



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

## IIC2233 — Programación Avanzada Programa de curso

Clases: Jueves, módulos 5 y 6  
Formato: Presencial  
Requisitos: IIC1103 — Introducción a la Programación  
Sitio web: [github.com/IIC2233/Syllabus/](https://github.com/IIC2233/Syllabus/)

Sección	Profesor	E-mail	Ayudantía
1	Hernán Valdivieso	hfvaldivieso@uc.cl	Martes, módulo 4
2	Daniela Concha	daconcha@uc.cl	Martes, módulo 4
3	Francisca Ibarra	faibarra1@uc.cl	Martes, módulo 5
4	Lucas Van Sint Jan	lfvansintjan@uc.cl	Martes, módulo 5
5	Francisca Cattán	fpcattan@uc.cl	Martes, módulo 6

### 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 y aplicar técnicas básicas de mantención de código** incluyendo uso de guía de estilo, modularización y sistemas de manejo de versiones (Git).
2. **Inferir un modelo de objetos** para resolver problemas realistas e **implementar esta solución** usando técnicas de programación orientada a objetos (OOP).
3. **Utilizar y diseñar objetos iterables** para resolver problemas de programación.
4. **Conocer el Paradigma de Programación Funcional** a través de las herramientas de Python.
5. **Comprender y aplicar el concepto de *threading*** para la modelación de problemas de concurrencia.
6. **Construir interfaces gráficas** funcionales utilizando módulos apropiados.

7. **Utilizar las formas de manejo de excepciones** en un programa, para construir código robusto.
8. **Utilizar el concepto de serialización** para construir codificadores y decodificadores.
9. **Utilizar el concepto de paso de mensajes** para construir una aplicación distribuida básica.
10. **Conocer** diferentes herramientas, prácticas y/o algoritmos actuales en el contexto de desarrollo de software.

## Contenidos

### Fundamentos de programación

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

### 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-parallelismo, creación y sincronización de *threads*, concurrencia.
- **Interfaces gráficas:** introducción a las interfaces gráficas usando la librería PyQt5. Uso de señales para comunicación *backend* y *frontend*.
- **I/O:** manejo de *bytes*, serialización binaria, serialización en formato JSON.
- **Networking:** *sockets*, modelo cliente-servidor y paso de mensajes, entre otros.
- **Aplicaciones:** *Web services*, consumo y generación de API, expresiones regulares, entre otros.

## Metodología

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

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 y las actividades se encontrarán disponibles en el sitio del curso (al menos) desde el viernes anterior a la clase. El material de estudio 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 resolverá dudas con respecto a los contenidos estudiados. Luego, se destinará el resto del horario de cátedra para desarrollar la actividad previamente publicada en la cual se aplican los nuevos conocimientos, o bien implementan 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 repaso 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.

## Evaluación

La evaluación será efectuada mediante tres (3) instrumentos: actividades de programación, evaluaciones individuales escritas y tareas individuales de programación de mayor extensión. Las fechas son las siguientes:

### ■ Evaluaciones Escritas

- **Midterm:** 3 octubre a las 17:30
- **Examen:** 5 diciembre a las 13:30

### ■ Tareas

- **T<sub>1</sub>:** 13 agosto - 26 agosto (9 días hábiles)
- **T<sub>2</sub>:** 30 agosto - 13 septiembre (11 días hábiles)
- **T<sub>3</sub>:** 1 octubre - 15 octubre (11 días hábiles)
- **T<sub>4</sub>:** 6 noviembre - 20 noviembre (11 días hábiles)

**Actividades de programación (AC) [10 %]** Periódicamente se publicarán actividades que el estudiantado dispondrá de al menos 4 días hábiles 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 desarrollarla y/o realizar consultas. Se publicarán un total de cinco (5) actividades en las fechas detalladas en el calendario del curso. Además, al inicio del semestre se publicará una Actividad 0, de carácter formativo no evaluado. Dado el carácter acumulativo del curso, cada actividad incluye el contenido de las semanas anteriores a menos que se explicita lo contrario, pero el foco será el contenido de la semana.

Publicación y entrega: Se subirá la actividad el día viernes anterior a la clase. Esta debe ser entregada como máximo hasta el día jueves de la clase a las 23:59. Dada la naturaleza de esta actividad que contempla tiempo en clases para su desarrollo, extensión de al menos 4 días hábiles y la posibilidad de realizarla en grupos para fomentar la metodología *flipped classroom*, no se podrá optar a actividad recuperativa o a la corrección de estas.

Obtención de puntaje: El objetivo de estas actividades es poner en práctica y evaluar el aprendizaje acumulado de cada estudiante. Cada trabajo será evaluado y se asignará un puntaje en función del nivel de logro alcanzado. La corrección de estas actividades serán automatizadas mediante dos tipos de casos de prueba o *tests*: privados y públicos. El puntaje obtenido se reflejará en la cantidad total de *tests* de acceso privado correctos. Se les entregará un conjunto de *tests* de acceso público junto a la actividad para que puedan probarla.

Cálculo de nota: Cada actividad otorgará un máximo de cuatro (4) puntos y el puntaje total a obtener entre todas las 5 actividades será de veinte (20) puntos. Posteriormente, se transformará el puntaje obtenido en una nota. El 7.0 se alcanzará al llegar a 12 puntos. La Actividad 0 no será evaluada puntaje, y no contará en el cálculo de puntaje.

**Evaluaciones Escritas (EE) [30 %]** El curso realizará dos (2) evaluaciones presenciales escritas a desarrollar con lápiz y papel: *midterm* y examen. Ambas evalúan 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:

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

Uno de los requisitos de aprobación del curso es que **EE** debe ser mayor o igual a 3,5.

**Tareas (T) [60 %]** Se publicarán cuatro (4) tareas de programación las que deberán ser resueltas **individualmente** por cada estudiante. Sólo se permitirá discutir sobre la evaluación y su solución con el cuerpo docente, y los casos de falta a la integridad académica entre el estudiantado serán sancionados de acuerdo a los lineamientos de la Universidad. La dificultad, extensión y formato de corrección de cada tarea será informado al inicio del curso.

Cálculo de nota: La nota de cada tarea se especifica como **T<sub>1</sub>**, **T<sub>2</sub>**, **T<sub>3</sub>** y **T<sub>4</sub>**, y la nota ponderada total de tareas se calcula como:

$$T = \frac{1 \times T_1 + 2 \times T_2 + 2 \times T_3 + 3 \times T_4}{8}$$

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

**Política de atraso.** Cada instrumento de evaluación cuenta con políticas de atraso particulares.

- **Actividades:** **No se aceptarán actividades entregadas fuera de plazo.** A toda actividad no entregada se le asignará 0 puntos.
- **Tareas:** Se aceptarán entregas **hasta 48 horas** después de la hora de oficial, aplicándose una **penalización a la nota máxima** dependiente de la cantidad de días de atraso desde la hora de entrega oficial. Toda entrega atrasada debe ser notificada vía formulario que se publicará durante el desarrollo de la evaluación. En caso no de notificar, se revisará lo último entregado dentro del plazo original.

Las tareas entregadas con hasta 24 horas (un día) de atraso tendrán nota máxima **6.0**, mientras que las tareas con hasta 48 horas (dos días) 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:

$$T_i = \min(T_i^a, \text{techo}^a)$$

Donde **T<sub>i</sub><sup>a</sup>** es la nota obtenida en la entrega atrasada y **techo<sup>a</sup>** 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.

**Inasistencias.** La no entrega de cualquier instrumento de evaluación implicará la asignación de nota 1,0. Sólo las evaluaciones escritas tendrán posibilidad de justificar máximo **una** inasistencia durante el semestre. Se debe justificar ante la Dirección de Pregrado correspondiente, quienes entregarán una carta de inasistencia. Finalmente, es responsabilidad del estudiante enviar esta carta vía correo electrónico a la coordinación del curso. El plazo para justificar una inasistencia es de 5 días normales (incluyendo días no hábiles) desde ocurrida la inasistencia. En el caso de tener licencia médica, el tiempo comienza a correr desde el reintegro a la universidad. La recuperación de la evaluación justificada será discutida por el equipo docente.

**Proceso de entrega.** Tanto para las actividades como 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 evaluación especificará el nombre de la carpeta en que debe entregarse, dentro de su repositorio.**

**Proceso de corrección.** Luego de publicadas las notas de tareas y evaluaciones escritas, se dará un periodo de una semana para recibir solicitudes de corrección vía formulario. 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 corrección, 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:

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

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 3,5.
- **T** debe ser mayor o igual a 3,95.

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

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

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 semestre actual del curso, y su publicación sea previa a la liberación del enunciado**. Este debe ser debidamente referenciado y su uso debe ser producto de un proceso de investigación, integración y adaptación. 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.

Está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, **siempre y cuando se incluya la referencia correspondiente**. Además de estar debidamente citado, su uso debe ser producto de un proceso de investigación, integración y adaptación.

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.