Lecture 24: Programming with Subclasses

CS 1110
Introduction to Computing Using Python

[E. Andersen, A. Bracy, D. Fan, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White]
Announcements

• Labs 17 & 18 released – treat your dis section this week as “study hall” for the labs and A5. *Bonus:* if you attend this week, your lab instructor will give you credit for one lab that you missed in the past, if you missed any

• **Assignment 5** due Wedn May 5\(^{th}\)

• Remember *academic integrity*!

• **Prelim 2** feedback released

• Lec 23 slides updated (added link to documentation on class object—*optional*—and corrected class diagrams on slide 12)

• **WICC** (student org Women in Computing At Cornell) Board Applications now open. For info see https://www.facebook.com/CornellWomenInComputing
Put Me in the Zoo

- Develop classes: Animal, Bird, Fish, Penguin, Parrot
- Instances can swim, fly, and speak based on class membership
- Track:
  - # of animals created (Q1)
  - name, tag #, weight for each animal (w/default weights)
- Methods:
  - print words if animal speaks
  - animal eats: print eating sounds and gain 1 pound
- Read the skeleton zoology.py
Questions to ask

- What does the class hierarchy look like?
- What are class attributes? What are instance attributes? What are constants?
- What does the `__init__` function look like?
- How do we support default weights?
- How do we implement the methods?
- What does a "stringified" Animal look like? `str(a)`
Q1: What is the best way to keep track of the number of Animals that have been created?

A: a global variable that you increment each time you call the Animal constructor
B: a class attribute inside the Animal class that is incremented by the Animal's __init__ method
C: an instance attribute inside each Animal that is incremented by the Animal's __init__ method
D: A & B both work, but B is better
E: A & B & C all work, but C is best
If `speak` is defined by the `Animal` class like this:

```python
def speak(self, words):
    if self.CAN_SPEAK:
        print(words)
```

Q2: Which subclasses need to provide their own version of this method?

A: Bird, Fish, Penguin, and Parrot
B: Bird and Parrot
C: just Parrot
D: none
E: I don’t know
If `eat` is defined by the `Animal` class like this:

```python
def eat(self):
    print("NOM NOM NOM")
    self.weight += 1
```

Q3: We want `Fish` to say nothing and `Birds` to make a pecking sound. Which subclasses need to provide their own version of this method?

A: `Bird`, `Fish`, `Penguin`, and `Parrot`
B: `Bird` and `Fish`
C: just `Bird`
D: just `Fish`
E: I don’t know
After lecture

• Implement class **Penguin**
  ▪ Penguins cannot fly but can swim
  ▪ Let’s say the default weight is 25 units
  ▪ You decide what it sound it makes when it eats
• Experiment! It’s the best way to learn
• *In lieu of pre-lecture reading for Thurs*, read, run, and experiment with module **zoo**, which sets up a **Zoo** and lets you interact with the animals. Check out how the module uses **Animal** and its subclasses