Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2021-2

16 de Septiembre de 2021
Actividad Formativa

Actividad Formativa 2

Excepciones

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AF2/`
- **Hora del *push*:** 16:40

Importante: Antes de comenzar, comprueba que Git este funcionando correctamente en tu repositorio privado. Para esto, **sube los archivos base de la actividad de inmediato** (*add*, *commit*, *push*). Se espera que en esta actividad (así como en las demás actividades y tareas) utilices Git a lo largo de **todo tu desarrollo** como una herramienta, no sólo como un método de entrega. Es por esto que recomendamos enfáticamente que vayas subiendo tus cambios constantemente (***push***), ya que **problemas de último minuto** relacionados con la entrega y Git **no serán considerados**.

Introducción

Después de dos años de clases online, les ayudantes de Programación Avanzada finalmente tendrán la oportunidad de conocerse en persona. Para eso, tendrán la mejor fiesta en la historia de la universidad: el **DCCarrete**. Todes les estudiantes de la universidad se mueren por atender, pero únicamente les afortunades recibieron una invitación.

Este evento fue organizado durante meses por les ayudantes jefes, planeando hasta el más mínimo detalle con mucho cariño. Sin embargo, en el mismo día del **DCCarrete**, hackers (enojades por no haber sido invitades) corrompieron el archivo con la lista oficial de asistentes. ¡Solo tú, estudiante de Programación Avanzada y experte en excepciones, puedes salvar la fiesta, evitar que personas cuelen y asegurar un ambiente libre de COVID-19!

Flujo del programa

Para esta actividad, deberás utilizar todos tus conocimientos de excepciones y así poder salvar al gran **DCCarrete**. Lo primero que debes hacer es revisar los datos corruptos, verificando que está todo en orden y corrigiendo lo que no. Cuando logres esto, la lista de invitades volverá a quedar perfecta.

Luego, deberás ayudarnos con el control de riesgos de COVID-19 en el evento. A pesar de que ya no estamos en cuarentena, la pandemia sigue, por lo que necesitamos asegurarnos que ningún invitade está enferme antes de dejarle entrar. Para ello, deberás crear una excepción personalizada, que verifique los posibles síntomas en les invitades.

¡Mucha suerte! Estamos contando contigo.

Archivos

- `data/lista_oficial.csv`: Este archivo contiene los datos de todes les invitades al **DCCarrete**. Pero, cuidado, ¡este archivo fue hackeado y alterado! Es posible que existan datos erróneos. No debes modificarlo
  Su formato es:

  ```
  nombre,edad,pase_movilidad,temperatura,tos,dolor_cabeza,mail
  ```

  Estos valores coinciden con los atributos de la clase `Invitade` por lo que pueden asumir que su tipo de dato coincide con el del atributo homónimo según lo indicado más abajo.
- `data/asistentes.txt`: Este archivo contiene los nombres de todas las personas esperando para entrar al **DCCarrete**. Eso sí, no todes fueron invitades. No debes modificarlo
- `invitades.py`: Este archivo contiene la clase `Invitade` que se explica más adelante. No debes modificarlo
- `cargar_archivos.py`: Este archivo contiene dos funciones para cargar los datos necesarios para la actividad (desde `lista_oficial.csv` y `asistentes.txt`). No debes modificarlo
- `verificar_invitades.py`: Este archivo contiene funciones para verificar y corregir los datos de les ayudantes invitades y también para identificar a asistentes colados. Deberás trabajar en este archivo durante la primera sección **Parte 1: Levantar y capturar excepciones**. Debes modificarlo
- `excepciones_covid.py`: Este archivo contiene la clase `RiesgoCovid`, utilizada en la segunda sección de la actividad: **Parte 2: Excepción personalizada**. Debes modificarlo
- `permitir_entrada.py`: Este archivo permite la entrada de invitades, verificando que se cumplan las condiciones para un **DCCarrete** seguro. Debes modificarlo
- `main.py`: Este es el archivo principal a ejecutar. El objetivo de este código es cargar los datos, corregir los errores de información de los ayudantes invitades, identificar colados y encontrar posibles riesgos de coronavirus. Por último, muestra la lista de les invitades aceptades. No debes modificarlo

Clase Invitade

Tendrás que trabajar con los objetos de la clase `Invitade`, que se encuentra definida en `invitades.py`. Esta clase **ya está implementada** y tiene los siguientes atributos, algunos de los cuales deberás revisar y posiblemente corregir mediante excepciones en la **Parte I**.

- `nombre`: Un `str` que corresponde al nombre del invitade.
- `edad`: Un `int` que corresponde a la edad del invitade. **Debe ser revisado.**

- `mail`: Un `str` que corresponde al mail del invitade. **Debe ser revisado.**
- `pase_movilidad`: Un `bool` que indica si el invitade tiene o no pase de movilidad. Cuidado aquí! Algunos de estos no son `bool`, debido a los hackers. **Debe ser revisado.**
- `temperatura`: Un `float` que corresponde a la temperatura del invitade.
- `tos`: Un `bool` que indica si el invitade presenta tos.
- `dolor_cabeza`: Un `bool` que indica si el invitade presenta dolor de cabeza.

Parte I: Levantar y capturar excepciones

En esta parte deberás manejar distintos tipos de excepciones para poder llevar a cabo correctamente el **DCCarrete**. Para esto deberás completar las siguientes funciones que se encuentran en el archivo `verificar_invitades.py` y utilizar `try/except` o levantar excepciones con `raise` donde corresponda.

Importante: Para esta parte no se considerará correcto el uso de `except` sin argumentos o de `except Exception`, sino que debes identificar el tipo de error y trabajar con este.

Hint: Se espera que puedas reconocer cuando utilizar *ValueError*, *TypeError* y *KeyError*.

- `def verificar_edad(invitade: Invitade):` **Debes modificarlo**

  Recibe una instancia de la clase `Invitade` y deberás verificar que el valor de su edad sea positiva. Si no se cumple esta condición, debes **levantar el error que corresponda** con un mensaje que lo indique, de la forma:

```
f"Error: la edad de {invitade.nombre} es negativa"
```

- `def corregir_edad(invitade: Invitade):` **Debes modificarlo**

  Recibe una instancia de la clase `Invitade`. Usando la función anterior, debes verificar que la edad del invitade sea positiva. Si **capturas** un error debes imprimirlo y corregir el problema utilizando el valor absoluto. En caso de haber corregido el valor, debes imprimir un mensaje avisando que el error ha sido corregido, de la forma:

```
print(f"El error en la edad de {invitade.nombre} ha sido corregido")
```

- `def verificar_pase_movilidad(invitade: Invitade):` **Debes modificarlo**

  Recibe una instancia de la clase `Invitade` y deberás verificar que el pase de movilidad sea de tipo `bool`. Si no se cumple esto, se debe **levantar el error que corresponda** con un mensaje que indique que el pase de movilidad no es un bool, como el siguiente:

```
f"Error: el pase de movilidad de {invitade.nombre} no es un bool"
```

- `def corregir_pase_movilidad(invitade: Invitade):` **Debes modificarlo**

  Recibe una instancia de la clase `Invitade`. Usando la función anterior, debes verificar que el pase de movilidad sea tipo `bool`, considerando que **solo fueron hackeados los ayudantes con pase**. Por

esto, solo en caso de que el tipo no sea `bool`, debes cambiar este atributo a `True`. Si **capturas** un error debes imprimirlo y corregir el problema. En caso de haber corregido el valor, debes imprimir un mensaje avisando que el error ha sido corregido, de la forma:

```
print(f"El error en el pase de movilidad de {invitade.nombre} ha sido corregido")
```

- `def verificar_mail(invitade: Invitade):` **Debes modificarlo**

  Recibe una instancia de la clase `Invitade` y deberás verificar que el mail del invitade esté en el formato correcto, ya que les hackers dieron vuelta algunos mails. Es decir, verificar que el mail tenga la forma `nombre@uc.cl` y **no** `uc@nombre.cl`. Si no se cumple esta condición, debes **levantar el error que corresponda** con un mensaje que indique el error, de la forma:

```
f"Error: El mail de {invitade.nombre} no está en el formato correcto"
```

- `def corregir_mail(invitade: Invitade):` **Debes modificarlo**

  Recibe una instancia de la clase `Invitade`. Usando la función anterior, debes verificar que el mail esté escrito en la forma correcta y no invertido, es decir, en el formato `nombre@uc.cl` y no `uc@nombre.cl`. Si **capturas** un error debes imprimirlo y corregir el mail, escribiendolo en el formato que corresponde. Al terminar de arreglarlo, debes imprimir un mensaje indicando que el mail fue corregido, de la forma:

```
print(f"El error en el mail de {invitade.nombre} ha sido corregido")
```

- `def dar_alerta_colado(nombre_asistente: str, diccionario_invitades: dict):` **Debes modificarlo**

  Recibe un `str` que representa el nombre de un asistente que está esperando para entrar al carrete y un `dict` en que sus llaves corresponden a los nombres de los invitades y sus valores a las instancias de estos invitades. Deberás obtener la instancia de asistente con `nombre_asistente` en `diccionario_invitades` y **utilizando excepciones**, **capturar** un posible error en caso de que el nombre no esté en el diccionario, imprimiendo el mensaje del error y otro mensaje como el siguiente:

```
print(f"Error: {nombre_asistente} se está intentando colar al carrete")
```

  En caso de que sí este en el diccionario, deberás crear una variable `asistente` que tenga la instancia de asistente y deberás imprimir el siguiente mensaje:

```
print(f"{asistente.nombre} esta en la lista y tiene edad {asistente.edad}")
```

Parte II: Excepción personalizada

En esta segunda parte deberás asegurarte que ningún invitade tenga COVID-19. Para ello, debes usar la excepción personalizada `RiesgoCovid` (ubicada en el archivo `excepciones_covid.py`), que se encargará de revisar los datos arreglados de la lista oficial de ayudantes invitados.

Clase RiesgoCovid

La clase `RiesgoCovid` corresponde a una **excepción personalizada** que permite detectar posibles casos COVID de acuerdo a los síntomas que presentan los invitades al **DCCarrete**. Esta tiene el método `alerta_de_covid` que indica el tipo de síntoma que presenta el invitade.

Debes asegurarte que `RiesgoCovid` sea una excepción personalizada.

- `def __init__(self, sintoma: str, nombre_invitade: str)`: Recibe dos `str`. El primero de ellos es el tipo de síntoma que presenta el invitade y el segundo corresponde al nombre del invitade. Debes guardar estos parámetros como atributos de la instancia. **Debes modificarlo**
- `def alerta_de_covid(self)`: En este método se deberá imprimir un mensaje de acuerdo al síntoma que presente el invitade. **Debes modificarlo**

  Los posibles síntomas son:

  - Fiebre: El `self.sintoma` corresponderá a `"fiebre"`. Debe imprimir un mensaje indicando que el invitade tiene fiebre y tiene prohibido el ingreso al **DCCarrete**.
  - Dolor de cabeza: El `self.sintoma` corresponderá a `"dolor_cabeza"`. Debe imprimir un mensaje indicando que el invitade tiene dolor de cabeza y tiene prohibido el ingreso al **DCCarrete**.
  - Tos: El `self.sintoma` corresponderá a `"tos"`. Debe imprimir un mensaje indicando que el invitade tiene tos y tiene prohibido el ingreso al **DCCarrete**.

Por último, en el archivo `permitir_entrada.py` se encuentran las siguientes funciones para verificar si las personas que van a participar en el **DCCarrete** presentan o no síntomas:

`def verificar_sintomas(invitade: Invitade)`: Esta función recibe una instancia de la clase `Invitade` y busca asegurar que quienes asistan al **DCCarrete** no presenten sintomas de COVID, ya sea fiebre, dolor de cabeza o tos. En caso de que el invitade tenga alguno de los síntomas se debe levantar una `Exception` del tipo `RiesgoCovid`. **No debes modificarlo**

`def entregar_invitados(diccionario_invitades: dict)→list`: Recibe un diccionario, tal que sus llaves son los nombres de los invitades y sus valores son instancias de la clase `Invitade`. Esta función esta encargada de verificar uno a uno si los invitades presentan síntomas covid, y **retorna una lista con las instancias de invitades** que no presentan riesgo de tener coronavirus, es decir, sin síntomas. En esta función se deberá recorrer el diccionario de invitades, trabajar con las instancias y para cada una, realizar un `try/except`, de tal forma de capturar la excepción `RiesgoCovid`. Para esto se debe usar la función `verificar_sintomas`, y así revisar quienes tienen síntomas. En el caso de que no haya error, se debe agregar al invitade a la lista oficial final de quienes pueden ingresar al **DCCarrete**. Si es que hay error del tipo `RiesgoCovid` se deberá llamar al método `alerta_de_covid` de la clase `RiesgoCovid`. En el caso de que el invitade tenga síntomas y se levante esta excepción, no se debe agregar este invitade a la lista oficial final del **DCCarrete**. **Debes modificarlo**

Notas

- Esta actividad está diseñada para ser implementada en el orden que se presenta en este enunciado.
- Recuerda que debes hacer `git push` en la rama principal de tu repositorio.

Objetivos

- Conocer los tipos de excepciones más comunes y relevantes para el curso.
- Ser capaz de levantar y capturar excepciones.
- Ser capaz de definir excepciones personalizadas.