LENGUAJES DE PROGRAMACIÓN 2

TALLER PARCIAL (25%)

En este examen parcial de lenguajes de programación 2 vamos a colocar en práctica lo visto en el curso. El parcial será solucionado en los grupos que ustedes han conformado.

Los códigos serán **sustentados** el **13 de octubre** de forma aleatoria entre los miembros de cada grupo en un pequeño espacio de tiempo. Y hará parte de la calificación final del parcial.

El parcial debe ser enviado como máximo el 13 de octubre hasta las 10am;

Parcial

Requisitos.

- Use el PEP8 para la creación de variables y clases
- La validación de errores debe hacerse con try, except, rise, else
- Solo se puede importar los módulos especificados en este documento
- Colocar comentarios y docstrings
- Usar Programación Orientada a Objetos

Paquetes

Instalar las siguientes bibliotecas que serán empleadas para el desarrollo del parcial

- requests (https://pypi.org/project/requests/)
- selectorlib (https://pypi.org/project/selectorlib/)
- fpdf (https://pyfpdf.readthedocs.io/en/latest/)

Web scraping o raspado web, es una técnica utilizada mediante programas de software (Python) para extraer información de sitios web. Usualmente, estos programas simulan la navegación de un humano en la World Wide Web ya sea utilizando el protocolo HTTP manualmente, o incrustando un navegador en una aplicación.

Población y temperatura de las principales ciudades del mundo

Cree un script en Python empleando todo lo aprendido hasta este punto en el curso, dando un especial énfasis en la programación orientada a objetos y al tratamiento de errores donde se realice una búsqueda de la población y la temperatura actual de una ciudad.

El script de Python debe generar un informe sobre esto para eso debe pedirle al usuario la cantidad de ciudades que serán ingresados al informe junto con el nombre del país y una ciudad importante de dicho país. Con base a esa información el script debe hacer un web scraping para extraer los datos de la población actual de esa ciudad junto con su temperatura actual. Adicionalmente, el script realizará un cálculo de la diferencia poblacional entre los países del informe, donde se hará la resta de la mayor población menos la de menor población. Toda la información debe desplegarse tanto en el terminal, como generar un reporte en formato PDF.

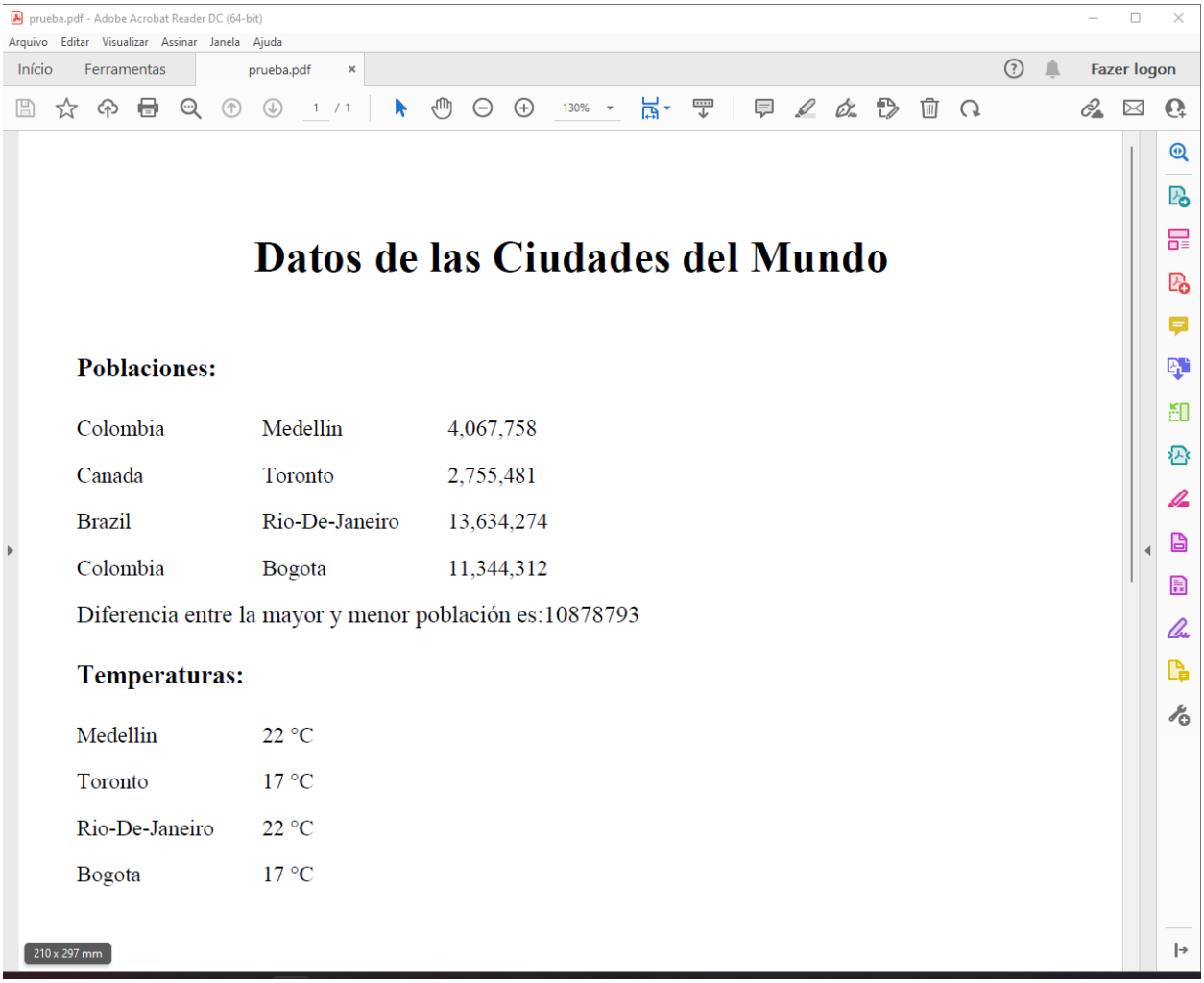
```
Cuantas ciudades desea ingresar: 4
Ingrese el país 1: Colombia
Ingrese la ciudad 1: Medellin
Ingrese el país 2: Canada
Ingrese la ciudad 2: Toronto
Ingrese el país 3: Brazil
Ingrese la ciudad 3: Rio de Janeiro
Ingrese el país 4: Colombia
Ingrese la ciudad 4: Bogota
La población de medellin/colombia es 4,067,758
Su temperatura a esta hora es de: 22 °C

La población de toronto/canada es 2,755,481
Su temperatura a esta hora es de: 17 °C

La población de rio-de-janeiro/brazil es 13,634,274
Su temperatura a esta hora es de: 22 °C

La población de bogota/colombia es 11,344,312
Su temperatura a esta hora es de: 17 °C

Diferencia de Población: 10878793
```



Toda esta información, la tomaremos de dos sitios web en específico:

- Para la población: https://worldpopulationreview.com/world-cities
- Para la temperatura: https://www.timeanddate.com/weather/

Antes de aventurarnos a resolver este problema debemos introducirnos un poco en el concepto de Web Scraping, para eso vea el siguiente video:

https://www.youtube.com/watch?v=aS-NRcdoc_Y

Una vez vimos el video anterior, podemos ver el siguiente video de explicación del examen parcial:

https://youtu.be/E5Pz2zCLYyA

Procedimiento sugerido:

1. Crear un script principal, que contenga el *Entry Point* donde únicamente se llame una función principal. Crear en el mismo script la función principal donde se harán todos los cálculos, los llamados de funciones, la creación de instancias, el despliegue de resultados, etc.
2. Pedirle al usuario que ingrese por el terminal los datos de:
    - Cantidad de ciudades a ingresar
    - Nombre de los países
    - Nombre de las Ciudades
3. Crear un módulo llamado cities dentro del cual se deben programar las clases relacionados a la obtención de datos via web scraping de la ciudad y país.
4. Importe las siguientes bibliotecas:

```
from selectorlib import Extractor
import requests
```

5. En el módulo anterior, cree una clase padre **Population** que tenga tres variables estáticas:
    - headers: Diccionario con los headers necesarios para hacer web scraping usando la biblioteca request

```
headers = {
    'pragma': 'no-cache',
    'cache-control': 'no-cache',
    'dnt': '1',
    'upgrade-insecure-requests': '1',
    'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36',
    'accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.9',
    'accept-language': 'en-GB,en-US;q=0.9,en;q=0.8',
    }
```

    - url_estatica: La parte de la URL que no cambia sin importar el tipo de ciudad que se selecciones
    - yml_path: la dirección del archivo YML para poder hacer web scraping.

6. El método constructor de **Population** debe recibir el país y la ciudad.
7. Crear un método de instancia oculto que construya un string con la dirección URL tomando como base la url_estatica y adicionando al final las partes de la URL que van cambiando conforme se va seleccionando una ciudad diferente.

8. Crear un método de instancia oculto que haga el **scrape**. En este método de instancia se emplean las funcionalidades de requests y Extractor, parecido al video del profesor Sergio donde se muestra como extraer los datos de una página web y que retorne un Diccionario con el dato que fue extraído dentro de ese diccionario.
9. Crear un método de instancia **get** que sea convertido en una **propiedad**. Este método get debe entregar el value del diccionario obtenido en scrape, pero antes debe hacer la validación de no tener ningún dato del Tipo None. Caso exista un dato None, levantar una excepción por Value Error y detener el programa.
10. Cree un método **str** que muestre la ciudad y el país en caso de que se coloque algún objeto instanciado por la clase **Population** dentro de un print.
11. Cree un método estático **difference** que reciba una lista con objetos del tipo Population. Con base en esta lista buscar cual es la ciudad con mayor y menor población y retornar la diferencia.
12. Crear una clase hija **Temperature** que herede de **Population** sus métodos.

13. Modificar dentro de **Temperature** las variables estáticas url_estatica y yml_path, junto con el método de construcción de la URL específica para la extracción de la temperatura de la ciudad.
14. Cree un nuevo módulo llamado **reports** donde se importen las siguientes bibliotecas:

```
import webbrowser
from fpdf import FPDF
import os
```

15. Crear un nuevo módulo, (otro script) llamado **reports** donde se debe crear una nueva clase llamada *ReportePdf* que como argumento en su constructor reciba el nombre del archivo del pdf.
16. Crear un método de instancia llamado *generar* dentro de *ReportePdf* que sea capaz de generar un archivo PDF con el informe mostrando los países, sus ciudades, la diferencia entre la mayor y menor población y las temperaturas en el tiempo de la consulta. Para eso este método puede recibir como argumentos de entrada un objeto del tipo Population, un objeto del tipo Temperature y un entero que contenga la diferencia entre la mayor y menor población del estudio.
17. Para generar el PDF se usa la librería FPDF. Es necesario que usted como futuro Pythonista aprenda a leer e interpretar diversas librerías que le pueden ser de utilidad, por lo tanto, entre a la documentación para que aprenda como utilizar cada uno de los métodos para generar el PDF. https://pyfpdf.readthedocs.io/en/latest/
18. Cuando es generado el PDF, este se almacena en la carpeta del proyecto automáticamente. Pero el objetivo del script es que este archivo además de guardarse se abra automáticamente cada que se ejecuta el script, por lo tanto en el módulo *reports* se importa la biblioteca *webbrowser* y dentro del método de instancia *generar* invoque la función *webbrowser.open(filename)* para que el PDF se abra automáticamente. Recuerde que debe cerrar el PDF siempre que vuelva ejecutar el script, porque caso contrario le va a aparecer un error.

Los pasos anteriores es un paso a paso de las clases que le permitirá cumplir el desafío del parcial, sin embargo, usted es libre de implementar su propia lógica para llevar a cabo el reto siempre y cuando respete que toda la aplicación debe hacerse con programación orientada a objetos.