M4_AE5_ABPRO-Ejercicio grupal

Paso 1: Conceptualización y Análisis

1. ¿Qué es una excepción en programación y por qué es importante manejarla correctamente?
Una excepción es un error que ocurre durante la ejecución de un programa y que interrumpe su flujo normal.Manejar excepciones es importante porque permite:

- Evitar que el programa se detenga abruptamente.
- Mostrar mensajes claros al usuario.
- Controlar situaciones inesperadas (como datos incorrectos, archivos inexistentes, conexiones fallidas).
- Hacer que el sistema sea más robusto y confiable.

2. ¿Cuáles son los tipos de excepciones más comunes?
En Python, algunas excepciones frecuentes son:

- ValueError – Cuando un valor no tiene el formato esperado.
- TypeError – Cuando se usa un tipo de dato incorrecto.
- IndexError – Acceso a un índice inexistente en una lista.
- KeyError – Cuando una clave no existe en un diccionario.
- ZeroDivisionError – Intentar dividir por cero.
- FileNotFoundError – Archivo no encontrado.
- PermissionError – Falta de permisos.
- AttributeError – Intentar acceder a un atributo inexistente.

3. ¿Cómo funciona la sentencia try/except y cuándo se debe utilizar?
La estructura try/except permite intentar ejecutar un bloque de código que podría fallar.
Se debe usar cuando se desea:

- Evitar que errores detengan el programa.
- Detectar y controlar fallas esperadas (por ejemplo, entrada de usuario incorrecta).
- Dar mensajes precisos según el tipo de error.

4. ¿Cómo se pueden capturar múltiples excepciones en un solo bloque de código?
Hay dos formas:

- Capturar varias excepciones en una tupla

```
try:
    operación_riesgosa()
except (ValueError, TypeError):
    print("Hubo un error de tipo o valor.")
```

- Usar múltiples except

```
try:
    operación_riesgosa()
except ValueError:
    print("Error de valor.")
except TypeError:
    print("Error de tipo.")
```

5. ¿Qué es el uso de raise en Python y cómo se utiliza para generar excepciones en validaciones?
`raise` permite lanzar una excepción de forma manual para detener la ejecución cuando algo no cumple una regla. Se usa cuando:

- Se deben validar datos.
- Se quiere evitar que el programa continúe con información incorrecta.
- Se aplican reglas de negocio (como evitar reservas inválidas).

6. ¿Cómo se pueden definir excepciones personalizadas y en qué casos sería útil?
Se crean heredando de Exception. Son útiles cuando:

- Se necesita representar errores específicos del negocio.
- Se quiere identificar claramente un tipo de error propio del sistema.
- Se requiere un manejo diferenciado según la situación.

7. ¿Cuál es la función de finally en el manejo de excepciones?
`finally` ejecuta código siempre, ocurra o no una excepción. Se usa para:

- Liberar recursos
- Cerrar archivos
- Finalizar conexiones
- Asegurar que el proceso quede en un estado consistente

8. ¿Cuáles son algunas acciones de limpieza que deben ejecutarse después de un proceso que puede generar errores?

- Cerrar archivos (`file.close()`)
- Cerrar conexiones a bases de datos
- Liberar memoria o buffers
- Desconectar APIs
- Guardar logs de errores
- Confirmar o revertir transacciones