Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2019-2

3 de octubre de 2019
Actividad Formativa

Actividad 06

Excepciones

Introducción

Últimamente, en las distintas plataformas móviles para transporte de personas han estado ingresando masivamente conductores falsos y malvados, furiosos por su derrota en **Initial P**. Detrás de ellos, está el malévolo **DrPinto** encomendándolos a atacar al DCC. Por suerte, el Líder Supremo **Tini Tamburini** sabe que alguien como tú puede identificarlos y derrotarlos. Por lo anterior, te designa a ti como desarrollador para crear un programa que sea capaz de chequear los distintos datos que el registro requiere, pudiendo filtrar y seleccionar a los conductores verdaderos.

Figura 1: Logo DCConductor

DCConductor

El objetivo de esta actividad es que puedas verificar los datos ingresados por los conductores en un registro. Primero, deberás verificar los posibles errores que aparezcan según los requerimientos de las plataformas; y luego deberás capturar estos errores y **manejarlos** de forma correcta. Te entregaremos el módulo `dcconductor.py` el cual tiene una estructura básica que deberás completar con el código adecuado.

Archivos

- `main.py` es el archivo principal a ejecutar. En este se llaman a todas las funciones a implementar.
- `dcconductor.py` contiene la clase `DCConductor`, la que posee los siguientes atributos:
  - `registro_oficial`: diccionario del registro oficial de conductores.
  - `conductores`: lista con los conductores que postulan
  - `seleccionados`: lista con los conductores que serán seleccionados. Inicialmente vacía.
- `carga_archivos.py` este archivo es el encargado de cargar los datos entregados.
- `conductores.py` contiene la clase `Conductor`. No debes hacer nada con este archivo. Sólo es importado a `dcconductor.py`. La clase posee los siguientes atributos:
  - `nombre`: `str` con el nombre del conductor
  - `rut`: `str` con el rut del conductor
  - `mail`: `str` con el mail del conductor
  - `celular`: `str` con el celular del conductor
  - `patente`: `str` con la patente del conductor
- `postulaciones.csv` contiene los datos ingresados por los conductores que postulan a la aplicación.
- `registro_oficial.json` contiene los datos con el registro oficial del país de los conductores. No necesitas abrirlo, el método `cargar_registro_oficial` se encarga de ello.
- `excepcion_patente.py` contiene la excepción personalizada `ErrorPatente`, la que será explicada más adelante.

Parte I: Poblar DCConductor

No tan solo te encomendaron trabajar en la validación, sino que también en poblar el módulo con la información de las solicitudes enviadas y los registros oficiales de conductores. Deberás completar el siguiente método en el módulo `carga_archivos.py` para lograrlo:

- `def cargar_datos(archivo_registro_oficial, archivo_conductores)`: recibe las rutas de los archivos nombrados e intenta cargarlos al diccionario de registro oficial y a la lista de conductores de `DCConductor` a través de los métodos especializados `cargar_registro_oficial` y `cargar_conductores` respectivamente. Estos últimos te los entregamos implementados y no deben ser modificados. En caso de que uno de los archivos no existan, tu programa levantará un error de Python, el cual debes capturar y luego imprimir un mensaje indicando este error e indicando el nombre del archivo que está causando problemas. Ejemplo: `"Error: El archivo <nombre> no existe."`

Parte II: Identificar errores

El líder supremo Tini Tamburini te encomienda a identificar los campos ingresados por los conductores. Para esto, deberás completar los siguientes métodos de la clase `DCConductor`, encargados **solamente**

de levantar excepciones de Python cuando el *input* no sea el esperado, es importante considerar que la excepción levantada **debe** tener relación con el error asociado.

- `def chequear_rut(conductor)`: recibe una instancia de la clase `Conductor` y chequea que el formato de escritura del RUT del conductor sea válido, en caso contrario levanta una excepción. En este caso nos interesa que no tenga puntos pero sí guión antes del último dígito, es decir, que sea de la forma: `"19644911-6"`. Los siguientes RUT levantarían excepciones: `"19.644.911-6"` porque contiene puntos; y `"196449116"` porque no tiene el guión final. Puedes suponer que el dígito verificador concuerda con el RUT, y que todos serán superiores a los 10 millones. Podrás ver ejemplos de los mensajes en la siguiente sección.
- `def chequear_nombre(conductor)`: recibe una instancia de la clase `Conductor` y chequea que el nombre del conductor se encuentre en el registro oficial. En caso que no se encuentre debe levantar una excepción.
- `def chequear_celular(conductor)`: recibe una instancia de la clase `Conductor` y chequea que el número de celular sea válido. Es válido si contiene únicamente números[1], es de largo 9 y comienza con `"9"`. Si no se cumple alguna de estas condiciones, levanta una excepción.
- `def chequear_patente(conductor)`: recibe una instancia de la clase `Conductor` y levanta la excepción `ErrorPatente` si la patente de este conductor no es la que tiene asociada en `registro_oficial`.[2] En esta función puedes asumir que el conductor se encuentra en el registro oficial.

Parte III: Capturar errores

Seleccionar conductores de `DCConductor`

En esta sección deberán manejar las excepciones levantadas en la parte anterior para todos los conductores que solicitaron inscribirse, para esto debes utilizar el archivo `main.py`:

- Debes verificar a cada conductor de la lista `conductores`, es decir, aplicar los cuatro métodos de la Parte I sobre cada conductor. En caso de que algún chequeo falle tu programa debe ser capaz de **manejar tales excepciones** con `try/except`, imprimir el error asociado al conductor y contar la cantidad total de errores. En caso de que no haya errores tendrás que agregarlo a la lista `seleccionados`.

A continuación veremos un ejemplo del funcionamiento de este método:

```
El rut 197879294 no contiene guion.
El conductor Albert León Salvador no está en el registro
El conductor Clara Soledad Pons Perez no está en el registro
La patente FLHI48 no es la registrada para Jose Manuel Salvà Pol.
El conductor Ignacio Jaume Castells no está en el registro
La patente OK2798 no es la registrada para Manuela Puga.
El conductor Celia Córdoba-Almeida no está en el registro
El celular 55971573 no comienza con 9.
El celular +56972773242 contiene caracteres no numéricos.
El rut 20.619.668-2 contiene puntos.
```

[1] Para esto puede serte de utilidad `isnumeric()`.

[2] Pueden revisar en el archivo `excepcion_patente.py` qué parametros recibe esta excepción.

Parte IV: Construir excepción propia

Hasta ahora hemos levantado y capturado excepciones, usando una tal `ErrorPatente` en algunas ocasiones. Tu deber ahora, es personalizar esta excepción, con lo que se detalla en el siguiente apartado.

Construir la excepción `ErrorPatente`

En el archivo `excepcion_patente.py` deberás rellenar la excepción personalizada, la cual recibirá como argumento al conductor. Aquí se debe guardar en un atributo un mensaje explicativo indicando el nombre del conductor y la patente errónea. Un ejemplo de mensaje explicativo que debe ir dentro de esta clase se muestra en el ejemplo de la parte III.

Requerimientos

- (1.00 pts) Parte I: Poblar DCConductor
- (3.00 pts) Parte II: Identificar errores Se levanta excepción si
  - (0.75 pts) Se ingresa un RUT con punto o sin guión.
  - (0.75 pts) Se ingresa un nombre inválido.
  - (0.75 pts) Se ingresa un celular inválido
  - (0.75 pts) Al conductor no le corresponde esa patente o la patente no existe.
- (1.50 pts) Parte III: Capturar excepciones
- (0.50 pts) Parte IV: Construir excepción propia

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AC06/`
- **Hora del *push*:** 16:30

Auto-evaluación

Como esta corresponde a una actividad formativa, te extendemos la instancia de responder, después de terminada la actividad, una auto-evaluación de tu desempeño. Esta se habilitará a las **16:50 de jueves 3 de octubre** y tendrás plazo para responderla hasta las **23:59 del día siguiente**. Puedes acceder al formulario mediante el siguiente enlace:

`https://forms.gle/ueFr9E2EgusSsskB9`

El asistir, realizar la actividad y responder la auto-evaluación otorgará como bonificación al alumno 2 décimas para sumar en su mejor actividad sumativa del semestre.