Измерение времени выполнения программы

Очень часто нам бывает необходимо измерить время выполнения всей программы, либо отдельных ее частей. Чтобы найти данную величину, достаточно посчитать разницу в секундах между точкой старта и местом, где она завершает свою работу.

В следующем примере демонстрируется применение функции `time()` для получения текущего времени, чтобы в итоге выявить, как долго работал блок кода.

Приведенный ниже код:

```
import time

start_time = time.time()

for i in range(5):
    print(i)
    time.sleep(1)

end_time = time.time()

elapsed_time = end_time - start_time
print(f'Время работы программы = {elapsed_time}')
```

выводит (время работы программы может незначительно отличаться):

```
0
1
2
3
4
Время работы программы = 5.022884845733643
```

Несмотря на простоту вышеописанного подхода, использовать его в серьезных целях, где требуется точный и независимый от операционной системы (ОС) результат, не рекомендуется. Все дело в том, что числовое значение времени, получаемое таким образом, может иметь погрешности за счет внутренних особенностей работы компьютера, в среде которого выполняется программа. Более того, системные часы могут быть подкорректированы вручную пользователем во время выполнения программы.

Может случиться такая ситуация, что очередной вызов функции `time()` вернет значение меньше, чем значение, полученное при предыдущем вызове.

Функция monotonic()

Для измерения времени выполнения программы идеально подходит функция `monotonic()`, доступная на всех ОС (начиная с Python 3.5), так как ее результат не зависит от корректировки системных часов.

Функция `monotonic_ns()` похожа на `monotonic()`, но возвращает время в наносекундах. Работает не на всех операционных системах.

Используемый таймер в функции `monotonic()` никогда не вернет при повторном вызове значение, которое будет меньше значения, полученного при предыдущем вызове. Это позволяет избежать многих ошибок, а также неожиданного поведения.

В следующем примере демонстрируется применение функции `monotonic()` для получения текущего времени, чтобы в итоге выявить, как долго работал блок кода.

Приведенный ниже код:

```
import time

start_time = time.monotonic()

for i in range(5):
    print(i)
    time.sleep(0.5)

end_time = time.monotonic()

elapsed_time = end_time - start_time
print(f'Время работы программы = {elapsed_time}')
```

выводит (время работы программы может незначительно отличаться):

```
0
1
2
3
4
Время работы программы = 2.547000000020489
```

Принцип работы и применения функции `monotonic()` такой же, как и у функции `time()`. Однако функция `monotonic()` дает результат, который обладает гарантированной точностью и не зависит от внешних условий.

Функция perf_counter()

Для самого точного измерения времени выполнения программы следует использовать функцию `perf_counter()`. Данная функция использует таймер с наибольшим доступным разрешением, что делает эту функцию отличным инструментом для измерения времени выполнения кода на коротких интервалах.

В следующем примере демонстрируется применение функции `perf_counter()` для получения текущего времени, чтобы в итоге выявить, как долго работал блок кода.

Приведенный ниже код:

```
import time

start_time = time.perf_counter()

for i in range(5):
    print(i)
    time.sleep(1)

end_time = time.perf_counter()

elapsed_time = end_time - start_time
print(f'Время работы программы = {elapsed_time}')
```

выводит (время работы программы может незначительно отличаться):

```
0
1
2
3
4
Время работы программы = 5.042140900040977
```

В Python версии 3.7 добавлена функция `perf_counter_ns()` – работает так же, но длительность выводится в наносекундах, что удобнее для совсем малых интервалов времени и быстро исполняемых команд.

Примечания

Примечание 1. О сравнении функций `time()`, `monotonic()` и `perf_counter()` можно почитать тут.

Примечание 2. Для измерения времени работы программы мы также можем использовать встроенный модуль `timeit`. Документация по данному модулю доступна по ссылке.

Примечание 3. Небольшая статья с общими рекомендациями по оптимизации и сравнениями времени выполнения разных вариаций программ от Гвидо Ван Россума.