УДК 004.423 **Информационные технологии**

Халевин Тимофей Анатольевич, *студент направления подготовки информатики и вычислительной техники, Хакасский государственный университет имени Н.Ф. Катанова, г. Абакан, Россия*

Голубничий Артем Александрович, *научный руководитель, старший преподаватель кафедры ПОВТиАС, Хакасский государственный университет имени Н.Ф. Катанова, г. Абакан, Россия*

ИСПОЛЬЗОВАНИЕ ГЕНЕРАТОРОВ СПИСКОВ В ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON

Аннотация: Данная статья рассматривает генераторы списков в Python, описывает их возможности и преимущества перед другими методами создания списков.

Ключевые слова: Python, генератор списков, цикл for.

Annotation: This article looks at list generators in Python, describing their features and advantages over other list generation methods.

Keywords: Python, list comprehensions, for loop.

Генераторы списков – это способ создания списков в Python, который позволяет создавать списки на основе некоторого шаблона или условия. Генераторы списков являются компактной и элегантной альтернативой циклам *for* и функциям *map()* и *filter()*, которые также используются для создания списков [1].

Синтаксис генератора списка очень прост и похож на синтаксис обычного списка. Вместо того, чтобы создавать список путем добавления элементов с помощью цикла, мы создаем список путем определения выражения, которое будет использоваться для генерации элементов списка [2].

Генераторы списков могут быть использованы для создания списков с любым количеством элементов, которые можно сгенерировать на основе заданного шаблона или условия. Они также могут быть использованы для фильтрации и преобразования существующих списков.

Генератор списка имеет синтаксис вида *[выражение for элемент in итерируемый_объект if условие]*, где:

1. выражение – это выражение, которое вы хотите применить к каждому элементу итерируемого объекта;

2. элемент – это переменная, которая будет использоваться для итерации по итерируемому объекту;

3. итерируемый_объект – это итерируемый объект, который вы хотите использовать для создания списка;

4. условие – не обязательный параметр, который определяет, какие элементы будут включены в окончательный список.

Например, вы можете использовать генератор списка для создания списка, содержащего квадраты четных чисел от 0 до 10. Пример такого генератора списка представлен на рисунке 1.

```
>>> squares = [x*x for x in range(10) if x % 2 == 0]
>>> squares
[0, 4, 16, 36, 64]
```

Рисунок 1 – Генератор списка с квадратами четных чисел.

В этом примере мы используем *range(10)* для создания списка чисел от 0 до 9. Затем мы используем условие *if x % 2 == 0*, чтобы выбрать только четные числа. Наконец, мы используем выражение *x*x* для создания списка квадратов четных чисел.

Генераторы списков могут быть использованы для более сложных преобразований и фильтраций данных. Например, можно использовать генератор списка для создания списка слов в тексте, содержащих более 5 букв. Пример такого генератора списка представлен на рисунке 2.

```
>>> text = "Язык программирования Python - один из самых популярных языков программирования
в мире"
>>> words = [word for word in text.split() if len(word) > 5]
>>> words
['программирования', 'Python', 'популярных', 'языков', 'программирования']
```

Рисунок 2 – Генератор списка с фильтрацией по условию

В этом примере мы используем *text.split()* для создания списка слов в тексте. Затем мы используем условие *if len(word) > 5*, чтобы выбрать только слова, содержащие более 5 букв. Наконец, мы используем переменную *word* для сохранения каждого слова в списке *words*.

Генераторы списков также могут быть вложенными, что позволяет создавать более сложные структуры данных. Например, можно использовать генератор списка для создания списка списков. Пример такого генератора списков представлен на рисунке 3.

```
>>> matrix = [[i*j for j in range(5)] for i in range(5)]
>>> matrix
[[0, 0, 0, 0, 0], [0, 1, 2, 3, 4], [0, 2, 4, 6, 8],
[0, 3, 6, 9, 12], [0, 4, 8, 12, 16]]
```

Рисунок 3 – Генератор списка вложенный в генератор списка

В этом примере мы используем два генератора списка: внешний генерирует строки матрицы, а внутренний – генерирует элементы каждой строки. Мы используем переменные *i* и *j* для итерации по строкам и столбцам матрицы, а выражение *i*j* используется для вычисления значения каждого элемента.

Генераторы списков могут также быть использованы для создания словарей и множеств. Например, можно использовать генератор списка для создания словаря, содержащего значения из списка в качестве ключей и их индексы в качестве значений. Пример создания словаря с использованием генератора представлен на рисунке 4.

```
>>> fruits = ['яблоко', 'клубника', 'банан']
>>> fruit_dict = {fruit: index for index, fruit in enumerate(fruits)}
>>> fruit_dict
{'яблоко': 0, 'клубника': 1, 'банан': 2}
```

Рисунок 4 – Создание словаря с использованием генератора списка

В этом примере мы используем генератор списка для создания словаря *fruit_dict*. Ключами словаря являются элементы из списка *fruits*, а значениями являются их индексы, полученные с помощью функции *enumerate()*.

Генераторы списков могут быть более эффективными и быстрыми, чем обычные циклы for, особенно при работе с большими объемами данных. Они также могут упростить код и сделать его более читаемым и понятным.

Однако, в некоторых случаях, когда требуется более сложная логика и обработка данных, использование обычного цикла for может быть более предпочтительным. Это может быть связано с тем, что в цикле for вы можете использовать более сложные условия и операции, а также выполнять более сложные действия внутри цикла.

Вычислительная эффективность генераторов списков может быть высокой благодаря их особенностям. При использовании генератора списка, можно применять к каждому элементу списка функцию или операцию в одной строке кода, что уменьшает количество операций, которые необходимо выполнить, и увеличивает скорость обработки данных.

Генераторы списков также используют меньше памяти, чем создание списка с помощью цикла *for* и метода *append()*. Это связано с тем, что генератор списка создает элементы по мере необходимости, в отличие от метода *append()*, который создает весь список в памяти заранее.

Кроме того, генераторы списков могут использоваться в комбинации с другими функциями, такими как *filter()* и *map()*, чтобы создать более сложные выражения. Генераторы списков могут быть использованы для улучшения производительности кода во многих случаях. Разберем несколько таких примеров.

Фильтрация списка по определенному условию с использованием цикла

for представлена на рисунке 5.

```
>>> numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> even_numbers = []
>>> for number in numbers:
...     if number % 2 == 0:
...         even_numbers.append(number)
...
...
>>> even_numbers
[2, 4, 6, 8, 10]
```

Рисунок 5 – Фильтрация списка по условию четности в цикле *for*

И пример точно такой же фильтрации по условию четности, но уже с использованием генератора списков представлен на рисунке 6.

```
>>> numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> even_numbers = [number for number in numbers if number % 2 == 0]
>>> even_numbers
[2, 4, 6, 8, 10]
```

Рисунок 6 – Фильтрация списка по условию четности с использованием генератора списков

Из примера видно, что генератор списков занимает меньше строк кода и выглядит более лаконично.

Разберем также пример возведения чисел в квадрат с использованием цикла *for* и генератора списков. На рисунке 7 представлен пример с использованием цикла *for*.

```
>>> numbers = [1, 2, 3, 4, 5]
>>> squared_numbers = []
>>> for number in numbers:
...     squared_numbers.append(number ** 2)
...
...
>>> squared_numbers
[1, 4, 9, 16, 25]
```

Рисунок 7 – Возведение чисел в квадрат с использованием цикла *for*

Возведение чисел в квадрат, но уже с использованием генератора списков

представлено на рисунке 8.

```
>>> numbers = [1, 2, 3, 4, 5]
>>> squared_numbers = [number ** 2 for number in numbers]
>>> squared_numbers
[1, 4, 9, 16, 25]
```

Рисунок 8 – Возведение чисел в квадрат с использованием генератора списков

Заключение

Генераторы списков представляют собой мощный инструмент в Python, который позволяет создавать списки более кратко и эффективно, чем с использованием циклов for и функций map/filter. Использование генераторов списков позволяет уменьшить количество кода и увеличить производительность. Однако, в некоторых случаях, генераторы списков могут быть менее читабельными, чем эквивалентный код с использованием циклов for.

Библиографический список:

1. Бэрри П. Изучаем программирование на Python [Текст] / П. Бэрри. – М.: Вильямс, 2014. – 243 с.

2. Гэддис Т. Начинаем программировать на Python [Текст] / Т. Гэддис. – СПб.: БХВ-Петербург, 2021. – 768 с.