Inheritance

Announcements

Attribute Lookup Practice

Discussion Question: Class Attribute Assignment

Implement the **Place** class, which takes a **name.** Its **print_history**() method prints the **name** of the **Place** and then the names of all the **Place** instances that were created before it.

```
class Place:
    last = None
    def __init__(self, n):
        self.name = n
        self.then = Place.last
        Place.last = self

    def print_history(self):
        print(self.name)
        if self.then is not None:
            self.then.print_history()
```

OK to write **self.last** or **type(self).last**

Not ok to write **self.last**

```
>>> places = [Place(x*2) for x in range(10)]
>>> places[4].print_history()
8
6
4
2
0
>>> places[6].print_history()
12
10
8
6
4
2
0
```

Example: Friends

```
class Friend:
    def __init__(self, name: str):
        self.name = name
        self.just_talked = None
        self.talked_to = 0

    def talk(self, other):
        self.just_talked = other

        other.just_talked = self

        other.talked_to += 1

    def is_bestie(self, other):

        return self.just_talked is other and other.just_talked is self

    def has_bestie(self):

        return self.just_talked and self.is_bestie(self.just_talked)
```

How many times someone else talked to this Friend

```
>>> john = Friend('John')
>>> kay = Friend('Kay')
>>> josh = Friend('Josh')
>>> john.talk(kay)
>>> josh.talk(kay)
>>> john.talked_to
0
>>> kay.talked_to
2
>>> kay.is_bestie(josh)
True
>>> kay.is_bestie(john)
False
```

Inheritance

Inheritance Example

A CheckingAccount is a specialized type of Account

```
>>> ch = CheckingAccount('Tom')
>>> ch.interest     # Lower interest rate for checking accounts
0.01
>>> ch.deposit(20)  # Deposits are the same
20
>>> ch.withdraw(5)  # Withdrawals incur a $1 fee
14
```

Most behavior is shared with the base class Account

```
class CheckingAccount(Account):
    """A bank account that charges for withdrawals."""
    withdraw_fee = 1
    interest = 0.01
    def withdraw(self, amount):
        return Account.withdraw(self, amount + self.withdraw_fee)
```

or

```
        return super().withdraw(      amount + self.withdraw_fee)
```

Looking Up Attribute Names on Classes

Base class attributes *aren't* copied into subclasses!

To look up a name in a class:

1. If it names an attribute in the class, return the attribute value.

2. Otherwise, look up the name in the base class, if there is one.

```
>>> ch = CheckingAccount('Tom')  # Calls Account.__init__
>>> ch.interest     # Found in CheckingAccount
0.01
>>> ch.deposit(20)  # Found in Account
20
>>> ch.withdraw(5)  # Found in CheckingAccount
14
```

Where’s Waldo

(Demo)