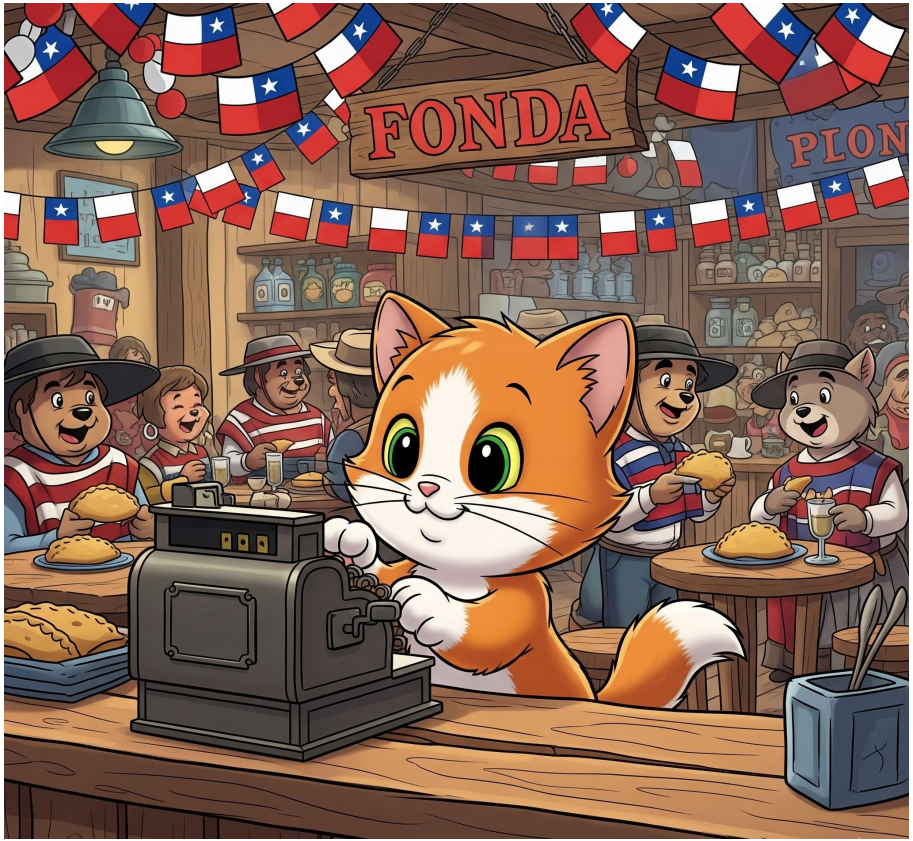
Programación Avanzada
IIC2233 2025-2

Cristian Ruz - Pablo Araneda - Francisca Ibarra - Tamara Vidal - Daniela Concha



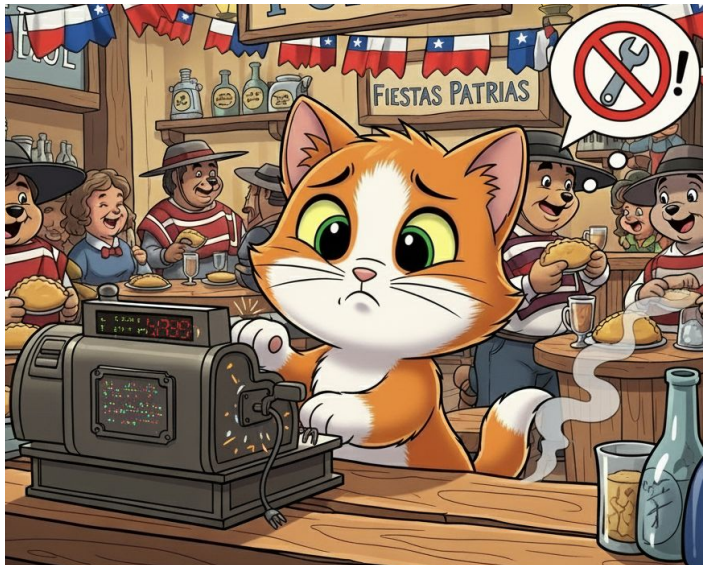
Experiencia 2

Excepciones

Experiencia 2: ¿Qué vamos a hacer?

1. Veremos un código que hace uso de un **módulo externo**, cuyo código fuente es **desconocido** y del cual solo tenemos acceso a su **documentación**.
2. Corregiremos los errores presentes en el código base para poder ejecutar el flujo completo del código sin detener la ejecución.
3. Para cada caso, discutiremos si hacer la corrección del código con un bloque **`if`**/**`else`** o un bloque **`try`**/**`except`**.

La DCCaja de Cuchito



Con las Fiestas Patrias acercándose, **Cuchito** el gato decidió crear su propia Fonda en Python.

Para ahorrar tiempo, compró ~~en Temu~~ una caja registradora para llevar las cuentas.

Al usarla, se dio cuenta de que tenía algunos bugs…

Como esta semana vamos a ver Excepciones, nos pide que lo ayudemos a corregir la caja registradora!

¿Qué esperamos haga nuestro programa?

Como mínimo, deberíamos lograr que nuestro programa pueda:

- Cargar datos de los productos disponibles para vender a partir de un archivo encriptado, ignorando los productos que causen problemas en la caja registradora.
- Procesar las compras de los clientes, indicando cuando no es posible adquirir un producto.
- Cerrar la caja al final del día, obteniendo el resumen de compras del día.

Para lograrlo, tenemos…

El módulo “`dccaja.pyc`”

Módulo externo **compilado** (es decir, no podemos fácilmente abrir el archivo y ver su código fuente) que contiene la implementación de la clase **`CajaRegistradora`.**

La clase posee los siguientes métodos públicos que puedes usar:

- **`ingresar_producto`**`(self, producto: str, precio: float) -> None:`
  Guarda un producto en la caja, y le asocia un precio.
- **`procesar_compra`**`(self, nombre_cliente: str,`
  `lista_productos: list, rut: str) -> None:`
  Procesa la compra del cliente, la añade al historial de la caja y genera un archivo **.txt** con la boleta de la compra.

El módulo “`dccaja.pyc`”

Módulo externo **compilado** (es decir, no podemos fácilmente abrir el archivo y ver su código fuente) que contiene la implementación de la clase **`CajaRegistradora`.**

La clase posee los siguientes métodos públicos que puedes usar:

- **`mostrar_productos`**`(self) -> None:`
  Muestra los productos ingresados.
- **`cuadrar_caja`**`(self) -> None:`
  Cierra la caja por el día, dando estadísticas de las compras del día.
- **`cerrar_caja`**`(self) -> None:`
  Cierra los registros de la caja para ese día y activa el siguiente día.

El archivo “`productos.topsecret`”

Es un archivo que contiene un **listado de los productos (“`nombre, precio`”)** que ofrecerá tu Restaurante.

Está almacenado en un formato (*bytes*) que no permite ver ni editar manualmente de forma rápida sus contenidos 👀, esto para que no puedas saber tan fácilmente qué productos vienen al restaurante.

(De todas formas, no te preocupes, el método que decodifica su contenido ya viene implementado en esta experiencia.)

Finalmente, **existen productos cuyo valor no es del formato adecuado para almacenar en la `CajaRegistradora`**. Por lo que durante el trabajo deberemos lidiar con estos casos.

El archivo “`fonda.py`”

El archivo a modificar y ejecutar en esta experiencia. Contiene una instancia de **Fonda**, clase que deberemos editar para hacer que la fonda funcione.

Posee como atributos:

- **`nombre`:** El nombre del restaurante.
- **`caja`:** Una instancia de la clase **`CajaRegistradora`.**

El archivo “`fonda.py`”

El archivo a modificar y ejecutar en esta experiencia. Contiene una instancia de **Fonda**, clase que deberemos editar para hacer que el restaurante funcione.

Posee como métodos:

- **`_desencriptar_archivo`**`(self, ruta_archivo: str) -> list:`
  Desencripta el archivo de productos y retorna una lista de *strings* con el formato “`producto,precio`”.
  NO DEBES MODIFICAR ESTE ARCHIVO.
- **`cargar_inventario`**`(self, ruta_productos) -> None:`
  Carga los productos a la caja, a partir de un archivo de productos.

El archivo “`fonda.py`”

El archivo a modificar y ejecutar en esta experiencia. Contiene una instancia de **Fonda**, clase que deberemos editar para hacer que el restaurante funcione.

Posee como métodos:

- **`cliente_comprar`**`(self, cliente: dict) -> list:`
  Procesa la lista de compras de un cliente.
- **`cerrar_por_el_dia`**`(self) -> None:`
  Da las estadísticas de las compras diarias y cierra la caja por el día.

Vamos a Programar

Vamos a programar la **Parte 1:**

“Cargar los datos de los productos en la Caja Registradora de la Fonda”

Vamos a Programar

Vamos a programar la **Parte 2:**

“Lograr que un cliente pueda adquirir su lista de compras, generando una boleta. Además, avisar cuando un producto de la lista no pudo ser adquirido.”

Vamos a Programar

Vamos a programar la **Parte 3:**

“Obtener estadísticas de las compras del día y cerrar el día.”

Programación Avanzada
IIC2233 2025-2

Cristian Ruz - Pablo Araneda - Francisca Ibarra - Tamara Vidal - Daniela Concha