Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2020-1

10 de septiembre de 2020
Actividad Formativa

Actividad Formativa 02

Levantamiento y manejo de excepciones

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AF02/`
- **Hora del *push*:** 16:50

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

Pasa otro día en que te acuestas a altas horas de la madrugada.

– ¿Cuándo podré volver a jugar con mis **DCCriaturas**?

La carga académica del semestre no te deja tiempo para realizar tus actividades favoritas. Miras el techo de la habitación, intentando conciliar el sueño, sabiendo que tendrás que levantarte en 3 horas más... 😴. Pero de repente, recibes un mensaje 😳.

– ¿Quién estará despierto a esta hora? – te preguntas, mientras desbloqueas tu teléfono.

¡No es nada más ni nada menos que tu gran amigo, **Gym Pro**! Lees su mensaje, sin poder creer lo que te dice:

> *Necesito tu ayuda. Logré obtener la base de datos donde se almacenan las notas de Programación Avanzada. Podemos modificar las notas de todos los alumnos... Piénsalo.*

Miras tu reflejo en la pantalla oscurecida de tu celular. Esta es tu oportunidad. Esto es lo que estabas esperando. Las dudas ya no cruzan tu mente. Te preparas para cometer el **DCCrimen del Siglo**[1].

[1] No aprobamos la comisión de crímenes en la vida real.

DCCrimen del Siglo

En esta actividad, ayudarás a tu amigo **Gym Pro** a *hackear* la base de datos de Programación Avanzada, con el objetivo de que todos los alumnos del ramo salgan automáticamente con promedio `7.0`. Afortunadamente para ti, **Gym Pro** ya extrajo la base de datos, pero te cuenta que la desencriptación de la información fue muy complicada, por lo que es muy probable que haya múltiples errores de tipeo o de formato. Por lo tanto, necesitarás tus vastos conocimientos sobre **manejo de excepiones** para poder sobreescribir la base de datos sin ningún problema, y así el equipo docente no los descubra. 👀

Archivos

- `verificar.py`: Deberás trabajar en este archivo durante la Parte I: Levantar y capturar excepciones.
- `dccrimen.py`: Este archivo contiene a la clase `GymPro`, que se explicará en la Parte II: Excepción personalizada. Es el archivo principal a ejecutar.
- `estudiante.py`: Este archivo contiene a la clase `Estudiante` que se explica a continuación. Además, contiene algunas funciones relacionadas con la carga de los datos. **No debes modificar este archivo**.
- `alumnos.txt`: Este archivo contiene los datos de los estudiantes y **no debe ser modificado**. El formato del archivo es:

```
nombre;n_alumno;año_de_ingreso;carrera;promedio
```

Clase Estudiante

Como verás más adelante, debes trabajar con objetos de la clase `Estudiante`, la que se encuentra definida en `estudiante.py`. Esta clase **ya viene implementada** y tiene los siguientes atributos:

- `nombre`: Un `str` que representa el nombre del estudiante.
- `n_alumno`: Un `str` equivalente al número de alumno del estudiante. **Este atributo deberá ser corregido.**
- `generacion`: Un `int` que corresponde al año de ingreso a la universidad.

- `carrera`: Un `str` que indica la carrera a la que pertenece. Este atributo puede corresponder a `"Profesor"`, lo que será relevante **solo** para tu excepción personalizada.
- `promedio`: Debe ser un `float` que corresponda al promedio del estudiante en Programación Avanzada. **Este atributo deberá ser corregido.**

Parte I: Levantar y capturar excepciones

En esta parte tendrás que **manejar excepciones** en algunas funciones que necesitarás para realizar efectivamente el **DCCrimen del Siglo**. Deberás completar las siguientes funciones, las que se encuentran en el archivo `verificar.py`, y utilizar `try/except` donde corresponda.

Importante: No se considerará correcto el uso de `except` sin argumentos o `except Exception`.

- `def verificar_numero_alumno(alumno)`: recibe una instancia de la clase `Estudiante`. Deberás verificar que el número de alumno de la instancia esté bien escrito, es decir, que siga las siguientes normas:
  - Si tiene una letra, esta debe encontrarse en la última posición y debe ser una `"J"`.
  - Los primeros dos caracteres deben corresponder a los dos últimos dígitos de su año de ingreso a la Universidad.
  - El tercer y cuarto carácter deben corresponder a su código de carrera: `"63"` para Ingeniería y `"61"` para College.

  En caso de que alguna de estas reglas no se cumpla, debes levantar una excepción del tipo `ValueError` con el siguiente mensaje:

```
"El numero de alumno es incorrecto"
```

- `def corregir_alumno(alumno)`: recibe una instancia de la clase `Estudiante`. Se debe verificar que su número de alumno sea válido, utilizando la función `verificar_numero_alumno`. En caso de no estar correcto, debes capturar la excepción, imprimir el objeto que la representa y **corregir** el número de alumno según las normas especificadas en la sección anterior. Una vez corregido (o de haber estado correcto en un principio), se debe imprimir la frase:

```
"<nombre del alumno> está correctamente inscrite en el curso, todo en orden...\n"
```

- `def verificar_inscripcion(n_alumno, base_de_datos)`: recibe un `str` que corresponde al número de alumno y un `dict` con la base de datos de los alumnos de Programación Avanzada, donde cada elemento tiene como `key` el número de alumno, y como `value` una instancia de la clase `Estudiante`. Este método verifica que el alumno se encuentre en la base de datos del curso. Si el número de alumno no se encuentra en la base de datos deberás **levantar una excepción** del tipo `KeyError` con el siguiente mensaje:

```
"El numero de alumno no se encuentra en la base de datos."
```

  En caso de no levantarse la excepción, la función retorna la instancia de la clase `Estudiante`.

- `def inscripcion_valida(n_alumno, base_de_datos)`: recibe un `str` que corresponde al número de alumno y un `dict` con la base de datos de los alumnos de Programación Avanzada, donde cada elemento tiene como `key` el número de alumno, y como `value` una instancia de la clase `Estudiante`. La función debe revisar que el alumno asociado al número que recibe esté efectivamente inscrito en el curso mediante la función `verificar_inscripcion()`. Si esto no ocurre, deberás imprimir el objeto que representa a la excepción capturada junto con el siguiente mensaje:

```
"¡Alerta! ¡Puede ser Dr. Pinto intentando atraparte!\n"
```

- `def verificar_nota(alumno)`: recibe una instancia de un `Estudiante` de Programación Avanzada. Este método verifica que el atributo `promedio` del alumno tenga el tipo correcto, es decir, que sea un `float`. Si esto se cumple, tiene que retornar `True`. Si no, deberás levantar una excepción del tipo `TypeError` con el siguiente mensaje:

```
"El promedio no tiene el tipo correcto"
```

- `def corregir_nota(alumno)`: recibe una instancia de la clase `Estudiante`, donde tendrás que verificar que su atributo `promedio` tenga el formato correcto a través de la función `verificar_nota()`. En caso de no estar correcto, debes capturar la excepción, imprimir el objeto que la representa y **corregir** la nota para que sea del tipo indicado. Una vez corregida (o de haber estado correcta desde un principio), se debe imprimir la frase:

```
"Procediendo a hacer git hack sobre <nota>...\n"
```

  Donde <`nota`> corresponde al promedio del estudiante.

Parte II: Excepción personalizada

En esta última parte, deberás finalmente *hackear* las notas del ramo sin levantar sospechas. Para esto, debes completar una excepción personalizada llamada `GymPro` (encontrada en el archivo `dccrimen.py`) que será llamada al recorrer la base de datos corregidos (**el recorrido de la base de datos viene implementado**). Esta excepción se encargará de que se puedan cambiar los promedios de los estudiantes a `7.0` sin que los profesores descubran el *hackeo*.

Primero, deberás completar el `__init__` de la excepción personalizada `GymPro`. Esta clase recibe como argumento una instancia de la clase `Estudiante`, y entrega como mensaje:

```
"Wait a minute... Who are you?"
```

Además, debe tener un atributo `profesor`, que corresponde al atributo `nombre` de la instancia de la clase `Estudiante` que recibe.

También deberás completar el método `evitar_sospechas`, que debe imprimir un mensaje que diga:

```
"¡Cuidado, viene <nombre>! Solo estaba haciendo mi último push..."
```

Donde <`nombre`> corresponde al atributo `profesor` de la clase.

Finalmente, deberás completar el `try/except` correspondiente[2]. En esta última parte, si es que durante

[2]Esto se encuentra al final del código en el módulo `dccrimen.py`

la iteración sobre la base de datos se encuentra a un profesor infiltrado (es decir, que su atributo `carrera` sea igual a `"Profesor"`), deberás levantar la excepción personalizada `GymPro` completada anteriormente, de lo contrario, deberás corregir su promedio por un `7.0`, e imprimir el siguiente mensaje:

```
"Hackeando nota..."
```

En el caso de atrapar la excepción, deberás llamar al método `evitar_sospechas` que completaste anteriormente.

Notas

- Al arreglar el formato de los números de alumno, puedes asumir que todos los que deban tener una letra tendrán solamente una y esta estará en mayúscula. Además, te recomendamos arreglar el formato en el mismo orden presentado en la Parte I: Levantar y capturar excepciones.
- Al arreglar el tipo de los atributos `promedio`, ten presente que inicialmente pueden ser de tipo `int`, `str` o `float`.
- Esta actividad está diseñada para ser implementada en el orden que se presenta en este enunciado.
- En el archivo `verificar.py` podrán probar los métodos que completaron para algunos casos. Para probar con toda la base de datos, deberán ejecutar el archivo `dccrimen.py`.
- Recuerda que debes hacer *push* en la rama principal de tu repositorio.

Objetivos

- Ser capaces de capturar y levantar excepciones en Python.
- Ser capaces de definir e implementar una excepción personalizada.
- Objetivo adicional: Ser bacán. ✨