

```

Employee.__doc__: Common base class for all employees
Employee.__name__: Employee
Employee.__module__: __main__
Employee.__bases__: ()
Employee.__dict__: {'__module__': '__main__', 'displayCount':
<function displayCount at 0xb7c84994>, 'empCount': 2,
'displayEmployee': <function displayEmployee at 0xb7c8441c>,
'__doc__': 'Common base class for all employees',
'__init__': <function __init__ at 0xb7c846bc>}
```

## Destroying Objects (Garbage Collection)

Python deletes unneeded objects (built-in types or class instances) automatically to free the memory space. The process by which Python periodically reclaims blocks of memory that no longer are in use is termed Garbage Collection.

Python's garbage collector runs during program execution and is triggered when an object's reference count reaches zero. An object's reference count changes as the number of aliases that point to it changes.

An object's reference count increases when it is assigned a new name or placed in a container (list, tuple, or dictionary). The object's reference count decreases when it's deleted with *del*, its reference is reassigned, or its reference goes out of scope. When an object's reference count reaches zero, Python collects it automatically.

```

a = 40      # Create object <40>
b = a      # Increase ref. count  of <40>
c = [b]    # Increase ref. count  of <40>

del a      # Decrease ref. count  of <40>
b = 100    # Decrease ref. count  of <40>
c[0] = -1  # Decrease ref. count  of <40>
```

You normally will not notice when the garbage collector destroys an orphaned instance and reclaims its space. But a class can implement the special