KA9010: Copy Management

Problem Domains:	Memory & Performance Management
Topic Summary:	Copying / Referencing Related Objects
Prerequisites:	Python 1000 & 2000 Series or Equivalent
Related Code:	KA9010.py
Related Topics:	KA9011, KA9012, KA9013, KA9014, KA9015
Version:	1.0 – ROUGH DRAFT

Question

What is a deep copy?

Answer

A deep copy is when objects collected by one class, are themselves copied, whenever the *containing* class is cloned.

Nagy's Notes

Most useful objects either contain, encapsulate, and / or refer to, other objects.

In Python, as in every object-oriented programming language of which I am aware, in order to save memory, as well as time, related objects can either be copied, or referred to.

Likewise, whenever we create a *copy* of any object *using* other objects, our choice is to either save space / time by *referring* to our dependent(s), or to simply *make a new copy* of each.

The related code demonstrates how we can use Python's build-in identification operations to see if two objects share the same copy.

Caveats: Whenever we choose to make `deeper` copies of any dependents, in addition to taking more space and time to clone each, we also risk making changes to one copy, which - by default - remain unapplied to any other copy.

Conversely, whenever we *share* references to but a single, common, *mutable* object instance – and again by default – updates to any *changeable* shared object(s) will instantly become available, to all.

Caveat concurrencies.