Inheritance

Reading: Sec 8.1—8.5

Overview

What is it?

Why use it?

Class Syntax

basic class definition:

    class Card:

    ...

creates a new namespace

statements in the body of the class add names (assignment, def, class, ...)

[demo: ClassSyntax.py]
**Derived class syntax:**

```python
class BlackjackCard(Card):
    ...
```

creates a new namespace that is linked to Card

new instances of BlackjackCard can refer to own names plus names defined in Card

![Diagram of class relationships between Card and BlackjackCard]
How does Python evaluate the following expressions?

```python
>>> c1 = Card(12)
>>> c2 = BlackjackCard(25)
```

**Example**

```python
>>> c1._id
>>> c1.points()
>>> c2._id
>>> c2.points()
>>> c1.suit()
>>> c2.suit()
```

Make sure you understand the “big picture”

A Blackjack card is a Card

Anything you can do with a Card you can also do with a Blackjack card (unless that behavior is overridden in the Blackjack class)
**Terminology**

base/derived, parent/child, super/sub

stress: base is general structure, derived is specialization

More examples:

animals (from Perkovic)

```
class Animal
    setSpecies
    speak
    setLanguage

class Bird
    speak

object tweety
```

publications [.bib]

```
Publication
    Book
    Tech Report
    Thesis
    Paper
        Textbook
        Monograph
        Ph.D.
        M.S.
        Journal Article
        Conference Paper
            Peer-Reviewed
            Invited
```
Example Program: GiftCard.py and PhoneCard.py

Not very realistic, but it illustrates concepts from Ch 8 (and a few more OOP topics)

A gift card is created with an initial balance. A method named ‘use’ will subtract an amount from the balance if there are sufficient funds.

A phone card is like a gift card, except (a) it can be “topped up” and (b) the balance can go negative. The card cannot be used if the balance is negative.

Things to try (in the order shown):

```python
>>> x = GiftCard(25)
>>> x

>>> y = PhoneCard(25)
>>> y

>>> x.use(10)
>>> x.use(20)

>>> y.use(10)
>>> y.use(20)

>>> y.topup(50)
>>> y.use(20)
```
Class Variables and Class Methods

We've already seen class variables: a reference to an object that is shared by all instances of the class.

The variable is accessed using a qualified name (Classname.varname).

Qualified names are used inside and outside the class.

Example in GiftCard.py: the currency symbol (csym)

We can also have class methods: methods that operate on the class as a whole instead of any single instance of the class.

Again use qualified name syntax: Classname.foo(args).

Note that (a) when the method is defined it does not have 'self' as one of its arguments and (b) when the method is called no instance is passed as the first argument.

An example in GiftCard.py: set_currency

>>> GiftCard.set_currency('euro')

Question: is the new currency symbol used by PhoneCard objects?