Computer Programming (FIN2017), S20

Practice: Dog

20200605

Dogs that Bark!

Add a `bark()` method to the `Dog` class. To invoke the `bark()` method, you have to specify the number of times the dog instance ***woofs***.

For example, let `buddy` be a Dog instance, the output of `buddy.bark(3)` will be

```
woof! woof! woof!
```

Dogs that Do More!

Consider the following `Dog` class.

```
class Dog:
    count = 0
    def __init__(self,name):
        self.name = name
        self.tricks = list()
        Dog.count += 1
    def addTrick(self,trick):
        tricks.append(trick)
```

Modify the above `Dog` class to add the following functionalities.

1. The id attribute
   Add a data attribute *id* to each `Dog` object, so that *id* represents the order that a `Dog` instance is created. For example, the *id* of the first `Dog` instance is 1. At the same time, implement the method `get_id()` which returns a dog's *id*. Hint: When instantiating a "Dog" instance, derive its *id* from the class variable *count*.
2. The tricks
   Modify the `Dog` class so that if you add the same trick to a `Dog` object more than once, only one copy is saved.
3. The `__str__()` method
   Implement the `__str__()` method to format a `Dog` instance as `dog_name (dog_id): trick1, trick2, ...` — sort the tricks with the built-in Python function `sorted()` or method `sort()`.

A code template to verify your program is as follows.

```python
class Dog:
    # your implementation

if __name__ == '__main__':
    fido = Dog('fido')
    buddy = Dog('buddy')

    fido.add_trick('sit')
    fido.add_trick('play dead')

    buddy.add_trick('roll over')
    buddy.add_trick('sit')
    buddy.add_trick('sit')

    print(fido)
    print(buddy)
```

The output should be:

```
fido (1): play dead, sit
buddy (2): roll over, sit
```