BME 646/ ECE695DL: Homework 1

Shaswata Roy

17 January 2022

1 Introduction

The code highlights the use of classes, subclasses and concepts such as inherintance and polymorphism.

2 Methodology

The classes, functions and variables were defined as given in the specific problem. Specific variables were printed and test cases were run to verify the validity of the code.

3 Implementation

```
class Countries():
    #Answer 1
    def __init__(self,capital,population):
        self.capital = capital
        self.population = population

    #Answer 3
    def net_population(self):
        birth = self.population[0]
        death = self.population[1]
        last_count = self.population[2]
        current_net = birth-death+last_count
        return current_net

class GeoCountry(Countries):
    #Answer 4
    def __init__(self,capital,population,area):
        super().__init__(capital,population)
        self.area = area
        self.density = 0
```

```
    #Answer 6
    def density_calculator1(self):
        if len(self.population)==3:
            self.density = super().net_population()/self.area
        else:
            self.density = self.net_population()/self.area

    def density_calculator2(self):
        birth = self.population[0]
        death = self.population[1]


        if len(self.population)==3:
            self.population[2] = self.population[2]-birth+death
            self.density = super().net_population()/self.area
        else:
            self.population[3] = self.population[3]-birth+death
            self.density = self.net_population()/self.area

    def net_density(self,choice):
        if choice == 1:
            return self.density_calculator1
        if choice == 2:
            return self.density_calculator2

    #Answer 7
    def net_population(self):
        last_count = super().net_population()
        if len(self.population)==3:
            self.population.append(last_count)
        birth = self.population[0]
        death = self.population[1]
        second_last_count = self.population[2]
        last_count = self.population[3]
        current_net = birth-death+(second_last_count+last_count)/2
        return current_net

if __name__ == "__main__":
    country = Countries("Piplipol",[40,30,20]) #Answer 2
    obj = GeoCountry("Polpip",[55,10,70],230) #Answer 5
    fn = obj.net_density(1)
    print(fn())
```

The output for the above print function is None as the function itself is being returned and printed without being executed.

Test Cases

```
ob1 = GeoCountry('YYY', [20,100, 1000],5)
print(ob1.density)#0
print(ob1.population)#[20,100,1000]
ob1.density_calculator1()
print(ob1.density)#184.0
ob1.density_calculator2()
print(ob1.population)#[20, 100, 1080]
print(ob1.density)#200.0
ob2 = GeoCountry('ZZZ', [20, 50, 100], 12)
fun = ob2.net_density(2)
print(ob2.density)#0
fun()
print("{:.2f}".format(ob2.density))#8.33
print(ob1.population)#[20,100, 1080]
print(ob1.net_population())#960.0
print(ob1.population)#[20,100,1080,1000]
print(ob1.density)#200.0 (the value of density still uses the previous value of population f
############################
#You need not stress about this portion below. If your results match, then well and good. If
#density_calculators were not modified to take care of both length = 4 and length =3 of popu
#The homework did not ask you to do so, but if you want to hone your skills, go for it!!
############################
ob1.density_calculator1()
print(ob1.population)#[20, 100, 1080, 1000]
print(ob1.density)#192.0
ob1.density_calculator2()
print(ob1.population)#[20, 100, 1080, 1080]
print(ob1.density)#200
```

Output

```
0
[20, 100, 1000]
184.0
[20, 100, 1080]
200.0
0
8.33
[20, 100, 1080]
960.0
[20, 100, 1080, 1000]
200.0
[20, 100, 1080, 1000]
192.0
```

```
[20, 100, 1080, 1080]
200.0
```

This matches the expected output.

4 Lessons Learned

- The use of inheritance and consequently the use of subclasses along with the associated syntax such as super()
- Polymorphism and it's use in overwriting functions based on the requirements

5 Suggested Enhancements

Don't believe there was the need for the other sections for this particular assignment.