Python Programming

EDUCATION FOR PEACE

(ESTD-1995)

RGM College of Engineering & Technology (Autonomous)

Department of Computer Science & Engineering

AY:2021-2022

PYTHON’S OBJECT ORIENTED PROGRAMMING - 7

Guido Van Rossum

Agenda:

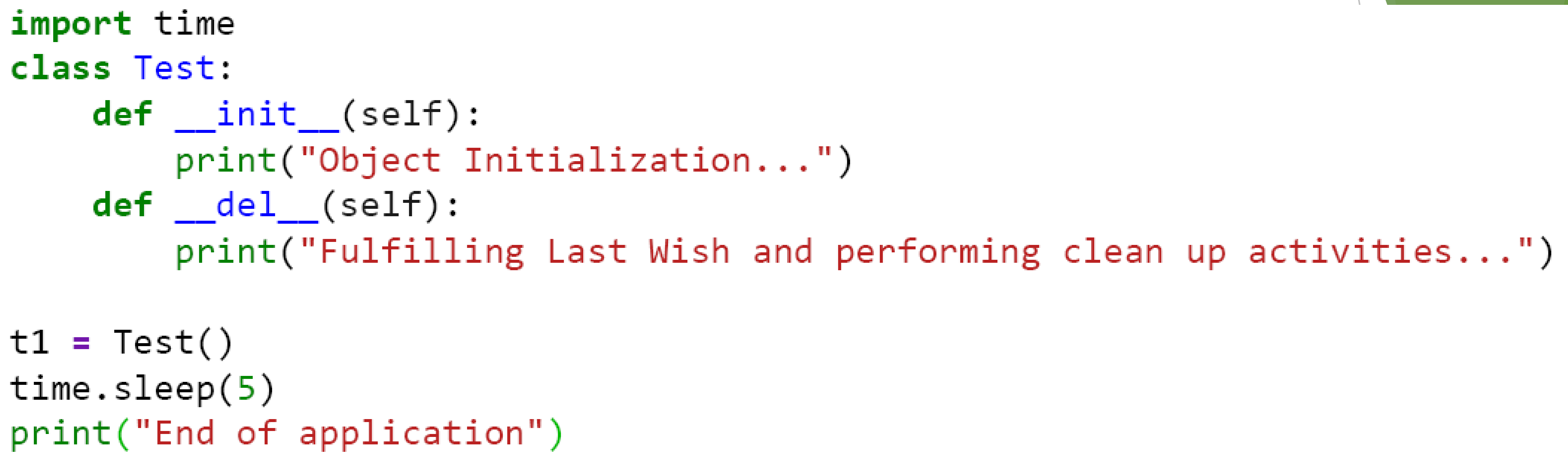
1. Destructor
2. How to find the number of references of an object?
3. Important Interview Questions

1. Destructor

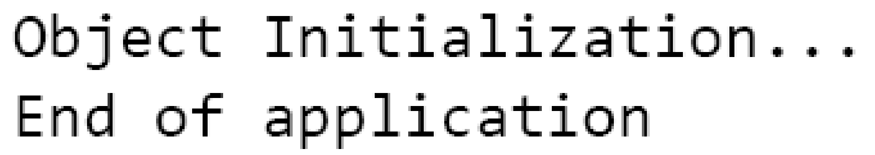
- Destructor is a special method and the name should be **_ _del_ _()**
- Just before destroying an object, Garbage Collector always calls destructor to perform clean up activities (like Resource de-allocation activities like close database connection etc.)
- Once destructor execution completes then Garbage Collector automatically destroys that object.

Note: The job of destructor is not to destroy object and it is just to perform clean up activities.

Demo Program on Destructor - I:



```
import time
class Test:
    def __init__(self):
        print("Object Initialization...")
    def __del__(self):
        print("Fulfilling Last Wish and performing clean up activities...")

t1 = Test()
time.sleep(5)
print("End of application")
```



```
Object Initialization...
End of application
```

Demo Program on Destructor - II:

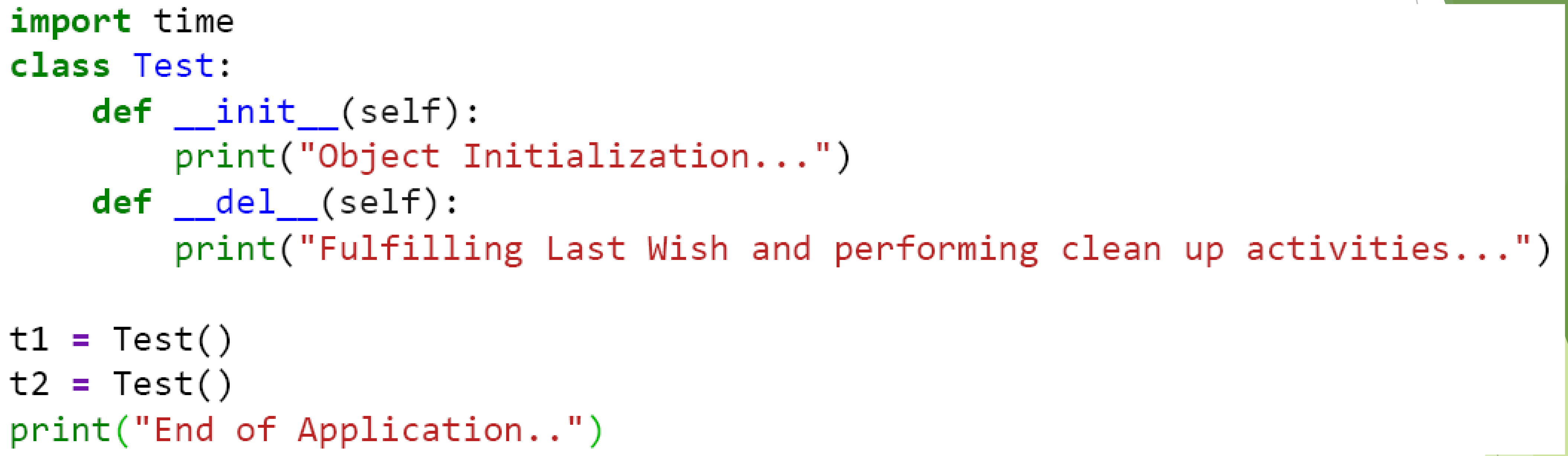
```python
import time
class Test:
    def __init__(self):
        print("Object Initialization...")
    def __del__(self):
        print("Fulfilling Last Wish and performing clean up activities...")

t1 = Test()
t1 = None
time.sleep(5)
print("End of application")
```

```
Object Initialization...
Fulfilling Last Wish and performing clean up activities...
End of application
```

Note:

- If the object does not contain any reference variable then only it is eligible for Garbage Collection. i.e., if the reference count of the object is zero, then only that object is eligible for Garbage Collection.

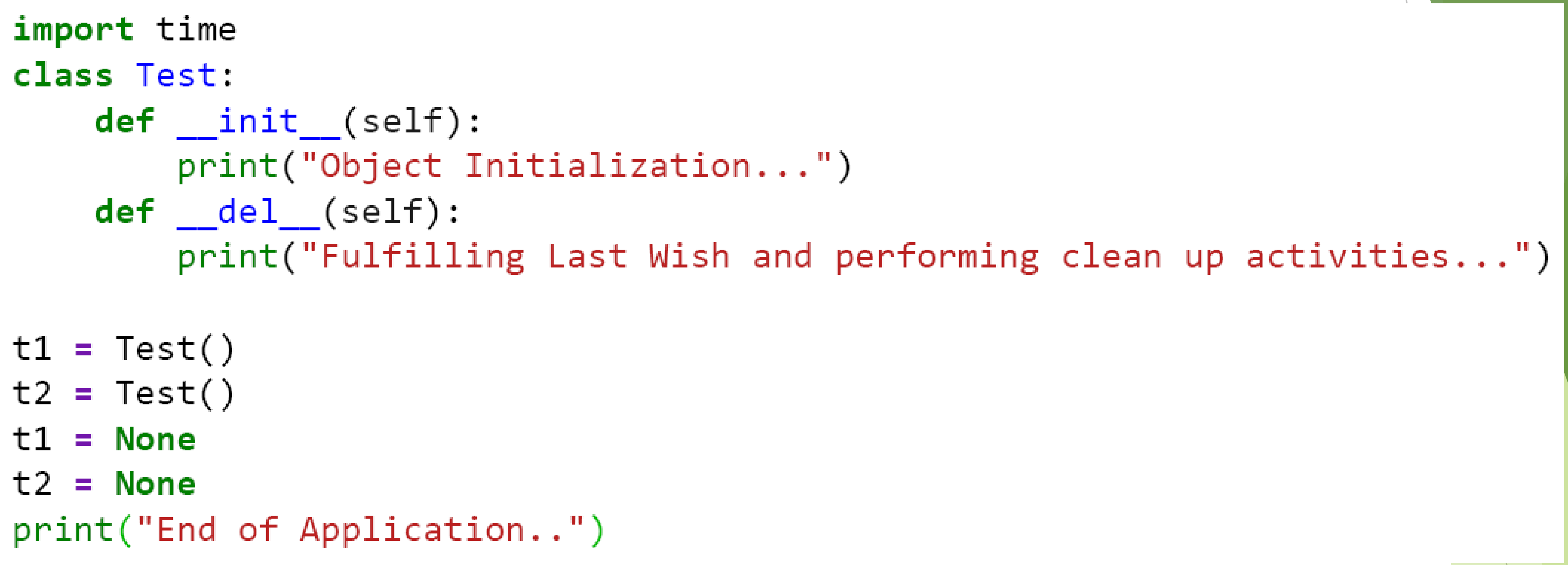
Demo Program on Destructor - III:



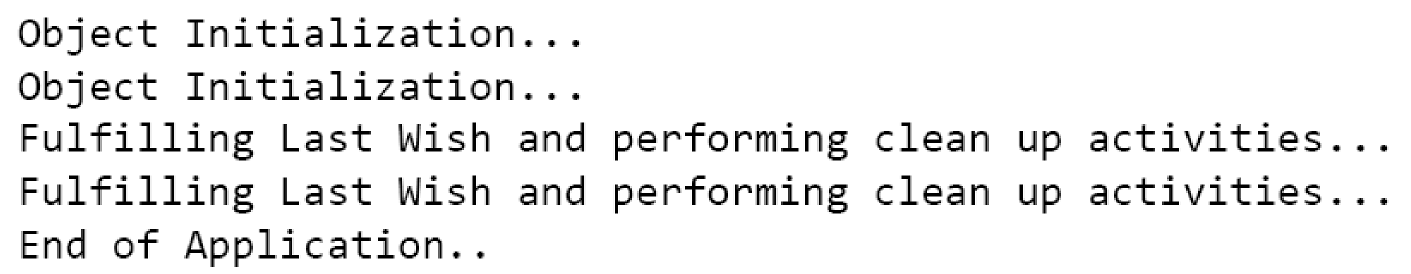
```python
import time
class Test:
    def __init__(self):
        print("Object Initialization...")
    def __del__(self):
        print("Fulfilling Last Wish and performing clean up activities...")

t1 = Test()
t2 = Test()
print("End of Application..")
```

```
Object Initialization...
Fulfilling Last Wish and performing clean up activities...
Object Initialization...
Fulfilling Last Wish and performing clean up activities...
End of Application..
```

Demo Program on Destructor - IV:



```
import time
class Test:
    def __init__(self):
        print("Object Initialization...")
    def __del__(self):
        print("Fulfilling Last Wish and performing clean up activities...")

t1 = Test()
t2 = Test()
t1 = None
t2 = None
print("End of Application..")
```



```
Object Initialization...
Object Initialization...
Fulfilling Last Wish and performing clean up activities...
Fulfilling Last Wish and performing clean up activities...
End of Application..
```

Important Conclusions -Destructors:

- The job of destructor is not to destroy object and it is just to perform clean up activities.
- Destroying object will take care by PVM.
- Once control reaches end of the program, all objects which are created as the part of that program are by default eligible for Garbage Collection.
- If the object does not contain any reference variable then only it is eligible for Garbage Collection. i.e., if the reference count is zero then only object eligible for Garbage Collection.

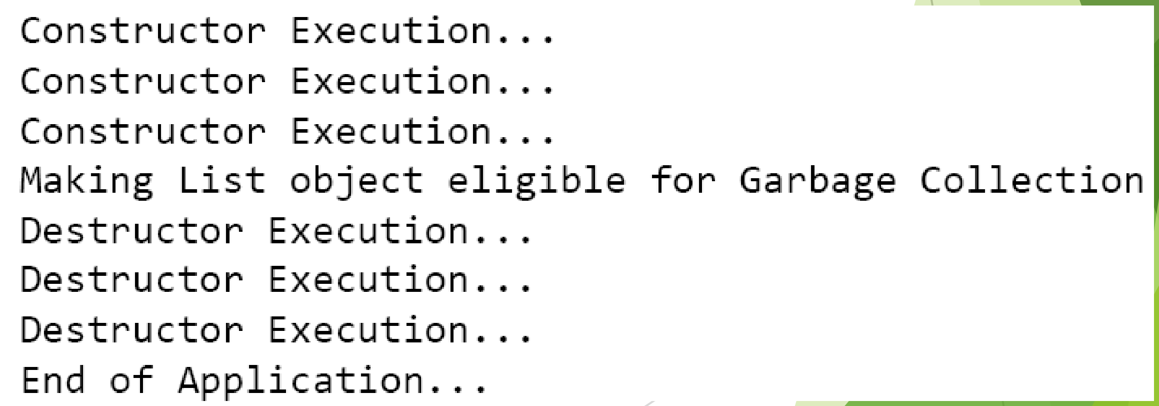
Demo Program on Destructor - V:

```python
import time
class Test:

    def __init__(self):
        print("Constructor Execution...")

    def __del__(self):
        print("Destructor Execution...")

t1=Test()
t2=t1
t3=t2
del t1
time.sleep(5)
print("object not yet destroyed after deleting t1")
del t2
time.sleep(5)
print("object not yet destroyed even after deleting t2")
print("I am trying to delete last reference variable...")
del t3
print("End of Application...")
```

```
Constructor Execution...
object not yet destroyed after deleting t1
object not yet destroyed even after deleting t2
I am trying to delete last reference variable...
Destructor Execution...
End of Application...
```

Demo Program on Destructor - VI:

```python
import time
class Test:

    def __init__(self):
        print("Constructor Execution...")

    def __del__(self):
        print("Destructor Execution...")

l = [Test(),Test(),Test()]
print("Making List object eligible for Garbage Collection ")
del l
time.sleep(5)
print("End of Application...")
```



```
Constructor Execution...
Constructor Execution...
Constructor Execution...
Making List object eligible for Garbage Collection
Destructor Execution...
Destructor Execution...
Destructor Execution...
End of Application...
```

Demo Program on Destructor - VII:

```python
import time
class Test:

    def __init__(self):
        print("Constructor Execution...")

    def __del__(self):
        print("Destructor Execution...")

l = [Test(),Test(),Test()]
time.sleep(5)
print("End of Application...")      # Clar
```

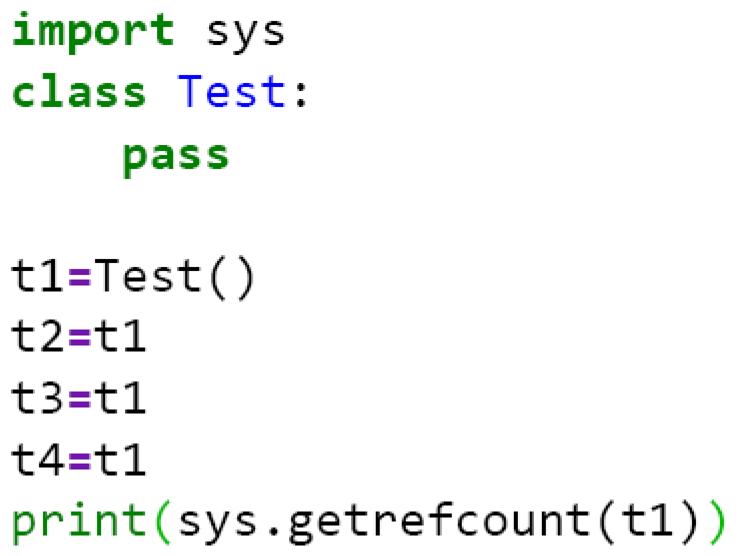
```
Constructor Execution...
Constructor Execution...
Constructor Execution...
End of Application...
```

2. How to find the number of references of an object?

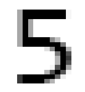
- sys module contains getrefcount() function for this purpose.

  sys.getrefcount(objectreference)

Eg:



```
import sys
class Test:
    pass

t1=Test()
t2=t1
t3=t1
t4=t1
print(sys.getrefcount(t1))
```



```
5
```

Note: For every object, Python internally maintains one default reference variable self.

3. Important Interview Questions

Q1. What is the difference between del t1 and t1=None ?

i) del t1:

- 't1' deleted and corresponding object eligible for Garbage Collection.
- We cannot use t1.
- If we don't want object and corresponding reference variable then we have to use this approach.

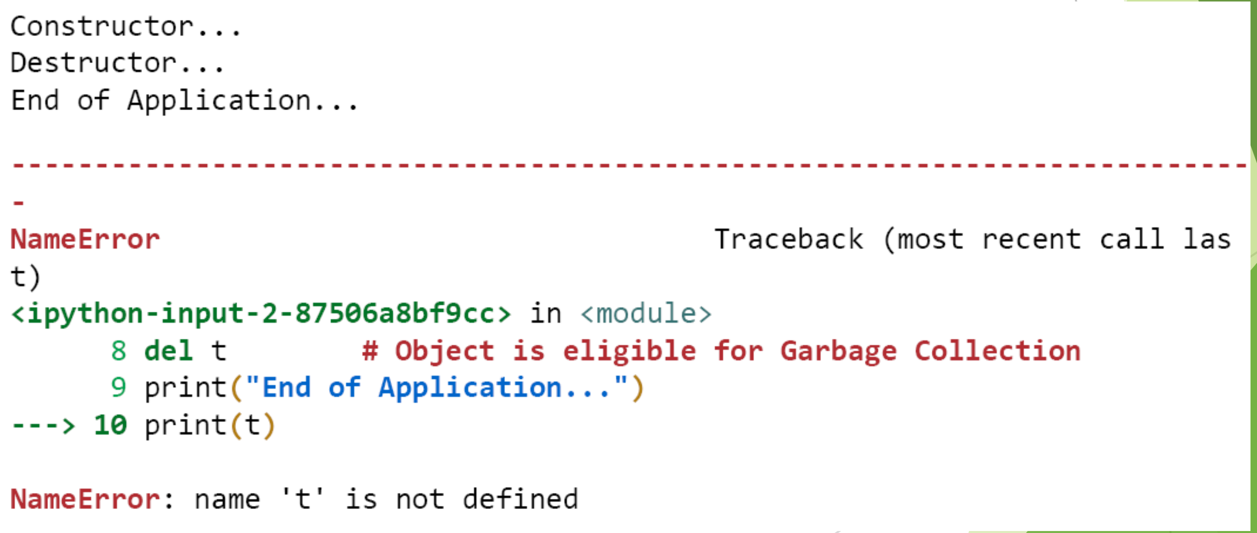
Eg 1:

```
class Test:
    def __init__(self):
        print("Constructor...")
    def __del__(self):
        print("Destructor...")

t = Test()
del t          # Object is eligible for Garbage Collection
print("End of Application...")
```

```
Constructor...
Destructor...
End of Application...
```

Eg 2:

```python
class Test:
    def __init__(self):
        print("Constructor...")
    def __del__(self):
        print("Destructor...")

t = Test()
del t        # Object is eligible for Garbage Collection
print("End of Application...")
print(t)
```



```
Constructor...
Destructor...
End of Application...

---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
<ipython-input-2-87506a8bf9cc> in <module>
      8 del t        # Object is eligible for Garbage Collection
      9 print("End of Application...")
---> 10 print(t)

NameError: name 't' is not defined
```

ii) t1 = None

- 't1' now onwards pointing to None object.
- Just it is reassigning the reference variable.
- The Test object is eligible for Garbage Collection.
- We can use t1 (but it's value is None).
- If we want reference variable but not object then we have to use this approach.

Eg:

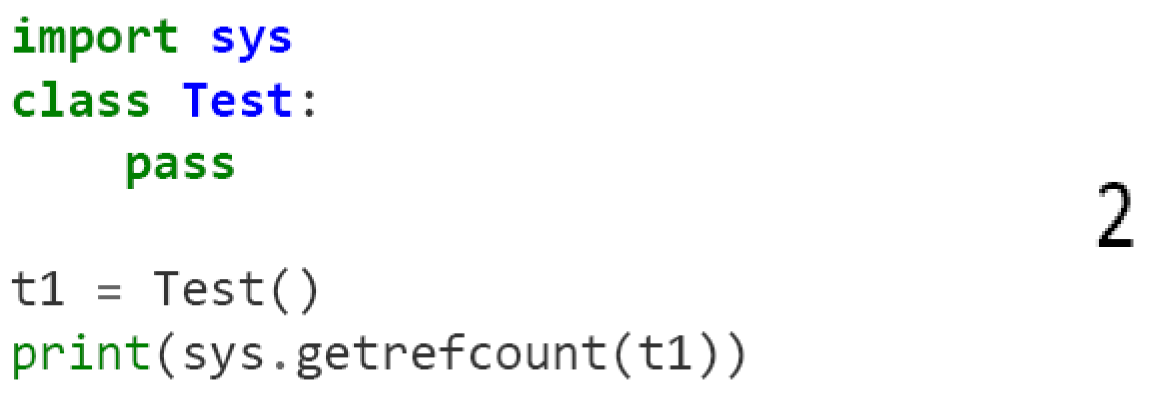
```
class Test:
    def __init__(self):
        print("Constructor...")
    def __del__(self):
        print("Destructor...")

t = Test()
t = None         # Object is eligible for Garbage Collection
print("End of Application...")
print(t)
```

```
Constructor...
Destructor...
End of Application...
None
```

Q2. How to find the number of references of an object?

- For this, we have to use sys module.
- sys module contains one function (i.e.,getrefcount()),by using this function we can able to know that how many reference variables are there for the given object.

Eg 1:



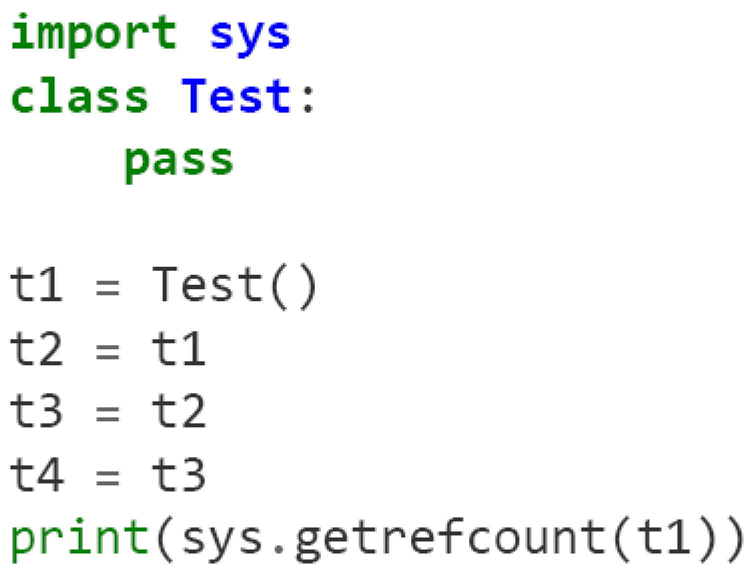
```
import sys
class Test:
    pass

t1 = Test()
print(sys.getrefcount(t1))
```

```
2
```

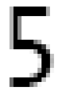
Note:

- For every object, internally PVM will maintain a reference variable known as self, that's why for the above program Test() object is referenced by two reference variables (t1 and self).

Eg 2:



```
import sys
class Test:
    pass

t1 = Test()
t2 = t1
t3 = t2
t4 = t3
print(sys.getrefcount(t1))
```



```
5
```

Eg 3:

```
import sys
class Test:
    pass

t1 = Test()
t2 = t1
t3 = t2
t4 = t3
del t3,t4
print(sys.getrefcount(t1))
```

```
3
```

Q3. What is the difference between Constructor and Destructor?

Constructor	Destructor
1. The name of the constructor should be __init__()	1. The name of the destructor should be __del__()
2. The main objective of constructor is to perform initialization activities of an object. Initialization means assigning the values to instance variables.	2. The main objective of destructor is to perform cleanup activities of an object. cleanup activities means resource deallocation activities like close database connection etc
3. Just after creating an object, PVM will execute constructor automatically to perform initialization activities.	3. Just before destroying an object, Garbage Collector will execute destructor automatically to perform cleanup activities.

Q4. Which of the following are TRUE?

A) The main purpose of constructor is to create an object.

B) The main purpose of constructor is to perform initialization of an object.

C) The main purpose of destructor is to destroy an object.

D) The main purpose of destructor is to perform clean up activities before destroying an object.

Ans: B,D

Q5. Which of the following are TRUE?

A) Constructor is responsible to create object where as Destructor is responsible to destroy object.

B) Constructor will be executed just after creating an object to perform initialization activities.

C) Destructor will be executed just before destroying an object to perform clean up activities.

D) All The above

Ans: B,C

Any question?

If you try to practice programs yourself, then you will learn many things automatically

Spend few minutes and then enjoy the study

Thank You