Lecture 28

Review

20 Floréal, Year CCXXX

Song of the day: ***Enti Warraks*** *by Mohammed Jamal (1992).*

Eeveelutionary Theory

As you may remember from the review, the Pokémon **Eevee** has several type-specific **evolutions**. Eevee will evolve into some of these Pokémon depending on which **evolution stone** it is given. For the purposes of this problem, we'll focus on the original three Eeveelutions:

- **Vaporeon** (`"water"` *-type*): If Eevee is given a *Water Stone* (`"water_stone"` in code).
- **Jolteon** (`"electric"` *-type*): If Eevee is given a *Thunder Stone* (`"thunder_stone"` in code).
- **Flareon** (`"fire"` *-type*): If Eevee is given a *Fire Stone* (`"fire_stone"` in code).

Define an `Eevee` class that will accept two parameters: `nickname`, a string containing the nickname given to the `Eevee` object, and `description_filepath`, a string containing the address of a `txt` file containing a short description of this specific `Eevee` object. The file, if it exists, will always look slightly different depending on the Eevee's specific stats, but it will always be of the following general format:

```
Eevee
Nature: friendly
Level: 5
```

The `Eeevee` class will have three instance attributes:

- `nickname`: The nickname passed by the user.
- `eeveelution_status`: The current Eeveelutionary status of the `Eevee` object. The value of this attribute is `None` when the Eevee is un-evolved, or either `"Vaporeon"`,

`"Jolteon"`, or `"Flareon"` if it has been evolved. We can assume that all newly instantiated `Eevee` objects start un-evolved.

- `type`: A string containing the current type of the `Eevee` object. We can assume that all newly instantiated `Eevee` objects are of type `"normal"`.
- `nature`: A string containing the nature denoted in the file. Will be `None` if file is not successfully opened.
- `level`: An integer containing this object's level as denoted in the file. Will be `1` if file is not successfully opened.

The `Eevee` class will have an instance method associated to it called `evolve()` (***sig:*** `str => None`). `evolve()` will accept one parameter, `stone_name`, a string containing the name of the evolutionary stone that the user wishes to give to this `Eevee` object. When appropriately called, `evolve()` will update the `eeveelution_status` and `type` of the `Eevee` object **if and only if**:

1. The `Eevee` object is un-evolved,
2. The `stone_name` is one of the three valid evolutionary stones mentioned above

Once this method is called, the `eeveelution_status` and the `type` of the `Eevee` object will change to their appropriate new values.

You may assume that the following dictionary is already defined at the top of your file, which you may use if you find it useful:

```
INFO_PER_STONE = {
    # Water stone information
    "water_stone": {
        "eeveelution": "Vaporeon",
        "type": "water"
    },

    # Thunder stone information
    "thunder_stone": {
        "eeveelution": "Jolteon",
        "type": "electric"
    },

    # Fire stone information
    "fire_stone": {
        "eeveelution": "Flareon",
        "type": "fire"
    }
}
```

Next, define a `get_stats()` method (***sig:*** `None => list`) that will return a list of the values of all the object's non-`None` attributes. If you can, do this using list comprehension—only then will you become a Python deity.

Sample behaviour:

```
# If the file exists...
camille = Eevee("Camille", "description.txt")
print(camille.get_stats())  # pre-evolution

camille.evolve("thunder_stone")
print(camille.get_stats())  # post-evolution

# If the file doesn't exist...
fryderyk = Eevee("Fryderyk", "not_description.txt")
print(fryderyk.get_stats())  # pre-evolution

fryderyk.evolve("fire_stone")
print(fryderyk.get_stats())  # post-evolution
```

Output:

```
['Camille', 'friendly', 5, 'normal']
['Camille', 'Jolteon', 'friendly', 5, 'electric']
['Fryderyk', 5, 'normal']
['Fryderyk', 'Flareon', 5, 'fire']
```