n

```
Time to create 1 million NormalClass instances: 1.161 seconds
Time to create 1 million SlotsClass instances: 1.109 seconds
Time to create 1 million WeakrefClass instances: 2.957 seconds
```

Мы можем видеть, что классы с атрибутами slots и normal создаются быстрее, чем класс с атрибутами weakref.

```
Time to read and change attributes of NormalClass instances: 0.175 seconds
Time to read and change attributes of SlotsClass instances: 0.175 seconds
Time to read and change attributes of WeakrefClass instances: 0.179 seconds
```

Мы можем видеть, что классы с normal атрибутами и slots быстрее читают и изменяют атрибуты, чем класс с атрибутами weakref.

Итак, если вас беспокоит производительность, лучше использовать классы со slots или normal атрибутами вместо классов с атрибутами weakref.

```
         3000007 function calls in 14.440 seconds

   Ordered by: cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    1.435    1.435   14.440   14.440 {built-in method builtins.exec}
        1    1.099    1.099   13.005   13.005 <string>:1(<module>)
        1    0.971    0.971   11.905   11.905 my_profile.py:67(test)
        1    0.982    0.982    8.237    8.237 my_profile.py:80(<listcomp>)
  1000000    7.255    0.000    7.255    0.000 my_profile.py:22(__init__)
        1    0.618    0.618    2.021    2.021 my_profile.py:74(<listcomp>)
  1000000    1.402    0.000    1.402    0.000 my_profile.py:16(__init__)
        1    0.328    0.328    0.676    0.676 my_profile.py:68(<listcomp>)
  1000000    0.348    0.000    0.348    0.000 my_profile.py:8(__init__)
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

Мы видим, что большая часть времени тратится на __init__ методы классов. Остальное время тратится в основном на list comprehention.

```
Filename: /home/ggeasy/PycharmProjects/VK2023/08/my_profile.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    91    721.0 MiB    721.0 MiB           1   @profile
    92                                         def test_normal():
    93    909.0 MiB    187.9 MiB     1000003       normal_instances = [NormalClass('abc', i) for i in range(1000000)]
    94
    95    909.0 MiB      0.0 MiB     1000001       for instance in normal_instances:
    96    909.0 MiB      0.0 MiB     1000000           instance.s
    97    909.0 MiB      0.0 MiB     1000000           instance.n += 1
```

```
Filename: /home/ggeasy/PycharmProjects/VK2023/08/my_profile.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
   100    728.9 MiB    728.9 MiB           1   @profile
   101                                         def test_slots():
   102    908.9 MiB    179.9 MiB     1000003       slots_instances = [SlotsClass('abc', i) for i in range(1000000)]
   103
   104    908.9 MiB      0.0 MiB     1000001       for instance in slots_instances:
   105    908.9 MiB      0.0 MiB     1000000           instance.s
   106    908.9 MiB      0.0 MiB     1000000           instance.n += 1
```

```
Filename: /home/ggeasy/PycharmProjects/VK2023/08/my_profile.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
   109    729.9 MiB    729.9 MiB           1   @profile
   110                                         def test_weakref():
   111   1031.6 MiB    301.7 MiB     1000003       weakref_instances = [WeakrefClass(TestClass(), TestClass()) for i in range(1000000)]
   112
   113   1031.6 MiB      0.0 MiB     1000001       for instance in weakref_instances:
   114   1031.6 MiB      0.0 MiB     1000000           instance.s()
   115   1031.6 MiB      0.0 MiB     1000000           instance.n()
```

Можно видеть, что обычные атрибуты и slots-атрибуты потребляют примерно одинаковое количество памяти, меньшее чем weakref-атрибуты.