Working with Garbage Collection
When to delete an Object ?

- Whenever an Object is no longer in use. Then is recommended to delete an Object.
- Advantages are reusability of the memory
- Inorder to delete an object then we have to assign "None" to the reference .Whenever "None" assign to the reference then the object become "unreferenced object"
- The Unreferenced Objects will maintain and erased by the Garbage collection
- By Chance if we are trying get anything by the reference which is assigned with "None" type then PVM will raise "AttributeError"

Example:

```
class Sample:
   def __init__(self):
      print("Const is invoked ")
      self.x=10

   def getData(self):
      print("x val is : ",self.x)


#calling
s1=Sample( )
s1.getData( )
s1=None
#s1.getData( ) AttributeError

print("Object 2...")
s2=Sample()
s2.getData()
```

s2=None

Note: Whatever the resources are allocated by the time of object instantiation by using constructor. It is all our responsibility to perform the resource DE-Allocation by the time of object destruction

- In order to perform resource DE-Allocation then we have to use destructors
- Resource DE-Allocation is nothing but
  - Closing the Files
  - Closing the Databases
  - Network disconnections....

- In Python destructors are defined by using "__del__(self)"
- The destructor will be executed automatically by the PVM whenever an Object become unreferenced.

Example:

```
class Sample:
   def __init__(self):  #Constructor
      print("Const is invoked ")
      self.x=10

   def __del__(self):   #Destructor
      print("======================")
      print("Dest is Invoked ")
      print("Resource DE-Allocation")
      print("Object is Object Deleted ....!!!")

   def getData(self):
      print("x val is : ",self.x)
```

```
#calling
s1=Sample( )
s1.getData( )
s1=None
#s1.getData( ) AttributeError

print("Object 2...")
s2=Sample()
s2.getData()
s2=None
```

- In Order get Reference count of an Object then we have to use "getrefcount(Object)" from sys module

Example:

```
import sys
import time

class Sample:
   def __init__(self):
      print(" const is invoked ...")

   def __del__(self):
      print(" Dest is involed ....")

#calling
s1=Sample()
s2=s1  #ref.copy
s3=s2 #ref.copy
print("Ref.count is : ",sys.getrefcount(s1))
time.sleep(2)
```

s1=None
s2=None
s3=None

Note: By default Garbage Collection is all ways enabled state only.to ensure the state of the garbage Collection then we have to use “isenabled()” from “gc” module

2.Based on our application requirements we can also enable or disable the garbage collection by using “enable” and “disable” attributes of “gc” module.

Example :

```
import gc

print("GC is Enabled ? : ",gc.isenabled())
gc.disable()
print("GC is Enabled ? : ",gc.isenabled())
gc.enable()
print("GC is Enabled ? : ",gc.isenabled())
```