Herramientas Computacionales para Ciencias
Homework 5

Mauricio Sevilla*

04/03/2019

Rules

This week we are going to concentrate on classes on `python`.

The first point have to be saved on a file `problemN.py` where `N` symbolizes the number of the problem, the three files must be compressed on a file `UserUniandes.zip` or `UserUniandes.rar`, for example on my case it should be `j.sevillam.zip` or `j.sevillam.rar`.

Problem 1: Classes - Taking a Dog for a Walk [3/5]

On this part, we will focus on the classes and methods structures.

We are going to construct a small game, our class will construct a `Dog` and it will have some methods to make the program interactive.

This class will receive 3 variables, the `name`[1], `posx` and `posy`, where the last two correspond to the coordinates on x and y.

On the constructor, we have created variables that were passed as arguments, but there you can also create variables with arbitrary values. On this case you may also create and initialize

`self.awaken=False`	We are going to initiate the Dog as `slept`.
`self.hungry=False`	The Dog won't walk if is hungry.
`self.counter=0`	This will help us to know when the dog gets hungry.

We are going to use three methods:

- `Awake(self)`: If the dog is slept, wakes it up.
- `Move(self,x1,y1)`: This is the longest,
  - If the dog is hungry it won't move.
  - Then, if the dog is not hungry and it is not slept either, given the values of x_1 and y_1 it updates the positions by adding them to `self.posx` and `self.posy`, and the variable counter is increased by 1, but if it is slept it shouldn't move.
  - Additionally, if the variable `self.counter` is grater or equal than 3, the dog gets hungry (`self.hungry=True`).
- `Feed(self)`: resets the variable `counter`, and the dog is not hungry anymore.

Write your code to print the state of the dog, for instance: `is slept`, `no longer slept`, `hungry` and so on.

Test

To test our code, let us do the following

- Create a Dog at a given position, on my case the dog is called `Lambda` and it starts at $(0,0)$

```
MyDog=Dog('Lambda',0,0)
```

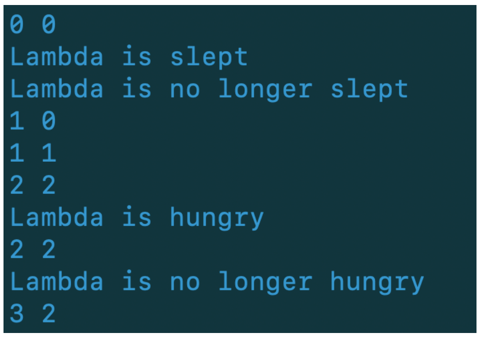
- Print the coordinates.
- Move it $(1,1)$ using the method `Move`, for example `MyDog.Move(1,1)`. it shouldn't move because the Dog s slept.
- Wake the Dog up.

*email=j.sevillam@uniandes.edu.co

[1] `name` must be a `str` with the name of the Dog we are creating.

- Now move it $(1, 0)$.
- Print the coordinates.
- Now move it $(0, 1)$
- Print the coordinates.
- Now move it $(1, 1)$
- Print the coordinates.
- Now move it $(1, 1)$. It shouldn't move because now is hungry.
- Print the coordinates.
- Feed the Dog.
- Now move it $(1, 0)$
- Print the coordinates.

The output should be something like



```
0 0
Lambda is slept
Lambda is no longer slept
1 0
1 1
2 2
Lambda is hungry
2 2
Lambda is no longer hungry
3 2
```

Figure 1: output for the problem 2

Problem 3: Inheritances [2/5]

To have an inheritance, we must have a class before as we did on class.

1. Create a class `vehicle`.
   - Color
   - Wheels
   - Max. Velocity

   Add the following methods
   - Move: print the value of a random velocity from 0 to Max. Velocity, to do so use the function `v=random.random()*self.VMax`[2]
   - Park: print a message: "The vehicle is parked"
2. Create the following inheritances of the class `vehicle` with the methods described,
   - Bicycle
     - Do some exercise: print the message "Doing exercise".
   - Motorcycle: also define a variable on the constructor for the size of the motor.
     - Put the helmet on: print "Helmet's on".
   - Car: also define a variable on the constructor for the size of the motor.
     - Turn on the radio: print "Radio's on".

[2]Don't forget to use `import random` at the beginning of the program