Classes, Dataclasses and f-strings

Classes form the heart of the object-oriented programming (OOP) philosophy that powers the Python language. The syntax of, and the philosophy behind, the language is still actively evolving.

You may have noticed there are some annoying boiler-plate aspects of writing your own classes. Or you have been bitten by the dynamically typed aspect of Python. Here, we will explore why the Dataclass arose and what its potential is.

Similarly, you may have found the string formatting methods that Python offers fairly clumsy in syntax, and restrictive in function. These criticisms are what f-strings are designed to mitigate.

---

1. We have been asked to write an application for a Hollywood movie studio. Define a class, `Actor`, that has `name`, `age` and `acted_in` attributes.

2. The acted_in property should be a list of strings. Define a method, `num_acted_in`, that returns the number of movies this actor acted in.

3. Instantiate this class with, e.g. the values `Kevin Bacon, 50, ['Footloose', 'Tremors', 'Sleepers']`

4. Obtain the number of movies your actor acted in.

5. Define the same class using a `Dataclass`. Age should be an integer, Name a string (you may split into first_name, surname) and acted_in a list of strings.

6. Create a print method, print, for your class to produce output following this format "Kevin Bacon is 50 years old and has acted in 3 movies". Use the .format() method.

7. Rewrite the method in 6) above using an f-string.

8. Use an f-string to write a short method , plus_n_years(n), which outputs "In 3 years, Kevin Bacon will be 53".