PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2233 — Programación Avanzada

Programa de curso

Actualización: 14 de julio de 2023

Profesores: Hernán Valdivieso (hvaldivieso@ing.puc.cl)
Daniela Concha (daconcha@uc.cl)
Francisca Cattan (fpcattan@uc.cl)
Joaquín Tagle (jtagle2@uc.cl)
Francisca Ibarra (faibarra1@uc.cl)

Clases: Jueves, módulos 5 y 6
Ayudantías: Martes, módulo 4
Formato: Presencial
Requisitos: IIC1103 — Introducción a la Programación
Sitio web: github.com/IIC2233/Syllabus/

Introducción

El desarrollo de la computación ha hecho que estemos rodeados de dispositivos que ejecutan programas computacionales todo el tiempo. Un aspecto importante para ser partícipes del desarrollo de estas herramientas es comprender su funcionamiento y cómo construir desde programas simples a *software* más complejo. Este curso profundiza aspectos de la programación que se introdujeron en cursos anteriores, en particular en las estructuras y técnicas de diseño que permiten construir herramientas más complejas y prácticas.

Objetivo General y Competencias

A lo largo de este curso, el alumno desarrollará técnicas para diseñar, implementar, ejecutar y evaluar herramientas de *software* que resuelvan problemas algorítmicos a partir de especificaciones detalladas. El alumno será capaz de desarrollar construcciones avanzadas de programación orientada a objetos y estructuras de datos fundamentales, construir código robusto, construir interfaces gráficas, y utilizar conceptos como *threading*, serialización y paso de mensajes.

Al finalizar el curso, el estudiante será capaz de:

1. **Comprender técnicas básicas de mantención de código** incluyendo uso de guía de estilo, modularización y sistemas de manejo de versiones.
2. **Inferir un modelo de objetos** para resolver problemas realistas e **implementar esta solución** usando técnicas de programación orientada a objetos.
3. **Usar estructuras de datos básicas** para resolver problemas de programación.
4. **Utilizar objetos iterables** para resolver problemas de programación.
5. **Aplicar el concepto de *threading*** para la modelación de problemas de colas.
6. **Construir interfaces gráficas** funcionales utilizando bibliotecas apropiadas.
7. **Aplicar las formas de manejo de excepciones** en un programa, para construir código robusto.

8. **Implementar estructuras de datos basadas en nodos**, como listas ligadas y grafos.
9. **Utilizar el concepto de serialización** para construir codificadores y decodificadores.
10. **Utilizar el concepto de paso de mensajes** para construir una aplicación distribuida básica.

Contenidos

Fundamentos de programación

- **Estructuras de datos básicas:** tuplas, *named tuples*, *stacks*, colas, diccionarios, *sets*.
- **Programación orientada a objetos:** objetos, herencia, herencia múltiple, polimorfismo, clases abstractas.
- **Iterables:** objetos iterables; generadores; y funciones de mapeo, filtro y reducción.
- **Manejo de excepciones**: tipos de excepciones, control de excepciones.
- **Estructuras de datos basadas en nodos:** listas ligadas y grafos.

Herramientas de programación

- **Técnicas básicas de mantención de código:** concepto y uso herramientas de sistemas de manejo de versiones, uso de guías de estilo y modularización.
- ***Threading:*** concepto de pseudo-paralelismo, creación y sincronización de *threads*, concurrencia.
- **Interfaces gráficas:** introducción a las interfaces gráficas usando la librería PyQt6.
- **I/O:** manejo de *bytes*, serialización binaria, serialización en formato JSON.
- ***Networking:*** *sockets*, modelo cliente-servidor y paso de mensajes.
- ***Web services***: consumo y generación de API.

Metodología

El curso seguirá una metodología *blended* en la cual se utilizará el modelo de clase invertida (*flipped classroom*) con actividades de programación en clase, tareas individuales de programación, sesiones de apoyo en ayudantía y sesiones expositivas en clases.

La modalidad de la clase de cada semana será *flipped classroom*. En estas, se asume que los alumnos ya estudiaron los contenidos de manera previa a la clase, para luego aplicarlos en ella mediante actividades prácticas de programación o ejemplos aplicados que desarrollará el docente. Para este fin, el material de estudio se encontrará disponible en el sitio del curso (al menos) desde el viernes anterior a la clase, y consiste en apuntes donde se describen detalladamente los tópicos.

En el horario de cátedra, el profesor hará un repaso sobre el material de estudio a modo de introducción para la clase y para resolver dudas con respecto a los contenidos estudiados. Luego, se destinará el horario de cátedra para desarrollar una actividad donde se apliquen los nuevos conocimientos, o para implementar los contenidos con ejemplos aplicados. Tanto el profesor como ayudantes del curso estarán presentes durante toda la duración de esta actividad para resolver dudas y guiar la actividad.

El curso contará con ayudantías semanales con el fin de aplicar los contenidos antes de la sesión de discusión en clases. Es importante notar, que las sesiones de reforzamiento de contenidos también asumen revisión previa del material y no están orientadas a ser un reemplazo de estudio previo, sino que son una instancia de resolución de dudas, previa a la clase.

El detalle de la modalidad de cada clase estará indicado desde el inicio del semestre en el calendario del curso.

Evaluación

La evaluación será efectuada mediante actividades de programación, evaluaciones individuales escritas y tareas individuales de programación de mayor extensión.

Actividades de programación (AC) [10 %] Periódicamente se publicarán actividades que el estudiantado dispondrá de al menos 5 días para resolver y entregar. Para fomentar la metodología de *flipped classroom*, esta actividad puede ser discutida y realizada en grupos, pero cada estudiante deberá entregar su propio trabajo en la plataforma del curso. Además, se destinará un tiempo, durante las clases, para desarrollar y/o realizar consultas sobre dicha actividad de programación. En total el curso constará con seis (6) actividades en las fechas detalladas en el calendario del curso. Dado el carácter acumulativo del curso, cada actividad incluye el contenido de las semanas anteriores a menos que se explicite lo contrario, pero el foco será el contenido de la semana.

El objetivo de estas actividades es poner en práctica y evaluar el aprendizaje acumulado de cada estudiante mediante una actividad práctica. Cada trabajo será evaluado y se asignará un puntaje en función del nivel de logro alcanzado. La corrección de estas actividades serán automatizadas. Para esto, la actividad realizada por el estudiantado deberá cumplir con diversos *tests* de acceso privado. No obstante, se les entregará un conjunto de *tests* de acceso público junto a la actividad para que puedan probar su actividad.

Cálculo de nota: Cada actividad otorgará un máximo de cuatro (4) puntos y el puntaje total a obtener entre todas las actividades será de veinticuatro (24) puntos. Posteriormente, se transformará el puntaje obtenido en una nota. Quedará a criterio del cuerpo docente si al momento de calcular la nota se espera que el 7.0 sera con 24 puntos o con menos.

Dada la naturaleza de esta actividad que contempla tiempo en clases para su desarrollo, extensión de al menos 5 días y la posibilidad de realizarla en grupos para fomentar la metodología *flipped classroom*, no se podrá optar a actividad recuperativa o a la recorrección de estas.

Evaluaciones Escritas (EE) [30 %] El curso realizará dos (2) evaluaciones escritas durante el desarrollo de este: *midterm* y examen. Ambas evaluaciones serán desarrolladas con lápiz y papel, y buscan evaluar que cada estudiante interiorizó correctamente los contenidos básicos del curso, es capaz de aplicar los contenidos en diversos problemas, y es capaz de identificar posibles errores de código junto con proponer mejoras.

La primera evaluación escrita, **MIDTERM**, será a mediados del semestre y la segunda evaluación escrita, **EXAMEN**, será a finales del semestre. Ambas consistirán principalmente de preguntas de alternativas y/o desarrollo.

Cálculo de nota: La nota de las evaluaciones escritas se calcula como el promedio de ambas evaluaciones. Es decir

$$\mathbf{EE} = \frac{\mathbf{MIDTERM} + \mathbf{EXAMEN}}{2}$$

Uno de los requisitos de aprobación del curso es que **EE** debe ser mayor o igual a 3,95.

Tareas (T) [60 %] Se publicarán tres (3) tareas de programación las que deberán ser resueltas **individualmente** por cada estudiante. La dificultad, extensión y formato de corrección de cada tarea será informado al inicio del curso. La segunda tarea $\mathbf{T_2}$ contemplará una entrega parcial evaluada, correspondiente a un porcentaje del total de esa nota.

Cálculo de nota: La nota de cada tarea se especifica como $\mathbf{T_1}$, $\mathbf{T_2}$ y $\mathbf{T_3}$, y la nota ponderada total de tareas se calcula como:

$$\mathbf{T} = \frac{1 \times \mathbf{T_1} + 6 \times \mathbf{T_2} + 3 \times \mathbf{T_3}}{10}$$

Uno de los requisitos de aprobación del curso es que **T** debe ser mayor o igual a 3,95.

Política de atraso.

- Actividades: **No se aceptarán actividades entregadas fuera de plazo**. Toda actividad no entregada se les asignarán 0 puntos.
- Tareas: Se aceptarán entregas de estas evaluaciones **hasta 48 horas** después de la hora de entrega, aplicándose una **penalización a la nota máxima** dependiente de la cantidad de días de atraso.

  Las tareas entregadas con hasta un día (24 horas) de atraso tendrán nota máxima **6.0**, mientras que las tareas con hasta dos días (48 horas) de atraso tendrán nota máxima **4.0**. Cabe señalar que estas penalizaciones son a la nota máxima y **no son un descuento directo**, es decir, en caso de entregar tarde y obtener una calificación menor a la nota máxima penalizada, la nota obtenida no se verá afectada.

  En otras palabras, la nota final de cada tarea atrasada se calculará como:

$$\mathbf{T_i} = \text{mín}\left(\mathbf{T_i^a}, \texttt{techo}^{\mathbf{a}}\right)$$

  Donde $\mathbf{T_i^a}$ es la nota obtenida en la entrega atrasada y $\texttt{techo}^{\mathbf{a}}$ es la nota máxima dado los días de atraso **a** (definida anteriormente).
- Cupón: Este cupón le permite al estudiante disminuir la penalización del atraso de una tarea en 24 horas. De este modo, si el estudiante entrega con dos días de atraso, tiene la posibilidad de usar el cupón para que se le aplique la penalización correspondiente a solo un día y, de la misma manera, si entrega con un día de atraso, puede utilizar el cupón para eliminar del todo la penalización y acceder a la nota máxima. En consecuencia, este cupón **no extiende el plazo de entrega**, solo disminuye la penalización.

  Durante el semestre, cada estudiante dispondrá de dos (2) cupones que podrán ser utilizados en conjunto para una sola tarea (eliminando la penalización de dos días de atraso), o cada uno en tareas distintas.

Proceso de entrega. Para cada actividad, se indicará un archivo **.py** a entregar. Este deberá ser subido a la plataforma del curso: Canvas.

Para las tareas de programación, estas se entregan **únicamente** a través de un repositorio privado y personal del estudiante. Este se alojará en la plataforma **GitHub** y será provisto por el equipo docente a cada estudiante. **Este es el medio de entrega oficial y único del curso. Cada tarea especificará la carpeta en que debe entregarse, dentro de su repositorio.**

Proceso de recorrección. Luego de publicadas las notas de tareas y evaluaciones escritas, se dará un periodo de una semana para recibir solicitudes de recorrección. Cada solicitud debe estar debidamente justificada, y debe ser enviada por los canales que el curso disponga para este propósito.

Cabe destacar que en el proceso de recorrección, el cuerpo docente puede revisar completamente el trabajo enviado y no solo los ítems que se están apelando. Por lo cual, la nota puede subir, bajar o mantenerse si se considera que uno o más ítems no fueron corregidos como corresponde según la rúbrica y los contenidos aplicados.

Nota final y aprobación. Sea **NP** la nota de presentación del curso, que se calcula como:

$$\mathbf{NP} = \frac{6 \times \mathbf{T} + 3 \times \mathbf{EE} + 1 \times \mathbf{AC}}{10}$$

Para aprobar el curso, el estudiantado debe cumplir con:

- **NP** debe ser mayor o igual a 3,95.
- **EE** debe ser mayor o igual a 3,95.
- **T** debe ser mayor o igual a 3,95.

La nota final del curso **NF** se calculará como:

$$\mathbf{NF} = \begin{cases} \mathbf{NP} & \text{si cumple los 3 requisitos} \\ \text{mín}(\mathbf{NP}; 3{,}9) & \text{en otro caso.} \end{cases}$$

Todas las notas y promedios del curso serán calculadas con un redondeo a **dos decimales**, salvo la nota final del curso que se calculará con un redondeo a **un decimal**.

Integridad Académica

Cualquier situación de **falta a la ética** detectada en alguna evaluación tendrá como **sanción un 1,1 final en el curso**. Esto sin perjuicio de sanciones posteriores que estén de acuerdo a la Política de Integridad Académica de la Escuela de Ingeniería y de la Universidad, que sean aplicables al caso. Rige para este curso tanto la política de integridad académica del Departamento de Ciencia de la Computación (ver anexo) como el **Código de honor de la Escuela de Ingeniería**.

Debido a la naturaleza de la disciplina en la que se enmarca el curso, está permitido el uso de código escrito por una tercera parte, pero solo bajo ciertas condiciones y **siempre debe estar correctamente referenciado**, indicando la fuente de donde se obtuvo. En primer lugar, **se permite el uso de código encontrado en internet** u otra fuente de información similar, **siempre y cuando su autor sea externo al curso, y su publicación sea previa a la liberación del enunciado**. En segundo lugar, está permitido utilizar todo el código publicado en los repositorios oficiales del curso, como el presente en los contenidos o en las ayudantías. Por último, el **compartir o utilizar código correspondiente a una evaluación actual o pasada** del curso y el uso de herramientas de generación de código se encuentran estrictamente prohibidos y serán considerados como faltas a la ética.

Bibliografía

- K. Pichara, C. Pieringer. *Advanced Programming in Python*, disponible en **Amazon**.

Anexo: Política de integridad académica del Departamento de Ciencia de la Computación

Se espera los alumnos de la Escuela de Ingeniería de la Pontificia Universidad Católica de Chile mantengan altos estándares de honestidad académica, acorde al Código de Honor de la Universidad. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Académica publicado por la Dirección de Pregrado de la Escuela de Ingeniería (Disponible en SIDING, en la sección Pregrado/Asuntos Estudiantiles/Reglamentos/Reglamentos en Ingeniería/Integridad Académica).

Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente *política de integridad académica*. Todo trabajo presentado por un alumno para los efectos de la evaluación de un curso debe ser hecho **individualmente** por el alumno, **sin apoyo en material de terceros**. Por "trabajo" se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros.

En particular, si un alumno copia un trabajo, o si a un alumno se le prueba que compró o intentó comprar un trabajo, **obtendrá nota final 1,1 en el curso** y se solicitará a la Dirección de Pregrado de la Escuela de Ingeniería que no le permita retirar el curso de la carga académica semestral.

Por "copia" se entiende incluir en el trabajo presentado como propio, partes hechas por otra persona. En caso que corresponda a "copia" a otros alumnos, la sanción anterior se aplicará a todos los involucrados. En todos los casos, se informará a la Dirección de Pregrado de la Escuela de Ingeniería para que tome sanciones adicionales si lo estima conveniente.

Obviamente, está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, **siempre y cuando se incluya la referencia correspondiente**.

Lo anterior se entiende como complemento al Reglamento del Alumno de la Pontificia Universidad Católica de Chile (**http://admisionyregistros.uc.cl/alumnos/informacion-academica/reglamentos-estudiantiles**). Por ello, es posible pedir a la Universidad la aplicación de sanciones adicionales especificadas en dicho reglamento.