Guía de ejercicios - Manejo de errores y archivos

 ¡Hola! Te damos la bienvenida a esta nueva guía de estudio.

¿En qué consiste esta guía?

La siguiente guía de estudio tiene como objetivo practicar y ejercitar los contenidos que hemos visto en clase.

¡Vamos con todo!

Tabla de contenidos

¡Comencemos!

¡Manos a la obra! - Herencia

Desde el área de electrónica de una tienda de retail, se te ha solicitado diseñar la estructura de clases de sus monitores, donde cada monitor tendrá su propia resolución. Por ahora, se solicita considerar que, dentro de los monitores, existen los de tipo LED, y también existen los monitores multifunción. En este caso, considere crear una clase que permita crear monitores / televisor 2 en 1 (debes considerar una clase Televisor).

¡Manos a la obra! - Sobreescritura de métodos

Tu amigo nuevamente ha solicitado tu ayuda, esta vez, para agregar tres pequeñas mejoras en el programa ya entregado.

Mejora 1

Necesita que, en el caso de crear un mapper de Pokemon, se solicite además de la URL base el límite de registros a obtener al consultar por el listado completo, y almacenar este valor en un atributo de instancia.

Mejora 2

Además, te comunica que su otro amigo (el que hizo la API), solicita que se aplique encapsulamiento de los atributos, tanto del que ya existía que contenía la URL base, como del nuevo solicitado que almacena el límite de registros (solo para pokemones).

Mejora 3

Finalmente, te solicita colocar un pequeño mensaje que aparezca en la terminal cada vez que se solicite el listado de personajes, indicando el listado que se está obteniendo.

¡Manos a la obra! - Manejo de excepciones con try/except

Continuando con el ejercicio guiado "Validación de datos" del archivo "**Presentación - Manejo de Errores y Excepciones**", le solicitan esta vez agregar el atributo “fecha” a las reuniones. Se debe validar que la fecha tenga el formato “DD/MM/AAAA”.

Tip: Puede usar la siguiente expresión regular:

```
^([0-3]?\d\/{1})([01]?\d\/{1})([12]{1}\d{3}?)$
```

¡Manos a la obra! - Modos de apertura de archivos

Continuando con el ejercicio guiado "Generación de objetos a partir de un archivo" del archivo "**Presentación - Manejo de archivos con Python**", le solicitan envuelva todo lo que está dentro del ciclo `while` dentro de un bloque `try/except`.

- Dentro de la cláusula `except`, escriba el contenido de la excepción (como `str`) en un archivo `error.log`.
- El contenido debe ser añadido al final del archivo `error.log`, sin borrar su contenido previo.
- La siguiente línea del documento de productos debe leerse al final del bloque `try/except`.

Solucionario

Herencia

1. En un archivo televisor.py, crear la clase `Televisor`

```
class Televisor():
    pass
```

2. En un archivo monitor.py, importar la clase `Televisor`, y crear la clase `Monitor`, con el atributo de instancia resolucion.

```
from televisor import Televisor

class Monitor():
    def __init__(self, resolucion) -> None:
        self.resolucion = resolucion
```

3. A continuación, en el mismo archivo, crear la clase `MonitorLED`, la cual hereda de la clase `Monitor`.

```
class MonitorLED(Monitor):
    pass
```

4. Finalmente, en el mismo archivo, crear la clase `MonitorTelevisor`, la cual hereda tanto de `Monitor` como de `Televisor`.

```
class MonitorTelevisor(Monitor, Televisor):
    pass
```

Sobreescritura de métodos

1. En la clase `MonMapper`, modificar el constructor de forma tal que la url base se asigne a un atributo de instancia privado. Agregar también el getter y setter de este atributo.

```
    def __init__(self, base_url: str) -> None:
        self.__base_url = base_url

    @property
    def base_url(self) -> str:
        return self.__base_url

    @base_url.setter
    def base_url(self, base_url) -> None:
        self.__base_url = base_url
```

2. Sobrescribir el constructor en la clase `PokemonMapper`, de forma tal que dentro de éste se haga el llamado al constructor de la clase base, y luego se asigne el límite al atributo de instancia privado (se dejó con valor por defecto 20, que es el mismo que tiene la api).

```
class PokemonMapper(MonMapper):

    def __init__(self, base_url: str, limit: int = 20) -> None:
        super().__init__(base_url)
        self.__limit = limit
```

3. A continuación, en la misma clase, definir la propiedad con getter y setter del atributo de instancia limit.

```
    @property
    def limit(self) -> int:
        return self.__limit

    @limit.setter
    def limit(self, limit) -> None:
        self.__limit = limit
```

4. Luego, modificar el llamado al listado de todos los pokemones, de forma tal que incluya el límite de la instancia.

```
    def obtener_mon_por_nombre(self, nombre: str) -> dict:
        respuesta = requests.get(
            f"{self.base_url}/{nombre}?limit={self.__limit}"
        )
```

5. Finalmente, en el script demo.py, hacer uso de `isinstance` al recorrer los mappers y mostrar el mensaje solicitado.

```
for mapper in mappers:
    if isinstance(mapper, PokemonMapper):
        print("Obteniendo listado de Pokemones...")
    elif isinstance(mapper, DigimonMapper):
        print("Obteniendo listado de Digimones...")

    mons += mapper.obtener_todos_los_mon()
```

Manejo de excepciones con try/except

1. Agregar la clase `FechaException` en el archivo error.py.

```
class FechaError(Error):
   pass
```

2. Agregar el atributo fecha en la clase `Reunion` en el archivo reunion.py.

```
class Reunion():
   def __init__(self, titulo: str, hora: str, fecha: str) -> None:
       self.titulo = titulo
       self.hora = hora
       self.fecha = fecha
```

3. En el archivo demo.py, agregar el `import` de la clase `FechaException`, definir la variable fecha como `None`, y la variable `date_re` con la expresión regular entregada.

```
from error import HoraError, LargoTextoError, FechaError

titulo = None
hora = None
fecha = None

time_re = "^(?:(?:([01]?\d|2[0-3]):)?([0-5]?\d):)?([0-5]?\d)$"
date_re = "^([0-3]?\d\/{1})([01]?\d\/{1})([12]{1}\d{3}?)$"
```

4. Dentro del bloque try, agregar el ingreso de fecha.

```
       if fecha is None or re.search(date_re, fecha) is None:
           fecha = input("\nIngrese fecha de la reunión"
           " (Formato: DD/MM/AAAA):\n")

           if re.search(date_re, fecha) is None:
               raise FechaError("Formato de fecha debe ser DD/MM/AAAA.")
```

5. Agregar el valor de fecha a la creación de la instancia.

```
r = Reunion(titulo, hora, fecha)
print("\nReunión creada correctamente.")
```

Modos de apertura de archivos

1. Opcionalmente, añadir `import` de datetime para incluir la fecha y hora del error en el log generado.

```
from datetime import datetime
```

2. Envolver el contenido dentro del ciclo `while` en un bloque `try`/`except`, obteniendo la instancia de la excepción en una variable.

```
while linea:
    try:
        producto = json.loads(linea)
        instancias.append(
            Producto(producto.get("nombre"), producto.get("precio"))
        )
        linea = productos.readline()
    except Exception as e:
```

3. Dentro de la cláusula except, incluir el bloque que permite añadir la excepción generada en un archivo de log.

```
    except Exception as e:
        with open("error.log", "a+") as log:
            now = datetime.now()
            log.write(f"[{now.strftime('%Y-%m-%d %H:%M:%S')}]"
            f" ERROR: {e}\n")
```

4. Finalmente, añadir una cláusula finally, y dejar dentro de ella la actualización del contenido de la variable linea.

```
while linea:
    try:
        producto = json.loads(linea)
        instancias.append(
            Producto(producto.get("nombre"), producto.get("precio"))
        )
    except Exception as e:
        with open("error.log", "a+") as log:
            now = datetime.now()
            log.write(f"[{now.strftime('%Y-%m-%d %H:%M:%S')}]"
          " ERROR: {e}\n")
    finally:
        linea = productos.readline()
```