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)

Facilitation Notes

This activity involves a lot more source code than usual, but it's doable if students don't get lost in the details. Encourage teams to manage their time well and not to think too deeply about any of the questions. Point out that UML is helpful to summarize the classes without looking at all the source code. Be sure to provide students with all the files in advance (see `animals.zip`).

On #5, have them quickly write a single word for each table cell. It's not that important what they write, as long as it connects back to the source code. The important part of Model 1 is the last question. Have more than one team share their answers with the class to facilitate discussion about object-oriented design.

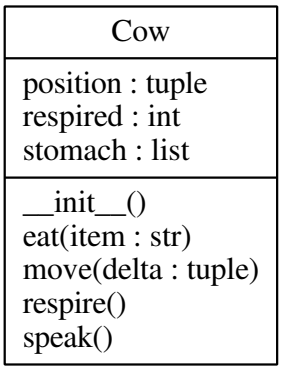
The attributes and methods in the source code (and UML diagrams) are sorted alphabetically, but the parameters of the `__init__` method are in a different order. You might want to point out this detail when students examine the source code from Model 2. Spend time reporting out the last two questions about the advantages and disadvantages of the single-class design.

As with the other models, the most important question for Model 3 is the last one. Be sure to save time for reporting out the advantages of using inheritance. If time permits, show other examples of inheritance (with UML class diagrams). In particular, GUI libraries make extensive use inheritance (see e.g., https://docs.python.org/3/library/tk.html).

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.

```
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"
```



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`?

The attributes are `position`, `respired`, and `stomach`. The methods are `__init__`, `eat`, `move`, `respire`, and `speak`.

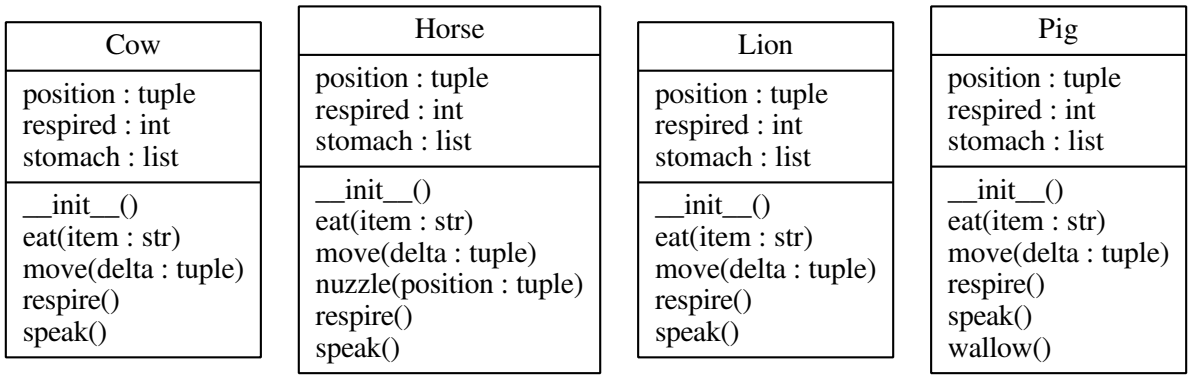
3. What is listed in each section of the UML class diagram?

a) Top section: class name

b) Middle section: attributes

c) Bottom section: methods

4. Consider the following class diagrams:



a) What attributes do the classes have in common?

All have the same three: `position`, `respired`, and `stomach`.

b) What methods do the classes have in common?

All five have `__init__`, `eat`, `move`, `respire`, and `speak`.

c) What methods are unique to a particular class?

Horse has `nuzzle`, and Pig has `wallow`.

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	grass	meat	everything
move	walked	galloped	walked	ran
nuzzle	N/A	moved	N/A	N/A
respire	nostrils	muzzle	nostrils	snout
speak	moo	nay	ROOOAR	oink
wallow	N/A	N/A	N/A	in water

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.

Write the method in one class, and copy/paste into the others.

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

Fix the problem in one class, and copy/paste into the others.

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

The amount of copying and pasting and other maintenance increases.

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.
Circled: position, respired, and stomach

8. Circle the two methods that were NOT common to all four classes in Model 1.
Circled: nuzzle and wallow

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

```
simba = Animal("Lion", "meat", "walked", "nostrils", "ROOOAR")
```

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

nuzzle and wallow

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

There's only one class. You don't have several copies of the same code lying around.

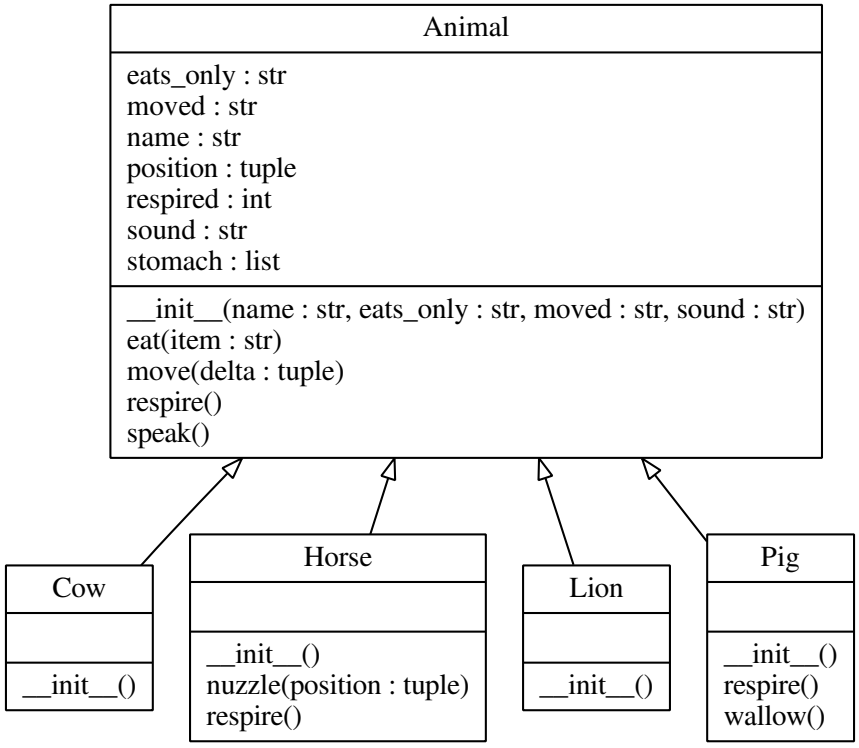
12. Describe 1–2 disadvantages the Model 2 design has compared to Model 1.

It's awkward to use the constructor. And not every object should have every method.

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.

Just one: the `__init__` method.

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)
```

In addition to the `__init__` method, it lists the methods inherited from animal.Animal (the base class).

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

```
simba = Lion()
```

It's much easier than before, because the `Lion` constructor is doing the work for you.

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

"ROOOAR". The constructor passed this value to `Animal`, which prints it out later.

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

No, it does not. It only inherits the same four that it had before.

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

`Animal` defines and stores the value of all the parameters. The other classes simply call `Animal`'s constructor.

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

It refers to the base class and is used to call the `__init__` method of `Animal`.

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

Every class has its own name, and objects don't have methods they're not supposed to have. The code is also easier to extend when adding new animals.