7501|9501 – Computación

Planificación

OBJETIVOS

El Marco Curricular recientemente aprobado expresa que “entre los rasgos que caracterizan a una persona graduada en FIUBA, con las especificidades que corresponda establecer en cada carrera, se pueden mencionar:

- Formación académica (científica y tecnológica) y profesional sólida y actualizada que le permita interpretar y procesar los cambios de paradigmas, extender la frontera del conocimiento e intervenir en las políticas públicas.
- Competencia para seleccionar y utilizar de manera efectiva las técnicas y herramientas propias de su carrera, tanto para la actividad profesional de excelencia como para iniciarse en la docencia, la investigación y el desarrollo.
- Capacidad de diseñar, planificar, realizar, evaluar, mejorar y gestionar proyectos y de generar e implementar soluciones a problemas profesionales complejos de naturaleza tecnológica, que sean acordes a los requerimientos del mundo actual y a las necesidades de la sociedad y del país, que les permita contribuir al desarrollo económico, ambiental y social con una perspectiva de accesibilidad y sustentabilidad.
- Formación integral que habilite el ejercicio profesional con una visión interdisciplinaria y amplia del país y del contexto, de acuerdo con principios éticos, compromiso social y responsabilidad cívica.
- Competencias para desempeñarse con creatividad, emprendedorismo y espíritu crítico, integrando y liderando equipos diversos.
- Capacidad para el aprendizaje continuo y autónomo y el desarrollo profesional en contextos de cambios sociales y tecnológicos.
- Competencias comunicacionales para desempeñarse en contextos interdisciplinarios, interculturales e internacionales; en redes virtuales y en dinámicas de trabajo grupal; utilizando tanto el español como el inglés.”

Enseñanza

Objetivos Generales

- Que el alumno adquiera una visión global de la Computación para comprender el aspecto científico de la actual sociedad informatizada.
- Que el alumno comprenda conceptos y técnicas de la disciplina que en el ejercicio profesional le posibiliten la interacción con profesionales en Informática sin problemas de comunicación.
- Que el alumno logre compenetrarse con las tecnologías y herramientas fundamentales de la Computación para usar la computadora como instrumento de trabajo, conociendo su precisión, capacidad y limitaciones.
- Que el alumno se familiarice con el modo de pensar en Ingeniería.

Objetivos Específicos

- Que el alumno tome conciencia de la importancia de la Algoritmia como paradigma de resolución de problemas y de la Programación como práctica en la resolución de problemas con la computadora.
- Que el alumno desarrolle la capacidad de relacionar esquemas de solución de problemas con la resolución de problemas algorítmicos, con hincapié en el método científico.
- Que el alumno desarrolle la capacidad de Análisis, Sistematización, Programación y Procesamiento de distintos problemas de tipo técnico-científicos para utilizar dichos conocimientos en su formación académica actual y en su ejercicio profesional futuro.

Aprendizaje

Competencias Generales de Ingeniería

- Conocimiento de tecnologías y métodos básicos para adquirir capacidad para el aprendizaje de nuevos métodos y tecnologías, y versatilidad para la adaptación a nuevas situaciones.
- Capacidad de resolver problemas con iniciativa, toma de decisiones, creatividad y sentido crítico.
- Capacidad de comunicar y transmitir conocimientos, habilidades y destrezas, comprendiendo la responsabilidad profesional de la actividad del Ingeniero.

Competencias Específicas de Computación

Cognitivas (Saber conocer)

- Conocimiento general sobre Algoritmia y conocimientos básicos sobre el uso y programación de computadoras, y de la sintaxis de un lenguaje de programación.
- Conocimiento de la organización, capacidades y funcionamiento de las computadoras y los fundamentos de su programación, y su aplicación para la resolución de problemas propios de la Ingeniería.
- Conocimiento y aplicación de los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad de los algoritmos propuestos, y conocimiento de los tipos y estructuras de datos fundamentales y su utilización apropiada para la resolución de un problema (programa eficiente).

Procedimentales (Saber hacer)

- Capacidad de resolución de problemas mediante algoritmos y programas que permitan ejecutarlos en una computadora (programa eficaz).
- Capacidad para descomponer un problema real en subproblemas para su posterior codificación en un programa.
- Capacidad para documentar programas con claridad y sencillez (programa inteligible) y, comprender documentación técnica y reutilizar código desarrollado por terceras partes.

Actitudinales (Saber ser)

- Motivación por la claridad, sencillez, eficacia y eficiencia algorítmica de problemas y su traducción a programas.
- Capacidad para debatir y concluir las distintas soluciones algorítmicas a un problema traducidas a programas.

Competencias Transversales o Genéricas

- Capacidad para la autoorganización y planificación del trabajo individual y del proceso de aprendizaje.
- Capacidad para el trabajo en grupo.
- Capacidad de análisis y síntesis.
- Motivación por la calidad del resultado.

- Disposición para el compromiso, la responsabilidad, la colaboración, el empeño, la dedicación, la solidaridad, la honestidad y el respeto en el trabajo individual y grupal durante el proceso de construcción del conocimiento.

PROGRAMA SINTÉTICO

Alcance de las Ciencias de la Computación. Técnicas para representar y almacenar información y forma en que las máquinas digitales manipulan los datos. Software de sistema, de aplicación y de traducción. Lenguajes de programación. Algoritmia y programación básicas.

PROGRAMA ANALÍTICO

Introducción a la Computación

Algoritmos. Alcance de las ciencias de la computación. Arquitectura de computadoras. Sistemas de numeración binario y hexadecimal.

Representación y almacenamiento de datos

Unidad central de almacenamiento. Memoria secundaria. Dispositivos periféricos. Códigos: para representar y almacenar símbolos (ASCII y UNICODE), números enteros (complemento a dos) y números reales (punto flotante normalizado). Confiabilidad: métodos de detección y corrección de errores.

Manipulación de datos

Unidad central de procesamiento. Codificación y almacenamiento de programas. Lenguaje de máquina. Ejecución de programas.

Nociones de Software

Software de sistema, de aplicación y de desarrollo. El sistema operativo: funciones, interfaz basada en caracteres e interfaz gráfica. Redes de computadoras y software de comunicación.

Introducción a la Algoritmia y a la Programación

Desarrollo de algoritmos: teoría de resolución de problemas de Pólya aplicada a la algoritmia. Enfoques descendente *(top down)* y ascendente *(bottom up)*. Pensamiento computacional. Tipos estándar y variables globales. Primitivas de especificación de algoritmos: asignación, entrada y salida estándar de datos, expresiones, estructuras de control selectivas (simples y múltiples), repetitivas (indefinidas y definidas) y de invocación de subalgoritmos (transferencia/retorno). Procesamiento de secuencias. Eficiencia, generalidad y corrección de algoritmos. Lenguajes de programación: historia, la programación imperativa o por procedimientos como paradigma de comunicación de algoritmos a computadoras; traducción e interpretación de programas; paradigmas de programación abstracta (programación orientada a objetos, programación lógica, programación funcional) y lenguajes que las sustentan.

El lenguaje de programación Python

Estructura de un programa Python y entorno integrado de desarrollo y aprendizaje IDLE. Modelo de Programa Tipo. Tipos de datos y variables: declaraciones. Funciones de librería. Enunciados de documentación interna y de entradas y salidas. Definición de tipos estructurados: secuencias. Mutabilidad, inmutabilidad y representación de la información en memoria en programas. Archivos de texto (memoria persistente): estructura, caracteres de control, funciones predefinidas y aplicaciones para captura de datos y comunicación de resultados de programas.

Unidades de programación

Funciones como estructuras de control de transferencia-retorno y como recursos de programación. Parámetros: formales y reales, por referencia y por valor. Definición de variables locales; reglas de alcance. Reusabilidad del software. Principios de modularización: cohesión y acoplamiento.

Agrupamiento de datos

Listas simples y compuestas. Listas como parámetros. Búsqueda y ordenamiento de elementos en listas. Aplicaciones: aritmética de alta precisión, álgebra de polinomios y matrices, resolución algebraica de sistemas de ecuaciones. Tuplas y diccionarios.

BIBLIOGRAFÍA

1 INTRODUCCIÓN A LA COMPUTACIÓN. J. Glenn Brookshear. Pearson Educación, S. A. Madrid. Undécima edición, 2012. ISBN: 97884782911397.

2 ALGORITMOS Y PROGRAMACIÓN I CON LENGUAJE PYTHON. Rosita Wachenchauzer, Margarita Manterola, Maximiliano Curia, Marcos Medrano y Nicolás Paez. 9 de marzo de 2011. En https://librosweb.es/libro/algoritmos_python/

3 INTRODUCCIÓN A LA PROGRAMACIÓN CON PYTHON 3. Andrés Marzal Varó, Isabel Gracia Luengo y Pedro García Sevilla. Departamento de Lenguajes y Sistemas Informáticos. Códigos de Asignaturas: EI1003 y MT1003. Publicaciones Universitat Jaume I. Primera edición, 2014. ISBN: 978-84-697-1178-1. En http://repositori.uji.es/xmlui/handle/10234/102653

4 GUÍAS TEÓRICAS Y PRÁCTICAS DEL CURSO DE COMPUTACIÓN (Elaboración Propia).

5 PYTHON SOFTWARE FOUNDATION. En https://www.python.org/about/

6 EL TUTORIAL DE PYTHON. Guido van Rossum. En http://docs.python.org.ar/tutorial/pdfs/TutorialPython3.pdf

7 GUÍA DE ESTILO DEL CÓDIGO PYTHON. Guido van Rossum y Barry Warsaw. Traducción al castellano por Raúl González Duque. 10 de agosto de 2007. En http://mmc.geofisica.unam.mx/edp/Herramientas/Lenguajes/Python/Convenciones.pdf

RÉGIMEN DE CURSADO

Metodología de enseñanza

La metodología de enseñanza adoptada para la materia es la de “Aprendizaje Guiado por (o basado en) Proyectos” (AGxP). Esta metodología se adoptó tomando en cuenta:

- El perfil buscado para el ingeniero FIUBA.
- Las opiniones recogidas en las reuniones de la coordinación de la materia con las comisiones curriculares.
- El asesoramiento brindado desde la Secretaría de Planificación Académica para alcanzar los objetivos buscados.

En rasgos generales, AGxP se basa en tomar aspectos de tres pilares:

- Aula invertida: se suplanta mayormente la clase expositiva tradicional por materiales que los alumnos ven por su cuenta antes de la clase, en lecturas, videos, etc., y se deja el espacio áulico para la resolución asistida de problemas basados en la teoría vista con anterioridad.
- “Conocimiento guiado por necesidades”: el conocimiento teórico que se imparte (en modalidad de aula invertida o tradicional) está **guiado** mayormente por la **necesidad** de ser usado en proyectos prácticos y concretos, que sean lo más parecido a un proyecto de ingeniería que sea posible a esta altura de las carreras.
- Trabajo en equipo: siempre que se pueda, se fomentará el trabajo en equipo, y se les brindarán herramientas para que aprendan a hacerlo mejor.

Para mantener la gradualidad en el abordaje de los temas, se desarrollan **dos proyectos en la segunda mitad de la cursada** (de la semana 9 a la semana 16), manteniendo las 8 primeras semanas con una modalidad tradicional de clases teóricas y prácticas y ejecución de trabajos prácticos de computadora. Esto supone una fuerte carga de

trabajos prácticos y, por lo tanto, requiere que los alumnos **adquieran conocimientos en estrategias de trabajo**, tanto metodológicas como actitudinales: organización del estudio, del trabajo, tanto individual como en equipo, etc.

Entre los aspectos relacionados con el perfil del ingeniero FIUBA definido en el Marco Curricular, esta materia pretende contribuir al desarrollo de competencias para seleccionar y utilizar de manera efectiva las técnicas y herramientas propias de su carrera, la adquisición de habilidades para integrar equipos diversos, la capacitación en el aprendizaje continuo y autónomo, y comunicacionales.

Estrategia pedagógica

La estrategia pedagógica para el desarrollo de contenidos de Computación se sintetiza en la experiencia de pensar para crear juntos. Integra enseñar a pensar y enseñar contenidos. El desarrollo de los contenidos teóricos y prácticos se aborda desde tres núcleos centrales (algoritmo, programa y computadora) en forma interrelacionada, iterativa e incremental y se organiza en espiral. El descubrimiento de algoritmos y el desarrollo de programas se complejiza y el conocimiento de la computadora se profundiza durante las 16 semanas de clase.

Para solucionar problemas con la computadora los estudiantes utilizan el Modelo de Solución de Problemas de Pólya aplicado al ámbito de la construcción de programas que consta de cuatro fases:

1. **Análisis**. Deben comprender en qué consiste el problema a resolver con la construcción del programa.
2. **Diseño**. Deben diseñar una estrategia para definir recursos y descubrir el algoritmo.

   El análisis del problema y el diseño de la solución, son las fases algorítmicas y constituyen el desafío creativo de aprendizaje porque requiere que los estudiantes desplieguen el pensamiento computacional, competencia clave de aprendizaje en la formación de los estudiantes de ingeniería.

3. **Codificación**. Deben implementar la estrategia para construir el programa.
4. **Evaluación**. Deben ejecutar el programa construido para comprobar que la solución es correcta. La Codificación y la Evaluación son las fases de programación.

EVALUACIONES

La evaluación tiene dos instancias:

- Una evaluación final individual que tiene como finalidad acreditar la asignatura. Esta evaluación, comúnmente llamada examen final o integrador, se realiza una vez terminada la cursada del cuatrimestre, en las fechas que defina la Facultad.
- Una evaluación procesual, cuyo objetivo es el de obtener información para el docente y para cada alumno acerca de la apropiación de los contenidos enseñados, y acreditarlos. Esta evaluación se desarrolla de manera continua e incremental durante la cursada, y define la aprobación de la cursada.

Forma de evaluación de la cursada

Los elementos con los que se va a evaluar a los estudiantes son dos trabajos prácticos que se desarrollan en la primera mitad del cuatrimestre, y dos proyectos que se desarrollan en la segunda mitad:

- TP individual, evaluado de manera incremental.

  Trabajo introductorio de programación para que los alumnos se familiaricen con las herramientas y el lenguaje de programación.

- TP grupal, evaluado de manera incremental.

  Trabajo de algoritmia y programación para resolver en equipo y de manera coordinada un problema complejo, aplicando estructuras de control y de datos específicas.

- Proyecto bajo la modalidad AGxP, evaluado de manera incremental.

Desarrollo en etapas con selección de estructuras de datos y de control según el problema a resolver e incorporación de técnicas y herramientas nuevas en base a material suministrado por los docentes.

- Proyecto bajo la modalidad AGxP, evaluado de manera incremental.

  Desarrollo en etapas con selección de estructuras de datos y de control según el problema a resolver e incorporación de técnicas y herramientas nuevas en base a material suministrado por los docentes.

  Este proyecto se evaluará individualmente para garantizar que todos los alumnos del equipo adquirieron los conocimientos suficientes para aprobar la cursada. Esto puede realizarse con una instancia de evaluación oral, o una exposición del equipo con participación de todos sus miembros.

Durante el desarrollo de los trabajos prácticos y los proyectos, los docentes harán un seguimiento semanal de los alumnos, tanto para auxiliarlos ante las dificultades que se les presenten como también para saber el grado de participación de los integrantes de cada equipo. Se trabajará con una lista de control conocida por los alumnos para que sirva también como autoevaluación. Esto está fuertemente relacionado con el propósito de evaluación continua e incremental.

Condiciones de aprobación de la cursada

La aprobación de la cursada se hace mediante 2 trabajos prácticos y 2 proyectos o casos, con notas individuales.

Para aprobar la cursada, un alumno debe aprobar desarrollar los dos trabajos prácticos y los dos proyectos. Los trabajos prácticos son formativos y tendrán devoluciones de ese carácter. Los proyectos tendrán calificaciones en consideración de los siguientes ítems:

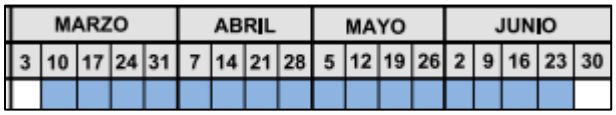
- Uso de variables y constantes
- Uso de estructuras de control
- Uso de funciones predefinidas
- Definición de funciones como medio de modularización
- Uso de comentarios y documentación
- Organización para el trabajo
- Calidad y claridad del código

Evaluación final

Individual, que comprende el desarrollo de un programa en Python y la respuesta a planteos conceptuales o procedimentales sobre computadoras, datos, sistemas operativos y redes.

CRONOGRAMA

Las clases son teórico-prácticas obligatorias, desarrolladas en un laboratorio con computadoras, y de 4 horas de duración.



Los días de la segunda fila corresponden a los lunes de cada semana. Las clases del curso son los martes, es decir, hay que sumarle uno al número de día.

Semana	Computadoras, Datos, SSOO y Redes	Algoritmia y Programación
1	Conceptos de algoritmo y programación. Arquitectura básica de una computadora.	IDLE de Python. Plantilla de programa tipo con documentación de datos de análisis y prueba y secciones algorítmicas. Teoría de resolución de problemas de Pólya aplicada a la algoritmia. Instrucciones de entrada y salida y operaciones con números enteros, en punto flotante y complejos. Edición o formato de salidas según tipos de datos **f'…'**. Biblioteca math.

Semana	Computadoras, Datos, SSOO y Redes	Algoritmia y Programación	
2	Sistemas de numeración binario y hexadecimal. Cambios de base.	Estructura de selección simple **if[-else]**. Estructura de selección múltiple **if-elif[-elif[…[-else]]…]**.	
3	Representación de enteros en complemento a 2.	Estructura de control de iteraciones **while**. Procesamiento de secuencias con condición de continuación simple. Procesamiento de secuencia con condición de continuación compuesta (problemas con sucesiones y series numéricas). Leyes de De Morgan.	TP1 (individual)
4	Representación de números reales en punto flotante.	Definición y uso de funciones. Parámetros, variables globales y variables locales. Funciones que devuelven uno o más resultados. Funciones como órdenes (sin retorno de resultados).	
5	Memoria RAM y sistemas de almacenamiento persistente.	Funciones recursivas. Resolución de problemas iterativos ya vistos, en forma recursiva. Equivalencia de soluciones iterativas y recursivas. Riesgos de la recursión.	
6	Arquitecturas RISC y CISC. Representación de instrucciones y ciclo de máquina. Programas almacenados en RAM.	Archivos de texto. Lectura de datos desde archivos de texto. Estructura **for** para recorrido de líneas. Listas. Creación e inserción de elementos. Estructura iterativa **for**, y funciones **len()** y **range()**. Creación de listas desde archivos de texto.	TP2 (grupal)
7	Sistemas operativos.	Copia de listas. Eliminación de elementos. Representación de polinomios como listas. Listas como parámetros de funciones. Biblioteca random.	
8	Redes: topologías, Internet y protocolos.	Lectura de archivos de texto línea por línea. Edición de datos heterogéneos en archivos de texto. Matrices como listas de listas y carga desde archivos de texto. Escritura de resultados en archivos de texto.	
9		Proyecto grupal 1	
10			
11			
12		Proyecto grupal 2	
13			
14			
15			
16			