

OPTIMAL KOD YOZISH SAN'ATI

Камалова Нилуфар Илхомовна

Бухарский государственный университет преподаватель кафедры"
Прикладная математика и технологии программирования", доктор
философских наук (PhD)

Гулмуродов Мухриддин Рахматулла ўғли

Бухарский государственный университет студент 1-1КИДТМ-23

Annotatsiya:

В статье представлена информация о том, что делается для написания оптимального кода на языке программирования Python, выводы исследователей, осуществлявших научную деятельность в этой области, способы занять меньше места в памяти при написании оптимального кода и способы увеличения скорости выполнения программы. Приведем примеры написания оптимального кода при применении операторов.

Kalit so'zlar: Optimal kod, Python, sikllar, iteratsiyalar, generatorlar, operatorlar

В настоящее время во всем мире существует множество языков программирования. Эти языки программирования предназначены для облегчения жизни человека. Языки программирование служат для выполнения команд между человеком и компьютером. Программа, на которой мы сейчас собираемся остановиться, - это язык программирования "Python". Эта программа признается большинством пользователей. "Python" очень удобен для написания программ в то же время написать оптимальный код в этой программе несколько проще. Обратимся к написанию оптимального кода через переменные. Чтобы написать оптимальный код, вы должны сначала разумно использовать переменные, которые у вас есть. Это положит конец недопониманию между программистом и пользователем.

Вопрос написания оптимального кода исследовался многими учеными. Например: Кирдяев М. М. - Можно сказать, что каждый пользователь, использующий программу Python, пытается оптимизировать код, экономя время и используя меньше операторов. Если производительность недостаточна, эту часть кода заменяют готовыми функциями. Это приводит к интеграции приложения [1]

Сааков Вячеслав Валерьевич считает, что для оптимизации программного кода необходимо сначала полностью выполнить и протестировать программу. После успешного запуска необходимо применить методы оптимизации. Он утверждал, что методы должны включать изменение параметров алгоритма, применение эволюционных стратегий, или использование эвристических методов для достижения более высокой эффективности [2].

S.S. Абдукаримов подчеркнул, что для написания оптимального кода необходимо использовать тестирующие системы[3].

Чтобы написать оптимальный код, необходимо выполнить следующее. Прежде всего, необходимо досконально разобраться в условии вопроса. Только тогда удастся избежать лишних операций и упростить код. После этого необходимо составить оптимальный алгоритм. Потому что выбор правильного алгоритма является ключом к оптимальному коду. Иногда даже небольшое изменение алгоритма может значительно повысить производительность. Если в алгоритме есть итерации, их необходимо устранить. Дублирование кода может привести к ошибкам и затруднить его обслуживание. Следовательно, необходимо использовать функции, классы или модули, чтобы избежать дублирования кода. Необходимо уменьшить сложность кода. Сложные задачи необходимо разбить на простые и логические блоки. Это упрощает чтение кода и его тестирование. Как только эти процессы будут выполнены, пространство, выделяемое из памяти, должно быть уменьшено. Особенно при работе с большими объемами данных или встроенными системами необходимо эффективно использовать имеющуюся память. Если в программном коде есть циклы, их необходимо оптимизировать. Циклы в коде — одна из самых ресурсоемких операций. Помимо выбора эффективных алгоритмов, циклы можно оптимизировать, например, чтобы избежать ненужных итераций.

Рассмотрим следующие примеры:

Нахождение оптимальных решений в приложениях с оператором if.

Оптимальный код	Неоптимальный код
<pre>sonlar = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] s = 0 for i in sonlar: if i % 2 == 0: s += i print(s)</pre>	<pre>sonlar = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] s = 0 for i in sonlar: if i % 2 == 0: s += i else: s += 0 print(s)</pre>
Обе программы возвращают правильный результат. Вторая реализация не оптимальна потому что он добавляет 0 к общей сумме, когда условие не выполняется. Это будет ненужный расчет.	

Поиск оптимальных решений в приложениях с оператором for.

Оптимальный код	Неоптимальный код
<pre>nums = [1, 2, 3, 4, 5] total = 0 for num in nums: total += num print(total)</pre>	<pre>nums = [1, 2, 3, 4, 5] total = 0 for i in range(len(nums)): total = total + nums[i] print(total)</pre>

В оптимальном решении используется числовое значение непосредственно в переменной “num”, а затем оно добавляется к “total”. Это приемлемо, потому что мы предотвращаем повторное обращение к списку для получения значения числа.

Поиск оптимального решения в приложениях с оператором While.

Оптимальный код	Неоптимальный код
<pre>i = 0 while i < 10: print(i) i += 1</pre>	<pre>while True: print(i) i += 1 if i >= 10: break</pre>

Оба экземпляра выполняют одну и ту же задачу, но в оптимальном решении код написан более понятным способом. В нем переменной “i” присваивается начальное значение перед циклом while, а сам цикл выполняется, когда переменная “i” меньше 10. После каждой итерации цикла переменная “i” увеличивается на 1. Это точное и эффективное решение.

С другой стороны, неоптимальное решение использует бесконечный цикл (‘While True’) и проверяет состояние внутри цикла. Блок кода, выполняемый на каждой итерации, трудно читать и непонятно, потому что неясно, когда и как цикл завершается. Это затрудняет поддержку и компиляцию программы, а также снижает ее эффективность.

Для написания оптимального кода требуется эффективное использование памяти. Оптимизация использования памяти в Python заключается в уменьшении объема занимаемой программой оперативной памяти.

Представим, у нас есть список чисел ‘my_list’, который занимает много памяти:

```
my_list = [i for i in range(1000000)]
```

Для оптимизации использования памяти можно использовать генераторы вместо создания списка:

```
my_generator = (i for i in range(1000000))
```

При использовании генератора `my_generator` значения будут генерироваться по мере необходимости, что сильно сокращает использование памяти. В данном случае, генератор будет занимать только небольшое количество памяти, в отличие от списка `my_list`, который может занимать значительное количество памяти.

Рекомендуется удалить неиспользуемые данные, чтобы они занимали меньше места в памяти. Язык Python автоматически удаляет объекты, на которые нет ссылок, с помощью сборщика мусора. Однако, в некоторых случаях, особенно при работе с большими объемами данных, может быть полезно явно удалять объекты, чтобы освободить память. Это можно сделать с помощью ключевого слова `del`, например: `del my_object`.

Также необходимо эффективно использовать время для написания оптимального кода. Оптимизация времени выполнения в Python заключается в улучшении производительности программы, чтобы она выполнялась быстрее и более эффективно.

Один из наиболее важных способов оптимизации времени выполнения - выбор правильного алгоритма для решения задачи. Некоторые алгоритмы имеют более низкую сложность времени выполнения, что может привести к значительной экономии времени.

Нужно использовать более эффективных структур данных. Выбор оптимальных структур данных может существенно улучшить временную сложность операций в программе. Информирование о подходящих структурах данных, таких как словари, множества или массивы, позволяет избежать затрат на проход по всем элементам или поиск в списке.

Еще один метод экономии времени в программировании - параллельное выполнение. Параллельное или распределенное выполнение задач может значительно сократить время выполнения программы.

Если программа часто выполняет дорогостоящие вычисления с одинаковыми входными данными, можно использовать кэширование результатов, чтобы избежать повторных вычислений и сэкономить время выполнения.

Важно понимать, что оптимизация памяти и времени выполнения является процессом, требующим тестирования, измерения производительности и



экспериментирования с различными методами. Конкретные методы оптимизации будут зависеть от специфики проекта, доступных ресурсов и требований производительности.

Чтобы научиться писать оптимальный код в целом, необходимо постоянное обучение и саморазвитие. Потому что технологии и методы программирования постоянно развиваются, поэтому важно постоянно обновлять свои знания и изучать новые подходы и методы написания оптимального кода..

Использованная литература

1. Кирдяев М.М. Обзор языка программирования Python для решения задач математического моделирования // НиКа. 2016.
2. Сааков В. В. и др. Разработка и оптимизация алгоритмов на python // молодой учёный 3. – 2023. – С. 36.
3. Abdukarimov Sirojiddin Sayfiddin o'g'li Auditoriyadan tashqari mashg'ulotlarda talabalarga dasturlashni o'rgatish metodikasi NamDu ilmiy axborotnomasi 2021 yil 8-son.
4. Nilufar K. DASTURLASH TILLARINI O 'QITISHDA INTELLEKT TESTLARINI INTEGRATSIYALASH //Involta Scientific Journal. – 2022. – Т. 1. – №. 8. – С. 37-45.

