Attributes

Announcements

Iterable vs Iterator

- First and foremost, there are data structures
- Python doesn’t know how to iterate over data structures, especially if *you* defined the data structure and it was not built in
- It is the job of a data structure/type itself to decide how it should be iterated
- A data structure that implements the **`__iter__()`** function is an **iterable**
- The **`__iter__()`** function creates an **iterator:** a generator that iterates through the data structure (however **`__iter__()`** says it should)
- Whenever Python needs to iterate over something (such as in a **`for`** loop), it calls the **`iter()`** function on it and lets the data structure tell it how it should be iterated over (by getting an **iterator** from it via its **`__iter__()`** function)

Looking Up Attributes by Name

Both instances and classes have attributes that can be looked up by dot expressions

<expression> . <name>

To evaluate a dot expression:

1. Evaluate the <expression> to the left of the dot, which yields the object of the dot expression

2. <name> is matched against the instance attributes of that object; if an attribute with that name exists, its value is returned

3. If not, <name> is looked up in the class, which yields a class attribute value

4. That value is returned unless it is a function, in which case a bound method is returned instead