Extending Classes

A major benefit of object-oriented programming is the ability to inherit classes and eliminate duplicate code. Inheritance allows you to define a new class based on an existing class.

Content Learning Objectives

After completing this activity, students should be able to:

- Read and interpret UML class diagrams for an existing code base.
- Evaluate pros and cons for designs with multiple similar classes.
- Define inheritance and demonstrate how to extend a base class.

Process Skill Goals

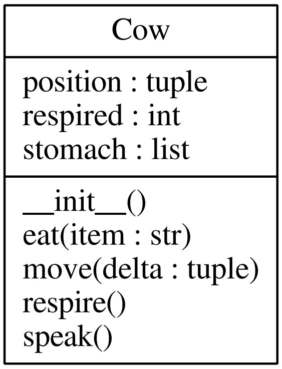
During the activity, students should make progress toward:

- Working with all team members to reach consensus on hard questions. (Teamwork)

Model 1 UML Class Diagrams

Unified Modeling Language (UML) provides a standard way of visualizing how programs are designed (http://www.uml.org/what-is-uml.htm). For example, a ***class diagram*** is a graphical summary of the attributes and methods of a class.

```python
class Cow:

    def __init__(self):
        self.position = (0, 0)
        self.respired = 0
        self.stomach = []

    def eat(self, item):
        if item == "grass":
            self.stomach.append(item)
            return "ate " + item
        else:
            return "This herbivore doesn't eat " + item

    def move(self, delta):
        self.position = (self.position[0] + delta[0],
                         self.position[1] + delta[1])
        return "This quadruped walked to " + str(self.position)

    def respire(self):
        self.respired += 1
        return "This cow respired through its nostrils."

    def speak(self):
        return "moo"
```



Cow

position : tuple
respired : int
stomach : list

__init__()
eat(item : str)
move(delta : tuple)
respire()
speak()

Questions (15 min) Start time: ________

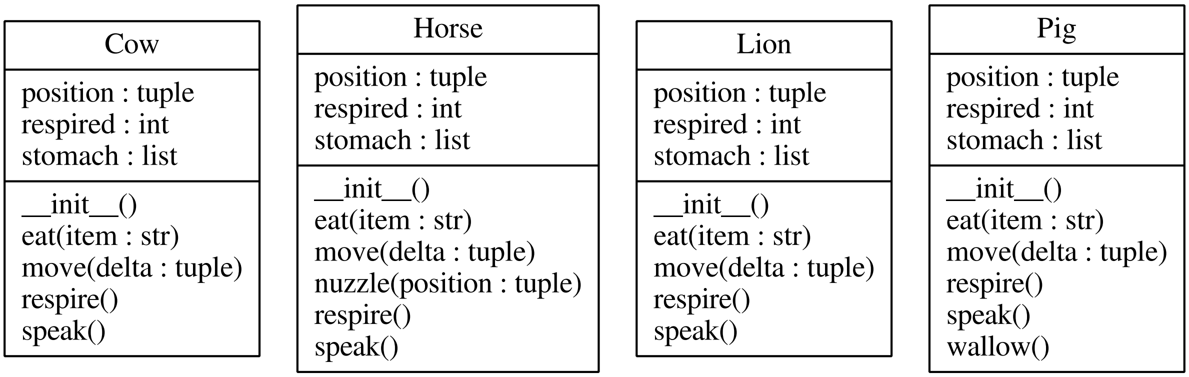
1. Draw an arrow from each name in the diagram to where it's defined in the code.

2. What are the attributes of `Cow`? What are the methods of `Cow`?

3. What is listed in each section of the UML class diagram?

a) Top section: b) Middle section: c) Bottom section:

4. Consider the following class diagrams:



a) What attributes do the classes have in common?

b) What methods do the classes have in common?

c) What methods are unique to a particular class?

5. Quickly examine the source code for each of the classes to identify similarities and differences. Write one or two words in each table cell to summarize your findings.

	Cow	**Horse**	**Lion**	**Pig**
eat	grass			
move	walked			
nuzzle	N/A			
respire	nostrils			
speak	moo			
wallow	N/A			

6. Consider what it would take to add a new method named `sleep` to each of the classes.

a) Describe the process of adding the same method to each source file.

b) If a mistake is found later on, how would you correct the method?

c) What problems do you see with this approach as more classes are added?

Model 2 Single-Class Approach

Given that the classes from Model 1 are similar, we could try combining them into a single class. In order to keep track of differences, we would need to store additional attributes. We could provide the information from Question #5 when creating an object:

```
angus = Animal("Cow", "grass", "walked", "nostrils", "moo")
```

The UML diagram below outlines this approach. As a team, discuss this design and become familiar with the accompanying source code.

Animal
eats_only : str moved : str name : str nasal : str position : tuple respired : int sound : str stomach : list

__init__(name : str, eats_only : str, moved : str, nasal : str, sound : str)
eat(item : str)
move(delta : tuple)
nuzzle(position : tuple)
respire()
speak()
wallow()

Questions (10 min) Start time: ________

7. Circle the three original attributes that were defined in Model 1.

8. Circle the two methods that were NOT common to all four classes in Model 1.

9. Write a statement that creates an `Animal` object representing a lion. Assign it to a variable named `simba`.

10. What methods does `simba` now have that it did not have in Model 1?

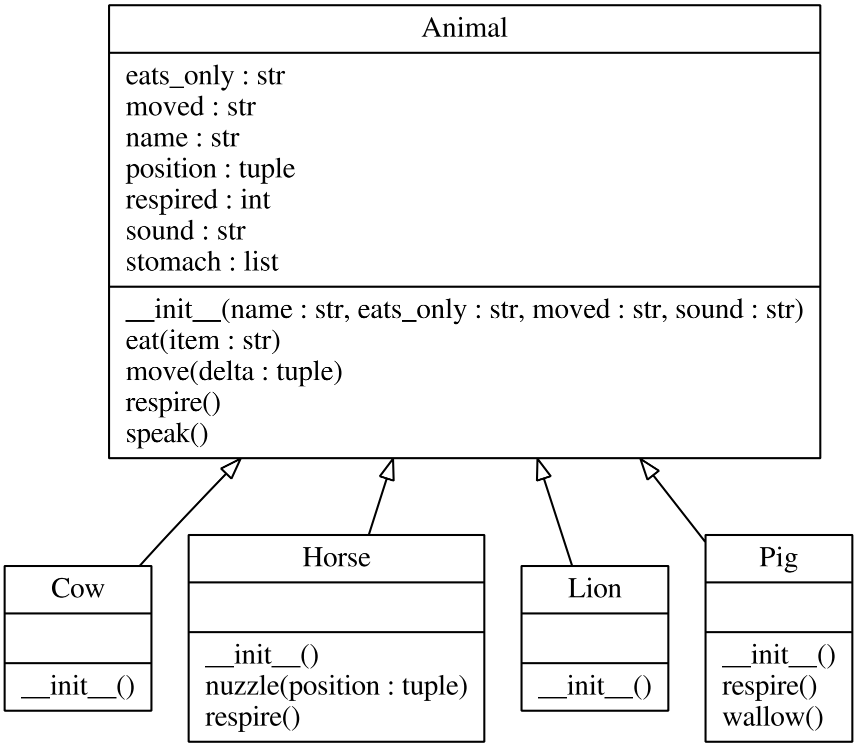
11. Describe 1–2 advantages the Model 2 design has compared to Model 1.

12. Describe 1–2 disadvantages the Model 2 design has compared to Model 1.

Model 3 Derived Classes

We can improve the code from Model 2 by using ***derived classes*** for `Cow`, `Horse`, `Lion`, and `Pig`. These classes only contain the attributes and methods specific to them. `Animal` is a ***base class*** that contains attributes and methods they all have in common. This language feature is known as ***inheritance***, because derived classes "inherit" attributes and methods from the base class.

The UML diagram below outlines this approach. As a team, discuss this design and become familiar with the accompanying source code.



Questions (20 min) Start time: ________

13. Open the lion.py source file. How many methods are defined in the Lion class? List the name of each one.

14. Type the following code into a Python Shell (in the same location as the source files). What methods are listed in the help?

```
from lion import Lion
help(Lion)
```

15. Write a statement that creates a Lion object. Assign it to a variable named simba. How is this statement different from Question #9?

16. In a Python Shell, what is the value of simba.speak()? Where did this value come from?

17. Does simba have any methods that it did not have in Model 1? Justify your answer.

18. Based on the source files, how does the __init__ method of Animal differ from the __init__ methods of the derived classes?

19. What is the meaning of the built-in function super() that is used in the derived classes?

20. Describe 1–2 advantages the Model 3 design has compared to Model 2.