Problem

Using the del keyword on a Python object does not seem to really invoke the destructor

Solution

Q: After doing a del obj from a Python script where obj can be any Python object the destructor, __del__, of obj is never invoked.

Calling del on a Python object does not guarantee the destructor will be called! That simply decrements a counter internal to the object. It will invoke the destructor if (and only if) the internal counter, after it is decremented, has reached zero.

Q: Why doesn't the Python interpreter call the __del__ method of all Python objects contained within it before exiting?

There is not even a guarantee that the Python interpreter will invoke the destructors of all objects residing in it before exiting! This is part of the Python specification.

Workaround:

The only guaranteed way to ensure a destructor is called in Python is to call it in your own code like this:
obj.__del__()

Related articles

- How can more people do development with ACS on the same machine without disturbing each other?
- Which ports are used by ACS?
- Problems connecting to ACS servers on a remote machine: bad /etc/hosts
- Why does the getComponent method of ZLegacy/ACS.ContainerServices return an object of type None?
- Why are some of my print statements not showing up in the container output section of acscommandcenter?