Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2022-1

16 de Junio de 2022
Actividad Formativa

Actividad Formativa 5

DCCoordinando IIC2233

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AF5/`
- **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

¡Has sido seleccionado como el nuev@ coordinador de programación avanzada! Dentro de tu nuevas labores se encuentra recolectar las entregas y publicar la *issue* con el *commit* a revisar de esta misma actividad. Sin embargo, hay un pequeño problema: no sabes cómo hacerlo.

Es en ese momento que recuerdas tus conocimientos aprendidos de *Webservices*, por lo que decides intentar usar la API de Github para publicar las *issues*, pero no sin primero hacer una pequeña prueba de que tu código funciona en un repositorio vacío, usando como fuente de datos una API de Pokémon que encontraste. ¡El futuro de IIC2233 está en tus manos, programador avanzad@!

Flujo del programa

El programa deberá inicialmente interactuar con PokeAPI, una API que te entregará estadísticas sobre un Pokémon, el cual será definido según tu número de alumno. Una vez recibidos los datos, deberás seleccionar ciertas estadísticas de las entregadas, las cuales compondrán el contenido de la *issue* que crearás.

En la segunda parte, deberás hacer uso de la API de Github para completar el proceso de manipulación de *issues*: deberás completar los métodos que permitan crear una *issue*, bloquear la *issue* creada para que no pueda recibir más comentarios, y luego desbloquearla. ¡Recuerda que para utilizar esta API, tienes que demostrar que eres el dueño de tu cuenta mediante un token de acceso personal!

Archivos

Archivos de código

- `parametros.py`: Contiene parámetros a usar en la actividad tales como rutas a archivos de texto, urls de las API a utilizar, y datos que necesitarás para realizar las *requests*. **Debes modificarlo**
- `api.py`: Contiene las funciones que realizarán los llamados a las API de la actividad. **Debes modificarlo**
- `main.py`: Realiza todo el flujo del programa e imprime los resultados en la consola. **No debes modificarlo**
- `utils/`: Carpeta con archivos que proveen funciones auxiliares para el funcionamiento correcto del menú de `main.py` **No debes modificarlo**

Archivos de datos

- `data/tokens.csv`: Una vez has ingresado por primera vez en el menú de la actividad, este archivo almacenará tu token de acceso personal a Github para que puedas ejecutar la actividad sin ingresarlo nuevamente.
- `data/issues.csv`: Cada vez que crees una issue en tu repositorio, este archivo almacenará el número de dicha issue en una nueva fila.

Parte I: Consiguiendo a tu Pokémon

En esta primera parte conseguirás los datos que usarás como contenido para crear una *Issue*. El id de tu Pokémon asignado corresponderá a **los dos últimos dígitos de tu número de alumno** (en caso de terminar en J, debes reemplazar dicha letra por un dígito 0). Por ejemplo:

- Si tu número de alumno es **12345678**, el id de tu pokémon asignado es **78**.
- Si tu número de alumno es **1234567J**, el id de tu pokémon asignado es **70**, ya que se reemplazó la J por un 0.

Deberás modificar la variable `POKEMON_NUMERO` del archivo `parametros.py` con el id de tu Pokémon asignado. Luego, deberás modificar la siguiente función del archivo `api.py`, procurando utilizar los parámetros `POKEMON_BASE_URL` y `POKEMON_NUMERO` al momento de trabajar:

- `def get_pokemon() -> list:` Este archivo deberá llamar a la API de Pokémon y obtener los datos de tu Pokémon asignado. La documentación de la función expuesta, la cual llamaremos *endpoint* de aquí en adelante, que te servirá para obtener la información, se encuentra en la siguiente página (debes dirigirte en la tablas de *Contents* a **Pokemon >Pokemon**).

  Una vez obtenidos los datos, deberás retornar una lista que incluya (siguiendo este mismo orden) el **nombre del pokémon** (*name*), su **peso** (*weight*), **altura** (*height*) y una **lista con los nombres de los tipos** (*type*) de dicho Pokémon. **Debes modificarlo**

  La lista que retorna debe quedar con el siguiente formato:

```
[nombre, peso, altura, [tipo1, tipo2, ...]]
```

Una vez completada esta parte, puedes ejecutar la opción *Get Pokémon* del menú contenido en `main.py` y deberías poder ver los datos obtenidos de tu Pokémon en la terminal.

Parte II: Crear y modificar *issues* de Github

En esta segunda parte deberás trabajar con los datos del Pokémon obtenido anteriormente para generar una *issue* en Issues, un repositorio dedicado especialmente para esta actividad.

Es importante saber que la API de Github requiere *headers* de Autenticación para cualquiera de las *requests* pedidas en esta actividad. Para esto, necesitarás tu *token* de acceso personal para tu cuenta de Github. Podrás obtener tu token desde esta página, y durante los primeros minutos de la actividad tu profesor mostrará cómo configurarlo adecuadamente.

TIP: el formato necesario para crear un diccionario que posteriormente puede ser pasado como *headers* a la función de *request* es la siguiente:

```
headers = {
    'llave': 'valor de la llave',
    'llave 2': 'otro valor',
    ...
}
```

Además, podrás encontrar la documentación de la API de github respecto al manejo de Issues en la siguiente página. Deberás buscar, para cada función pedida en esta parte, la documentación del *endpoint* apropiado que te permitirá realizar lo pedido.

Las funciones a implementar son:

- `def post_issue(token: str, pokemon: list) -> (int, int):` Recibe tu **token de acceso personal de Github**, y el **listado con información de tu Pokémon** obtenido en la parte anterior. Con estos datos, deberás realizar una *request* de tipo `POST`, para generar una *issue* cuyo título sea tu **nombre de usuario de Github** y cuyo contenido o *body* sea el listado con la **información del Pokémon**. Una vez realizada la *request*, deberás retornar una tupla que contenga el código de estado de la respuesta y el *number* de la *issue* creada. **Debes modificarlo**
- `def put_lock_issue(token: str, numero_issue: int) -> int:` Recibe tu **token de acceso personal de Github**, y un *número de issue a bloquear*. Debes realizar una *request* que permita bloquear (agregar un *lock*) a la *issue* mediante el método `PUT` y retornar el código de estado que entregue la respuesta. **Debes modificarlo**
- `def delete_lock_issue(token: str, numero_issue: int) -> int:` Recibe tu **token de acceso personal de Github**, y un **número de *issue* a desbloquear**. Deberás realizar una *request* que permita desbloquear (o eliminar el *lock*) de la *issue* ingresada mediante el método `DELETE` y retornar el código de estado que entregue la respuesta. **Debes modificarlo**

Una vez completada cada función de esta parte, podrás ejecutar los métodos respectivos en el menú del archivo `main.py` y comprobar en la sección de *issues* del repositorio de la actividad si tu *issue* fue creada, bloqueada o desbloqueada según corresponda.

Notas

- Esta actividad será corregida de **manera automática**, así que procura seguir al pie de la letra las instrucciones de cada método respecto a qué argumentos reciben y qué deberían retornar.
- Siéntete libre de agregar nuevos `print()` en cualquier lugar de tu código para encontrar errores. Es una herramienta muy útil para comprobar tu desarrollo, pero recuerda borrarlos antes de hacer el commit final, así evitas problemas con el corrector automático.

- En general, las APIs poseen *rate limiters*, lo que significa que en caso de detectar un flujo inesperadamente alto de *requests* de una misma dirección, denegarán el uso de sus servicios a dicha dirección temporalmente. ¡Ten cuidado con que tu código no envíe cientos de requests en un segundo![1]
- **Importante**: En esta actividad estarás haciendo uso de tu *Personal Access Token de Github*, lo que para fines prácticos equivale a tu contraseña de acceso a la aplicación. El menú de la actividad solicita la contraseña como *input* de terminal con el fin de no almacenarla como parámetro en el código, y la almacena dentro de un archivo de extensión `.txt` en la carpeta `data`. Como medida de seguridad, incluimos un archivo `.gitignore` que evitará que dichos archivos de la carpeta se suban al momento de subir tu código. Te recordamos que seas cuidados@ con no dejar tu *token* como variable en el código, o incluirla al momento de subirlo.

Requerimientos

- (0.5 pts) 2 funciones completadas correctamente.
- (0.5 pts) Otras 2 funciones completadas correctamente.

[1] En esta actividad no deberías necesitar hacer *requests* dentro de un flujo como `while` o `for`