УДК 004.423 **Информационные технологии**

Дрянкова Дарья Александровна, *студент факультета информатики и вычислительной техники, Хакасский государственный университет имени Н.Ф. Катанова, г. Абакан, Россия*

Замулин Иван Сергеевич, *заведующий кафедрой ПОВТиАС, Хакасский государственный университет имени Н.Ф. Катанова, г. Абакан, Россия*

ЛЯМБДА-ФУНКЦИИ В ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON

Аннотация: В статье рассматривается лямбда-функция языка программирования Python.

Ключевые слова: Python, лямбда-функция, функция.

Annotation: The article discusses the lambda function of the Python programming language.

Keywords: Python, lambda function, function.

Лямбда-функция - это функция, которая может быть определена без имени и может быть передана как аргумент в другую функцию или присвоена переменной. Она является одним из инструментов функционального программирования и широко используется в языках программирования, поддерживающих функциональное программирование.

Синтаксис лямбда-функции обычно состоит из ключевого слова "lambda", за которым следует список параметров функции через запятую, затем двоеточие, и, наконец, выражение, которое должна выполнить функция. В листинге 1 приведён пример синтаксиса лямбда-функции в языке Python:

```
lambda x, y: x + y
```

Листинг 1 – Синтаксис лямбда-функции в языке Python

Эта функция принимает два аргумента x и y, и возвращает их сумму.

Обычные функции и лямбда-функции имеют ряд различий.

Основное различие между обычной функцией и лямбда-функцией заключается в том, что лямбда-функции определяются без имени и не требуют объявления с помощью ключевого слова "def". Они могут быть использованы внутри других функций в качестве аргументов и не требуют явного присваивания имени. Обычные функции, с другой стороны, требуют объявления с помощью ключевого слова "def" и имеют явное имя. Они могут быть вызваны из других функций, но не могут быть переданы как аргументы без использования некоторых дополнительных конструкций языка.

Еще одно отличие между обычными функциями и лямбда-функциями заключается в их объеме кода. Обычные функции могут содержать произвольное количество строк кода, в то время как лямбда-функции обычно используются для определения коротких функций, которые можно определить в одной строке кода [1].

Лямбда-функции используются в языках программирования в целях упрощения и сокращения кода, а также для улучшения читаемости и стиля кода. Они часто используются в функциональном программировании, где функции являются основным строительным блоком приложений. Лямбда-функции могут использоваться в качестве аргументов для других функций, фильтрации и сортировки списков, создания анонимных функций для обработки событий и т.д.

Лямбда-функции в Python широко используются для различных задач, в том числе для фильтрации и сортировки списков, создания анонимных функций для обработки событий, создания функций высшего порядка, передачи кода в другие функции и т.д. Рассмотрим наиболее распространенные варианты использования лямбда-функций в Python.

1) Фильтрация и сортировка списков.

Лямбда-функции в Python часто используются для фильтрации и сортировки списков. Например, для фильтрации списка чисел, чтобы оставить только числа, которые больше 5, можно использовать код, представленный на рисунке 1.

```
>>> numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> filtered_numbers = list(filter(lambda x: x > 5, numbers))
>>> filtered_numbers
[6, 7, 8, 9, 10]
```

Рисунок 1 – Фильтрация списка чисел с помощью лямбда-функции

Для сортировки списка строк по длине, можно использовать код, представленный на рисунке 2.

```
>>> strings = ['this', 'is', 'a', 'list', 'of', 'strings']
>>> sorted_strings = sorted(strings, key=lambda x: len(x))
>>> sorted_strings
['a', 'is', 'of', 'this', 'list', 'strings']
```

Рисунок 2 – Сортировка списка строк с помощью лямбда-функции

2) Создание функций высшего порядка.

Лямбда-функции могут использоваться для создания функций высшего порядка - функций, которые принимают другие функции в качестве аргументов или возвращают функции в качестве результата. Например, для определения функции, которая будет применять заданную функцию к каждому элементу списка, можно использовать следующий код, изображенный на рисунке 3.

```
>>> def apply_to_each(func, lst):
...     return [func(x) for x in lst]
...
>>> numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> squared_numbers = apply_to_each(lambda x: x ** 2, numbers)
>>> squared_numbers
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Рисунок 3 – Использование лямбда-функции в функции высшего порядка

Здесь мы определили функцию apply_to_each, которая применяет заданную функцию func к каждому элементу списка lst. Затем мы использовали лямбда-функцию, чтобы определить функцию, которая будет возводить каждое число в квадрат, и передали ее в apply_to_each, чтобы получить список квадратов

чисел из исходного списка.

3) Передача кода в другие функции.

Лямбда-функции можно использовать для передачи кода в другие функции в качестве аргумента. Например, для сортировки списка по нескольким критериям можно использовать лямбда-функции следующим образом, показанным на рисунке 4.

```
>>> records = [
...     {'name': 'John', 'age': 25},
...     {'name': 'Jane', 'age': 30},
...     {'name': 'Bob', 'age': 20},
...     {'name': 'Alice', 'age': 27},
... ]
...
>>> sorted_records = sorted(records, key=lambda x: (x['age'], x['name']))
>>> sorted_records
[{'name': 'Bob', 'age': 20}, {'name': 'John', 'age': 25}, {'name': 'Alice', 'age
': 27}, {'name': 'Jane', 'age': 30}]
```

Рисунок 4 – Использование лямбда-функции в качестве аргумента в других функциях на примере сортировки списка

Здесь мы использовали лямбда-функцию, чтобы передать сортировочный ключ в функцию sorted, которая отсортировала список словарей по возрастанию возраста, а затем по имени.

4) Работа с функциями из модулей.

Лямбда-функции могут использоваться для работы с функциями из модулей, которые ожидают определенный формат функции-аргумента. Например, для применения функции map к списку строк для преобразования их в верхний регистр, можно использовать код, представленный на рисунке 5.

```
>>> strings = ['this', 'is', 'a', 'list', 'of', 'strings']
>>> upper_strings = list(map(lambda x: x.upper(), strings))
>>> upper_strings
['THIS', 'IS', 'A', 'LIST', 'OF', 'STRINGS']
```

Рисунок 5 – Использование лямбда-функции вместе с функциями из модулей на примере преобразования списка строк

Здесь мы использовали лямбда-функцию, чтобы передать функцию

str.upper в функцию map, которая применила эту функцию ко всем элементам списка strings.

Преимущества использования лямбда-функций:

Лямбда-функции являются мощным инструментом программирования, который может значительно упростить код и сделать его более читаемым и понятным. Они позволяют определять функции в одной строке кода без необходимости объявления имени функции. Это снижает количество переменных в коде и упрощает чтение и понимание кода.

Лямбда-функции обычно используются вместе с функциями высшего порядка, такими как map, reduce и filter, что позволяет сократить объем кода и уменьшить количество циклов в коде. Это делает код более понятным и улучшает его производительность.

Также лямбда-функции обладают высокой степенью гибкости и могут использоваться для различных задач, таких как фильтрация и сортировка списков, создание анонимных функций для обработки событий и т.д. Это позволяет программистам быстро реагировать на изменения в проекте и быстро реализовывать новые функции и возможности.

Одним из ключевых преимуществ лямбда-функций является их краткость и простота. Лямбда-функции позволяют создавать функции в одной строке кода, что делает код более легким для чтения и сокращает количество ошибок в коде. Это также позволяет быстро определять функции в режиме реального времени, что упрощает отладку кода.

Также лямбда-функции могут использоваться для передачи кода в другие функции, что позволяет программистам создавать более гибкие и универсальные приложения. Это также позволяет легко создавать адаптивный код, который может быть изменен и модифицирован без необходимости переписывать всю программу.

Недостатки использования лямбда-функций:

Главным недостатком лямбда-функций является их ограниченность. Лямбда-функции обычно используются для определения коротких функций,

которые можно определить в одной строке кода, что может привести к трудностям, когда необходимо создать функцию более сложной структуры. Например, лямбда-функции не могут содержать вложенные блоки кода или операторы return с несколькими значениями. Также они могут усложнить отладку кода, поскольку они не имеют имени и не могут быть прослежены в стеке вызовов при ошибке.

Еще одним недостатком лямбда-функций является их неявность. Лямбда-функции могут использоваться вместо именованных функций, что может привести к потере ясности в коде. Это может усложнить поддержку кода в будущем, особенно если разработчики, не знакомые с лямбда-функциями, пытаются понять, что происходит в коде [2].

Также важно отметить, что не все языки программирования поддерживают лямбда-функции или поддерживают их в разной степени. Например, лямбда-функции в Python могут быть использованы в любом месте, где ожидается функция, в то время как в языке Java они требуют некоторых дополнительных конструкций для использования внутри методов.

Заключение

Итак, мы рассмотрели лямбда-функции в Python, их синтаксис и преимущества. Лямбда-функции позволяют определить функцию в одной строке кода без необходимости использовать отдельное определение функции. Они могут быть использованы для передачи кода в другие функции, для работы с функциями из модулей, для работы с коллекциями данных и для создания анонимных функций.

Таким образом, лямбда-функции в Python широко используются для решения различных задач и позволяют создавать более компактный и читабельный код.

Библиографический список:

1. Доусон М. Программируем на Python [Текст] / Доусон М. – СПб.: Издательский Дом ПИТЕР, 2022. – 416 с. ISBN: 9785446113866.

2. Васильев А.Н. Программирование на Python в примерах и задачах [Текст] / Васильев А.Н. – М.: Бомбора, 2021, 2022. – 616 с. ISBN: 978-5-04-103199-2.