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

IIC2115 - PROGRAMACIÓN COMO HERRAMIENTA PARA LA INGENIERÍA

– Programa de curso –

Profesor	:	Francisco Garrido Valenzuela (`fgv@ing.puc.cl`)
Sitio Web	:	Siding y Syllabus en GitHub (`github.com/IIC2115/Syllabus`)
Clases	:	jueves, módulos 4 y 5 (14:00 - 16:50) - Sala BC24
Ayudantía	:	martes, módulo 4 (14:00 - 15:20) - Sala K200
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.

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 los seis capítulos de contenidos. Cada capítulo será desarrollado en cuatro cátedras (2 bloques de dos módulos) y una ayudantía, en una duración aproximada de dos semanas. Durante este periodo los alumnos deben estudiar los contenidos del capítulo y desarrollar un laboratorio práctico evaluado. Además, en la primera clase cada capítulo (primeros dos módulos), se realizará una consolidación de los contenidos para luego desarrollar una actividad de particiación evaluada que se entrega durante la clase. Adicionalmente, contaran con tres módulos (dos de clases y uno de ayudantía) para trabajar en la sala con la presencia de integrantes del cuerpo docente y resolver sus consultas.

Todas las actividades serán explicadas y discutidas al inicio de cada clase. Dicho esto, es fundamental que los alumnos asistan a las sesiones para recibir recomendaciones e indicaciones del cuerpo docente. De este modo puedan solucionar problemas y certificar el avance realizado.

El material de estudio para cada tópico se encontrará disponible en el sitio del curso una clase antes del inicio de cada laboratorio (con excepción del capítulo introductorio). **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**.

Las sesiones de cátedra no considerarán en ningún caso la revisión de materia, con excepción de las clases de consolidación de contenidos. Los laboratorios y actividades de participación se realizarán con la presencia de los profesores y ayudantes, quienes estarán disponibles para contestar dudas y aclarar conceptos. Adicionalmente, los alumnos podrán realizar consultas mediante las *issues* de GitHub de manera *online*. La asistencia a clases es "voluntaria", pero existe una fracción de la nota del curso que se asigna por concepto de trabajo en clases (asistencia y participación).

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 1.0 punto cada 4 horas, o fracción. La nota final de los laboratorios prácticos (**L**) está dada por el promedio de los 6 laboratorios. El profesor se reserva el derecho a modificar esta regla en casos excepcionales.

- Participación (20%): Durante el segundo módulo de la primera clase de cada laboratorio, los alumnos deberán resolver un problema que tiene directa relación con el laboratorio en curso. La participación se evaluará en base al problema propuesto realizado durante la sesión, y podrá tener notas 1.0, 4.0 ó 7.0. La nota final de las actividades de participación (**P**) está dada por el promedio de los 6 problemas resueltos.

- Asistencia (20%): 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 (**A**) se calculará como el promedio de las 6 notas de asistencia por laboratorio.

6 Exigencias de aprobación

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

$$\mathbf{F} = 0.6 \cdot \mathbf{L} + 0.2 \cdot \mathbf{P} + 0.2 \cdot \mathbf{A}$$

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

Fecha	Actividades	Tópicos
08/08	Introducción al curso	
22/08	Clase 1: L01*	Programación orientada a objetos
27/08	Ayudantía 1: L01	Programación orientada a objetos
29/08	Clase 2: L01	Programación orientada a objetos
05/09	Clase 3: L02*	Estructuras de datos
10/09	Ayudantía 2: L02	Estructuras de datos
12/09	Clase 4: L02	Estructuras de datos
26/09	Clase 5: L03*	Técnicas y algoritmos
01/10	Ayudantía 3: L03	Técnicas y algoritmos
03/10	Clase 6: L03	Técnicas y algoritmos
10/10	Clase 7: L04*	Bases de datos y archivos
15/10	Ayudantía 4: L04	Bases de datos y archivos
17/10	Clase 8: L04	Bases de datos y archivos
24/10	Clase 9: L05*	Análisis y visualización de datos
29/10	Ayudantía 5: L05	Análisis y visualización de datos
05/11	Ayudantía 6: L05	Análisis y visualización de datos
07/11	Clase 10: L05	Análisis y visualización de datos
14/11	Clase 11: L06*	Tópicos avanzados
19/11	Ayudantía 7: L06	Tópicos avanzados
21/11	Clase 12: L06	Tópicos avanzados

*: Clase con actividad de participación

Laboratorio	Publicación	Entrega
L01	19/08	01/09
L02	02/09	15/09
L03	16/09	06/10
L04	07/10	20/10
L05	21/10	10/11
L06	11/11	24/11

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.