Programación Avanzada
Daniel Gregorio Longino
Tarea 2

Al llegar a la línea 99 la variable **data** tiene una estructura de *diccionario*, este diccionario consiste de dos objetos: 'meta' y 'objects' (ver figura 1). El valor del objeto 'meta' es el diccionario

{'limit' : 5000, 'next' : None, 'offset' : 0, 'previous' : None, 'total_count' : 82}.

Por otro lado, el objeto 'objects' tiene una estructura de lista. Los elementos de esta lista son diccionarios de la forma

{'date' : 'x', 'frequency' : w, 'happiness' : 'z', 'timeseries' : '/api/v1/timeseries/3/'},

El valor del objecto 'date' es un cadena de caracteres 'x' del tipo str, donde x es una fecha entre el primero de enero de 2022 al 23 de marzo de 2022, y es de la forma 2022 − mm − dd. En el caso del objeto 'frequency', su valor es un número real w que depende de x. El valor del objeto 'happiness' es 'z', donde z es un número real. Y el orden en que aparecen estos diccionarios en la lista es en orden creciente, respecto a las fechas mencionadas anteriormente.

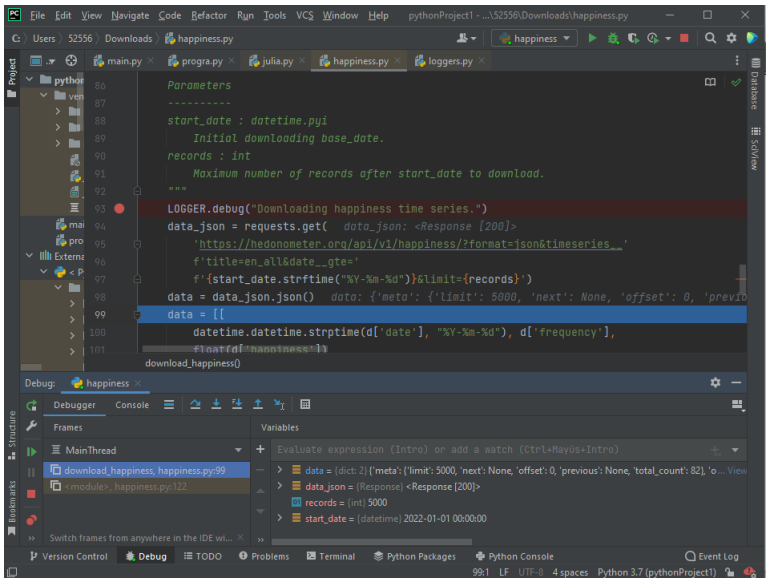

Figura 1: Estructura de la variable data al llegar a la línea 99.

Ahora que ya tenemos la variable data, lo que nos interesa es extraer los valores de los objetos 'date', 'frequency' y 'happiness' para cada fecha particular

y agregarlos en una lista. Esto se hace mediante las líneas de código 99-102 usando lo que se conoce como 'list comprenhensions'. List comprehensions, es una funcionalidad que nos permite crear listas avanzadas en una misma línea de código y tiene la ventaja de que son mas declarativos que los loops, lo que significa que son más fáciles de leer y entender. En lugar de crear una lista vacía y agregar cada elemento al final, simplemente podemos definir la lista y su contenido al mismo tiempo siguiendo este formato:

$$\text{new\_list} = [\textit{expresión} \text{ for } \textit{miembro} \text{ en } \textit{colección}]. \tag{1}$$

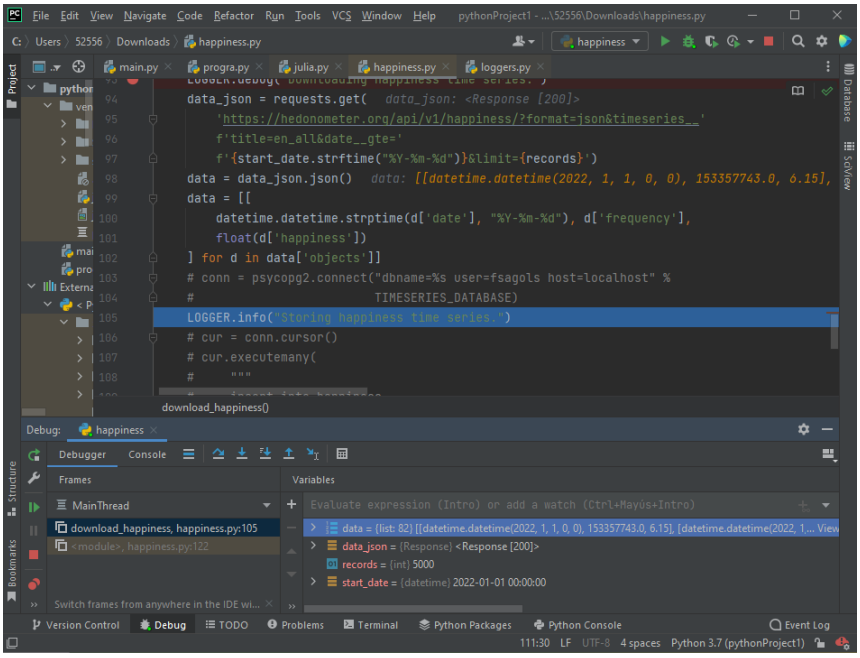

Figura 2: Estructura de la variable data al terminar de ejecutar la línea 102.

Cada list comprehension en Python tiene tres elementos

1. Expresión. Es el propio miembro, una llamada a un método o cualquier otra expresión válida que devuelva un valor.
2. Miembro. Es el objeto o valor en la lista o colección.
3. Colección. Es una lista, conjunto, secuencia, generador o cualquier otro objeto que puede devolver sus elementos uno a la vez.

En nuestro caso usando list comprenhension vamos a crear una lista de listas, en donde la expresión es

[datetime.datetime.strptime(d['date'], '%Y-%m-%d'), d['frequency'], float(d['happiness'])] (2)

Esta expresión es el miembro de nuestra nueva lista en sí. Esto es una lista que contiene tres elementos: los valores de los objetos 'date', 'frequency' y 'happiness', con algunos cambios particulares. Por ejemplo, el valor de 'happiness' lo estamos convirtiendo a un número real. Y el valor de 'date' lo estamos modificando usando el módulo datetime (ver figura 2). El miembro es d y la colección de donde estamos tomando este objeto es data['objects'], la cual es una lista, cuyos elementos ya los habíamos descrito al inicio.

Finalmente, la diferencia entre print(data) y print('\n'.join(str(d) for d in data)), es que el primero imprime el valor de data en una sola línea, estos es, como data es una lista, entonces print(data) imprime dicha lista, en una única línea (ver figura 3). Mientras que la segunda instrucción crea una cadena donde une todos los elementos la lista data, pero con la caracteristica de que imprime solo un elemento por línea (ver figura 4), y esto se debe a que el separador es '\n' (salto de línea en python).

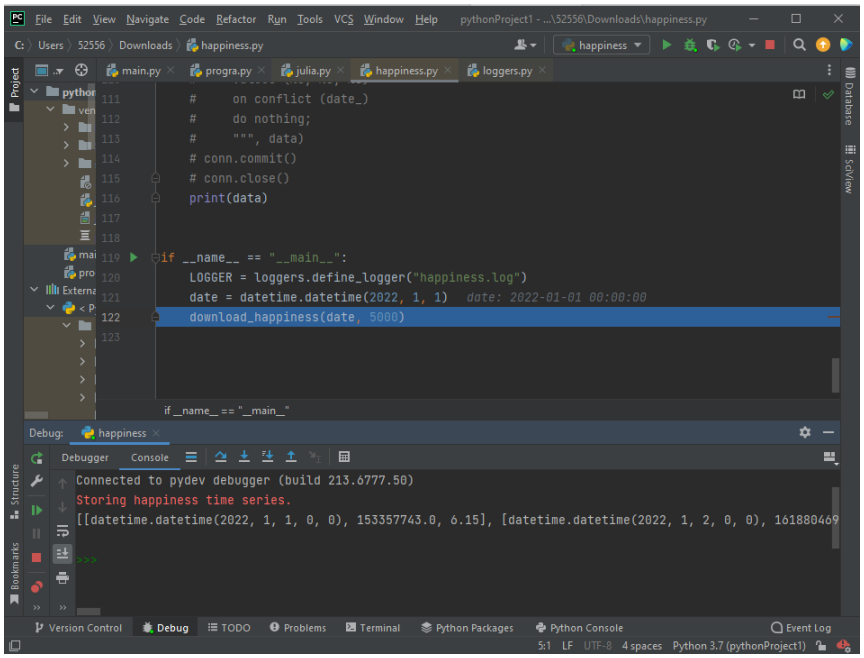

Figura 3: print(data).

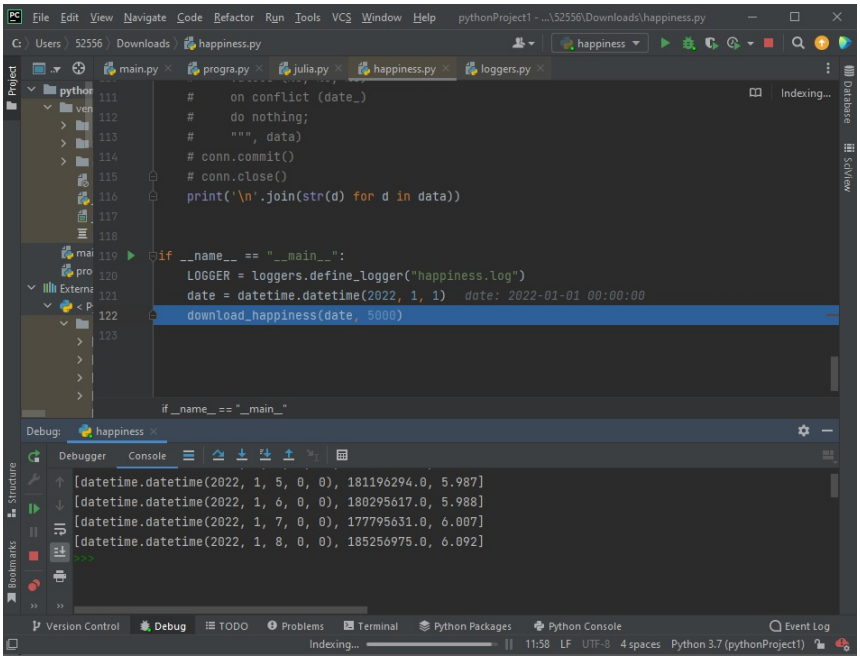

Figura 4: print('\n'.join(str(d) for d in data)).