**Course: Introduction to Data Science (DS2006) - Laboratory 08**

**Student:**

**Task 1**: Create a file named [dog.py](http://dog.py) and implement the code shown in Figure 1. What is the output of this code when you run it?

#Define a class for Dogs:

class Dog:

#Defines the attributes (properties dogs have) and initializes them:

#(Attributes are variables that belong to a given class)

def \_\_init\_\_(self, name, age, breed):

self.name = name

self.age = age

self.breed = breed

#Method example:

def bark(self):

print(f"{self.name} looks at you and barks: Woof Woof!")

#Create an object from Class Dog:

dog1 = Dog("Bidu", 1, "Mixed")

#Create another object from Class Dog:

dog2 = Dog("Pipoca", 5, "German Sheperd")

**Figure 1. Code Snippet**

**Task 2**: Considering the current code we have in [dog.py](http://dog.py), what is the output of:

a) print([dog1.name](http://dog1.name))

b) print([dog2.name](http://dog2.name))

c) print([dog3.name](http://dog2.name))

**Task 3**: Create a new object of the class of Dog in a variable named dog3. Feel free to choose the name, age and breed of this new dog object.

**Task 4**: Replace the method shown in Figure 2 with the method shown in Figure 3. What happens when you try to run the code?

#Method example:

def bark(self):

print(f"{self.name} looks at you and barks: Woof Woof!")

**Figure 2. Code Snippet**

#Method example:

def bark():

print(f"{self.name} looks at you and barks: Woof Woof!")

**Figure 3. Code Snippet**

**Task 5**: Replace the method shown in Figure 3 with the method shown in Figure 4. What happens when you try to run the code?

#Method example:

def bark(self):

print(f"{name} looks at you and barks: Woof Woof!")

**Figure 4. Code Snippet**

**Task 6**: What is the output of calling dog1.bark() and print(dog1.bark())? Is the output the same? If not, what's the difference between them?

**Task 7**: Create a new attribute called **address** in [dog.py](http://dog.py) class and Update the necessary parts of the code (Hint: \_\_init\_\_ method and constructor calls).

**Task 8**: Create a new function called **sleep** in your [dog.py](http://dog.py) class. It should have no parameters to be used and print a message related to the dog being asleep.

**Task 9**: Create a new function called **barkAt** in your [dog.py](http://dog.py) class. It should take one parameter to be used to print a message that the dog is barking at the string that was passed as a parameter.

**Task 10**: Create a new file named [cat.py](http://dog.py). In this file define a cat class. The attributes of this class should be: Name and Breed.

**Task 11**: Create an **\_\_init\_\_** method for the initialization of Cat objects.

**Task 12**: In the Cat class, create one method that takes no parameters to be used and is related to something that cats do (be creative 🙂)

**Task 13**: In the Cat class, create one method that takes at least one parameter to be used and is related to something that cats do (be even more creative 🙂)

**Task 14**: Using the cat class methods you created, create 5 objects of the class cat and make them use the two different methods you have created. Show the code and output of this process.

Now we want to think about how we could refactor our latest version of the multiplayer Battle of Dices project using object oriented programming.

* **Task 15: Think and reflect individually** what classes do you need to Refactor our latest version of Battle of Dices? Write the classes and their \_\_init\_\_ methods.
* **Task 16: Think and reflect within your group** what classes do you need to Refactor the latest version of Battle of Dices? Write the classes, their \_\_init\_\_ methods and prepare a short presentation for the class. For the presentation please also add the google slides link in this document.