TITULO DE LABORATORIO
PARCIAL (CONVERSOR DE TEMPERATURA)

HANNA KATHERINE ABRIL GÓNGORA
KAROL ASLEY ORJUELA MAPE

UNIVERSIDAD MANUELA BELTRÁN

PROGRAMACIÓN ORIENTADA A OBJECTOS

DOCENTE

DIANA MARCELA TOQUICA RODRÍGUEZ

BOGOTÁ DC VIERNES 22 DE MARZO

1. **EXPLICACIÓN DEL CÓDIGO EN PYTHON**

- *import requests:* Importa el módulo requests, que se utiliza para enviar solicitudes HTTP en Python.

```
1   import requests
2
```

- Se define la clase Ciudad que tiene dos atributos: *ciudad* y *temp_ciudad*. El método *__init__* inicializa estos atributos cuando se crea un objeto de esta clase.

```
4   class Ciudad:
5       def __init__(self, ciudad, temp_ciudad):
6           self.ciudad = ciudad
7           self.temp_ciudad = temp_ciudad
8
```

- *@staticmethod* indica que el método *obtener_temperatura* es estático, lo que significa que puede ser llamado directamente desde la clase sin necesidad de crear una instancia de la clase.

```
10      @staticmethod
11      def obtener_temperatura(ciudad):
```

- *def obtener_temperatura(ciudad):* Este método estático toma el nombre de una ciudad como argumento y utiliza la API de OpenWeatherMap para obtener la temperatura actual de cualquier ciudad ingresada en grados Celsius.

  Para eso

  Se crea una URL que se obtuvo un link de la API y una clave para que la API pueda usarse, esa clave se agrega al final de link para que la API pueda ser usada con normalidad, además de eso se le agrego a la URL este parámetro (&units=metric) para que la temperatura la diera en medida europea y no americana. Usando f-strings se incluye el nombre de la ciudad y una clave de API proporcionada por OpenWeatherMap en el link.

respuesta = requests.get(url): Envía una solicitud GET a la URL creada para obtener los datos de temperatura de la ciudad.

datos = respuesta.json(): Convierte la respuesta de la solicitud GET en un objeto JSON. (JSON es una herramienta esencial para el manejo eficiente de datos en aplicaciones web, programas computacionales y otros contextos digitales)

```
10      @staticmethod
11      def obtener_temperatura(ciudad):
12          url = f'https://api.openweathermap.org/data/2.5/weather?q={ciudad}&appid=06d4164c7de905
13          respuesta = requests.get(url)
14          datos = respuesta.json()
15
```

- *if respuesta.status_code == 200:* Verifica si la solicitud fue exitosa (código de estado 200). Si es así, extrae la temperatura de los datos JSON y la devuelve.

  else: En caso de que la solicitud falle, imprime un mensaje de error que indica la razón del fallo y devuelve None.

```
10      @staticmethod
11      def obtener_temperatura(ciudad):
12          url = f'https://api.openweathermap.org/data/2.5/weather?q={ciudad}&appid=06d4164c7de905
13          respuesta = requests.get(url)
14          datos = respuesta.json()
15
16          if respuesta.status_code == 200:
17              temp_ciudad = datos['main']['temp']
18              return temp_ciudad
19          else:
20              print(f'Error al obtener la temperatura: {datos["message"]}')
21              return None
22
```

- Se define la clase *ConvTemp* con dos atributos: kelvin y fahrenheit. Al igual que la clase Ciudad, también tiene un método estático llamado *convertir_temperatura*

```
24  class ConvTemp:
25      def __init__(self, kelvin, fahrenheit):
26          self.kelvin = kelvin
27          self.fahrenheit = fahrenheit
28
```

- *def convertir_temperatura(temp_ciudad, opcion_conversion):* Este método estático toma la temperatura de una ciudad la cual se obtuvo en la clase *Ciudad* de la API y la opción de conversión (Kelvin o Fahrenheit) como argumentos y realiza la conversión según la opción especificada.

```
29      @staticmethod
30      def convertir_temperatura(temp_ciudad, opcion_conversion):
31          if opcion_conversion == '1':
32              return temp_ciudad + 273.15
33          elif opcion_conversion == '2':
34              return (temp_ciudad * 9 / 5) + 32
35          else:
36              return None
37
```

- *def main():* Define la función principal del programa.

```
39  def main():
```

- *while True:* Comienza un bucle infinito para permitir al usuario realizar múltiples consultas sin salir del programa cada que se realiza la consulta.

  Se define la variable *ciudad = input('Ingresa el nombre de la ciudad...'*: La cual solicita al usuario que ingrese el nombre de la ciudad de la cual quiere obtener la información de temperatura o "exit" para salir del programa.

  temperatura_ac_ciudad = Ciudad.obtener_temperatura(ciudad): Llama al método estático *obtener_temperatura* de la clase Ciudad para obtener la temperatura actual de la ciudad ingresada por el usuario.

  Si se obtiene la temperatura (*temperatura_ac_ciudad is not None*), se muestra la temperatura actual en grados Celsius.

```
39  def main():
40      while True:
41          ciudad = input('''Ingresa el nombre de la ciudad para obtener su temperatura o
42                         escribe "exit" para salir: ''')
43
44          if ciudad.lower() == 'exit':
45              break
46
47          temperatura_ac_ciudad = Ciudad.obtener_temperatura(ciudad)
48
49          if temperatura_ac_ciudad is not None:
50              print(f'La temperatura actual en {ciudad} es de {temperatura_ac_ciudad}°C.')
51          else:
52              print('No se pudo obtener la temperatura.')
53
54          opcion_conversion = input("""Ingresa el tipo de conversión que deseas realizar:
55                      1. Celsius a Kelvin
56                      2. Celsius a Fahrenheit
57
58                      Ingrese tu opción: """)
59
```

- Se solicita al usuario que ingrese el tipo de conversión que desea realizar (Celsius a Kelvin o Celsius a Fahrenheit).

```
54          opcion_conversion = input("""Ingresa el tipo de conversión que deseas realizar:
55                      1. Celsius a Kelvin
56                      2. Celsius a Fahrenheit
57
58                      Ingrese tu opción: """)
59
```

- *temperatura_convertida = ConvTemp.convertir_temperatura(temperatura_ac_ciudad, opcion_conversion):* Llama al método estático *convertir_temperatura* de la clase *ConvTemp* para convertir la temperatura según la opción seleccionada por el usuario.

  Si se realiza la conversión correctamente (*temperatura_convertida is not None*), se muestra la temperatura convertida en la unidad correspondiente.

```
60          temperatura_convertida = ConvTemp.convertir_temperatura(temperatura_ac_ciudad, opcion_conversion)
61
62          if temperatura_convertida is not None:
63              if opcion_conversion == '1':
64                  print("")
65                  print(f'La temperatura convertida en {ciudad} es de {temperatura_convertida}°K.')
66                  print("")
67              elif opcion_conversion == '2':
68                  print("")
69                  print(f'La temperatura convertida en {ciudad} es de {temperatura_convertida}°F.')
70                  print("")
71          else:
72              print('Opción de conversión no válida.')
73
```

- El bloque *if __name__ == '__main__':* asegura que la función *main()* se ejecute solo cuando el script se ejecute directamente y no cuando se importe como un módulo en otro script.

```
74  if __name__ == '__main__':
75      main()
76
```

2. **EXPLICACION DEL CODIGO UML**

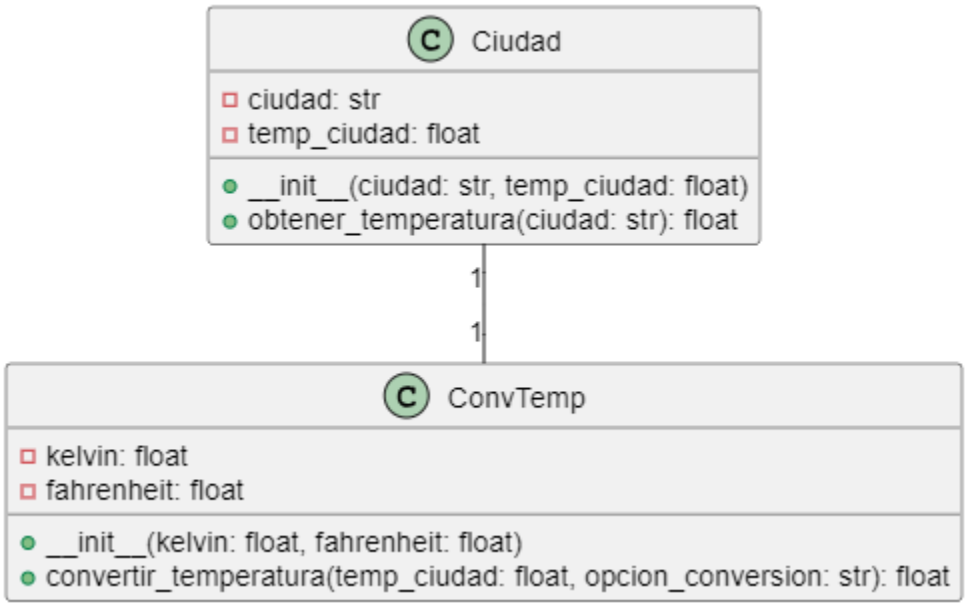
```
1   @startuml Conversor_Temperatura
2
3   class Ciudad {
4       - ciudad: str
5       - temp_ciudad: float
6       + __init__(ciudad: str, temp_ciudad: float)
7       + obtener_temperatura(ciudad: str): float
8   }
9
10  class ConvTemp {
11      - kelvin: float
12      - fahrenheit: float
13      + __init__(kelvin: float, fahrenheit: float)
14      + convertir_temperatura(temp_ciudad: float, opcion_conversion: str): float
15  }
16
17  Ciudad "1" -- "1" ConvTemp
18
19  @enduml
20
```

- **Línea 1:** Inicia un diagrama UML llamado “Conversor_Temperatura”.
- **Líneas 3-8:** Define una clase llamada “Ciudad” con los siguientes elementos:
    - Atributo “ciudad” de tipo cadena (str).
    - Atributo “temp_ciudad” de tipo flotante (float).
    - Método constructor “init” que inicializa la clase con los atributos “ciudad” y “temp_ciudad”.
    - Método “obtener_temperatura” que toma un parámetro de tipo cadena (ciudad) y retorna un valor flotante.
- **Líneas 10-15:** Define otra clase llamada “ConvTemp” con los siguientes elementos:
    - Atributo “kelvin” de tipo flotante (float).
    - Atributo “fahrenheit” de tipo flotante (float).
    - Método constructor “init” que inicializa la clase con los atributos “kelvin” y “fahrenheit”.
    - Método “convertir_temperatura” que toma dos parámetros, uno flotante y otra cadena, y retorna un valor flotante.
- **Línea 17:** Muestra una relación entre las clases “Ciudad” y “ConvTemp”,

indicando una asociación entre ellas.

- **Líneas 19:** Finaliza el diagrama UML.

3. DIAGRAMA DEL CODIGO EN UML



4. EXPLICACION Y USO

Este es un código relativamente simple en donde se implementan algunos pilares de POO para su desarrollo. El propósito de este código es obtener la temperatura de una ciudad en especifico que ingrese el usuario, y mediante los cálculos correspondientes pasar de la temperatura Celsius a Kelvin o Fahrenheit.

En este caso se uso un API externa para la obtención de temperatura, ya que no queríamos que el programa se limitara a algunas ciudades predeterminadas con posible información desactualizada respecto al clima.

En el código se le pide al usuario que ingrese una ciudad, cualquier ciudad, este parámetro será remplazado en la URL, mediante la cual se realizara la búsqueda de la temperatura en dicha ciudad, y mediante parámetros también definidos en el código se obtiene la temperatura de toda la información que ofrece la API y la imprime en grados Celsius para el usuario.

Después se le pide al usuario el tipo de conversión que desea realizar ya sea de Celsius a Kelvin o de Celsius a Fahrenheit. Después de que el usuario hace su elección se le da la temperatura

convertida de esa ciudad según la elección del usuario.

Ese es básicamente el funcionamiento del código, pero se debe tener en cuenta que si el usuario escribe el nombre de una ciudad que no existe se le enviara un anuncio diciendo que no existe, además de eso se tiene la opción de salir del programa mediante la palabra clave *exit*.

Y el tipo de entrada de texto no importa, se puede ingresar texto tanto en mayúsculas como en minúsculas como mixtas, e igualmente se realizará la búsqueda.

5. DEMOSTRACION DE PRUEBAS Y FUNCIONAMIENTO DEL CODIGO

5.1. Prueba 1.

```
Ingresa el nombre de la ciudad para obtener su temperatura o
escribe "exit" para salir: paris
La temperatura actual en paris es de 14.9°C.
Ingresa el tipo de conversión que deseas realizar:
                    1. Celsius a Kelvin
                    2. Celsius a Fahrenheit

                    Ingrese tu opción: 1

La temperatura convertida en paris es de 288.04999999999995°K.

Ingresa el nombre de la ciudad para obtener su temperatura o
escribe "exit" para salir: PARIS
La temperatura actual en PARIS es de 14.9°C.
Ingresa el tipo de conversión que deseas realizar:
                    1. Celsius a Kelvin
                    2. Celsius a Fahrenheit

                    Ingrese tu opción: 2

La temperatura convertida en PARIS es de 58.82°F.

Ingresa el nombre de la ciudad para obtener su temperatura o
escribe "exit" para salir: exit
```

En esta prueba se puede evidenciar la conversión de la temperatura de París a sus dos opciones tanto Kelvin, como Fahrenheit, además de eso se demuestra que el tipo de entrada de texto no difiere entre mayúsculas o minúsculas.

5.2. Prueba 2.

```
Ingresa el nombre de la ciudad para obtener su temperatura o
escribe "exit" para salir: madrid
La temperatura actual en madrid es de 18.6°C.
Ingresa el tipo de conversión que deseas realizar:
                    1. Celsius a Kelvin
                    2. Celsius a Fahrenheit

                    Ingrese tu opción: 1

La temperatura convertida en madrid es de 291.75°K.

Ingresa el nombre de la ciudad para obtener su temperatura o
escribe "exit" para salir: dubai
La temperatura actual en dubai es de 25.32°C.
Ingresa el tipo de conversión que deseas realizar:
                    1. Celsius a Kelvin
                    2. Celsius a Fahrenheit

                    Ingrese tu opción: 2

La temperatura convertida en dubai es de 77.576°F.

Ingresa el nombre de la ciudad para obtener su temperatura o
escribe "exit" para salir: EXIT
```

En esta prueba se realiza con la obtención de la temperatura de dos ciudades diferentes como lo es Madrid y Dubái, en donde cada una de estas arroja su respectiva temperatura, y se le realiza su correspondiente conversión según lo seleccionado, además de eso se muestra que el tipo de entrada de texto no difiere entre mayúsculas o minúsculas u ortografía tanto en la escritura de los nombres de las ciudades como al momento de pedir la salida del programa.

5.3. Prueba 3.

```
Ingresa el nombre de la ciudad para obtener su temperatura o
escribe "exit" para salir: bogota
La temperatura actual en bogota es de 22.73°C.
Ingresa el tipo de conversión que deseas realizar:
                    1. Celsius a Kelvin
                    2. Celsius a Fahrenheit

                    Ingrese tu opción: 2

La temperatura convertida en bogota es de 72.914°F.

Ingresa el nombre de la ciudad para obtener su temperatura o
escribe "exit" para salir: bogota
La temperatura actual en bogota es de 22.73°C.
Ingresa el tipo de conversión que deseas realizar:
                    1. Celsius a Kelvin
                    2. Celsius a Fahrenheit

                    Ingrese tu opción: 3
Opción de conversión no válida.
Ingresa el nombre de la ciudad para obtener su temperatura o
escribe "exit" para salir: exit
```

En esta prueba se demuestra la toma de temperatura con otra ciudad diferente a las anteriores, y el programa arroja su respectiva temperatura. Además de eso se demuestra que si al momento de elegir el tipo de conversión se da una opción invalida, este nos va a proporcionar un mensaje diciéndonos que la opción no es válida.

5.4. Prueba 4.

```
Ingresa el nombre de la ciudad para obtener su temperatura o
escribe "exit" para salir: jladfjd
Error al obtener la temperatura: city not found
No se pudo obtener la temperatura.
```

En esta prueba se demuestra que si no se da el nombre de una ciudad valida, este nos va a decir que no se encontró dicha ciudad y que la temperatura no se pudo obtener.

5.5. Prueba 5.

```
Ingresa el nombre de la ciudad para obtener su temperatura o
escribe "exit" para salir: oslo
La temperatura actual en oslo es de -1.24°C.
Ingresa el tipo de conversión que deseas realizar:
                    1. Celsius a Kelvin
                    2. Celsius a Fahrenheit

                    Ingrese tu opción: 2

La temperatura convertida en oslo es de 29.768°F.

Ingresa el nombre de la ciudad para obtener su temperatura o
escribe "exit" para salir: Vladivostok
La temperatura actual en Vladivostok es de -2.99°C.
Ingresa el tipo de conversión que deseas realizar:
                    1. Celsius a Kelvin
                    2. Celsius a Fahrenheit

                    Ingrese tu opción: 1

La temperatura convertida en Vladivostok es de 270.15999999999997°K.

Ingresa el nombre de la ciudad para obtener su temperatura o
escribe "exit" para salir: EXIT
```

En esta prueba podemos ver el ingreso de otras ciudades totalmente diferentes a las anteriores, la diferencia de esta prueba es que se demuestra las temperaturas bajo 0 de algunas ciudades, y el respectivo cálculo de conversión para estas.

6. DECISIÓN DE DISEÑO O CONSIDERACION IMPORTANTE

Para el diseño de este código como consideración importante tomamos que ciudades queríamos que se mostraran, en ese momento nos dimos cuenta de que no queríamos que el código se limitara a unas cuantas ciudades predefinidas con temperaturas predefinidas que podrían estar desactualizadas. Por tal motivo decidimos que la mejor opción para que este código fuera realmente de utilidad era usar un API externa que nos pudiera dar datos meteorológicos actualizados de cualquier ciudad que el usuario quiera revisar, por lo que en ese momento empezamos a realizar la investigación de cómo podríamos implementar un API gratuita y simple que nos diera dicha información, vimos que habían varias que podían prestar esos servicios pero nos decidimos por OpenWeatherMap ya que tenia una biblioteca de información bastante amplia en caso de que más adelante le quisiéramos adaptar mayor información meteorológica, además de

su simple uso y simple inscripción.

Por lo que de esta forma conseguimos que el programa fuera funcional totalmente con información útil, actualizada, y de cualquier ciudad del mundo soportada por el API.

7. PILARES DE POO IMPLEMENTADOS

7.1. Encapsulamiento:

Se utilizan atributos privados (self.ciudad, self.temp_ciudad, self.kelvin, self.fahrenheit) que solo son accesibles dentro de la clase.

Los métodos obtener_temperatura y convertir_temperatura son métodos estáticos (@staticmethod), lo que significa que no requieren una instancia de la clase para ser llamados. Esto puede considerarse como una forma de encapsulamiento, ya que estos métodos están asociados con la clase, pero no dependen de instancias específicas de esa clase.

7.2. Abstracción:

La clase Ciudad y ConvTemp representan conceptos abstractos de una ciudad y la conversión de temperatura, respectivamente. Estas clases encapsulan comportamientos relacionados con esos conceptos.

Los métodos obtener_temperatura y convertir_temperatura ocultan los detalles de implementación al usuario final, permitiendo que estos procesos complejos sean abstractos y utilizables mediante interfaces más simples.

8. CONCLUCIONES

Este es un código simple para obtener la temperatura actual de cualquier ciudad ingresada por el usuario mediante una API, y además de eso permite convertir la temperatura a Kelvin o Fahrenheit.

Podemos concluir que la realización de este código nos ayudo a reforzar nuestros conocimientos sobre POO y sus pilares, ya que en este código se implementaron algunos de ellos de manera sutil. Además de eso pudimos aprender el correcto manejo e implementación de una API como en este caso lo fue OpenWeatherMap con propósitos de la obtención de la temperatura, y que su implementación fuera exitosa para los propósitos del código. Además de eso se manejó la

interacción con el usuario mediante un menú simple de entender y manejar. Además, se reforzo lógica mediante los cálculos necesarios para realizar la conversión.

En resumen, el código proporciona una funcionalidad básica pero útil para obtener y convertir la temperatura de una ciudad, utilizando la API de OpenWeatherMap para obtener los datos de temperatura actuales.