mejora en términos de eficiencia, legibilidad para diseñar algoritmos eficientes y estructurados.
  - Aplica técnicas de verificación de software evaluando la calidad y funcionalidad de los algoritmos diseñados para garantizar la fiabilidad y eficiencia de las soluciones desarrolladas.

- Que el alumno desarrolle las siguientes competencias transversales:
  - Participa colaborativamente en equipos de trabajo.
  - Plantea la resolución de problemas que se resuelven con computadoras, con iniciativa, toma de decisiones y autonomía.
  - Fortalece la comunicación oral y escrita en el intercambio y transmisión de los conocimientos.

4- SELECCIÓN Y ORGANIZACIÓN DE CONTENIDOS

4.1. Contenidos mínimos establecidos en el Plan de Estudios para la Asignatura

Paradigmas y Lenguajes. Análisis y diseño de algoritmos. Tipos de datos simples. Estructuras de datos elementales. Tipos abstractos de datos. Paradigma de programación imperativo. Resolución de problemas y algoritmos. Estructuras básicas de control. Procedimientos y funciones. Algoritmos fundamentales: recorrido, búsqueda, ordenamiento, actualización. Verificación de Algoritmos: pruebas orientadas a la caja negra y pruebas orientadas a la caja blanca. Lenguaje de programación imperativo.

4.2. Programa Sintético sobre la base de los contenidos mínimos

Unidad 1: Lenguajes y *Paradigmas*

Lenguajes y Paradigmas. Lenguaje de programación imperativo. Paradigma de programación imperativo. Lenguaje de programación Python.

Unidad 2: R*esolución de problemas y Algoritmos*

Resolución de problemas y algoritmos. Etapas en la resolución de problemas con computadora. Algoritmos. Formas de expresar un algoritmo. Tipos de datos simples. Constantes y variables. Estructuras básicas de control. Expresiones

Unidad 3: *Estructuras de datos I*

Estructuras de datos: clasificación. Tipo de dato Arreglo: vectores y matrices. Registros. Algoritmos fundamentales: recorrido, búsqueda, ordenamiento y actualización.

Unidad 4: ***Programación modular y estructurada***

Programación modular y estructurada. Subprogramas o módulos: Procedimientos y Funciones. Ventajas de la modularización. Comunicación entre módulos: parámetros. Alcance de los datos: variables globales y locales.

Unidad 5: ***Verificación***

Estrategias de prueba del software. Verificación de Algoritmos: pruebas orientadas a la caja negra y pruebas orientadas a la caja blanca.

Unidad 6: ***Estructuras de datos II***

Abstracción en los lenguajes de programación. Tipos abstractos de datos (TAD. Diferencia entre tipo de dato y TAD. Pilas y colas. Implementación estática. Operaciones.

4.3- Articulación Temática de la Asignatura

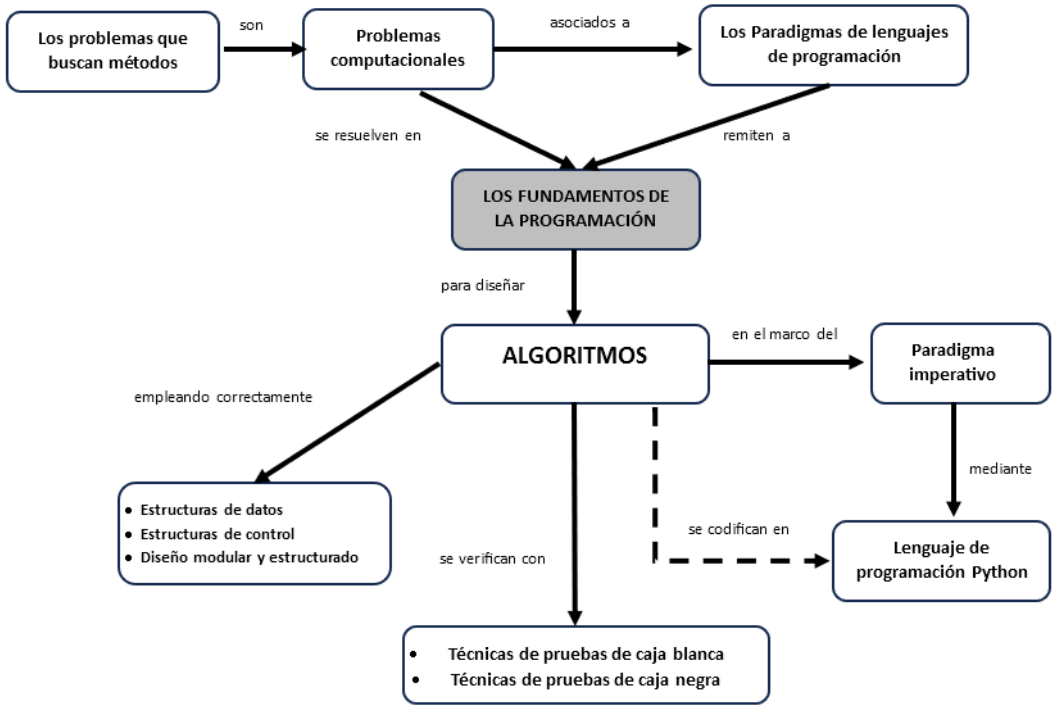

Figura 2. Articulación temática de la asignatura

4.4. Programa Analítico

Unidad 1: Lenguajes y *Paradigmas*

Lenguajes de programación: concepto, tipos de lenguajes y traductores. Paradigmas: concepto, tipos y características. Paradigma de programación imperativo: concepto y principales características. Lenguaje de programación imperativo: Python: entorno. Creación de programas.

Unidad 2: R*esolución de problemas y Algoritmos*

Etapas en la resolución de problemas con computadora: análisis del problema, diseño del algoritmo, codificación del algoritmo, verificación, mantenimiento y documentación. Algoritmos: concepto, propiedades, dominio, errores de compilación, errores de ejecución y errores de lógica. Formas de expresar un algoritmo: diagramas de flujo, pseudocódigo y lenguajes de programación. Tipos de datos simples. Constantes y variables. Estructuras básicas de control. Expresiones aritméticas, lógicas y relacionales.

Unidad 3: *Estructuras de datos I*

Estructura de datos elementales. Clasificación de las estructuras de datos: simples o compuestas, homogéneas o heterogéneas, estáticas o dinámicas. Tipo de dato arreglo: concepto, características. Arreglos unidimensionales: vectores. Arreglos bidireccionales: matrices. Registros: concepto, notación. Operaciones con arreglos: algoritmos fundamentales. Recorrido: concepto, acceso secuencial. Búsqueda: concepto, métodos de búsqueda lineal, binaria y por transformación de claves. Ordenación: concepto,métodos de ordenación selección, intercambio o burbuja. Actualización: concepto, añadir,insertar y eliminar.

Unidad 4: *Programación modular y estructurada*

Programación modular y estructurada: descomposición de problemas. Módulo: concepto y características. Procedimientos. Funciones. Ventajas de la modularización: Independencia entre módulos: la Independencia funcional: cohesión y acoplamiento, modificación de los módulos, reusabilidad y mantenimiento del código. Comunicación entre módulos:

parámetros. Correspondencia entre parámetros actuales y formales. Métodos para el pasaje de parámetros: por valor, por resultado, valor resultado y por referencia. Alcance de los datos: Variables locales y globales.

Unidad 5: *Verificación*

Estrategias de pruebas de software: concepto, organización, estrategias para arquitecturas convencionales. Verificación de algoritmos: concepto, objetivos y principios. Diseño de casos de prueba orientados a la caja blanca: concepto, método del camino básico. Diseño de casos de prueba orientados a la caja negra: concepto, método de particiones de equivalencia y método de análisis de valores límite.

Unidad 6: *Estructuras de datos II*

Abstracción en los lenguajes de programación: abstracciones de control. Abstracciones de datos. Tipos abstractos de datos (TAD): ventajas, implementación, especificación. Diferencia entre tipo de dato y TAD. Pilas y Colas. TAD pila, TAD cola. Implementación estática. Operaciones.

.

4.5- Cronograma para el desarrollo de las Unidades Temáticas

UNIDAD	CARGA HORARIA	CRONOGRAMA DE DICTADO
Unidad 1: Lenguajes y Paradigmas	2	1º semana (1º módulo)
Unidad 2: Resolución de problemas y Algoritmos	10	2º, 3º, 4º, 5º, 6º semana (1º módulo)
Unidad 3: Estructuras de datos I	12	7º, 8º, 9º, 10º, 11º y 12º semana (1º módulo)
Unidad 4: Programación modular y estructurada	6 14	13º, 14º y 15º semana (1º Módulo) 1º 2º, 3º, 4º, 5º, 6º y 7º semana (2º módulo)
Unidad 5: Verificación	8	8º, 9º, 10º y 11º semana (2º módulo)
Unidad 6: Estructuras de datos II	8	12º, 13º, 14º y 15º semana (2º módulo)
TOTAL	**60**	

Tabla 2: Cronograma para el desarrollo teórico de las unidades temáticas

5. FORMACIÓN PRÁCTICA

5.1. Descripción de las actividades de formación práctica

Los trabajos prácticos se derivan de las unidades temáticas de la signatura, considerando la complejidad de los contenidos y la necesidad de su aplicación práctica.

En estas actividades, los alumnos resuelven problemas básicos y problemas similares a los del mundo real elaborando soluciones en computadoras, aplicando el paradigma imperativo.

Los trabajos prácticos incluyen ejercicios que se resuelven mediante la construcción de:

- Diagramas de flujo
- Codificación en el lenguaje de programación Python,
- Verificación de la solución propuesta

La resolución de los ejercicios propuestos contribuye al desarrollo de competencias particulares en el contexto de la formación práctica en programación:

Diagramas de Flujo: Fomentan el **pensamiento algorítmico y la planificación**, esenciales para la creación de soluciones estructuradas.

Codificación en Python: Desarrolla **habilidades técnicas de programación**, preparando a los estudiantes para la implementación efectiva de soluciones.

Verificación: Enfatiza la **prueba y depuración**, asegurando que las soluciones sean funcionales y confiables, habilidades clave en la creación de software de calidad.

Las actividades prácticas se desarrollan tanto en forma individual, promoviendo el desarrollo de habilidades de autogestión y superación frente al error, como en modalidad grupal, fomentando la interacción con sus compañeros, la cooperación y la discusión productiva para resolver el problema planteado.

En las actividades de formación experimental que se realizan en grupo, en modalidad taller en el laboratorio de computadoras, donde se busca que el alumno participe activamente en su proceso de aprendizaje, involucrándose de manera integral en la resolución de problemas de una complejidad intermedia, en el diseño, y en la aplicación práctica de conceptos

teóricos. El objetivo es desarrollar habilidades criticas como la resolución de problemas, el pensamiento crítico, y la capacidad de trabajar en equipo.

En las actividades de formación práctica se espera que el alumno avance en el desarrollo de habilidades cognitivas relevantes para a la programación, con un enfoque en la organización y claridad en la codificación y documentación de los proyectos. Se fomenta la prueba y verificación del código elaborado en Lenguaje Python, utilizando el IDE PyCharm.

Las actividades prácticas contarán con un acompañamiento docente continuo, proporcionando retroalimentación oportuna y constructiva, para que los alumnos corrijan errores, mejoren sus habilidades y tengan claridad sobre las expectativas y criterios de éxito.

.

5.2.-Formación en Ejes Transversales

En la Tabla 3 se relacionan cada uno de los Ejes Transversales de Formación con lasactividades y los resultados de aprendizaje esperados.

Eje	Actividades	Resultados de Aprendizaje	Grado de Profundidaden el tratamiento
Identificación, formulación y resolución de problemas de informática (Computacionales)	• Lectura, interpretación y discusión de enunciados propuestos por el docente, en los trabajos prácticos para identificar los datos de entrada y salida. • Resolver problemas simples propuestos por el docente en los trabajos prácticos, diseñando el algoritmo solución expresado en diagrama de flujo y en lenguaje de programación Python; documentar cada paso del proceso y exponer su fundamentación ante sus compañeros.	**RA1:** Construye algoritmos solución empleando estructuras de control (selección, iteración) y diagramación estructurada para la resolución de problemas (simples y de complejidad intermedia) de manera eficiente, optimizando recursos y garantizando la corrección y claridad en el proceso.	Básico

Eje	Actividades	Resultados de Aprendizaje	Grado de Profundidaden el tratamiento
Utilización de técnicas y herramientas de aplicación en la informática	• Implementar algoritmos fundamentales de recorrido, búsqueda, ordenamiento y actualización en problemas planteados, utilizando las estructuras de datos, y organiza tanto el diagrama de flujo como el código en módulos bien definidos. • Diseñar casos de prueba basados en el problema planteado, incluyendo datos de entrada para validar los datos de salida (caja negra) y verifica la lógica del programa (caja blanca) utilizando el enunciado y el algoritmo solución expresado en diagrama de flujo y código, para identificar y corregir errores	**RA2:** Aplica adecuadamente las estructuras de control y las estructuras de datos, junto con algoritmos fundamentales de recorrido, búsqueda (lineal o binaria), ordenamiento (selección o burbuja) y actualización (modificación de los datos), para resolver problemas de computación utilizando las técnicas de programación modular y estructurada. **RA3:** Diseña y verifica casos de prueba basados en el problema planteado, evaluando el algoritmo solución expresado en diagrama de flujo y codificado en Python para identificar y corregir errores, aplicando técnicas de pruebas de software como caja blanca y caja negra.	Básico
Generación de desarrollos tecnológicos y/o innovaciones tecnológicas.	• Resolver el problema de complejidad intermedia propuesto como actividad de formación experimental, siguiendo el enfoque modular. El proceso abarca la planificación, diseño del sistema, implementación del código, pruebas unitarias e integrales, y la documentación del proyecto. Al final, presentar y defender la solución planteada ante la clase, explicando las decisiones técnicas y cómo garantizo la eficiencia y escalabilidad del software.	**RA4:** Desarrolla y documenta un sistema codificado en Python para resolver un problema de una complejidad intermedia, utilizando técnicas de programación modular y estructurada, aplicando buenas prácticas de diseño y pruebas para garantizar la eficiencia, la escalabilidad y la correcta funcionalidad del software.	Básico

Eje	Actividades	Resultados de Aprendizaje	Grado de Profundidaden el tratamiento
Fundamentos para el desempeño en equipos de trabajo	• Presentar el trabajo desarrollado como actividad de formación experimental, explicando al resto de la clase la colaboración individual para resolver el problema y cómo la dinámica grupal empleada contribuye al éxito del equipo, de manera responsable una comunicación clara, apoyo mutuo y una integración efectiva de ideas para asegurar un trabajo colaborativo y respetuoso.	**RA5:** Colabora eficazmente en equipos de trabajo, aportando de manera responsable conocimientos teóricos y prácticos para la revisión de conceptos y la resolución conjunta de problemas, favoreciendo la dinámica grupal y el logro de los objetivos planteados.	Básico

5.3 Cronograma de formación práctica

Las actividades prácticas incluyen los trabajos prácticos con su correspondiente desarrollo de solución mediante la creación de proyectos en lenguaje Python.

ACTIVIDAD	CARGA HORARIA	CRONOGRAMA DE DESARROLLO
T.P.1 Estructuras Básicas de Control	20	Semanas 1, 2, 3, 4 y 5
T.P.2 Estructura de Datos Arreglos	28	Semanas 7,8, 9, 10, 11, 12 y 13
T.P.3 Programación Estructurada y Modular	12	Semanas 13,14 y 15
T.P.4 Estructura de Datos Registro	24	Semanas 1,2,3,4,5 y 6
T.P.5 Verificación	20	Semanas 7, 8, 9 10 y 11
T.P.6 Estructura de Datos Pilas y Colas	16	Semanas 12,13,14 y 15
TOTAL	**120**	

Tabla 4: Cronograma para el desarrollo de las Actividades Prácticas

6.BIBLIOGRAFÍA.

TÍTULO	AUTORES	EDITORIAL	EJEMPLARES DISPONIBLES	AÑO DE EDICIÓN
Algoritmos, datos y programas	De Giusti, Armando	Prentice Hall	1 (uno)	2001, Argentina
Fundamentos de Programación	Joyanes Aguilar, Luis	McGraw-Hill	1 (uno)	2003, España
Introducción a la Informática	Abásolo Guerrero María J. Perales López Francisco J.	UIB	1 (uno)	2011, Palma
Estructuras de Datos	Cairo, Osvaldo y Guardati, Silvia	McGraw Hil	1 (uno)	2006, Mexico
Ingeniería del software	Pressman, Roger	McGraw Hil	1 (uno)	2006, España
Introducción a la Programación	Abásolo Guerrero, Perales Lopez	Universitat de les Illes Balears	1 (uno)	2011, España
Introducción práctica a la programación con Python	Algar Díaz, M. J. & Fernández de Sevilla Vellón, M.	Editorial Universidad de Alcalá	Biblioteca Digital eLibro.net	2019
	Link de acceso: https://elibro.net/es/lc/unsebiblio/titulos/124259			

TÍTULO	AUTORES	EDITORIAL	EJEMPLARES DISPONIBLES	AÑO DE EDICIÓN
Lógica de programación	Ramírez Marín, J. H	Ediciones de la U.	Biblioteca Digital eLibro.net	2017
	Link de acceso: https://elibro.net/es/lc/unsebiblio/titulos/70315			
Diseño y construcción de algoritmos	Mancilla Herrera, A.	Universidad del Norte	Biblioteca Digital eLibro.net	2015
	https://elibro.net/es/lc/unsebiblio/titulos/69931			
Algoritmos y programación: mejores prácticas	Ayala San Martín, G.	Fundación Universidad de las Américas Puebla (UDLAP).	Biblioteca Digital eLibro.net	2020
	https://elibro.net/es/lc/unsebiblio/titulos/180290			
Fundamentos iniciales de lógica de programación I. Algoritmos en PseInt y Python	Ramírez Marín, J. H	Institución Universitaria de Envigado	Biblioteca Digital eLibro.net	2019
	Link de acceso: https://elibro.net/es/lc/unsebiblio/titulos/226488			

Tabla 5: Bibliografía

7. ESTRATEGIAS METODOLÓGICAS.

7.1- Aspectos pedagógicos y didácticos

En el marco de una Pedagogía activa y de estrategias de enseñanza dinámicas, se propone una metodología de enseñanza interactiva que favorezca la comunicación entre el docente y el alumno y los alumnos entre sí; además se formaran grupos operativos que mediante el trabajo colaborativo y el apoyo de los ayudantes estudiantiles avancen en un aprendizaje significativo que tienda al logro de competencias para un aprendizaje autónomo.

Las actividades de estudio tanto prácticas como de tipo teórico serán apoyadas y desarrolladas mediante el aula virtual de la asignatura alojada en la plataforma Moodle del Centro Universitario Virtual (CUV - http://cuv.unse.edu.ar). A través de esta plataforma se acercarán alestudiante diferentes recursos didácticos creados por el docente tales como videos tutoriales, cuestionarios evaluativos, presentación digital de las actividades y proyectos

propuestos en las prácticas. Cada contenido desarrollado es mediado por el material digital de estudio para su consulta permanente.

Tanto en las clases teóricas como en las clases prácticas, se aplicará la metodología de aula invertida. Los alumnos realizarán actividades de aprendizaje y de evaluación formativa previamente a los encuentros con los docentes. Durante las clases se completarán las actividades requeridas para el abordaje de cada tema y se organizarán tareas individuales y grupales colaborativas que favorezcan la apropiación del contenido.

Los conceptos teóricos y los aspectos metodológicos de la práctica requerida son abordados tanto en los encuentros teóricos como en los prácticos, a fin de fortalecer las habilidades de resolución algorítmica. En la práctica, se plantea a los alumnos diferentes problemas en los que se solicita, la programación en lenguaje Python, de la resolución del problema a fin de fortalecer las habilidades de programación.

Se utiliza software libre en la tarea de desarrollar habilidades relacionadas con la programación en un lenguaje procedimental, habilidades que son necesarias para la resolución de problemas y el diseño de algoritmos; además permite preparar a los alumnos a enfrentar los distintos cambios del mundo tecnológico.

Las diversas actividades tienden a que, el alumno, vincule la teoría con la práctica y viceversa, fomentando el desarrollo de habilidades para la comunicación efectiva mediante la expresión oral y/o escrita de los conceptos fundamentales, que serán documentados para la evaluación formativa del alumno.

7.2- Mecanismos para la integración de docentes

Se realizarán actividades de revisión y coordinación en el área Algoritmos y Lenguajes en el marco de la Comisión de Seguimiento del Plan de Estudios de la carrera de Licenciatura en Sistemas de Información. Además, se efectuarán reuniones periódicas con las cátedras del área Algoritmos y Lenguajes con el fin de aunar tareas conjuntas de integración y articulación.

La coordinación entre las asignaturas se realiza mediante encuentros periódicos en los que

se revisan los logros alcanzados en la comprensión y aplicación del paradigma de programación estudiado y las actividades prácticas.

7.3- Recursos Didácticos

Se utilizarán como recursos didácticos:

i. Bibliografía actualizada impresa y digital (Biblioteca eLibro) y apuntes elaborados por el equipo docente de la asignatura, disponibles en la plataforma del CUV y en el Centro de Documentación del Departamento de Informática.

ii. Diapositivas y videos elaborados por los docentes de la asignatura, demos, tutoriales, entre otros.

iii. Como apoyo a las actividades académicas, se dispondrá del espacio que la asignatura tiene asignada en la plataforma Moodle del Centro Universitario Virtual (CUV), que permitirá al alumno acceder a los recursos digitales, apoyo de prácticas, entre otros. También brindará a los alumnos un canal de comunicación permanente donde podrán acceder a información actualizada de la asignatura: fechas de evaluaciones, resultados de parciales, condición final de la cursada, entre otros.

8. EVALUACIÓN

8.1- Evaluación Diagnóstica

Se recurrirá a la información del desempeño de los alumnos en el curso de ingreso.

8.2- Evaluación Formativa

Se implementará un proceso de evaluación formativa de carácter sistemático, cualitativo y continuo, se pretende resaltar el papel de la evaluación orientadora, reguladora y motivadora.

Se busca obtener información relevante respecto al proceso de enseñanza y el proceso de aprendizaje que permita comprender cómo se producen dichos procesos y tomar las decisiones pertinentes en vistas a mejorar tanto el desarrollo como los resultados del aprendizaje.

La evaluación formativa permitirá el reconocimiento, la observación y propuesta superadora de las dificultades y posibilidades de aprendizaje.

La evaluación formativa de los contenidos teóricos se llevará a cabo mediante la aplicación de las herramientas disponibles en la plataforma Moodle del CUV. En particular se utilizará la herramienta Cuestionario (CD), siguiendo la técnica de la pregunta, las lecciones, videos H5P, Taller, tarea, paquete Scorm entre otros.

En cuanto a las competencias relacionadas a la práctica, la evaluación formativa se llevará a cabo aplicando la técnica de resolución de problemas propuestos en cada trabajo práctico.

8.2.1. Programa y Cronograma de Evaluaciones Formativas.

		MESES				
ACTIVIDAD	**TEMA**	**Mayo**	**Julio**	**Agosto**	**Octubre**	**Noviembre**
Cuestionario 1	Estructuras de Datos: Arreglos	1º Semana				
Cuestionario 2	Programación Modular		3ª Semana			
Cuestionario 3	Estructura de Datos: Registro			3º Semana		
Cuestionario 4	Verificación				3º Semana	
Cuestionario 5	Estructuras de datos: Pilas y Colas					3º Semana

Tabla 7: Cronograma de evaluaciones teóricas formativas

Los cuestionarios se llevarán a cabo mediante la plataforma del CUV o en su defecto mediante prueba escrita y se realizarán en horarios de clase de teoría.

Se detalla a continuación objetivos, modalidad y criterios de evaluación de los trabajos prácticos.

TRABAJOS PRÁCTICOS	OBJETIVOS
TP 1 Estructuras básicas de control	Resolver problemas utilizando las estructuras de control y estructuras de datos
TP 2 Estructuras de Datos I: Arreglos	Resolver problemas utilizando las estructuras de datos adecuada aplicando para la solución de los problemas algoritmos fundamentales de recorrido, búsqueda, ordenamiento y actualización.
TP 3 Programación estructurada y modular	Aplicar el diseño estructurado y modular para la solución algorítmica identificando correctamente los subproblemas: procedimientos y funciones.
TP 4 Registros y Programación estructurada y modular	Aplicar el concepto de diseño estructurado y modular aplicando la estructura de datos registro.
TP 5 Verificación	Diseñar casos de prueba para los problemas planteados, aplicando la técnica de caja negra: Análisis de valores límites y la técnica de caja blanca: Método del camino básico
TP 6 Estructuras de datos II: pilas y Colas	Reconocer las principales características de las estructuras de datos Pilas y Colas, diseñando el algoritmo solución correspondiente mediante el desarrollo de sus procedimientos básicos.

ASPECTOS COMUNES A TODOS LOS TRABAJOS PRÁCTICOS	
OBJETIVOS GENERALES	Que el estudiante adquiera las competencias para: • Usar las estructuras de datos con sus operaciones básicas en la resolución deproblemas. • Desarrollar las soluciones algorítmicas mediante la utilización del paradigma procedimental.
MODALIDAD	• Resolución individual de problemas mediante la diagramación de flujo, aplicando el Paradigma procedimental. • Elaboración individual y grupal, de proyectos de programación que hacen uso de las estructurasde datos y las estructuras de control adecuadas
CRITERIOS DE EVALUACIÓN	Cada ejercicio seleccionado recibirá una calificación de aprobado o desaprobado. Para aprobar la presentación debe cumplir como mínimo con los siguientes ítems: • El trabajo práctico debe estar desarrollado completamente. • Los diagramas de flujo deben haber sido desarrollados por el alumno siguiendo las pautas fijadas en los Recursos Bibliográficos. • La codificación en lenguaje Python debe realizarse siguiendo los principios de la programación estructurada. • La presentación del código de los ejercicios solicitados deberá realizarse en tiempo y formaa través de la plataforma CUV.FCEyT.

8.3- Evaluación Parcial

8.3.1- Programa de Evaluaciones Parciales

En la siguiente tabla se muestra el programa de evaluaciones parciales para el presente año académico.

Evaluación	Tema	Modalidad	Fecha	Devolución de resultados
Parcial Práctico 1	Temas Estructuras básicas de control Estructuras de Datos I: Arreglos	Especialmente diseñada, individual, escrita, resolución de un problema propuesto mediante diagrama de flujo . Verificar la solución propuesta con una prueba de escritorio.	12 de junio (2 hs)	26 de junio
Parcial de Python 1	Temas Estructuras básicas de control Estructuras de Datos I: Arreglos	Individual Escrito en computadora. Resolución de un problema propuesto mediante: Diagrama de flujo Codificación aplicando lenguaje Python	28 de junio 90 minutos	4 de julio
Parcial Práctico 2	Temas Estructuras de Datos I: Arreglos Registros y Programación estructurada y modular. Algoritmos fundamentales de Búsqueda, ordenación, inserción, eliminación y actualización de datos.	Especialmente diseñada, individual, escrita, resolución de un problema propuesto mediante diagrama de flujo . Verificar la solución propuesta con una prueba de escritorio. Codificación aplicando lenguaje Python	30 de septiembre (2 hs)	9 de octubre

Evaluación	Tema	Modalidad	Fecha	Devolución de resultados
Recuperatorio 1 y/o Recuperatorio 2	Temas Estructuras básicas de control Estructuras de Datos I: Arreglos Registros y Programación estructurada y modular Algoritmos fundamentales de Búsqueda, ordenación, inserción, eliminación y actualización de datos	Especialmente diseñada, individual, escrita, resolución de un problema propuesto mediante diagrama de flujo . Verificar la solución propuesta con una prueba de escritorio. Codificación aplicando lenguaje Python	16 de octubre (2 hs)	30 de octubre
Recuperatorio Parcial de Python 1	Temas Estructuras básicas de control Estructuras de Datos I: Arreglos	Individual Escrito en computadora. Resolución de un problema propuesto mediante: Diagrama de flujo Codificación aplicando lenguaje Python	25 de octubre	31 de octubre
Parcial Práctico 3	Temas Verificación Estructuras de datos II: Pilas y Colas	Especialmente diseñada, individual, escrita, diseño de casos de prueba aplicando técnicas de verificación: Caja blanca y caja negra.	7 de noviembre (2 hs)	14 de noviembre.
Recuperatorio 3	Temas Verificación Estructuras de datos: Pilas y Colas	Diseño de diagrama de flujo utilizando Pilas y Colas .	20 de noviembre (2 hs)	25 de noviembre

Tabla 8: Programa de Evaluaciones Parciales

Evaluación	Tema	Modalidad	Fecha	Devolución de resultados
Presentación del Trabajo de formación experimental	Resolver el problema de complejidad intermedia, siguiendo el enfoque modular. El proceso abarca la planificación, diseño del sistema, implementación del código, pruebas unitarias e integrales, y la documentación del proyecto.	Presentar y defender la solución planteada ante la clase, explicando las decisiones técnicas y cómo garantizo la eficiencia y escalabilidad del software.	27 de octubre	28 de octubre

Tabla 9: Evaluación del trabajo de formación experimental

8.3.2- Criterios de Evaluación

Resultados de aprendizaje	Criterios de Evaluación	Evaluación en la que se aplica
RA1: Construye algoritmos solución empleando estructuras de control (selección, iteración) y diagramación estructurada para la resolución de problemas (simples y de complejidad intermedia) de manera eficiente, optimizando recursos y garantizando la corrección y claridad en el proceso.	• Resolver correctamente el problema planteado y los resultados deben coincidir con los esperados en los casos de prueba. • Evidenciar un sólido entendimiento de los principios teóricos aplicados en la construcción del algoritmo	Parcial práctico 1 Recuperatorio 1

Resultados de aprendizaje	Criterios de Evaluación	Evaluación en la que se aplica
RA2: Aplica adecuadamente las estructuras de control y las estructuras de datos, junto con algoritmos fundamentales de recorrido, búsqueda (lineal o binaria), ordenamiento (selección o burbuja) y actualización (modificación de los datos), para resolver problemas de computación utilizando las técnicas de programación modular y estructurada.	1 El algoritmo debe estar dividido en módulos o funciones bien definidos, con nombres descriptivos para variables y módulos, garantizando legibilidad y claridad. 2 Uso Eficiente de las estructuras de control (selección e iteración) y estructuras de datos (arreglos, registros, pilas, y colas) para resolver problemas de manera eficiente. 3 Implementación correcta de los algoritmos fundamentales de recorrido, búsqueda (lineal o binaria), y ordenamiento (selección o burbuja), asegurando eficiencia y exactitud. 4. Explicación Teórica: El estudiante debe justificar las decisiones tomadas durante la implementación del algoritmo y explicar cómo los principios teóricos se aplican en su solución.	Parcial práctico 2 Recuperatorio 2
RA3: Diseña y verifica casos de prueba basados en el problema planteado, evaluando el algoritmo solución expresado en diagrama de flujo y codificado en Python para identificar y corregir errores, aplicando técnicas de pruebas de software como caja blanca y caja negra.	1 Aplicación correcta de las Técnicas de Prueba: Caja Blanca y Caja Negra para verificar la funcionalidad y estructura interna del algoritmo asegurando que los resultados sean correctos en cada caso 2 Coherencia entre el diagrama de flujo y el código Python, y mejorar la solución, optimizando el algoritmo cuando sea necesario.	Parcial práctico 3 Recuperatorio 3

Resultados de aprendizaje	Criterios de Evaluación	Evaluación en la que se aplica
RA4: Desarrolla y documenta un sistema codificado en Python para resolver un problema de una complejidad intermedia, utilizando técnicas de programación modular y estructurada, aplicando buenas prácticas de diseño y pruebas para garantizar la eficiencia, la escalabilidad y la correcta funcionalidad del software.	1. Funcionalidad*: El sistema debe resolver correctamente el problema y cumplir con los requisitos funcionales. 2. *Estructura y modularidad*: El código debe ser modular y seguir principios de programación estructurada. 3. *Eficiencia y escalabilidad*: El sistema debe estar optimizado para el rendimiento y ser escalable. 4. *Pruebas y validación*: El sistema debe incluir pruebas unitarias e integración que validen su funcionalidad. 5. *Documentación*: El sistema debe estar bien documentado, tanto interna como externamente. 6. *Innovación y creatividad*: Se valorará si el sistema implementa soluciones innovadoras o creativas para abordar el problema planteado, demostrando un pensamiento crítico y una comprensión profunda de las técnicas utilizadas.	Trabajo de formación experimental
RA5: Colabora eficazmente en equipos de trabajo, aportando de manera responsable conocimientos teóricos y prácticos para la revisión de conceptos y la resolución conjunta de problemas, favoreciendo la dinámica grupal y el logro de los objetivos planteados.	1. Conocimientos y habilidades: Aplica conocimientos teóricos y prácticos para resolver problemas y apoyar al equipo. 2. Comunicación y colaboración: Contribuye con ideas y soluciones, fomentando una dinámica positiva en el grupo. 3. Responsabilidad y compromiso: Cumple tareas a tiempo, gestiona su tiempo bien y apoya al equipo.	Trabajo de formación experimental

Tabla 10: Criterios de evaluación

8.3.3- Escala de Valoración

La escala de valoración a emplear en los parciales y recuperatorios será cuantitativa politómica (escala de 1 a 100). El puntaje mínimo para aprobar los parciales es de cincuenta (50) puntos, y para los parciales recuperatorios es de sesenta (60) puntos. El trabajo experimental será evaluado con Aprobado/Desaprobado

8.4- Evaluación Integradora

No se prevee.

8.5- Evaluación Sumativa

8.5.1- Condiciones para lograr la promoción sin Examen Final de la signatura. (Rige la Resolución HCD N° 135/00)

No se prevee.

8.5.2- Condiciones para lograr la Regularidad de la Asignatura.

Para regularizar la asignatura el alumno deberá

- Aprobar los cuatro parciales de la práctica con un puntaje mayor igual a 5o sus correspondientes recuperatorios integrales, según tabla 6 con un puntaje mayor igual a 6
- Completar la presentación de una selección de los ejercicios de Trabajos prácticos solicitados
- Aprobar el trabajo experimental propuesto. En caso de desaprobar se otorgará la posibilidad de una nueva evaluación.

8.6- Examen Final

En el examen final los alumnos serán evaluados sobre los contenidos teóricos previstos en elprograma de la asignatura. El examen será individual y podrá ser oral o escrito.

8.7- Examen Libre

Para el examen Libre el alumno deberá aprobar las dos (2) instancias que se detallan a continuación, siendo cada una de ellas eliminatorias:

Primera etapa

Aprobar una evaluación escrita de tipo práctica aplicando la diagramación modular y estructurada, utilizando las estructuras de datos I y II y codificando en la computadora la solución planteada en Lenguaje Python. Aplicar técnicas de verificación: Caja blanca y Caja negra.

Segunda etapa

Aprobar una evaluación oral/escrita de los contenidos teóricos del programa analítico de la asignatura.

.....

Ing. Carmen Silva
Profesor responsable de la Asignatura