# MODALIDAD ACADÉMICA

| | | |
|---|---|---|
| **Asignatura** | **Algoritmos y Estructuras de Datos** | |
| **Carrera** | **INGENIERÍA EN SISTEMAS DE INFORMACIÓN** | |
| **Ciclo Lectivo** | *2018* | |
| **Vigencia del programa** | *Desde el ciclo lectivo 2015* | |
| **Plan** | *2008* | |
| **Nivel** | ■ 1er. Nivel<br>☐ 2do. Nivel<br>☐ 3er. Nivel<br>☐ 4to. Nivel<br>☐ 5to. Nivel | |
| **Coordinador de la Cátedra** | Ing. Valerio Frittelli | |
| **Área de Conocimiento** | ■ Programación<br>☐ Computación<br>☐ Sistemas de Información<br>☐ Gestión Ingenieril<br>☐ Modelos<br>☐ Complementaria | |
| **Carga horaria semanal** | 5 horas | |
| **Anual/ cuatrimestral** | Anual | |
| **Contenidos Mínimos,** según Diseño Curricular-Ordenanza 1150 (sólo para asignaturas curriculares) | **Contenidos Mínimos:** La Ordenanza 1150 de Rectorado que reglamenta el Diseño Curricular de la Carrera, establece un cierto conjunto de *contenidos mínimos* para cada asignatura del Plan. Específicamente, para la asignatura AED esos contenidos mínimos son:<br>[1] Concepto de Dato.<br>[2] Tipos de Datos Simples.<br>[3] Estructuras de Control Básicas: Secuencial, Condicional, Cíclica.<br>[4] Tipos Abstractos de Datos.<br>[5] Abstracciones con Procedimientos y Funciones.<br>[6] Pasaje de Parámetros.<br>[7] Estructuras de Datos: Registros, Arreglos y Archivos.<br>[8] Estructuras de Datos Lineales (Pilas – Colas).<br>[9] Recursividad.<br>[10] Estrategias de Resolución *[de Problemas]*.<br>[11] Algoritmos de Búsqueda, Recorrido y Ordenamiento.<br>[12] Archivos de Acceso Secuencial y Aleatorio: Organizaciones y Accesos.<br>[13] Procesamiento Básico *[de Archivos]*.<br>[14] Noción de Orden de Complejidad.<br>[15] Noción de Complejidad Computacional. | |
| **Correlativas para cursar** (según Diseño Curricular-Ordenanza 1150) | Regulares | Aprobadas |
| | • Ninguna (salvo curso de ingreso) | • Ninguna (salvo curso de ingreso) |
| **Correlativas para rendir** (según Diseño Curricular-Ordenanza 1150) | Regulares | Aprobadas |
| | • Ninguna (salvo curso de ingreso) | • Ninguna (salvo curso de ingreso) |

| | |
|---|---|
| **Objetivos de la Asignatura** | **a.) Objetivos Generales para la Asignatura:** Tal cual los prescribe la *Ordenanza 1150* de Rectorado, los *objetivos generales* para AED son en realidad *expectativas de logro* para los alumnos (es decir, se plantean como metas que deberían ser capaces de alcanzar los propios alumnos al terminar el cursado). Literalmente, son los siguientes:<br>● Identificar problemas algorítmicos.<br>● Conocer el proceso de diseño e implementación de software.<br>● Aplicar las herramientas fundamentales representativas de los procesos, integrando la sintaxis elemental de un lenguaje de programación en el laboratorio asociado.<br>**b.) Objetivos Particulares (propuestos desde la Cátedra):** Desde la Cátedra se propone mantener el espíritu de los Objetivos Generales antes citados, pero replanteándolos de manera más funcional. Los tres objetivos generales originales, están cubiertos completamente por los objetivos particulares que siguen:<br>**i.** Favorecer que los alumnos desarrollen *capacidad de razonamiento lógico*, abordando problemas reales de diversos dominios de aplicación y analizando esos problemas de forma de identificar datos, procesos posibles y resultados a obtener, de forma que esa capacidad de razonamiento lógico se convierta en una base sólida para el avance en asignaturas similares de la carrera y el futuro desarrollo profesional.<br>**ii.** Favorecer que los alumnos adquieran la habilidad de diseñar soluciones lógicas para los problemas estudiados, diseñando *algoritmos* y sus *estructuras de datos asociadas*, de forma que esa habilidad consolide en cada alumno la capacidad de razonamiento, acreciente la confianza en sí mismo, y fomente su independencia de criterio.<br>**iii.** Proveer a los alumnos los elementos introductivos fundamentales para que desarrollen la *capacidad básica para analizar la eficiencia de un algoritmo*, ya sea en términos de tiempo de ejecución, consumo de memoria o complejidad de código fuente, de forma que esa capacidad de análisis les permita diferenciarse como programadores y desarrolladores eficientes, además de convertirse en referentes sólidos y confiables.<br>**iv.** Favorecer que los alumnos incorporen *conocimiento y dominio de un lenguaje de programación de propósito general*, y sean capaces de plantear programas completos para las soluciones algorítmicas propuestas, verificando su correcto funcionamiento, de forma que ese dominio facilite la puesta en práctica de los conceptos y técnicas aprendidas, consolide esos aprendizajes, y encamine a los alumnos hacia la aplicación profesional. |

**Programa Analítico**

**Propuesta de Programa Analítico:** Se propone el siguiente esquema de unidades para el nuevo programa analítico (vjgente desde el ciclo 2015). Los números que figuran entre corchetes hacen referencia a los contenidos mínimos previstos por la Ordenanza 1150, que están siendo efectivamente cubiertos por el ítem nombrado y

resaltado en *cursiva*:

**Unidad Nro. 1: Fundamentos de programación – Estructuras secuenciales.**

**Objetivos Específicos:**

- Favorecer que los alumnos desarrollen *capacidad de razonamiento lógico*, abordando problemas reales de diversos dominios de aplicación y analizando esos problemas de forma de identificar datos, procesos posibles y resultados a obtener, de forma que esa capacidad de razonamiento lógico se convierta en una base sólida para el avance en asignaturas similares de la carrera y el futuro desarrollo profesional.
- Favorecer que los alumnos incorporen *conocimiento y dominio de un lenguaje de programación de propósito general*, y sean capaces de plantear programas completos para las soluciones algorítmicas propuestas, verificando su correcto funcionamiento, de forma que ese dominio facilite la puesta en práctica de los conceptos y técnicas aprendidas, consolide esos aprendizajes, y encamine a los alumnos hacia la aplicación profesional.

**Contenidos:**

Concepto de algoritmo, computadora (o máquina algorítmica) y programa. Memoria, celdas de memoria y representación de información en base al sistema binario. Lenguajes de programación. Descarga, instalación y configuración del kit de herramientas (SDK) de Python y de un entorno de desarrollo (o IDE). *Concepto de dato* [1] y tipo de datos. *Tipos de datos simples* [2] en Python. Variables. Instrucciones básicas: asignación, carga por teclado y visualización por consola estándar. Operadores aritméticos. Expresiones aritméticas (o numéricas). Tipos estructurados elementales en Python: secuencias de datos y cadenas de caracteres. *Estructuras de control de flujo básicas: la estructura secuencial de instrucciones* [3]. Concepto de script en Python. Estructura de un script, edición, depuración y ejecución a través de la consola de comandos de Python y a través de un IDE. Reglas y convenciones de escritura de código fuente del Lenguaje Python. Uso de comentarios en un script Python. Pasos en la resolución de problemas y estructura de un algoritmo. Técnicas de representación de algoritmos: pseudocódigo y diagrama de flujo. Definición de problema simple. Planteo y resolución de problemas simples en base a *secuencias de instrucciones simples* [3]. Problemas típicos: aplicaciones matemáticas (conversión de unidades de tiempo, cálculo de promedios y porcentajes directos, aplicaciones del operador resto de una división) o de gestión de cadenas de caracteres (a partir de invocación a métodos nativos).

**Bibliografía Obligatoria:**

[1] Frittelli, V., et al. (2016). Publicaciones Cátedra de AED. *Fichas de Estudio para AED (disponible en Aula Virtual AED)*. Cordoba: UTN – Facultad Regional Córdoba.

[2] Python Software Foundation. (2015). *Python Documentation*. Retrieved February 24, 2016, from https://docs.python.org/3/.

**Bibliografía Complementaria:**

[1] Pilgrim, M. (2004). *Dive Into Python - Python from novice to pro*. Retrieved March 6, 2016, from http://www.diveintopython.net/toc/index.html.

**Evaluación:**

Contenidos y prácticas evaluados a través de los Cuestionarios Teóricos 1, 2 y 3 y el Desafío de Programación 1 disponibles en el Aula Virtual.

**Unidad Nro. 2: Estructuras condicionales - Subproblemas.**

**Objetivos Específicos:**

- Favorecer que los alumnos desarrollen *capacidad de razonamiento lógico*, abordando problemas reales de diversos dominios de aplicación y analizando esos problemas de forma de identificar datos, procesos posibles y resultados a obtener, de forma que esa capacidad de razonamiento lógico se convierta en una base sólida para el avance en asignaturas similares de la carrera y el futuro desarrollo profesional.

- Favorecer que los alumnos adquieran la habilidad de diseñar soluciones lógicas para los problemas estudiados, diseñando *algoritmos* y sus *estructuras de datos asociadas*, de forma que esa habilidad consolide en cada alumno la capacidad de razonamiento, acreciente la confianza en sí mismo, y fomente su independencia de criterio.
- Favorecer que los alumnos incorporen *conocimiento y dominio de un lenguaje de programación de propósito general*, y sean capaces de plantear programas completos para las soluciones algorítmicas propuestas, verificando su correcto funcionamiento, de forma que ese dominio facilite la puesta en práctica de los conceptos y técnicas aprendidas, consolide esos aprendizajes, y encamine a los alumnos hacia la aplicación profesional.

**Contenidos:**

*Estructuras de control de flujo básicas: la instrucción condicional* [3]. Instrucción condicional simple, doble y múltiple en Python. Operadores relacionales y lógicos. Expresiones y proposiciones lógicas (o booleanas). Tablas de verdad. Condiciones anidadas. Planteo de problemas simples en base a *instrucciones condicionales* [3]. Concepto de Subproblema. Problemas compuestos (divisibles en subproblemas). Descomposición de un problema compuesto en subproblemas simples. Planteo y resolución de problemas compuestos en base a *instrucciones condicionales* [3]. Problemas típicos: determinación del menor o el mayor de una secuencia fija de números de entrada (dos, tres o cuatro números), filtrado de valores de entrada para evitar operaciones matemáticas no definidas, cálculo de raíces de una ecuación lineal o cuadrática, cálculo del valor de un polinomio.

**Bibliografía Obligatoria:**

[1] Frittelli, V., et al. (2016). Publicaciones Cátedra de AED. *Fichas de Estudio para AED (disponible en Aula Virtual AED)*. Cordoba: UTN – Facultad Regional Córdoba.

[2] Python Software Foundation. (2015). *Python Documentation*. Retrieved February 24, 2016, from https://docs.python.org/3/.

**Bibliografía Complementaria:**

[1] Pilgrim, M. (2004). *Dive Into Python - Python from novice to pro*. Retrieved March 6, 2016, from http://www.diveintopython.net/toc/index.html.

**Evaluación:**

Contenidos y prácticas evaluados a través de los Cuestionarios Teóricos 3, 4 y 5 y el Trabajo Práctico 1 disponibles en el Aula Virtual.

**Unidad Nro. 3: Estructuras repetitivas.**

**Objetivos Específicos:**

- Favorecer que los alumnos desarrollen *capacidad de razonamiento lógico*, abordando problemas reales de diversos dominios de aplicación y analizando esos problemas de forma de identificar datos, procesos posibles y resultados a obtener, de forma que esa capacidad de razonamiento lógico se convierta en una base sólida para el avance en asignaturas similares de la carrera y el futuro desarrollo profesional.
- Favorecer que los alumnos adquieran la habilidad de diseñar soluciones lógicas para los problemas estudiados, diseñando *algoritmos* y sus *estructuras de datos asociadas*, de forma que esa habilidad consolide en cada alumno la capacidad de razonamiento, acreciente la confianza en sí mismo, y fomente su independencia de criterio.
- Favorecer que los alumnos incorporen *conocimiento y dominio de un lenguaje de programación de propósito general*, y sean capaces de plantear programas completos para las soluciones algorítmicas propuestas, verificando su correcto funcionamiento, de forma que ese dominio facilite la puesta en práctica de los conceptos y técnicas aprendidas, consolide esos aprendizajes, y encamine a los alumnos hacia la aplicación profesional.

**Contenidos:**
Concepto de proceso repetitivo. *Estructuras de control de flujo básicas: la estructura cíclica o repetitiva* [3]. Tipos generales de ciclos: el ciclo 0-N y el ciclo 1-N. Los ciclos while y for en Python. Ciclos infinitos. Interrupción de un ciclo. Ciclos anidados. Variables de conteo y de acumulación. Variables centinela (o banderas). Uso de banderas en la condición de un ciclo. Noción de conteo de operaciones críticas y comparación de tiempos esperados de ejecución. Planteo y resolución de problemas de naturaleza repetitiva básica usando *instrucciones cíclicas* [3]. Problemas típicos: carga y procesamiento de sucesiones de datos desde la consola estándar, conociendo la cantidad de datos y/o mediante un proceso de carga por doble lectura, determinación de promedios y porcentajes, determinación del mayor y/o menor de una sucesión de valores, problemas generales de búsqueda de un valor único o de múltiples valores, conteo y acumulación de cantidades, generación y procesamiento de sucesiones de números a partir de un valor inicial dado, generación de números aleatorios y su aplicación en juegos y simulaciones simples, procesamiento básico de caracteres y cadenas (aplicación de métodos nativos para cadenas de caracteres de Python).

**Bibliografía Obligatoria:**

[1] Frittelli, V., et al. (2016). Publicaciones Cátedra de AED. *Fichas de Estudio para AED (disponible en Aula Virtual AED)*. Cordoba: UTN – Facultad Regional Córdoba.

[2] Python Software Foundation. (2015). *Python Documentation*. Retrieved February 24, 2016, from https://docs.python.org/3/.

**Bibliografía Complementaria:**

[1] Pilgrim, M. (2004). *Dive Into Python - Python from novice to pro*. Retrieved March 6, 2016, from http://www.diveintopython.net/toc/index.html.

**Evaluación:**
Contenidos y prácticas evaluados a través de los Cuestionarios Teóricos 7, 8 y 9 y el Trabajo Práctico 2 disponibles en el Aula Virtual.

**Unidad Nro. 4: Funciones – Programación estructurada.**

**Objetivos Específicos:**

- Favorecer que los alumnos desarrollen *capacidad de razonamiento lógico*, abordando problemas reales de diversos dominios de aplicación y analizando esos problemas de forma de identificar datos, procesos posibles y resultados a obtener, de forma que esa capacidad de razonamiento lógico se convierta en una base sólida para el avance en asignaturas similares de la carrera y el futuro desarrollo profesional.
- Favorecer que los alumnos adquieran la habilidad de diseñar soluciones lógicas para los problemas estudiados, diseñando *algoritmos* y sus *estructuras de datos asociadas*, de forma que esa habilidad consolide en cada alumno la capacidad de razonamiento, acreciente la confianza en sí mismo, y fomente su independencia de criterio.
- Proveer a los alumnos los elementos introductivos fundamentales para que desarrollen la *capacidad básica para analizar la eficiencia de un algoritmo*, ya sea en términos de tiempo de ejecución, consumo de memoria o complejidad de código fuente, de forma que esa capacidad de análisis les permita diferenciarse como programadores y desarrolladores eficientes, además de convertirse en referentes sólidos y confiables.
- Favorecer que los alumnos incorporen *conocimiento y dominio de un lenguaje de programación de propósito general*, y sean capaces de plantear programas completos para las soluciones algorítmicas propuestas, verificando su correcto funcionamiento, de forma que ese dominio facilite la puesta en práctica de los conceptos y técnicas aprendidas, consolide esos aprendizajes, y encamine a los alumnos hacia la aplicación profesional.

**Contenidos:**

Concepto elemental de subrutina para plantear un subproblema: introducción al uso de funciones. Funciones en Python: cabecera y estructura general. Funciones con retorno de valor y funciones sin retorno de valor. *Pasaje de parámetros a una función* [6]. Parámetros actuales y parámetros formales. Formas especiales de parametrización en Python. Concepto general/teórico de parametrización por valor y por referencia y el modelo en Python: parametrizacción por valor. Ámbito de una variable: variables locales y variables globales a un bloque de acciones. Concepto de Programación Estructurada: planteo de un programa en forma modular usando funciones. Generalización de código fuente: planteo de funciones parametrizadas de uso general. Reutilización de código. Módulos y paquetes de módulos en Python. Módulos de la librería estándar más comunes en Python. Importación de módulos. Creación de módulos y paquetes del programador. *Recursividad* [9]. Funciones recursivas bien planteadas. Determinación de la condición de corte de la recursividad. Seguimiento de la recursividad. Recursividad mutua o indirecta. Análisis básico de la eficiencia y la aplicabilidad de la recursividad en situaciones prácticas comunes.

**Bibliografía Obligatoria:**

[1] Frittelli, V., et al. (2016). Publicaciones Cátedra de AED. *Fichas de Estudio para AED (disponible en Aula Virtual AED)*. Cordoba: UTN – Facultad Regional Córdoba.

[2] Python Software Foundation. (2015). *Python Documentation*. Retrieved February 24, 2016, from https://docs.python.org/3/.

**Bibliografía Complementaria:**

[1] Pilgrim, M. (2004). *Dive Into Python - Python from novice to pro*. Retrieved March 6, 2016, from http://www.diveintopython.net/toc/index.html.

**Evaluación:**

Contenidos y prácticas evaluados a través de los Cuestionarios Teóricos 10, 11, 12 y 13 y el Desafío de Programación 2 disponibles en el Aula Virtual.

**Unidad Nro. 5: Arreglos.**

**Objetivos Específicos:**

- Favorecer que los alumnos desarrollen *capacidad de razonamiento lógico*, abordando problemas reales de diversos dominios de aplicación y analizando esos problemas de forma de identificar datos, procesos posibles y resultados a obtener, de forma que esa capacidad de razonamiento lógico se convierta en una base sólida para el avance en asignaturas similares de la carrera y el futuro desarrollo profesional.
- Favorecer que los alumnos adquieran la habilidad de diseñar soluciones lógicas para los problemas estudiados, diseñando *algoritmos* y sus *estructuras de datos asociadas*, de forma que esa habilidad consolide en cada alumno la capacidad de razonamiento, acreciente la confianza en sí mismo, y fomente su independencia de criterio.
- Proveer a los alumnos los elementos introductivos fundamentales para que desarrollen la *capacidad básica para analizar la eficiencia de un algoritmo*, ya sea en términos de tiempo de ejecución, consumo de memoria o complejidad de código fuente, de forma que esa capacidad de análisis les permita diferenciarse como programadores y desarrolladores eficientes, además de convertirse en referentes sólidos y confiables.
- Favorecer que los alumnos incorporen *conocimiento y dominio de un lenguaje de programación de propósito general*, y sean capaces de plantear programas completos para las soluciones algorítmicas propuestas, verificando su correcto funcionamiento, de forma que ese dominio facilite la puesta en práctica de los conceptos y técnicas aprendidas, consolide esos aprendizajes, y encamine a los alumnos hacia la aplicación profesional.

**Contenidos:**

Concepto de tipo compuesto o estructurado (estructuras de datos vs. variables de tipo simple) *Estructuras de datos* [7]: definición general y necesidades de uso. *Arreglos* [7] (variables subindicadas): concepto de acceso directo y acceso secuencial. Dimensión de un arreglo. Índices. Implementación del concepto de arreglo de tamaño dinámico mediante listas nativas en Python. Arreglos unidimensionales (o *"vectores"*): creación y asignación de valores. *Recorrido secuencial de un vector* [11]. Carga y visualización. *Ordenamiento: clasificación tradicional en métodos directos (o simples) y compuestos (o mejorados). Algoritmos directos (Bubble sort, Selection sort e Insertion sort)* [11]. Algoritmos de *Búsqueda Secuencial y Búsqueda Binaria* [11]. Fusión (o mezcla) de vectores ordenados. Elementos básicos de análisis de eficiencia mediante conteo de comparaciones. Vectores de conteo y vectores de acumulación. Vectores paralelos. Arreglos bidimensionales (o *"matrices"*) y multidimensionales. Conceptos y aplicaciones. Creación y asignación de valores. *Recorrido de matrices en orden de filas y en orden de columnas* [11]. Sumarización por filas y columnas.

**Bibliografía Obligatoria:**

[1] Frittelli, V., et al. (2016). Publicaciones Cátedra de AED. *Fichas de Estudio para AED (disponible en Aula Virtual AED)*. Cordoba: UTN – Facultad Regional Córdoba.

[2] Python Software Foundation. (2015). *Python Documentation*. Retrieved February 24, 2016, from https://docs.python.org/3/.

**Bibliografía Complementaria:**

[1] Pilgrim, M. (2004). *Dive Into Python - Python from novice to pro*. Retrieved March 6, 2016, from http://www.diveintopython.net/toc/index.html.

**Evaluación:**

Contenidos y prácticas evaluados a través de los Cuestionarios Teóricos 15, 16, 17 y 18 más el Desafío de Programación 3 y el Trabajo Práctico 3 disponibles en el Aula Virtual.

**Unidad Nro. 6: Registros.**

**Objetivos Específicos:**

- Favorecer que los alumnos desarrollen *capacidad de razonamiento lógico*, abordando problemas reales de diversos dominios de aplicación y analizando esos problemas de forma de identificar datos, procesos posibles y resultados a obtener, de forma que esa capacidad de razonamiento lógico se convierta en una base sólida para el avance en asignaturas similares de la carrera y el futuro desarrollo profesional.
- Favorecer que los alumnos adquieran la habilidad de diseñar soluciones lógicas para los problemas estudiados, diseñando *algoritmos* y sus *estructuras de datos asociadas*, de forma que esa habilidad consolide en cada alumno la capacidad de razonamiento, acreciente la confianza en sí mismo, y fomente su independencia de criterio.
- Proveer a los alumnos los elementos introductivos fundamentales para que desarrollen la *capacidad básica para analizar la eficiencia de un algoritmo*, ya sea en términos de tiempo de ejecución, consumo de memoria o complejidad de código fuente, de forma que esa capacidad de análisis les permita diferenciarse como programadores y desarrolladores eficientes, además de convertirse en referentes sólidos y confiables.
- Favorecer que los alumnos incorporen *conocimiento y dominio de un lenguaje de programación de propósito general*, y sean capaces de plantear programas completos para las soluciones algorítmicas propuestas, verificando su correcto funcionamiento, de forma que ese dominio facilite la puesta en práctica de los conceptos y técnicas aprendidas, consolide esos aprendizajes, y encamine a los alumnos hacia la aplicación profesional.

**Contenidos:**

*Registros* [7]: agrupamientos de variables de tipos diferentes. Definición y forma de creación de registros en Python. Campos de un registro. Acceso a campos. Concepto de abstracción de datos y empleo de registros para modelar esa abstracción. Tipos nativos de datos y *tipos abstractos de datos* [4]. Concepto de *abstracción funcional y empleo de funciones en Python para modelar esa abstracción* [5]. Uso combinado de arreglos y registros: vectores y matrices de registros. Problemas típicos: implementación de tipos abstractos que representen números complejos, puntos en un plano, segmentos de recta, fechas, entidades del dominio para sistemas de gestión administrativa (como clientes, cuentas, artículos, libros y otras entidades).

**Bibliografía Obligatoria:**

[1] Frittelli, V., et al. (2016). Publicaciones Cátedra de AED. *Fichas de Estudio para AED (disponible en Aula Virtual AED)*. Cordoba: UTN – Facultad Regional Córdoba.

[2] Python Software Foundation. (2015). *Python Documentation*. Retrieved February 24, 2016, from https://docs.python.org/3/.

**Bibliografía Complementaria:**

[1] Pilgrim, M. (2004). *Dive Into Python - Python from novice to pro*. Retrieved March 6, 2016, from http://www.diveintopython.net/toc/index.html.

**Evaluación:**

Contenidos y prácticas evaluados a través del Cuestionario Teórico 14 más el Desafío de Programación 3 y el Trabajo Práctico 3 disponibles en el Aula Virtual.

**Unidad Nro. 7: Estructuras lineales.**

**Objetivos Específicos:**

- Favorecer que los alumnos desarrollen *capacidad de razonamiento lógico*, abordando problemas reales de diversos dominios de aplicación y analizando esos problemas de forma de identificar datos, procesos posibles y resultados a obtener, de forma que esa capacidad de razonamiento lógico se convierta en una base sólida para el avance en asignaturas similares de la carrera y el futuro desarrollo profesional.
- Favorecer que los alumnos adquieran la habilidad de diseñar soluciones lógicas para los problemas estudiados, diseñando *algoritmos* y sus *estructuras de datos asociadas*, de forma que esa habilidad consolide en cada alumno la capacidad de razonamiento, acreciente la confianza en sí mismo, y fomente su independencia de criterio.
- Proveer a los alumnos los elementos introductivos fundamentales para que desarrollen la *capacidad básica para analizar la eficiencia de un algoritmo*, ya sea en términos de tiempo de ejecución, consumo de memoria o complejidad de código fuente, de forma que esa capacidad de análisis les permita diferenciarse como programadores y desarrolladores eficientes, además de convertirse en referentes sólidos y confiables.
- Favorecer que los alumnos incorporen *conocimiento y dominio de un lenguaje de programación de propósito general*, y sean capaces de plantear programas completos para las soluciones algorítmicas propuestas, verificando su correcto funcionamiento, de forma que ese dominio facilite la puesta en práctica de los conceptos y técnicas aprendidas, consolide esos aprendizajes, y encamine a los alumnos hacia la aplicación profesional.

**Contenidos:**

*Estructuras de datos nativas y estructuras de datos abstractas* [4]. *Estructuras de datos lineales* [8] y no lineales. *Estructuras lineales básicas: concepto general de pilas, cola* [8] y lista. Análisis de la implementación general de listas nativas ya provista por Python, en base a un arreglo de crecimiento dinámico como soporte. *Implementación de pilas y colas usando registros, arreglos/listas y funciones de Python como soporte para*

*modelar la abstracción de datos y la abstracción funcional* [5]. Aplicación de Pilas en problemas de control de simetría, control de retornos, e inversión de secuencias. Aplicación de colas en problemas de espera para acceso a puestos de servicio. Descripción de las estructuras predefinidas de Python para manejo de estructuras o colecciones abstractas (listas usadas como pilas o colas, uso general de listas nativas).

**Bibliografía Obligatoria:**

[1] Frittelli, V., et al. (2016). Publicaciones Cátedra de AED. *Fichas de Estudio para AED (disponible en Aula Virtual AED)*. Cordoba: UTN – Facultad Regional Córdoba.

[2] Python Software Foundation. (2015). *Python Documentation*. Retrieved February 24, 2016, from https://docs.python.org/3/.

**Bibliografía Complementaria:**

[1] Pilgrim, M. (2004). *Dive Into Python - Python from novice to pro*. Retrieved March 6, 2016, from http://www.diveintopython.net/toc/index.html.

**Evaluación:**

Contenidos y prácticas evaluados a través de los Cuestionarios Teóricos 19 y 20 disponibles en el Aula Virtual.

**Unidad Nro. 8: Elementos de análisis de algoritmos.**

**Objetivos Específicos:**

- Favorecer que los alumnos adquieran la habilidad de diseñar soluciones lógicas para los problemas estudiados, diseñando *algoritmos* y sus *estructuras de datos asociadas*, de forma que esa habilidad consolide en cada alumno la capacidad de razonamiento, acreciente la confianza en sí mismo, y fomente su independencia de criterio.
- Proveer a los alumnos los elementos introductivos fundamentales para que desarrollen la *capacidad básica para analizar la eficiencia de un algoritmo*, ya sea en términos de tiempo de ejecución, consumo de memoria o complejidad de código fuente, de forma que esa capacidad de análisis les permita diferenciarse como programadores y desarrolladores eficientes, además de convertirse en referentes sólidos y confiables.

**Contenidos:**

Motivaciones para el análisis formal de algoritmos. Elementos de análisis de eficiencia de un algoritmo: tiempo de ejecución, espacio de memoria empleado y complejidad del código fuente. Concepto de comportamiento asintótico. Formalización del *concepto de orden de complejidad* [14]. Notación *O mayúscula* (o *Big O*) para expresar el orden de un algoritmo. Clasificación básica de las principales funciones asociadas al concepto de orden de complejidad. Propiedades de la relación de orden y aplicaciones prácticas. Aplicaciones: *Análisis de tiempo de ejecución de los algoritmos de ordenamiento simples* [11][14]. *El ordenamiento de Shell como algoritmo mejorado de la inserción simple* [11][14]. Comparación del comportamiento asintótico de los tiempos de ejecución. *Análisis comparativo entre la búsqueda secuencial y la búsqueda binaria* [11][14].

**Bibliografía Obligatoria:**

[1] Frittelli, V., et al. (2016). Publicaciones Cátedra de AED. *Fichas de Estudio para AED (disponible en Aula Virtual AED)*. Cordoba: UTN – Facultad Regional Córdoba.

[2] Python Software Foundation. (2015). *Python Documentation*. Retrieved February 24, 2016, from https://docs.python.org/3/.

**Bibliografía Complementaria:**

[1] Pilgrim, M. (2004). *Dive Into Python - Python from novice to pro*. Retrieved March 6, 2016, from http://www.diveintopython.net/toc/index.html.

**Evaluación:**
Contenidos y prácticas evaluados a través de los Cuestionarios Teóricos 20 y siguientes, los Desafíos de Programación 3 y 4 y el Trabajo Práctico 4 disponibles en el Aula Virtual.

**Unidad Nro. 9: Archivos.**

**Objetivos Específicos:**

- Favorecer que los alumnos desarrollen *capacidad de razonamiento lógico*, abordando problemas reales de diversos dominios de aplicación y analizando esos problemas de forma de identificar datos, procesos posibles y resultados a obtener, de forma que esa capacidad de razonamiento lógico se convierta en una base sólida para el avance en asignaturas similares de la carrera y el futuro desarrollo profesional.
- Favorecer que los alumnos adquieran la habilidad de diseñar soluciones lógicas para los problemas estudiados, diseñando *algoritmos* y sus *estructuras de datos asociadas*, de forma que esa habilidad consolide en cada alumno la capacidad de razonamiento, acreciente la confianza en sí mismo, y fomente su independencia de criterio.
- Proveer a los alumnos los elementos introductivos fundamentales para que desarrollen la *capacidad básica para analizar la eficiencia de un algoritmo*, ya sea en términos de tiempo de ejecución, consumo de memoria o complejidad de código fuente, de forma que esa capacidad de análisis les permita diferenciarse como programadores y desarrolladores eficientes, además de convertirse en referentes sólidos y confiables.
- Favorecer que los alumnos incorporen *conocimiento y dominio de un lenguaje de programación de propósito general*, y sean capaces de plantear programas completos para las soluciones algorítmicas propuestas, verificando su correcto funcionamiento, de forma que ese dominio facilite la puesta en práctica de los conceptos y técnicas aprendidas, consolide esos aprendizajes, y encamine a los alumnos hacia la aplicación profesional.

**Contenidos:**
El problema de la volatilidad de los datos en memoria principal. Concepto y necesidad de medios de almacenamiento externo. *Archivos* [7]. Archivos de texto y binarios. Archivos de registros. Apertura. Modos de apertura. Lectura. Escritura. Verificación de final de archivo. Reposicionamiento. *Formas de organización y accesos: archivos de acceso secuencial y archivos de acceso aleatorio* [12]. Elementos de *procesamiento básico de archivos de registros (y archivos binarios en general)* [13]: altas, bajas (lógicas y físicas) y modificaciones (procesamiento *ABM*), listados de contenido (completos o filtrados), búsqueda secuencial, consultas. *Procesamiento básico de archivos de texto* [13]: archivos de texto como soporte de lotes de datos secuenciales para su uso en problemas de procesamiento masivo de datos. Creación, carga, recorrido, conversión de contenidos (de tipo texto a tipo numérico) y eventual creación de arreglos a partir de esos contenidos.

**Bibliografía Obligatoria:**

[1] Frittelli, V., et al. (2016). Publicaciones Cátedra de AED. *Fichas de Estudio para AED (disponible en Aula Virtual AED)*. Cordoba: UTN – Facultad Regional Córdoba.

[2] Python Software Foundation. (2015). *Python Documentation*. Retrieved February 24, 2016, from https://docs.python.org/3/.

**Bibliografía Complementaria:**

[1] Pilgrim, M. (2004). *Dive Into Python - Python from novice to pro*. Retrieved March 6, 2016, from http://www.diveintopython.net/toc/index.html.

**Evaluación:**
Contenidos y prácticas evaluados a través de los Cuestionarios Teóricos 21 al 24, el Desafíos de Programación 4 y el Trabajo Práctico 4 disponibles en el Aula Virtual.

**Unidad Nro. 10: Estrategias de resolución de problemas.**

**Objetivos Específicos:**

- Favorecer que los alumnos desarrollen *capacidad de razonamiento lógico*, abordando problemas reales de diversos dominios de aplicación y analizando esos problemas de forma de identificar datos, procesos posibles y resultados a obtener, de forma que esa capacidad de razonamiento lógico se convierta en una base sólida para el avance en asignaturas similares de la carrera y el futuro desarrollo profesional.
- Favorecer que los alumnos adquieran la habilidad de diseñar soluciones lógicas para los problemas estudiados, diseñando *algoritmos* y sus *estructuras de datos asociadas*, de forma que esa habilidad consolide en cada alumno la capacidad de razonamiento, acreciente la confianza en sí mismo, y fomente su independencia de criterio.
- Proveer a los alumnos los elementos introductivos fundamentales para que desarrollen la *capacidad básica para analizar la eficiencia de un algoritmo*, ya sea en términos de tiempo de ejecución, consumo de memoria o complejidad de código fuente, de forma que esa capacidad de análisis les permita diferenciarse como programadores y desarrolladores eficientes, además de convertirse en referentes sólidos y confiables.
- Favorecer que los alumnos incorporen *conocimiento y dominio de un lenguaje de programación de propósito general*, y sean capaces de plantear programas completos para las soluciones algorítmicas propuestas, verificando su correcto funcionamiento, de forma que ese dominio facilite la puesta en práctica de los conceptos y técnicas aprendidas, consolide esos aprendizajes, y encamine a los alumnos hacia la aplicación profesional.

**Contenidos:**

Motivación y antecedentes para el estudio de *estrategias de resolución de problemas (o de planteo de algoritmos)* [10]. Clasificación y concepto básico de las estrategias existentes más comunes: fuerza bruta, recursividad, backtraking, divide y vencerás, randomización, algoritmos ávidos y programación dinámica. Revisión de planteos algorítmicos conocidos basados en Fuerza Bruta. *Recursividad* [9] y *Backtraking*: planteo de algoritmos basados en aprovechar el proceso de vuelta atrás de la recursión para analizar soluciones parciales a un problema. Casos típicos: problema de las Ocho Reinas, problema de las Torres de Hanoi. *Recursividad* [9] y *Divide y Vencerás*: estructura general de la estrategia. Aplicaciones: *los algoritmos de ordenamiento Merge sort y Quick sort* [11]. *Análisis general de eficiencia en tiempo de ejecución, ocupación de memoria y complejidad de código fuente* [14].

**Bibliografía Obligatoria:**

[1] Frittelli, V., et al. (2016). Publicaciones Cátedra de AED. *Fichas de Estudio para AED (disponible en Aula Virtual AED)*. Cordoba: UTN – Facultad Regional Córdoba.

[2] Python Software Foundation. (2015). *Python Documentation*. Retrieved February 24, 2016, from https://docs.python.org/3/.

**Bibliografía Complementaria:**

[1] Pilgrim, M. (2004). *Dive Into Python - Python from novice to pro*. Retrieved March 6, 2016, from http://www.diveintopython.net/toc/index.html.

**Evaluación:**

Contenidos y prácticas evaluados a través de los Cuestionarios Teóricos 25 y 26 y el Trabajo Práctico 4 disponibles en el Aula Virtual.

**Unidad Nro. 11: Introducción a la Teoría de la Complejidad Computacional.**

**Objetivos Específicos:**

- Favorecer que los alumnos desarrollen *capacidad de razonamiento lógico*, abordando problemas reales de diversos dominios de aplicación y analizando esos problemas de forma de identificar datos, procesos posibles y resultados a obtener, de forma que esa capacidad de razonamiento lógico se convierta en una base sólida para el avance en asignaturas similares de la carrera y el futuro desarrollo profesional.
- Favorecer que los alumnos adquieran la habilidad de diseñar soluciones lógicas para los problemas estudiados, diseñando *algoritmos* y sus *estructuras de datos asociadas*, de forma que esa habilidad consolide en cada alumno la capacidad de razonamiento, acreciente la confianza en sí mismo, y fomente su independencia de criterio.
- Proveer a los alumnos los elementos introductivos fundamentales para que desarrollen la *capacidad básica para analizar la eficiencia de un algoritmo*, ya sea en términos de tiempo de ejecución, consumo de memoria o complejidad de código fuente, de forma que esa capacidad de análisis les permita diferenciarse como programadores y desarrolladores eficientes, además de convertirse en referentes sólidos y confiables.
- Favorecer que los alumnos incorporen *conocimiento y dominio de un lenguaje de programación de propósito general*, y sean capaces de plantear programas completos para las soluciones algorítmicas propuestas, verificando su correcto funcionamiento, de forma que ese dominio facilite la puesta en práctica de los conceptos y técnicas aprendidas, consolide esos aprendizajes, y encamine a los alumnos hacia la aplicación profesional.

**Contenidos:**

Motivación y antecedentes para el estudio de la *Teoría de la Complejidad Computacional* [15]. Algoritmos de tiempo de ejecución de orden polinomial. Algoritmos intratables (tiempo de ejecución de orden exponencial o factorial). Reducción de un problema a otro. Reducción polinómica. Nociones de máquina algorítmica determinista y de máquina algorítmica no determinista. Problemas de decisión y problemas de optimización. Clases de Complejidad. Las clases P, NP y NP Completo. Propiedades de un problema NP Completo. Aproximación conceptual al Teorema de Cook-Levin. Los problemas del milenio y la cuestión P vs. NP en la actualidad.

**Bibliografía Obligatoria:**

[1] Frittelli, V., et al. (2016). Publicaciones Cátedra de AED. *Fichas de Estudio para AED (disponible en Aula Virtual AED)*. Cordoba: UTN – Facultad Regional Córdoba.

[2] Python Software Foundation. (2015). *Python Documentation*. Retrieved February 24, 2016, from https://docs.python.org/3/.

**Bibliografía Complementaria:**

[1] Pilgrim, M. (2004). *Dive Into Python - Python from novice to pro*. Retrieved March 6, 2016, from http://www.diveintopython.net/toc/index.html.

**Evaluación:**

Contenidos y prácticas evaluados obligatoriamente en el Examen Final de la asignatura.

| | |
|---|---|
| **Metodología de enseñanza y aprendizaje** | ✓ En los primeros dos meses las clases teóricas se llevarán a cabo en las aulas comunes, y las prácticas en los laboratorios.<br>✓ Luego de esos dos primeros meses, todas las clases se desarrollarán en los laboratorios de informática.<br>✓ En cada clase se desarrolla un tema central, y el mismo se ejemplifica y analiza con modelos de programas presentados por los profesores. |

| | |
|---|---|
| | ✓ Los alumnos realizan modificaciones y variantes sobre los modelos, y realizan ejercicios nuevos en base a los temas tratados.<br>✓ Los trabajos prácticos y las tareas semanales (cuestionarios y desafíos) integran los conocimientos de las distintas unidades, y parte del tiempo de clase se usa para analizar dudas y elementos relevantes referidos a esos trabajos. |
| **Sistema de evaluación** | Se usará un esquema de regularidad ***común a todos los cursos***, con las siguientes evaluaciones que serán registradas en la libreta y se usarán para determinar la *condición final del alumno*:<br>✓ Cuatro *Parciales de Regularidad* (*P1*, *P2*, *P3 y P4*) a desarrollar en el laboratorio. Los dos primeros (P1 y P2) se evalúan en el primer semestre, y los dos últimos (P3 y P4) en el segundo semestre.<br>✓ Cuatro *Trabajos Prácticos* (*TP1*, *TP2*, *TP3* y *TP4*) a desarrollar en grupos.<br>✓ Un conjunto de veintinueve *Cuestionarios Teóricos* tipo preguntas de opciones múltiples, a desarrollar en el aula virtual,<br>✓ Cuatro *Desafíos de Programación* consistentes en resolver algún problema y sólo informar los resultados.<br>✓ Los cuatro *Trabajos Prácticos* se promediarán con los veintinueve *Cuestionarios* y con los cuatro *Desafíos*, para formar una única *Nota Final de Trabajos Prácticos* (abreviado como *NFTP*) que irá a la libreta junto con las notas de los tres Parciales.<br>✓ El promedio *NFTP* se calculará en forma ponderada: las notas que se obtengan en cada actividad tendrán pesos diferentes en el cálculo final. Cada uno de los cuatro Trabajos Prácticos tendrá un peso del 17.5% del total (con lo cual, entre los cuatro suman el 70% del total en el cálculo de la *NFTP*). El promedio entre los veintinueve Cuestionarios tendrá un peso del 10% sobre el total de la *NFTP*. Y el promedio entre los cuatro Desafíos tendrá un peso del 20% sobre el total.<br>✓ Con lo anterior, y designando al promedio de todos los Cuestionarios como *PC* y al promedio de todos los Desafíos como *PD*, entonces la fórmula de cálculo de la *NFTP* es la siguiente:<br>NFTP = 0.175*TP1 + 0.175*TP2 + 0.175*TP3 + 0.175*TP4 + 0.1*PC + 0.2*PD<br>✓ El valor final así obtenido, será redondeado al entero más próximo, y esa finalmente será la nota NFTP. |
| **Regularidad: condiciones** | o Para ***regularizar*** la materia, el alumno tendrá cinco notas al final del año, que irán a la libreta: los parciales P1, P2, P3 y P4 más la NFTP. Para obtener la condición de regular el alumno deberá aprobar con nota mayor o igual a 4(cuatro) al menos tres de los cuatro parciales O BIEN los dos parciales del segundo semestre (P3 y P4), y obtener un promedio final de 4(cuatro) o más en la NFTP (*no es obligatorio* que los cuatro trabajos prácticos se entreguen ni que se hagan todos los cuestionarios y desafíos, siempre que al final del cursado el promedio ponderado que conforma la NFTP sea mayor o igual a 4(cuatro)).<br>o Por reglamento aprobado en la UTN desde 2017, los alumnos que alcancen |

<table>
<tr>
<td></td>
<td>
la condición de regular en cualquier asignatura podrán rendirla dentro de un período de un ciclo lectivo sin tener en cuenta correlativas previas. A partir de cumplido el ciclo lectivo desde la regularización, sólo se podrá rendir si previamente se rindieron y aprobaron todas las previas. La asignatura AED no tiene materias previas, pero la norma se cita aquí de todos modos en carácter informativo (para ser aplicada en asignaturas posteriores).<br>
o Todo alumno que no llegue a la condición mínima de regular deberá recursar la materia. Sólo a los efectos estadísticos, estos alumnos serán clasificados como alumnos libres o como alumnos que abandonaron el cursado, en base a los siguientes criterios:<br>
o Se considerará que un alumno abandonó el cursado de la materia si hizo menos del 50% de las instancias de evaluación. Aquí sólo consideramos como tales a los cuatro parciales y los cuatro trabajos prácticos (o sea, ocho instancias en total). De esta forma, el estado final de un alumno que sólo haya cumplido con tres de las ocho instancias mencionadas se registrará como abandono.<br>
o Se considerará que un alumno quedó en estado de libre si hizo al menos el 50% de las siete instancias de evaluación, pero no alcanzó las callificaciones y promedios requeridos para regularizar. De esta forma, el estado final de un alumno que haya cumplido con cuatro o más de las ocho instancias mencionadas, pero no haya logrado las calificaciones mínimas para regularizar, se registrará como libre.<br>
Escala de notas para el proceso de regularidad(*)
<table>
<tr><th>NOTAS</th><th>PORCENTAJE</th><th>CALIFICACIÓN</th></tr>
<tr><td>1</td><td></td><td>No Aprobado</td></tr>
<tr><td>2</td><td></td><td>No Aprobado</td></tr>
<tr><td>3</td><td></td><td>No Aprobado</td></tr>
<tr><td>4</td><td>55% a 57%</td><td>Aprobado</td></tr>
<tr><td>5</td><td>58% a 59%</td><td>Aprobado</td></tr>
<tr><td>6</td><td>60% a 68%</td><td>Aprobado</td></tr>
<tr><td>7</td><td>69% a 77%</td><td>Aprobado</td></tr>
<tr><td>8</td><td>78% a 86%</td><td>Aprobado</td></tr>
<tr><td>9</td><td>87% a 95%</td><td>Aprobado</td></tr>
<tr><td>10</td><td>96% a 100%</td><td>Aprobado</td></tr>
</table>
(*) Escala acordada en reunión de Docentes Coordinadores
</td>
</tr>
<tr>
<td>Promoción: condiciones</td>
<td>
o Todo alumno que rinda los cuatro parciales, los apruebe con notas no menores a 7(siete) y promedio entre los cuatro mayor o igual a 8(ocho), y obtenga promedio en la NFTP mayor o igual a 8(ocho) pero con todos los Trabajos Prácticos entregados y aprobados (con nota de 7 o más), quedará en situación de promocionado, con lo cual no deberá rendir la parte práctica de la materia en el examen final. La condición de promocionado no se pierde con el transcurso del tiempo, pero sí se
</td>
</tr>
</table>

perderá si el alumno resultare aplazado al presentarse a rendir el examen final (después del aplazo en un final, la ***promoción*** ya no valdrá para los turnos siguientes y el alumno deberá rendir como ***regular***).

- Está prevista una instancia de ***recuperación para promoción***, que se describe en el bloque correspondiente a la condición de ***aprobación directa***.

**Escala de notas para el proceso de regularidad(*)**

| NOTAS | PORCENTAJE | CALIFICACIÓN |
|---|---|---|
| 1 | | No Aprobado |
| 2 | | No Aprobado |
| 3 | | No Aprobado |
| 4 | 55% a 57% | Aprobado |
| 5 | 58% a 59% | Aprobado |
| 6 | 60% a 68% | Aprobado |
| 7 | 69% a 77% | Aprobado |
| 8 | 78% a 86% | Aprobado |
| 9 | 87% a 95% | Aprobado |
| 10 | 96% a 100% | Aprobado |

(*) Escala acordada en reunión de Docentes Coordinadores

**Aprobación Directa: condiciones.**

- A partir del ciclo 2017 se implementa la situación final de ***aprobación directa***, la cual implica que todo alumno que la logre, quedará eximido por completo del examen final y sólo deberá inscribirse en un turno de exámenes y presentarse para registrar su nota final en actas y en la libreta.

- Las condiciones para obtener la ***aprobación directa*** son las siguientes:

  a. ***Rendir y aprobar los cuatro parciales con notas no menores a 8(ocho) y promedio entre los tres mayor o igual a 9(nueve).***

  b. ***Obtener un promedio en la NFTP mayor o igual a 8(ocho) pero con todos los Trabajos Prácticos entregados y aprobados (con nota de 7 o más) (lo mismo que para la condición de promoción).***

  c. ***Rendir y aprobar (con nota de 6 o más) un Coloquio Final Teórico (que designaremos como Quinto Parcial o P5).***

- La condición de ***aprobación directa*** no se pierde con el transcurso del tiempo. Como se dijo para la promoción, los alumnos podrán inscribirse a exámenes dentro del plazo de un ciclo lectivo completo sin que se controlen las correlatividades previas (en caso de haberlas). Pero luego de ese ciclo completo, sólo podrán inscribirse a examen para firmar su aprobación directa si antes aprobaron las materias correlativas previas.

- Para los alumnos que alcancen la condición de ***aprobación directa***, la nota final de la materia, que será registrada en el acta de examen y en la libreta

del alumno, será el promedio entre los cinco parciales (los cuatro normales de regularidad y el Quinto Parcial) y la nota final del práctico. Esta nota será también registrada en la planilla final de regularidad en el Sistema de Autogestión.

- Si el alumno cumple las condiciones para ***aprobación directa*** SALVO porque uno (y sólo uno) de los cuatro parciales de regularidad quedó aprobado pero con ***nota = 7***, podrá hacer de todos modos el ***Quinto Parcial*** a modo de recuperación para ***aprobación directa***. Pero en este caso, deberá aprobar ese Quinto Parcial con nota de 8 o más.

- Por otra parte, si el alumno cumple las condiciones para ***promoción*** SALVO porque uno (y sólo uno) de los cuatro parciales de regularidad quedó aprobado pero con ***nota = 4, 5, o 6***, podrá entonces hacer el ***Quinto Parcial*** a modo de recuperación para ***promoción***. Pero en este caso, deberá aprobar ese Quinto Parcial con nota de 7 o más.

**Escala de notas para el proceso de regularidad(*)**

| NOTAS | PORCENTAJE | CALIFICACIÓN |
|---|---|---|
| 1 | | No Aprobado |
| 2 | | No Aprobado |
| 3 | | No Aprobado |
| 4 | 55% a 57% | Aprobado |
| 5 | 58% a 59% | Aprobado |
| 6 | 60% a 68% | Aprobado |
| 7 | 69% a 77% | Aprobado |
| 8 | 78% a 86% | Aprobado |
| 9 | 87% a 95% | Aprobado |
| 10 | 96% a 100% | Aprobado |

(*) Escala acordada en reunión de Docentes Coordinadores

<table>
<tr><td>Modalidad de examen final</td><td>
<ul>
<li>Alumnos regulares: desarrollo de un programa en el laboratorio sobre todos los temas de la asignatura. Si aprueba esta instancia, un coloquio breve con un profesor de la cátedra, que incluye también los temas de las unidades números 10 y 11.</li>
<li>Alumnos promocionados: un coloquio sobre todos los temas de la asignatura, que incluye los temas de las unidades 10 y 11.</li>
<li>Alumnos con aprobacion directa: sólo deben inscribirse a rendir y pasar al aula del examen el día en que sustancie, para registrar su nota final en el acta del examen y en su libreta.</li>
</ul>
Escala de Notas para Examen Final (*)
<table>
<tr><th>NOTA</th><th>PORCENTAJE</th><th>CALIFICACIÓN</th></tr>
<tr><td>1</td><td></td><td>Insuficiente</td></tr>
<tr><td>2</td><td></td><td>Insuficiente</td></tr>
<tr><td>3</td><td></td><td>Insuficiente</td></tr>
<tr><td>4</td><td></td><td>Insuficiente</td></tr>
<tr><td>5</td><td></td><td>Insuficiente</td></tr>
<tr><td>6</td><td>60% a 68%</td><td>Aprobado</td></tr>
<tr><td>7</td><td>69% a 77%</td><td>Bueno</td></tr>
<tr><td>8</td><td>78% a 86%</td><td>Muy Bueno</td></tr>
<tr><td>9</td><td>87% a 95%</td><td>Distinguido</td></tr>
<tr><td>10</td><td>96% a 100%</td><td>Sobresaliente</td></tr>
</table>
(*) Escala acordada en reunión de Docentes Coordinadores
</td></tr>
<tr><td>Actividades en laboratorio</td><td>
<ul>
<li>✓ En los primeros dos meses las clases teóricas se llevarán a cabo en las aulas comunes, y las prácticas en los laboratorios.</li>
<li>✓ Luego de esos dos primeros meses, todas las clases se desarrollarán en los laboratorios de informática.</li>
<li>✓ Cada alumno debe instalar en su computadora personal las herramientas de software requeridas por la materia, y realizar en forma personal prácticas y pruebas sobre ellas para lograr dominio pleno de su uso.</li>
<li>✓ Es bienvenida la utlización por parte de los alumnos de sus propias notebooks, en el trancurso de cada clase.</li>
<li>✓ En varias clases se prevé la realización de trabajos y tareas pautadas para ser teminadas (en la medida de lo posible) en el transcurso de la misma clase. Esos trabajos se suben al Aula Virtual para su revisión por parte de los jefes de trabajos prácticos y auxiliares del curso, o son evaluados en forma automática mediante mecanismos del Aula Virtual.</li>
</ul>
</td></tr>
<tr><td>Horas/año totales de la asignatura (hs. cátedra)</td><td>160 horas (32 semanas de clase, 5 horas cátedra por semana).</td></tr>
<tr><td>Cantidad de horas prácticas totales (hs. cátedra)</td><td>80 horas.</td></tr>
<tr><td>Cantidad de horas teóricas totales (hs. cátedra)</td><td>80 horas.</td></tr>
<tr><td>Tipo de formación práctica (sólo si es asignatura curricular</td><td>☐ Formación experimental<br>■ Resolución de problemas de ingeniería</td></tr>
</table>

| | |
|---|---|
| -no electiva-) | ☐ Actividades de proyecto y diseño<br>☐ Prácticas supervisadas en los sectores productivos y /o de servicios |
| **Cantidad de horas cátedras afectadas a la formación práctica indicada en el punto anterior**<br>(sólo si es asignatura curricular -no electiva-) | 34 horas. |
| **Descripción de los prácticos** | • *Cuestionarios Teóricos*: Cuestionarios de opciones múltiples, uno por semana a modo de control de lectura y refuerzo de temas.<br>• *Desafíos de Programación*: Enunciados sobre temas diversos, que deben ser resueltos por los alumnos informando los resultados obtenidos (sin subir los propios programas). Se complementan con preguntas teóricas.<br>• *Trabajos Prácticos*: Enunciados situacionales generales. Los alumnos resuelven en grupos y entregan el programa completo y funcionando. |
| **Criterios generales**<br>(los cuales serán tenidos en cuenta en las correcciones) | Los trabajos prácticos se entregan a través del aula virtual, y son revisados y calificados por los docentes de la Cátedra. Los cuestionarios y desafíos son de corrección automática. En casos de parciales y trabajos prácticos, los alumnos deben presentar a través del aula virtual proyectos desarrollados en Python, que incluyan los archivos fuente implementados. Cuando se trate de cuestionarios y desafíos de programación, los alumnos simplemente deben subir sus respuestas mediante los recursos disponibles en el aula virtual. El enunciado de cada *trabajo práctico* incluye consignas a cumplir y recomendaciones adicionales, además de criterios de evaluación tales como: compilación, completitud de consignas cubiertas, diseño de clases, lógica aplicada en la solución de cada ítem, diseño de interfaz de usuario y elementos de control de carga e integridad. Cada criterio se evalúa y aporta un peso a la calificación final. De acuerdo a la suma de esos pesos, se aplica la escala de notas que ya fue indicada en secciones anteriores de este mismo documento (y que forma parte de la consigna mostrada a los estudiantes) Lo mismo vale para los parciales. |
| **Cronograma de actividades de la asignatura** | Ver achivo *Crono AED [2018].xls* anexo a esta presentación. |
| **Propuesta para la atención de consultas y mail de contacto.** | Todos los profesores, jefes de trabajos prácticos y auxiliares de la Cátedra están registrados y disponibles en el Aula Virtual de la asignatura, en la que todos los alumnos deben a su vez registrarse. Mediante el servicio de mensajería y los foros del Aula Vritual, todos los alumnos pueden realizar todas las consultas que deseen a cualquiera de los docentes. Además, la Cátedra (en conjunto con el Departamento de Ingeniería en Sistemas de Información) tiene implementado un esquema de *tutorías* para atención de alumnos, fuera de los horarios de clase. El esquema de *horarios de tutorías* es gestionado por Departamento de Sistemas y es atendido por docentes de la Cátedra. |
| **Plan de integración con otras asignaturas** | Esta asignatura es la correlativa previa natural de Paradigmas de Programación, y aporta elementos fundamentales para todas las asignaturas de programación de la carrera y para las asignaturas que directa o indirectamente apliquen sobre el diseño y desarrollo de software. Por lo tanto, la Cátedra en su conjunto ha realizado un gran esfuerzo por actualizar sus contenidos y prácticas, de forma de abarcar tan profundamente como sea posible los contenidos mínimos exigidos |

| | |
|---|---|
| | por la Ordenanza 1150 de Rectorado que regula la creación y funcionamiento de la Carrera. Dichos contenidos son requisitos previos indispensables en numerosas asignaturas posteriores, y de esta forma se ha tratado de garantizar que los alumnos que regularizan AED tengan la formación básica esperable. |
| **Bibliografía Obligatoria** | [1] Frittelli, V., et al. (2018). Publicaciones Cátedra de AED. *Fichas de Estudio para AED (disponible en Aula Virtual AED)*. Cordoba: UTN – Facultad Regional Córdoba.<br>[2] Python Software Foundation. (2017). *Python Documentation*. Retrieved February 24, 2016, from https://docs.python.org/3/. |
| **Bibliografía Complementaria** | [1] Pilgrim, M. (2004). *Dive Into Python - Python from novice to pro*. Retrieved March 6, 2016, from http://www.diveintopython.net/toc/index.html. |
| **Distribución de docentes** | Ver achivo *AED Planta Docente [2018].pdf* anexo a esta presentación. |

Firma: …….

Aclaración: …….....