Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2115 - PROGRAMACIÓN COMO HERRAMIENTA PARA LA INGENIERÍA

– Programa de curso –

Profesores : Francisco Garrido (`fgv@ing.puc.cl`) y Hans Löbel (`halobel@ing.puc.cl`)

Sitio Web : Siding y Syllabus en GitHub (`github.com/IIC2115/Syllabus`)

Clases : Lunes, módulos 4 y 5 (14:00 - 16:50) - Sala A7

Ayudantía : Jueves, módulo 4 (14:00 - 15:20) - Sala B13

Horario de atención : agendar cita por email

1 Presentación del curso

Durante los últimos años, el uso y desarrollo de software especializado en las distintas especialidades de la ingeniería se ha transformado en una constante, ya sea por lo complejo de las tareas a realizar, o por la gran cantidad de datos que es necesario analizar. Es por esto que el conocimiento y las habilidades relacionadas con la programación se han transformado no sólo en una ventaja, sino en una necesidad para los profesionales de la ingeniería.

El propósito de este curso es que el alumno se familiarice con la programación como una básica y poderosa herramienta, no sólo para solucionar de manera más eficiente y efectiva problemas clásicos en ingeniería, sino que además para desarrollar soluciones innovadoras a nuevos problemas. Para alcanzar este objetivo, el curso cubre una amplia variedad de tópicos, incluyendo lenguajes y herramientas que son fundamentales para enfrentar de manera satisfactoria problemas de ingeniería, tanto en el aspecto profesional, como en el académico.

2 Objetivos de aprendizaje

A nivel general, al finalizar el curso los alumnos serán capaces de:

- Evaluar y utilizar de manera efectiva distintos lenguajes y herramientas de programación para resolver problemas asociados a sus áreas de especialización, en base a los requerimientos de estos.
- Proponer y desarrollar soluciones novedosas utilizando la programación, ya sea para problemas tradicionales o para nuevos problemas en ingeniería.

A nivel particular, al finalizar el curso los alumnos serán capaces de:

- Utilizar herramientas modernas para el desarrollo de software.
- Modelar problemas utilizando técnicas de programación orientada a objetos.
- Crear soluciones a problemas, utilizando estructuras y técnicas avanzadas de programación.
- Modelar datos y sus relaciones, y realizar consultas sobre estos, mediante distintos modelos y lenguajes.
- Analizar, visualizar y presentar datos utilizando distintos lenguajes.
- Manejar, procesar y visualizar datos de sistemas de información geográfica (SIG) mediante lenguajes de programación.

3 Contenido

A continuación se presenta un desglose detallado de los contenidos del curso:

Capítulo 0: Introducción y herramientas básicas

- Sistema de control de versiones: git y GitHub
- Python, Jupyter Notebook y Google Colab
- Manejo de errores y debugging
- Introducción a consola de comandos

Capítulo 1: Programación orientada a objetos (OOP)

- Clases
- Agregación y composición

- Herencia y herencia múltiple
- Polimorfismo y clases abstractas
- Diagramas de clases

Capítulo 2: Estructuras de datos

- Stacks y colas
- Diccionarios
- Sets
- Árboles, listas ligadas y grafos

Capítulo 3: Técnicas y algoritmos

- Recursión y Backtracking
- Dividir y conquistar
- Ordenamiento y búsqueda en arreglos
- Búsqueda en grafos

Capítulo 4: Uso de bases de datos y archivos

- Manejo de archivos
- Modelo relacional de datos
- Consultas sobre datos usando SQL
- Uso de SQL en Python

Capítulo 5: Análisis de datos en Python

- Manipulación y limpieza de datos
- Visualización con herramientas externas
- Clasificación y regresión
- Programación funcional

Capítulo 6: Tópicos avanzados

- Manejo de GIS en python
- Simulación

- Web services y web scrapping
- Manejo de datos espaciales

4 Metodología

El curso sigue una metodología de clase invertida (*flipped classroom*), donde los alumnos deben estudiar y manejar los contenidos de manera previa a la clase, para luego aplicarlos en ella mediante actividades prácticas de programación. Estas actividades son individuales y son acumulativas en cuanto a contenidos a lo largo del semestre.

La dinámica del curso se desarrolla en el marco de seis capítulos de contenidos, donde cada uno tiene una duración aproximada de dos semanas. Cada capítulo considera 2 bloques de dos módulos, y una ayudantía. El material de cada capítulo es liberado una semana antes del inicio de este.

El primer bloque de cada capítulo considera dos actividades. La primera se desarrolla durante el primer módulo y consiste en una consolidación de los contenidos cubiertos por el material del capítulo, mediante una exposición por parte de los profesores del curso. Posteriormente, en el segundo modulo del primer bloque, los alumnos deberán realizar una actividad de programación evaluada de manera individual, donde deberán aplicar los contenidos estudiados.

El segundo bloque de cada capítulo considera el desarrollo de un laboratorio práctico individual. Este laboratorio tendrá mayor dificultad que las actividad del capítulo correspondiente. El enunciado de cada laboratorio se liberará una semana antes del inicio del segundo bloque de cada capítulo y su extensión es de aproximadamente dos semanas.

Las ayudantías se utilizarán para contestar dudas relacionadas con el desarrollo de los laboratorios. Tanto las ayudantías como las sesiones de cátedra no considerarán en ningún caso la revisión de materia, con excepción del módulo de consolidación de contenidos.

Cada una de las sesiones del curso considera el trabajo en la sala con la presencia de integrantes del cuerpo docente con el fin de resolver sus consultas. Así, es fundamental que los alumnos asistan a las sesiones, de este modo puedan recibir ayuda para solucionar problemas y certificar el avance realizado. Adicionalmente, los alumnos podrán realizar consultas mediante las *issues* de GitHub de manera *online*. **Se espera además que los alumnos utilicen otras fuentes para complementar y profundizar los contenidos, tales como los libros que se encuentran indicados en la bibliografía o sitios de internet**.

5 Evaluaciones

Las evaluaciones se dividen en tres tipos, cada una con su correspondiente nota final promedio:

- Laboratorios prácticos (60%): se realizarán 6 laboratorios prácticos evaluados, cuya calificación se basará en su completitud y la aplicación de los contenidos involucrados. Los laboratorios se realizarán de manera individual. Para la entrega se utilizará la plataforma GitHub y la fecha limité será las 23:59 del día indicado en el enunciado.

  La no entrega de un laboratorio será calificada con nota 1.0, mientras que por atraso se descontará un total $0.3 + 5.7/(1 + 0.8e^{12-2.2t})$) puntos, donde t corresponde a las horas de retraso. La nota final de los laboratorios prácticos ($\mathbf{L}$) está dada por el promedio de los 6 laboratorios.

- Participación (30%): Durante el segundo módulo de la primera clase de cada capítulo, los alumnos deberán resolver un problema que tiene directa relación con los contenidos del capítulo. Este podrá tener nota 1.0, 4.0 ó 7.0. La nota final de las actividades de participación ($\mathbf{P}$) está dada por el promedio de las notas obtenidas en los problemas.

- Asistencia (10%): La asistencia se contabilizará en algún momento del segundo módulo de todas las clases. Las ayudantías son voluntarias pero se recomienda fuertemente su asistencia. Por cada laboratorio, la nota de asistencia es 7.0 si asiste a todas las clases (4 módulos), 4.0 si asiste solo un día o 1.0 si no asiste a ninguna clase del laboratorio. La nota final de asistencia ($\mathbf{A}$) se calculará como el promedio de las 6 notas de asistencia por laboratorio.

6 Exigencias de aprobación

Para aprobar el curso, las notas $\mathbf{L}$, $\mathbf{P}$ y $\mathbf{A}$ deben ser mayores o iguales a 3.95. En caso de cumplir este criterio, la nota final del curso ($\mathbf{F}$) se calcula de la siguiente manera:

$$\mathbf{F} = 0.6 \cdot \mathbf{L} + 0.3 \cdot \mathbf{P} + 0.1 \cdot \mathbf{A}$$

En caso contrario, la nota final de reprobación ($\tilde{\mathbf{F}}$) será:

$$\tilde{\mathbf{F}} = min(3.9, \mathbf{F})$$

7 Retroalimentación y recorrecciones

Dada la naturaleza práctica de la metodología del curso, es fundamental la entrega de retroalimentación rápida en relación a lo realizado en los laboratorios, con el fin de contribuir de manera temprana

al correcto aprendizaje de los contenidos. Tomando esto en consideración, cada uno de los laboratorios tendrá retroalimentación, que se entregará junto con la nota. Consistirá en una descripción detallada, donde se indicarán todos los elementos que fueron relevantes para la corrección, además de la asignación de puntaje por cada uno de estos. En caso de no quedar conforme con la nota obtenida y/o la retroalimentación, se debe realizar una solicitud de recorrección **sólo** a través del formulario indicado en el sitio del curso.

8 Cronograma de actividades

Capítulo	Tópicos	Fecha	Actividades
Capítulo 0	Metodología, git, Jup. Notebooks	09/03	Introducción al curso, material cap. 1
Capítulo 1	Prog. orientada a objetos	16/03	Consolidación, P01, enunciado L01
		23/03	Clase taller L01, material cap. 2
		26/03	Ayudantía taller L01
Capítulo 2	Estructuras de datos	30/03	Consolidación, P02, enunciado L02
		06/04	Clase taller L02, material cap. 3
		09/04	Ayudantía taller L02
Capítulo 3	Técnicas y algoritmos	13/04	Consolidación, P03, enunciado L03
		20/04	Clase taller L03
		23/04	Ayudantía taller L03
		27/04	Clase taller L03, material cap. 4
		30/04	Ayudantía taller L03
Capítulo 4	Bases de datos y archivos	04/05	Consolidación, P04, enunciado L04
		11/05	Clase taller L04, material cap. 5
		14/05	Ayudantía taller L04
Capítulo 5	Análisis y visualización de datos	18/05	Consolidación, P05, enunciado L05
		25/05	Clase taller L05
		28/05	Ayudantía taller L05
		01/06	Clase taller L05, material cap. 6
		04/06	Ayudantía taller L05
Capítulo 6	Tópicos avanzados	08/06	Consolidación, P06, enunciado L06
		15/06	Clase taller L06
		18/06	Ayudantía taller L06
		22/06	Clase taller L06
		25/06	Ayudantía taller L06

Laboratorio	Publicación	Entrega
L01	16/03	29/03
L02	30/03	12/04
L03	13/04	03/05
L04	04/05	17/05
L05	18/05	07/06
L06	08/06	28/06

9 Política de Integridad Académica

Los alumnos de la Escuela de Ingeniería deben mantener un comportamiento acorde al Código de Honor de la Universidad:

"Como miembro de la comunidad de la Pontificia Universidad Católica de Chile me comprometo a respetar los principios y normativas que la rigen. Asimismo, prometo actuar con rectitud y honestidad en las relaciones con los demás integrantes de la comunidad y en la realización de todo trabajo, particularmente en aquellas actividades vinculadas a la docencia, el aprendizaje y la creación, difusión y transferencia del conocimiento. Además, velaré por la integridad de las personas y cuidaré los bienes de la Universidad."

En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un procedimiento sumario. Ejemplos de actos deshonestos son la copia, el uso de material o equipos no permitidos en las evaluaciones, el plagio, o la falsificación de identidad, entre otros. Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica en relación a copia y plagio: todo trabajo presentado por un alumno (grupo) para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno (grupo), sin apoyo en material de terceros. Si un alumno (grupo) copia un trabajo, se le calificará con nota 1.0 en dicha evaluación y dependiendo de la gravedad de sus acciones podrá tener un 1.0 en todo ese ítem de evaluaciones o un 1.1 en el curso. Además, los antecedentes serán enviados a la Dirección de Docencia de la Escuela de Ingeniería para evaluar posteriores sanciones en conjunto con la Universidad, las que pueden incluir un procedimiento sumario. Por "copia" o "plagio" se entiende incluir en el trabajo presentado como propio, partes desarrolladas por otra persona. Está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la cita correspondiente.

10 Bibliografía

- Apuntes del curso disponibles en el sitio.
- *Advanced Computer Programming in Python*; Pichara y Pieringer; 2017.
- *Database Management Systems*; Ramakrishnan y Gehrke; 2002.
- *LaTeX Beginner's Guide*; Kottwitz; 2011.
- *Introduction to Algorithms*; Cormen, Leiserson, Rivest y Stein; 2009 (3ª edición).
- *Python Data Science Handbook*; VanderPlass; 2016.