ДЗ 8. Метляков Андрей

1.

Класс, который будет атрибутом для каждого из тестируемых:

```
class TypecalInfo:
    def __init__(self,
                 serial_number,
                 age, vendor, model):
        self.serial_number = serial_number
        self.age = age
        self.vendor = vendor
        self.model = model
```

класс с обычными атрибутами:

```
class Device:
    def __init__(self, name, info):
        self.name = name
        self.info = info
```

класс со слотами:

```
class SlottedDevice:
    __slots__ = ('name', 'info')

    def __init__(self, name, info):
        self.name = name
        self.info = info
```

класс с атрибутами weakref:

```
class WeakRefDevice:
    def __init__(self, name, info):
        self.name = name
        self.info = weakref.ref(info)
```

или

```
class WeakRefDevice:
    def __init__(self, name, info):
        self.name = name
        self._info = weakref.ref(info)

    @property
    def info(self):
        return self._info()

    @info.setter
    def info(self, value):
        self._info = weakref.ref(value)
```

Измерение времени создания 500 000 экземпляров:

```
[Running] python -u "c:\hse_deep_python_aut_24\08\weakref_slots_compare.py"
{'Device_instance_creation': 0.8435167000279762, 'SlottedDevice_instance_creation': 0.7293621000135317, 'WeakRefDevice_instance_creation': 0.4462793999700807}
```

```
'Device_instance_creation': 0.8435167000279762,
```

```
'SlottedDevice_instance_creation': 0.7293621000135317,
```

```
'WeakRefDevice_instance_creation': 0.4462793999700807
```

Вывод:

Экземпляры обчного класса создаются медленнее всего. Обычный класс использует __dict__ для хранения атрибутов. Для каждого экземпляра создаётся отдельный словарь, что требует больше памяти и ресурсов.

Экземпляры класса со слотами создаются быстрее. Использование __slots__ позволяет избежать создания __dict__, снижая накладные расходы памяти. Вместо словаря атрибуты хранятся в статически определённых структурах, что улучшает производительность.

Экземпляры класса с атрибутами на основе weakref создаются быстрее всего. Слабые ссылки не увеличивают счётчик ссылок объекта. Из-за этого сборщик мусора может раньше удалять соответствующие объекты, но это может привести к неожиданным результатам, если связанные объекты удалятся раньше, чем ожидается.

Измерение времени чтения атрибутов 500 000 экземпляров:

```
[Running] python -u "c:\hse_deep_python_aut_24\08\weakref_slots_compare.py"
{'Device_access': 0.014009699982125312, 'SlottedDevice_access': 0.012487300031352788, 'WeakRefDevice_access': 0.009864100022241473}
```

```
'Device_access': 0.014009699982125312,
```

```
'SlottedDevice_access': 0.012487300031352788,
```

```
'WeakRefDevice_access': 0.009864100022241473}
```

Вывод: пока что результат аналогичен.

Измерение времени изменения атрибутов 500 000 экземпляров:

```
'Device_modify': 0.07270409999182448,
```

```
'SlottedDevice_modify': 0.07063620002008975,
```

```
'WeakRefDevice_modify': 0.11025029997108504}
```

ВАЖНО! На данном этапе если мы хотим изменять значение WeakRefDevice снова на weakref, то данный класс должен быть доплнен следующим образом:

```python
class WeakRefDevice:
    def __init__(self, name, info):
        self.name = name
        self._info = weakref.ref(info)

    @property
    def info(self):
        return self._info()

    @info.setter
    def info(self, value):
        self._info = weakref.ref(value)
```

Обычный класс и класс со слотами в данном эксперименте показали примерно одинаковый результат, а вот **WeakRefDevice** из-за работы со свойствами через декораторы оказался значительно медленнее.

2. Профилирование

```
[Running] python -u "c:\hse_deep_python_aut_24\08\weakref_slots_compare.py"
         8500040 function calls in 6.146 seconds

   Ordered by: cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        6    2.051    0.342    5.335    0.889 c:\hse_deep_python_aut_24\08\weakref_slots_compare.py:46(create_device_instances)
        9    0.109    0.012    3.415    0.379 c:\hse_deep_python_aut_24\08\weakref_slots_compare.py:54(measure_time)
  4500000    1.641    0.000    1.641    0.000 c:\hse_deep_python_aut_24\08\weakref_slots_compare.py:11(__init__)
  1000000    1.065    0.000    1.065    0.000 c:\hse_deep_python_aut_24\08\weakref_slots_compare.py:33(__init__)
        3    0.386    0.129    0.562    0.187 c:\hse_deep_python_aut_24\08\weakref_slots_compare.py:67(test_device_modify)
  1000000    0.551    0.000    0.551    0.000 c:\hse_deep_python_aut_24\08\weakref_slots_compare.py:27(__init__)
        3    0.077    0.026    0.141    0.047 c:\hse_deep_python_aut_24\08\weakref_slots_compare.py:61(test_device_access)
  1000000    0.127    0.000    0.127    0.000 c:\hse_deep_python_aut_24\08\weakref_slots_compare.py:19(__init__)
   500000    0.075    0.000    0.075    0.000 c:\hse_deep_python_aut_24\08\weakref_slots_compare.py:41(info)
   500000    0.063    0.000    0.063    0.000 c:\hse_deep_python_aut_24\08\weakref_slots_compare.py:37(info)
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
       18    0.000    0.000    0.000    0.000 {built-in method time.perf_counter}
```

Функция **create_device_instance** отвечает за создание экземпляров устройств, и она занимает значительное время в сравнении с другими. Хотя сама функция выполняется всего 6 раз, она вызывает другие функции, что приводит к большому накопленному времени. По данному отчету мы можем достаточно подробно описать все функции, которые были вызваны в результате работы программы.

Однако, все-таки основное время тратится на создание экземпляров устройств и их данных в функции **create_device_instances**.

Конструкторы классов создают большое количество объектов, не смотря на это, они занимают относительно мало времени по сравнению с более сложными функциями.

Далее – как раз 6 вызовов **create_device_instances. (1. Простой класс, 2. Класс с слотами, 3. Класс с weakref)**

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    46     29.4 MiB     29.4 MiB           1   @profile
    47                                         def create_device_instances(cur_cls, count):
    48    238.4 MiB   -529.6 MiB     1000002       return [cur_cls(f"SN{i}",
    49    238.4 MiB   -536.8 MiB     1000000                      TypecalInfo(100 * i,
    50    238.4 MiB   -268.4 MiB      500000                                  2 * i,
    51    238.4 MiB   -255.8 MiB      500000                                  f"vendor{i}",
    52    238.4 MiB   -526.3 MiB     1000002                                  f"model{3 * i}")) for i in range(count)]


Filename: .\weakref_slots_compare.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    46     36.6 MiB     36.6 MiB           1   @profile
    47                                         def create_device_instances(cur_cls, count):
    48    223.0 MiB   -937.2 MiB     1000002       return [cur_cls(f"SN{i}",
    49    223.0 MiB   -934.1 MiB     1000000                      TypecalInfo(100 * i,
    50    223.0 MiB   -465.5 MiB      500000                                  2 * i,
    51    223.0 MiB   -472.5 MiB      500000                                  f"vendor{i}",
    52    223.0 MiB   -940.1 MiB     1000002                                  f"model{3 * i}")) for i in range(count)]


Filename: .\weakref_slots_compare.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    46     40.2 MiB     40.2 MiB           1   @profile
    47                                         def create_device_instances(cur_cls, count):
    48    138.8 MiB  -5743.9 MiB     1000002       return [cur_cls(f"SN{i}",
    49    138.8 MiB  -5759.2 MiB     1000000                      TypecalInfo(100 * i,
    50    138.8 MiB  -2891.5 MiB      500000                                  2 * i,
    51    138.8 MiB  -2878.0 MiB      500000                                  f"vendor{i}",
    52    138.8 MiB  -5759.7 MiB     1000002                                  f"model{3 * i}")) for i in range(count)]


Filename: .\weakref_slots_compare.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    46     43.9 MiB     43.9 MiB           1   @profile
    47                                         def create_device_instances(cur_cls, count):
    48    238.4 MiB  -2560.6 MiB     1000002       return [cur_cls(f"SN{i}",
    49    238.4 MiB  -2564.4 MiB     1000000                      TypecalInfo(100 * i,
    50    238.4 MiB  -1293.9 MiB      500000                                  2 * i,
    51    238.4 MiB  -1261.2 MiB      500000                                  f"vendor{i}",
    52    238.4 MiB  -2558.6 MiB     1000002                                  f"model{3 * i}")) for i in range(count)]
```

```
Filename: .\weakref_slots_compare.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    46    238.4 MiB    238.4 MiB           1   @profile
    47                                         def create_device_instances(cur_cls, count):
    48    432.1 MiB  -1840.7 MiB     1000002       return [cur_cls(f"SN{i}",
    49    432.1 MiB  -1845.9 MiB     1000000                      TypecalInfo(100 * i,
    50    432.1 MiB   -945.4 MiB      500000                                  2 * i,
    51    432.1 MiB   -869.3 MiB      500000                                  f"vendor{i}",
    52    432.1 MiB  -1867.3 MiB     1000002                                  f"model{3 * i}")) for i in range(count)]


Filename: .\weakref_slots_compare.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    46    432.1 MiB    432.1 MiB           1   @profile
    47                                         def create_device_instances(cur_cls, count):
    48    541.6 MiB -11627.3 MiB     1000002       return [cur_cls(f"SN{i}",
    49    541.6 MiB -11646.0 MiB     1000000                      TypecalInfo(100 * i,
    50    541.6 MiB  -5836.6 MiB      500000                                  2 * i,
    51    541.6 MiB  -5822.1 MiB      500000                                  f"vendor{i}",
    52    541.6 MiB -11650.9 MiB     1000002                                  f"model{3 * i}")) for i in range(count)]
```

Как и ожидалось - использование памяти больше при создании экземпляров базового класса, дальше класс со слотами, меньше всего при создании экземпляров класса с weakref.

Вызовы функции **test device access и test device modify**

```
Filename: .\weakref_slots_compare.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    62    541.6 MiB    541.6 MiB           1   @profile
    63                                         def test_device_access(instances):
    64    541.8 MiB      0.1 MiB      500001       for instance in instances:
    65    541.8 MiB      0.1 MiB      500000           _ = instance.name
    66    541.8 MiB      0.0 MiB      500000           _ = instance.info


Filename: .\weakref_slots_compare.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    62    541.8 MiB    541.8 MiB           1   @profile
    63                                         def test_device_access(instances):
    64    541.8 MiB      0.0 MiB      500001       for instance in instances:
    65    541.8 MiB      0.0 MiB      500000           _ = instance.name
    66    541.8 MiB      0.0 MiB      500000           _ = instance.info


Filename: .\weakref_slots_compare.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    62    541.8 MiB    541.8 MiB           1   @profile
    63                                         def test_device_access(instances):
    64    541.8 MiB  -3818.1 MiB      500001       for instance in instances:
    65    541.8 MiB  -3818.1 MiB      500000           _ = instance.name
    66    541.8 MiB  -3818.1 MiB      500000           _ = instance.info


Filename: .\weakref_slots_compare.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    69    541.8 MiB    541.8 MiB           1   @profile
    70                                         def test_device_modify(instances):
    71    541.9 MiB   -258.8 MiB      500001       for instance in instances:
    72    541.9 MiB   -258.8 MiB      500000           instance.name = "NewName"
    73    541.9 MiB   -517.6 MiB     1000000           instance.info = TypecalInfo(100,
    74    541.9 MiB   -258.8 MiB      500000                                       2,
    75    541.9 MiB   -258.8 MiB      500000                                       "vendor_updated",
    76    541.9 MiB   -258.8 MiB      500000                                       "model_updated")
```

Третий замер (класс с weakref) как и ожидалось показывает лучший результат.

```
Filename: .\weakref_slots_compare.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    69    541.8 MiB    541.8 MiB           1   @profile
    70                                         def test_device_modify(instances):
    71    541.8 MiB  -2893.7 MiB      500001       for instance in instances:
    72    541.8 MiB  -2893.7 MiB      500000           instance.name = "NewName"
    73    541.8 MiB  -5787.4 MiB     1000000           instance.info = TypecalInfo(100,
    74    541.8 MiB  -2893.7 MiB      500000                                       2,
    75    541.8 MiB  -2893.7 MiB      500000                                       "vendor_updated",
    76    541.8 MiB  -2893.7 MiB      500000                                       "model_updated")


Filename: .\weakref_slots_compare.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    69    541.8 MiB    541.8 MiB           1   @profile
    70                                         def test_device_modify(instances):
    71    541.8 MiB      0.0 MiB      500001       for instance in instances:
    72    541.8 MiB      0.0 MiB      500000           instance.name = "NewName"
    73    541.8 MiB      0.0 MiB     1000000           instance.info = TypecalInfo(100,
    74    541.8 MiB      0.0 MiB      500000                                       2,
    75    541.8 MiB      0.0 MiB      500000                                       "vendor_updated",
    76    541.8 MiB      0.0 MiB      500000                                       "model_updated")
```

Первый и второй замеры функции **и test device modify** показывают аномалии с отрицательными изменениями памяти. Это может быть связано с оптимизациями в Python или особенностями работы инструмента подсчета памяти