Garbage Collection in Python

PyConZA 2022
Durban

Elijah Okello

Software Engineering Student
Makerere University, UG

Garbage !

Garbage is anything unwanted

Memory Management and Garbage Collection!

Any set of objects in memory that are no longer being used by a process is considered garbage.

Originally, programmers

did manual memory

Management.

Automatic Memory Management

Disadvantages

- Additional memory usage
- Additional computation

Advantages outweigh disadvantages

Moore's law is the observation that the number of transistors in a dense integrated circuit (IC) doubles about every two years.

How does CPython collect Garbage ?

CPython is the reference implementation of the Python Programming Language. Implemented in C and Python, it is the most widely used implementation of the Python Programming Language.

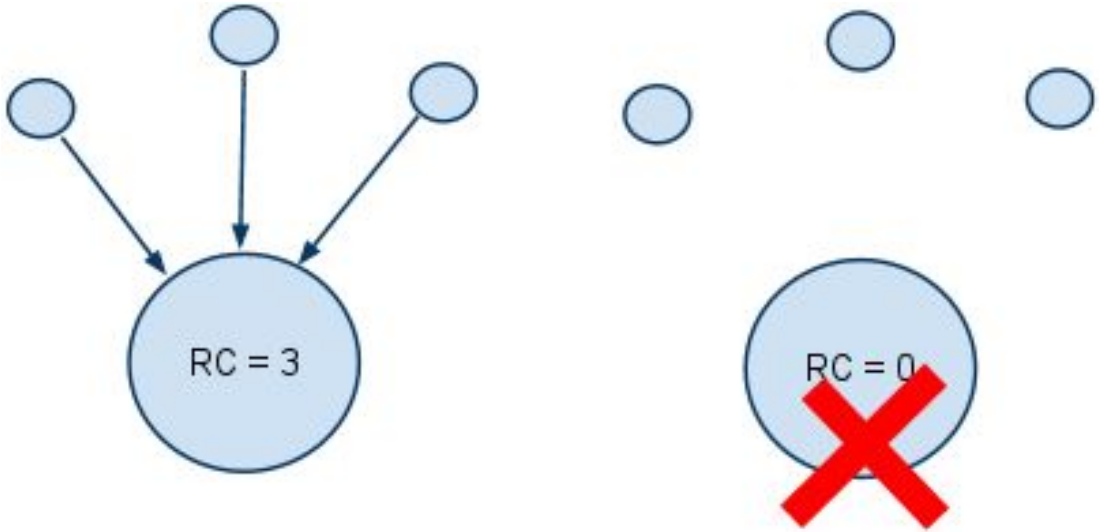

```
python -c 'import platform; print(platform.python_implementation())'
```

CPython GC Algorithms

- Reference Counting
- Generational Garbage Collection

Reference Counting

The runtime keeps track of all references to an object in memory.



```
>>> import sys

>>> a = 'my-string'

>>> sys.getrefcount(a)

2

>>> mylist = [a] # Make a list with a as an element.

>>> mydict = { 'key': a } # Create a dictionary with a as one
of the values.

>>> sys.getrefcount(a)

5
```

Generational Garbage Collector

Why do we need the generational garbage collector

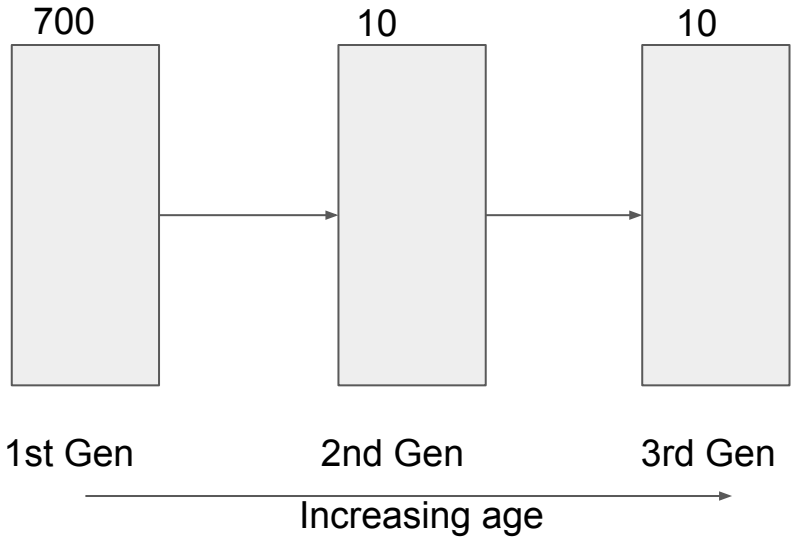
```
>>> class Student(object):
...     pass
...
>>> st1 = Student()
>>> st1.obj = st1
>>> del st1
```

Reference Cycle is when an object references itself.

Generational Garbage Collector Cont’d

Terminology

- Generation
- Threshold



How to interact with the CPython GC

```
>>> import gc
>>> gc.get_threshold()
(700, 10, 10)
>>> gc.get_count()
(410, 5, 7)
>>> gc.get_collect()
30
>>> gc.set_threshold(500,10,10)
>>>
```

BEST PRACTICES FOR GC in Cpython

Chances are you may not really need to change the GC behaviour in Cpython.

The GC is fine tuned for optimal results.

For special use cases you can manipulate it to your satisfaction using the previously stated methods.

Garbage Collection in pypy

PyPy is an alternative implementation of the Python programming language to CPython. PyPy often runs faster than CPython because PyPy uses a just-in-time compiler


```
pypy3 -c 'import platform; print(platform.python_implementation())'
```

Pypy GC Algorithms

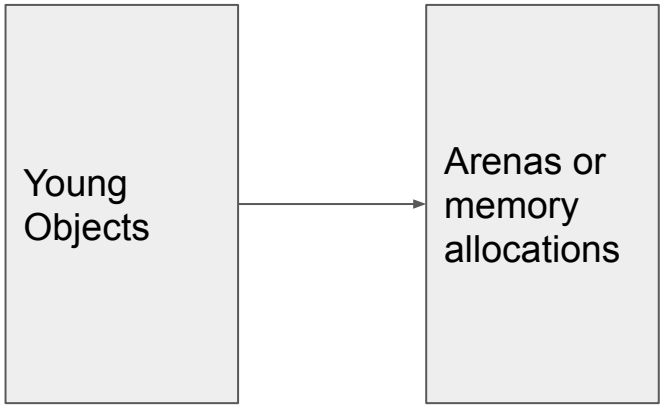
The pypy gc algorithm is chosen at the point of building the executable from source using the `--gc=NAME` option

```
pypy ../../rpython/bin/rpython --opt=jit —-gc=marksweep
```

- Semispace Copying GC
- Generational GC
- Hybrid GC
- Mark and Sweep
- Mark and Compact
- Incminimark GC [default]

Incminimark GC

PyPy's default garbage collector is called incminimark - it's an incremental, generational moving collector



Nursery

PYPY_GC_NURSERY

The collection for the older generations is done incrementally and this allows the applications not to have very long pause times as the gc is making a collection

How to interact with the Pypy GC. Incminimark

Incminimark GC is configurable through a set of environment variables

PYPY_GC_NURSERY : Set the size of the nursery beyond which collection happens

PYPY_GC_NURSERY_DEBUG : If set to non-zero, will fill nursery with garbage, to help debugging

https://doc.pypy.org/en/latest/gc_info.html#semi-manual-gc-management

Semi-manual GC management

```
>>> import gc
>>> gc.disable()
>>> gc.enable()
>>> gc.collect()
28
```

Best Practices in pypy

The nursery size is a very crucial variable - depending on your workload (one or many processes) and cache sizes you might want to experiment with it via *PYPY_GC_NURSERY* environment variable.

In low latency applications like games, you might want to control precisely when the gc runs. You do so by disabling the gc at certain point in your program `gc.disable()` and then collecting `gc.collect()`.

IronPython

IronPython is an open-source implementation of the Python programming language which is tightly integrated with .NET.

It uses the same GC algorithms as CPython ie Reference Counting and Generational GC with some changes

Jython

The Jython project provides implementations of Python in Java, providing to Python the benefits of running on the JVM and access to classes written in Java.

Jython uses Garbage Collection algorithms provided by the JVM.

Additional material for further study

https://docs.python.org/3/library/gc.html

https://doc.pypy.org/en/latest/gc_info.html#semi-manual-gc-management

https://rpython.readthedocs.io/en/latest/garbage_collection.html#mark-and-sweep


THANK
YOU!