Class name: Cat (from cat.py)

Class description:

For the purpose of this assignment, I have decided to create this class called “Cat” to simulate what it is like to own a pet cat. This class takes in 4 arguments: the name of the cat, its fur color, and its favorite toy and food. In addition to these factors, each cat has its own affection level count, as well as a boolean factor for if the cat is clean or not. The name and fur inputs are mostly to just make it easier for users to identify one cat from another if they choose to create multiple cat objects. The favorite food and toy inputs are there so that when users decide on doing an activity with their cats, choosing between their favorite food/toy or something else changes how many affection points they will be getting from that activity.

Description of class variables:

Affection

- This variable keeps track of how many affection points each cat object has obtained after going through a set of activities. Each cat always starts off at 0, and depending on the activity, gains or loses points from said activity.

Clean

- This variable stores a boolean value which determines if a cat is clean or dirty, and can be changed through the “outside” and “bath” activities. This status can be seen when returning the overview of a cat object under the category of “is clean” (returns “True” for clean, and “False” for dirty).

Description of data variables:

Self.name

- This variable stores the name of the cat. It mostly exists for identifying purposes.

Self.fur

- Like the name variable, the fur color variable mostly exists for identifying purposes so that it adds a visual aspect to the cat.

Self.fav_toy

- This variable stores the cat’s favorite toy, and can be used to gain more affection points when performing the “play” action.

Self.fav_food

- This variable stores the cat’s favorite food, and can be used when feeding the cat to earn more affection points.

Description of methods:

Play

- This action takes in a toy argument in which it will be compared to the favorite toy of the cat to see if it meets the requirements for receiving more affection points. If a user fails to put in the correct toy name, it will still add points to the total of affection points for the cat, but not as much.

Bath

- This activity, although it decreases the amount of affection points for the cat, it will change the boolean value status of the cat to “True”, regardless of if the cat is already clean.

Outside

- This activity is a very simple simulation of what happens when a cat goes out to play or explore outside. Its main purpose is to change the boolean value of the “is clean” category to “False”.

Feed

- This method takes in a food argument and compares it to the favorite food of the cat to determine the amount of affection points it will receive. (Same concept as “play”.)

Pet

- When called, this method adds a small amount to the total affection points of a cat.

Ignore

- When called, this method subtracts a significant amount of affection points from a cat.

Get_affection_advice

- This method not only allows users to see the name of the cat and their affection points, it also allows them to put into perspective how well they are treating their cat. If the affection points of that cat exceed 10, they are congratulated by a message, while if points are lower than -5, users are warned to treat their cat better.

Get_cat_overview

- This method is a useful tool that lets users see in one glance of how their cat is doing overall. When called upon, the method lists each category and status of the cat that they are analyzing.

Description and steps on how to run the program:

Although this class can be run through a program editor like the example shown in the file, since this class requires to be updated in real-time, I believe that the best way to run this program is through a terminal. In this tutorial, I will be explaining the steps on how to import the code onto your terminal, as well as creating and interacting with your new cats.

1. First, open the terminal. Make sure that you navigate to the correct file directory that has the cat.py file.
2. Next, type in “python3” and press enter. You will see three arrow keys on the left side with your text cursor next to it. Type in “from cat, import Cat”. (It is important to write the capitalization correctly, or else the terminal will print an error message. Press enter.

```
PS C:\Users\sarah\documents\ucsc_documents\CSE20> python3
Python 3.9.9 (tags/v3.9.9:ccb0e6a, Nov 15 2021, 18:08:50) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from cat import Cat
```

3. Create your first cat by setting a variable name (such as “cat1”). To initiate the cat object, you must type “Cat(‘name’, ‘fur color’, ‘favorite toy’, ‘favorite food’)”. These will be the identifying factors of your new cat. Users will be able to create multiple cats this way. Press enter.

```
>>> cat1 = Cat("Sammy", "red", "yarn", "chicken")
```

4. To perform activities with your cat, simply put your cat's variable name (in this example: cat1) and type in the method you would like to perform with it with a dot in between. After you press enter, the status of your cat should be up to date.

```
>>> cat1.pet()
>>> cat1.get_cat_overview()
 Name: Sammy
 Fur Color: red
 Favorite Toy: yarn
 Favorite Food: chicken
 Affection Level: 0.1
 Is Clean: True
```