Hyperion Development

Task: Object Orientated Programming (Part 2)

www.hyperiondev.com

Introduction

What is Inheritance?

Inheritance is a bit more complicated than the classes and objects examined previously. Suppose you had written a class with some properties and methods, and you wanted to define another class with most of the same properties and methods, plus some additional structure. Inheritance is the mechanism by which you can produce the second class from the first, without redefining the second class completely from scratch or altering the definition of the first class itself.

Alternatively, suppose you wish to write two related classes that share a significant subset of their respective properties and methods. Rather than writing two completely separate classes from scratch, you could encapsulate the shared information in a single class and write two more classes that inherit all of that information from the first class, saving space and improving code clarity in the long run.

A class that is inherited from is called the "base" class, and the class that inherits from the base class is called the "derived" class. In most major object oriented languages, objects of the derived class are also objects of the base class, but not vice-versa. This means that functions which act on base objects may also act on derived objects, a fact which is often useful when writing an object oriented program. (Note: this distinction is arguably more important in languages with a stronger type system; Python takes everything to an extreme by making all user-defined classes subclasses of a base type confusingly referred to as 'object'. If you are interested in the internals of the Python language (and you should be, because it is interesting!), master the material in this Task and then try looking online at some other material.)

To illustrate this concept better, read through the cow-themed demonstrations of composition, inheritance, and some other Python-specific object oriented language features located in **example.py**.

Now, open **exercise.py** and fill out the logic for the method definitions. Rename this file to **exercise_completed.py** when complete. You may leave any questions you have for your tutor in comments.txt.

-The Hyperion Team

Instructions

- Feel free at any point to refer back to previous material if you get stuck. Remember that if you require more assistance our tutors are always more than willing to help you!
- Complete the following two compulsory tasks to progress onto the next task!
- We highly recommend that you do the optional tasks located in **example.py**.

Compulsory Task 1

In this Task we're going to be creating classes to exhibit vectors and complex numbers.

- Open the file called **exercise.py**.
- Complete the instruction indicated in Part 1

Compulsory Task 2

In this Task we're going to be creating classes to create the game tic-tac-toe, also known as noughts and crosses.

- Open the file called **exercise.py**.
- Complete the instruction indicated in Part 2

Still need help?

Just write your queries in your comments.txt file and your tutor will respond.

Task Statistics

Last update to task: 08/01/2016.
Course Content Manager: Riaz Moola.
Task Authors: Riaz Moola and Brandon Haschick
Task Feedback link: Hyperion Development Feedback.