Lecture 21: Programming with Subclasses

CS 1110

Introduction to Computing Using Python

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

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

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)`

speak(words)

If speak is defined by the Animal class like this:

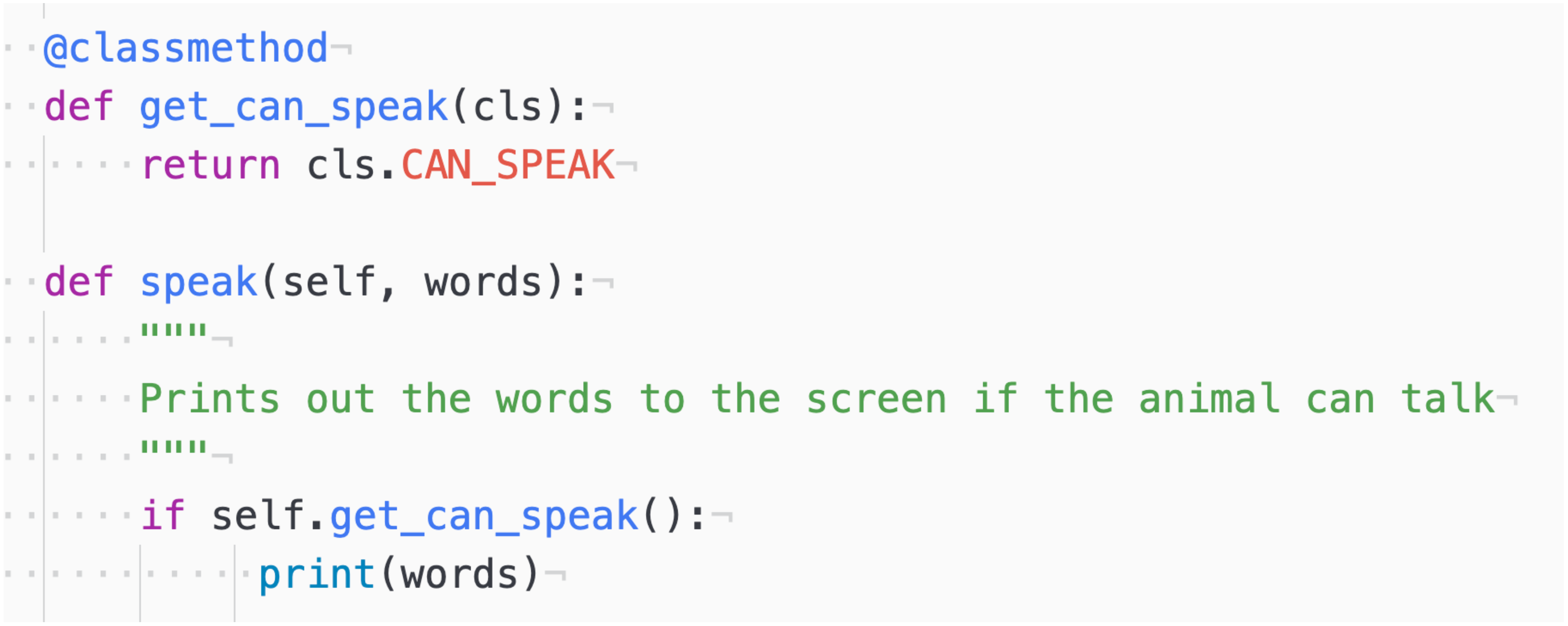
```
def speak(self, words):
    if self.CAN_SPEAK:
        print(words)
```

Q2: is this a good idea?

A: no, you're accessing a class attribute with self
B: looks good to me
C: I don’t know

@classmethod

- solution to the problem on the previous slide.



```
  @classmethod
  def get_can_speak(cls):
      return cls.CAN_SPEAK

  def speak(self, words):
      """
      Prints out the words to the screen if the animal can talk
      """
      if self.get_can_speak():
           print(words)
```

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
- 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