**Assignment 1 – Designing Classes Using Python**

Value: 40 Marks (worth 33% of Final Grade)

# Description

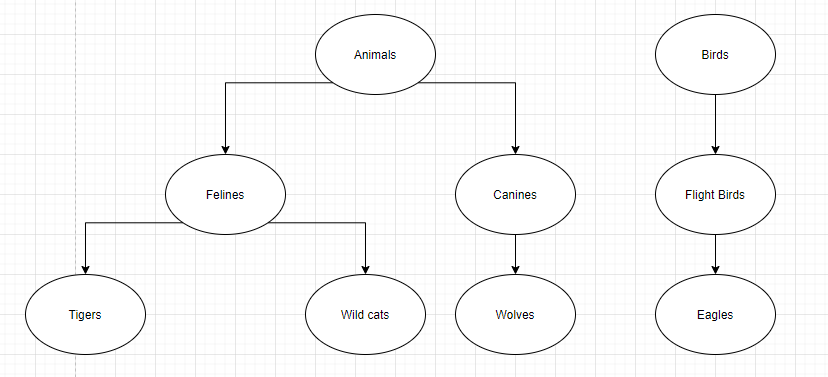
For this assignment, your goal is to apply various concepts for designing classes and creating proper parent child relationships to simulate real world scenarios.

This assignment is designed to measure and assess your ability to:

* Apply object-oriented programming techniques.
* Use the concept of inheritance.
* Produce quality code, including the way entities are modelled and how modelling represents the real world effectively. You will also be graded on how much code reuse is facilitated by your design.

# Instructions

For this assignment, you will create an ecosystem that represents a zoo. The zoo will have a variety of animals and birds. These animals and birds can be organized as shown in the following chart.



Each of these animal **(animal class should have properties/methods that will be inherited by its children. Example for properties number of legs and hands)** groups have some special characteristics that get passed on to the child. The child along with the characteristics derived from the parent will also have their own characteristics **(Example is felines could have a property representing the cat family. Canines also could have a property representing the dog family)**. This zoo **(zoo is a class as described later)** can save a fixed number of animals and birds **(When adding animal/bird object to the zoo class, you should not exceed this fixed number)** and is open to see the animals in them.

Design and represent the animals and birds in the above diagram as classes **(You must implement all the classes in the diagram and the zoo class)**in python. Design a zoo class to house these animals and birds. These classes should obey the requirements below.

**Requirements:**

1. Each animal has some common features **(These are the attributes/properties that should be defined in the animal class)** like the number of hands **(must be initialized to 0)** and legs **(must be initialized to 4)**.
   1. Felines and canines have 4 legs and no hands
2. Each bird has a number of legs and number of wings as a feature **(These are the properties that should be defined in the bird class)**.
   1. Flight birds have 2 legs and 2 wings (**Both properties must be initialized to 2)**.
3. Represent these features as attributes.
4. The characteristic for each of the animals are as follows
   1. Felines belong to the cat family **(an attribute should be declared in the Felines class and set to cat family)**.
   2. Canines belong to the dog family **(an attribute should be declared in the Canines class and set to dog family)**.
   3. Tigers can roar and are lethal predators **(an attribute should be declared in the Tigers class and set to roar and lethal predators)**
   4. Wild cats can climb trees **(Same as tiger)**
   5. Wolves hunt in packs and have a leader **(same as tiger and wild cats)**
   6. Flight birds fly and hunt for food **(an attribute should be declared in the FlightBird class and set to hunt for food)**
   7. Eagles fly extremely high and can see their prey from high up in the sky. **(same as FlightBird)**
5. Add the characteristics of animals and birds to their respective classes.
6. Create a zoo that can have 2 animals **(number of animals should not exceed 2 animals)** and 1 bird **(number of birds in the zoo should not exceed 1. When adding a bird/animal to the zoo, you must check these restrictions)**.
7. Zoo should be able to add **(add() method could be added to add animal/bird objects. Inside add(), you must check that number of animal objects does not exceed 2 and number of bird objects should not exceed 1 )**only an animal or a bird if it is not full
8. Zoo should be able to provide a way to look **(This is another possible method “looking() that display whether the zoo is empty or not. It also displays all animals and birds that are currently in the zoo)** at all the animals/birds it has.
9. Looking at animals/birds means you should be able to get all the features and characteristics of them **(looking() should display all the characteristics of each animal and bird object in the zoo)**.
10. **Possible methods in various classes**
    1. **Animal and bird classes could have a looking() that will display their characteristics**
    2. **Each child in the animal/bird tree could override looking() inherited from the animal class. They could also have a method to display the attribute that is specific to the child.**
    3. **All classes should have \_\_init\_\_()**
11. **After implementing all the classes, test them using the following:**

**zoo = Zoo()**

**zoo.add(Tiger())** # should display animal added

**zoo.add(Wolf())** # should display animal added

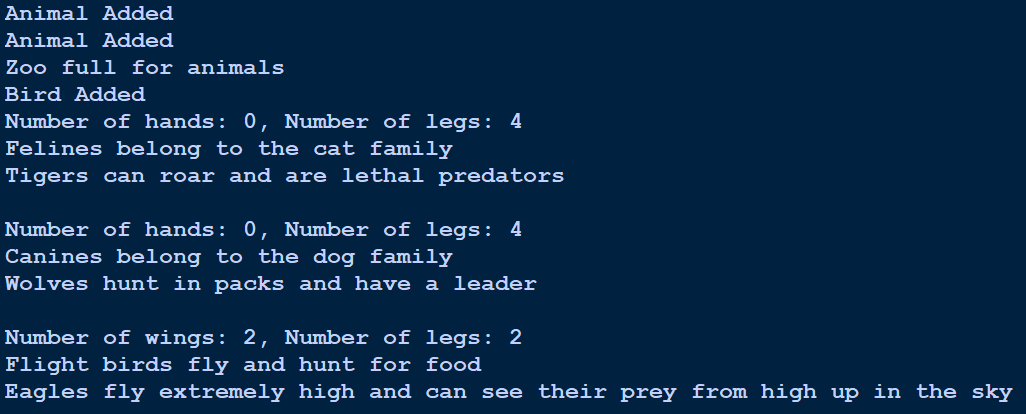
**zoo.add(WildCat())**# should display zoo full of animal

**zoo.add(Eagle())** # should display bird added

**zoo.looking()** # should display all the characteristics of animals/birds that have been

added to the zoo

**The following is a sample run. Please try to exactly generate it:**

****

1. Submit your python code and a screenshot of your zoo showing all its animals/birds to the instructor for grading:

* Go to the **Assignments** tab in Brightspace and click on **Assignment 1 – Designing Classes Using Python**.
* Click **Add a File** to upload your assignment to Brightspace.
* Click the**Submit** button.

# Assessment Criteria

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Criteria | Beginning (0-3) | Developing (4-6) | Accomplished (7-8) | Excellent  (9-10) | Score |
| Quality | Almost all the characteristics and features are missing for most of the classes. | A few features and characteristics are missing for most of the classes | Most of the features and characteristics are present with most of the class. A few are missing. | All features and characteristics are present/inherited by most of the classes. | /10 |
| Design | Classes are not designed well. No use of final, static, or private/protected access for attributes. | Classes are designed is fine. Noticeable misses in use of static and final and attributes access modifiers | Good design of classes. However, there are a few misses in use private/protected access for attributes and use of static/final. | The design is almost perfect. All the attributes are private with methods to access them. Static and final is used wherever required. | /10 |
| Output | The code does not run. | The code runs but the output does not match with the expectations. | The output is mostly correct but there are a few misses. | The output of the code is as per the expectation. | /10 |
| **TOTAL** | | | | | **/30** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Criteria | Beginning (0-1) | Developing (1-2) | Accomplished (3-4) | Excellent  (5) | Score |
| Efficiency | The code makes all the classes implement their features independently | The code makes most of the classes implement their features independently | A few classes where features could have been inherited implement them | Only a few classes implement the features and rest inherit them | 5 |
| Refactoring | The classes make no use of overriding | There is use of overriding but a lot of scenarios are missed | Most of the classes have overridden methods, however there are a few missed | Almost all the classes use overriding wherever possible. | /5 |
| **TOTAL** | | | | | **/10** |

**Total Grade = / 40**