CS 111 first name: Yuan last name: Hu
Quiz 5 BU ID: U26611932

Suppose that you are writing a class called `Temperature` that serves as a blueprint for objects that represent the temperature (in *degrees* and *scale*).

For example, the following lines:

```
t1 = Temperature(10, "C")
t2 = Temperature(50, "F")
```

would create `Temperature` objects representing the temperatures 10 degrees Celsius and 50 degrees Fahrenheit.

1. (5 points) Define a `Temperature` class header and a constructor (`__init__` method) that can be used as shown above and allows the scale to be only the values of “C” or “F”. If the scale is anything else, print out an error message. For example, if the client tries to construct this object: `Temperature(5, 'liters')` the error message should read: “You can’t measure temperature in liters!”.

2. (3 points) Write a `__repr__` method that will be used when printing `Temperature` objects. Use the form "d degrees s”. For example, given the two temperatures shown above,

   `print(t1,'\n',t2)` should output:

   ```
   10 degrees C
   50 degrees F
   ```

1.

```
class Temperature:
    def __init__(self, num, scale):
        """ construct a Temperature object
        """
        self.degrees = num
        if scale == 'C' or scale == 'F':
            self.scal = scale
        else:
            print("You can't measure temperature in" + scale + "!")
```

2.

```
def __repr__(self):
    """ print in standard format """
    print(str(degrees) + " degrees " + scal)
```

3.

```
def value_in_celsius(self):
    """ returns the value of the temperature in degrees Celcius """
    if self.scal == 'C':
        return self.degrees
    else:
        return 5/9*(self.degrees - 32)
```

3. (5 points) Write a method `value_in_celsius(self)` that returns the value of the temperature in degrees Celsius. It must not modify the data attributes of the object.

   The relationship of degrees Farenheit to Celsius is: C = 5/9 (F-32).

   For example, given the two `Temperature` objects shown above,

   `print(t1.value_in_celsius(), t2.value_in_celsius())`should output:

   `10 10`

4. (5 points) Given the two `Temperature` objects declared above, we want the expression `t1 == t2` to evaluate to `True`. Write the method required to enable this functionality. In particular, you should implement the method re-using the `value_in_celsius` method that you wrote in (3) and comparing the results; do not re-implement this calculation.

5. (2 points) Assume that the `Temperature` class has a method definition for a method `add_n(self, degrees)`. (You do not need to write this method.) Write three lines of client code to create a new `Temperature` object (for your favorite temperature); call the `add_n` method on your object.

4. def __eq__(self, other):

```
    """ return true if two temperatures are equal."""
    comp_one = self.value_in_celsius()
    comp_two = other.value_in_celsius()

    return comp_one == comp_two
```

5. import Temperature

```
def try_object():
    my_temp = Temperature(20, 'C')

    my_temp.add_n(20)
```