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

IIC2233 — Programación Avanzada
Programa de curso 2025-1

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

Sección	Profesor	E-mail	Ayudantía
1	Pablo Araneda	pharaneda@uc.cl	Martes, módulo 5
2	Daniela Concha	daconcha@uc.cl	Martes, módulo 5
3	Francisca Ibarra	faibarra1@uc.cl	Martes, módulo 6
4	Lucas Van Sint Jan	lfvansintjan@uc.cl	Martes, módulo 6
5	Francisca Cattan	fpcattan@uc.cl	Martes, módulo 4

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 técnicas para construir código robusto** en un programa, como las formas de manejo de excepciones, entre otros.
8. **Utilizar el concepto de serialización** para construir codificadores y decodificadores.
9. **Comprender y aplicar infraestucturas y tecnicas básicas** para crear y mantener **aplicaciones distribuidas**, asegurando el intercambio de datos y su persistencia, entre otros.
10. **Conocer** diferentes herramientas, prácticas y/o algoritmos actuales en el contexto de desarrollo de software.
11. **Comprender** los resultados obtenidos al ejecutar pruebas unitarias.

Contenidos

Fundamentos de programación

- **Estructuras de datos básicas:** listas, tuplas, *named tuples*, *stacks*, colas, diccionarios, *sets*, empaquetado y desempaquetado de datos.
- **Programación orientada a objetos:** clases, objetos, herencia, polimorfismo, clases abstractas.
- **Iterables:** objetos iterables, objetos iteradores, generadores y listas ligadas.
- **Programación funcional:** paradigma, funciones de mapeo, filtro y reducción.
- **Manejo de errores**: análisis de errores, 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-paralelismo, concurrencia, creación y sincronización de *threads*.
- **Interfaces gráficas:** introducción a las interfaces gráficas usando la librería PyQt. Uso de señales para comunicación *backend* y *frontend*.
- **I/O:** *bytes*, serialización binaria, serialización en formato JSON.
- ***Networking*:** protocolos y arquitecturas de comunicación, *sockets*, modelo cliente-servidor e intercambio de información, sistemas distribuidos, 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, tareas individuales de programación y evaluaciones escritas.

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. El material de estudio consiste en apuntes donde se describen detalladamente los tópicos.

En el horario de cátedra, el profesor hará un breve repaso sobre el material de estudio a modo de introducción para la clase y comentará con los estudiantes respecto a los contenidos estudiados. Luego, se destinará el resto del

horario de cátedra para desarrollar la actividad publicada durante la misma clase en la cual se aplican los nuevos conocimientos, o bien implementar los contenidos junto al profesor con ejemplos aplicados. Tanto el profesor como los ayudantes del curso estarán presentes durante toda la duración de esta actividad para resolver dudas y guiarla.

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 cuatro (4) instrumentos: actividades de programación, controles de salida, evaluaciones individuales escritas, y tareas individuales de programación de mayor extensión. Las fechas son las siguientes:

- **Evaluaciones Escritas**
  - **Midterm**: 11 abril a las 17:30
  - **Examen**: 1 julio a las 13:30
- **Tareas**
  - $\mathbf{T_1}$: 17 marzo - 26 marzo (8 días hábiles)
  - $\mathbf{T_2}$: 4 abril - 17 abril (10 días hábiles)
  - $\mathbf{T_3}$: 16 mayo - 30 mayo (10 días hábiles)
  - $\mathbf{T_4}$: 9 junio - 23 junio (10 días hábiles)

Nota de Cátedra (C) [20%] Para evaluar el método *flipped classroom*, se realizaran doce (12) evaluaciones en el horario de cátedra. Estas serán del tipo **Actividades de programación (AC)** y **Controles de Salida (CS)**, las cuales serán explicadas con mayor detalle en sus secciones correspondientes.

Cálculo de nota: Cada actividad de programación y control de salida otorgará cuatro (4) puntos al puntaje de **AC** y **CS**, respectivamente con la única excepción de la primera actividad llamada **AC0** la cual no aportará puntaje. Sumando los dos puntajes, cuyo valor sumado puede llegar a 44 puntos, se calculará la nota de Cátedra (**C**) mediante la siguiente fórmula:

$$\mathbf{C} = min(7, \frac{\mathbf{AC} + \mathbf{CS}}{36} \times 6 + 1)$$

Actividades de programación (AC) Con el fin de controlar el que se esté usando la metodología de *flipped classroom*, periódicamente se publicarán actividades que el estudiantado deberá resolver de forma individual durante la clase. Además, durante las clases se permitirá realizar consultas. Se publicarán un total de nueve (9) 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.

Publicación y entrega: Se subirá la actividad durante el periodo de la clase. Esta debe ser entregada como máximo a las 17:20 del mismo día. Dada la naturaleza de esta actividad que contempla tiempo en clases para su desarrollo, y para fomentar la metodología *flipped classroom*, no se podrá optar a actividad recuperativa o a la recorrección 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, a excepción de la primera actividad llamada **AC0** que no aportará puntaje, otorgará cuatro (4) puntos dando a que el puntaje total a obtener entre todas las 9 actividades será de treinta y dos (32) puntos. Posteriormente, el puntaje aportará a la nota de Cátedra (**C**).

Controles de Salida (CS) En las clases en las que no haya actividad, se realizará una Experiencia, donde se aplicará un control de salida en los últimos 15 minutos de la clase. Se realizarán un total de tres (3) experiencias en las fechas detalladas en el calendario del curso. Para fomentar la metodología de *flipped classroom*, la experiencia será expuesta frente a todo el curso, permitiendo la discusión sobre la forma de abordar el problema propuesto. Dado el carácter acumulativo del curso, cada experiencia incluye el contenido de las semanas anteriores a menos que se explicite lo contrario, pero el foco será el contenido de la semana.

Publicación y entrega: El control se realizará en los últimos 15 minutos de la clase, es decir, este será subido en la plataforma Canvas a las 17:05 y cerrará a las 17:20. Dada la naturaleza de la Experiencia que contempla tiempo en clases para su desarrollo, y discusión sobre la misma para fomentar la metodología *flipped classroom*, no se podrá optar a control recuperativo o a la recorrección de estos. El material utilizado durante la clase será subido posterior a la realización del control de salida.

Obtención de puntaje: El objetivo de la experiencia es poner en práctica y evaluar el aprendizaje de cada estudiante. Cada respuesta del control será evaluada y se asignará un puntaje en función del nivel de logro alcanzado. La corrección de estos controles serán automatizadas al ser de alternativas. El puntaje obtenido se reflejará en la cantidad total de respuestas correctas.

Cálculo de nota: Cada control otorgará cuatro (4) puntos y el puntaje total a obtener entre todos los 3 controles será de doce (12) puntos. Posteriormente, el puntaje aportará a la nota de Cátedra (**C**).

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:

$$\mathbf{EE} = \frac{4 \times \mathbf{MIDTERM} + 6 \times \mathbf{EXAMEN}}{10}$$

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

Tareas (T) [50 %] 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 $\mathbf{T_1}$, $\mathbf{T_2}$, $\mathbf{T_3}$ y $\mathbf{T_4}$, y la nota ponderada total de tareas se calcula como:

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

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.
- Controles: **No se aceptarán controles entregados fuera de plazo**. A todo control no entregado 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:

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

  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.

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. Para los controles de salida, estos se responderán en un cuestionario presente en el Canvas del curso. 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. A excepción de los controles de salida, **GitHub es el medio de entrega oficial y único del curso, donde cada evaluación especificará el nombre de la carpeta en que debe entregarse dentro del repositorio del estudiante.**

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 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 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:

$$\mathbf{NP} = \frac{5 \times \mathbf{T} + 3 \times \mathbf{EE} + 2 \times \mathbf{C}}{10}$$

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:

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

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 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.