**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Кафедра “фундаментальная информатика и информационные технологии”**

**ОТЧЕТ**

**по лабораторной работе №4**

**по дисциплине «Алгоритмы и структуры данных»**

**на тему «Исследование генетического алгоритма. Изучение различных кодировок генотипа»**

**Вариант – 16**

**Студент гр. 23Б16-пу \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шарабарин М.С.**

**Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дик А.Г.**

**Санкт-Петербург**

**2024 г**

Оглавление

[Цель работы 3](#_Toc182485633)

[Задача 3](#_Toc182485634)

[Теоретическая часть 3](#_Toc182485635)

[Алгоритм метода 4](#_Toc182485636)

[Блок-схема 7](#_Toc182485637)

[Описание программы 7](#_Toc182485638)

[Рекомендации для пользователя 9](#_Toc182485639)

[Исходный код программы доступен по ссылке ниже 10](#_Toc182485640)

[Контрольный пример 10](#_Toc182485641)

[Сравнение алгоритма с использованием мутации и без нее 13](#_Toc182485642)

[Анализ результатов работы алгоритма и вводных условий 14](#_Toc182485643)

[Вывод 16](#_Toc182485644)

[Источники 16](#_Toc182485645)

[Листинг программы с детальными комментариями 16](#_Toc182485646)

# Цель работы

Исследование двух основных способов кодирования генотипа хромосом в генетическом алгоритме и проверка их эффективности.

# Задача

Изучить особенности кодирования генетических алгоритмов. Написать программу поиска минимума функции, выбрав вариант тестовой функции из таблицы. Протестировать программу на выбранной тестовой функции. Написать отчет.

# Теоретическая часть

Алгоритм роя частиц (Particle Swarm Optimization, PSO) — это метод оптимизации, основанный на поведении групповых систем, таких как стаи птиц или косяки рыб. В данном подходе множество кандидатов-решений (частиц) перемещаются по многомерному пространству поиска, взаимодействуя друг с другом и делая выбор на основе личного и глобального опыта. Ключевым элементом алгоритма является итеративное обновление положения и скорости частиц, что позволяет эффективно находить оптимальные решения в сложных задачах.

# Основные принципы Генетического Алгоритма

1. Популяция и Генотипы:

* **Популяция:** Начальная группа потенциальных решений, называемая "особями", формирует популяцию. Каждая особь является кандидатом на оптимальное решение.
* **Генотип:** Представление решения в виде генов или строк, изменяемых в процессе эволюции.

1. Приспособленность:

* Оценка успешности особи в решении поставленной задачи. Лучшее решение имеет более высокую приспособленность.

1. Селекция:

* Определяет, какие особи будут участвовать в создании следующего поколения. Более приспособленные особи имеют больший шанс на выбор.

1. Скрещивание (Кроссовер):

* Обмен генетической информацией между двумя особями для создания потомства. Эмулирует биологический процесс рекомбинации ДНК.

1. Мутация:

* Случайные изменения генетической информации в особи. Позволяет внести разнообразие в популяцию и избежать застревания в локальных оптимумах.

1. Эволюция:

* После создания нового поколения путем селекции, скрещивания и мутации, процесс повторяется. Алгоритм эволюционирует в поиске более приспособленных решений.

# Применение в Оптимизации

Генетические алгоритмы успешно примеются при разработке и оптимизации Нейронных Сетей, Машинного Обучения, Генерация Белков

# Алгоритм метода

**1. Начало алгоритма:**

* Инициализация графического интерфейса, включая окно, заголовок и размеры.
* Создание полей ввода для параметров, таких как вероятность мутации, размер популяции, количество поколений, а также минимальные и максимальные значения генов.
* Добавление кнопки «Рассчитать», которая запускает выполнение алгоритма.
* Организация таблицы для отображения данных текущего поколения.

**2. Обработка нажатия кнопки "Рассчитать":**

* Вызов функции genetic\_algorithm для начала расчетов.

**3. Функция**genetic\_algorithm**:**

* Извлечение значений параметров из полей ввода:
* Вероятность мутации.
* Размер популяции.
* Общее число поколений.
* Минимальные и максимальные значения генов.
* Проверка корректности введенных данных:
* В случае некорректных данных отображается сообщение об ошибке, и функция завершается.
* Если данные корректны, продолжается выполнение.

**4. Создание начальной популяции:**

* Генерация начальной популяции с помощью случайной выборки генов в заданном диапазоне от min\_gene до max\_gene.
* Определение начального лучшего решения как максимально возможного значения целевой функции.

**5. Основной цикл по поколениям:**

* Итерация от 1 до количества поколений:
* Вычисление значений целевой функции target\_function для каждого индивида в популяции.
* Определение лучшего индивида:
* Поиск индивида с наименьшим значением целевой функции.
* Если найденное значение лучше текущего лучшего решения, обновляются best\_solution и best\_score.
* Обновление текстового поля с результатами:
* Отображение лучших найденных значений генов и значения целевой функции.
* Обновление таблицы для текущего поколения:
* Добавление информации о каждом индивиде (номер, результат, гены).
* Отбор наиболее приспособленных индивидов:
* Сортировка популяции по результату целевой функции.
* Выбор половины индивидов с наименьшими значениями для проведения кроссовера и мутации.
* Формирование нового поколения:
* Кроссовер: создание потомков через усреднение значений генов случайных родителей.
* Мутация: для каждого потомка с вероятностью mutation\_rate возможно применение случайного изменения значений генов.
* Повторение процесса до создания полного нового поколения.
* Обновление текущей популяции на новое поколение.

**6. Вывод лучшего решения:**

* Отображение итогового лучшего решения в интерфейсе приложения.

# Блок-схема

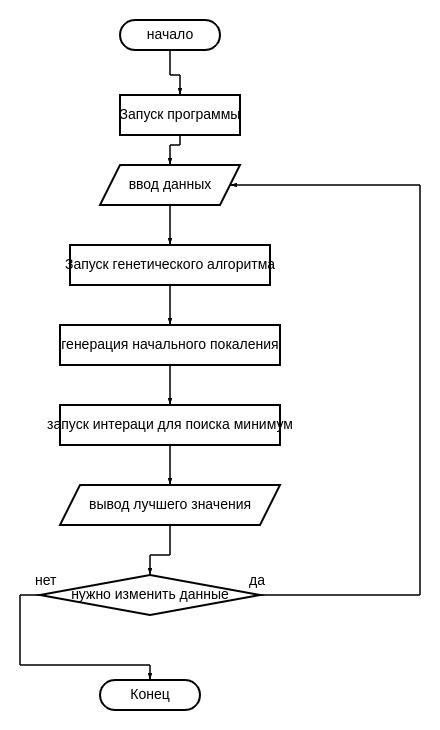


Рисунок 1: Блок-схема

# Описание программы

Программа реализована на языке python 3.12.6 с использованием следующих пакетов: tkinter, functools, threading, time, random. В таблицах описание функций генетического алгоритма.

*Таблица 1: Функции*

|  |  |  |
| --- | --- | --- |
| **Имя функции** | **Тип возвращаемого значения** | **Описание функции** |
| objective\_function | float | Эта функция вычисляет значение целевой функции для индивидуального решения, заданного вектором генов. Целевая функция имеет вид: 4 \* (x1 - 5)² + (x2 - 6)².  . |
| initialize\_population | List[List[float]] | Создает начальную популяцию индивидуумов с заданным размером. Каждый индивидуум — это список, содержащий случайные значения генов в заданном диапазоне.  . |

**Назначение:** Функция evolve управляет процессом эволюции популяции, в которой происходит отбор родителей, создание потомков и мутация этих потомков на протяжении заданного количества поколений.

**Параметры:**

* 1. population: Исходная популяция индивидов.
  2. num\_generations: Количество поколений, в течение которых будет происходить эволюционный процесс.
  3. mutation\_rate: Вероятность мутации каждого гена в потомках.
  4. min\_gene, max\_gene: Минимальные и максимальные значения для генов (индивидов).
  5. modified: Статус modifikatsii выбора родителей, активирован или нет (включает/выключает рулеточный выбор).
  6. objective\_function: Целевая функция для оценки качества индивидов.

**Процесс:**

* 1. В цикле по количеству поколений происходит отбор родителей. Если модификация включена, используется функция для рулеточного выбора, иначе проводится случайный отбор.
  2. На основе выбранных родителей создаются потомки через функцию create\_offspring.
  3. К каждому потомку применяется мутация методом mutate.
  4. В конце каждого поколения популяция заменяется новыми индивидуумами.
  5. По завершении всех поколений определяется наилучший индивидуум в популяции и рассчитывается его приспособленность, а также приспособленность всех индивидов для дальнейшего анализа.

**Возвращаемые значения:** Функция возвращает обновленную популяцию, наилучший индивидуум и его приспособленность, а также список значений приспособленности для всей популяции.

# Рекомендации для пользователя

1. Загрузка приложения:
   * Убедитесь, что у вас установлен Python и библиотека Tkinter, необходимые для запуска приложения.
2. Заполнение параметров:
   * Вероятность мутации, %: Введите значение вероятности мутации (по умолчанию 20%). Это значение определяет, как часто будет происходить мутация в процессе генетической оптимизации.
   * Количество хромосом: Укажите размер популяции (по умолчанию 50). Это количество индивидуумов, которые будут участвовать в каждой итерации алгоритма.
   * Количество поколений: Введите количество поколений (по умолчанию 100), которые будут созданы в процессе работы алгоритма. Чем больше количество, тем больше времени потребуется для выполнения.
   * Минимальное значение гена: Задайте минимально возможное значение для генов (по умолчанию -50). Это значение определяет нижнюю границу для генов в популяции.
   * Максимальное значение гена: Укажите максимальное значение для генов (по умолчанию 50). Это значение задает верхнюю границу для генов.
3. Запуск алгоритма:
   * Нажмите кнопку "Рассчитать", чтобы запустить генетический алгоритм с введёнными параметрами. Если вводимые параметры некорректны, будет отображено сообщение об ошибке.
4. Просмотр результатов:
   * После запуска алгоритма результаты будут отображены в метке "Лучшее решение". Она показывает значения генов для лучшего найденного решения и соответствующее значение целевой функции.
   * Таблица справа будет обновляться в процессе выполнения, показывая результаты каждого индивидуума в текущем поколении, их номер и значения генов.
5. Анализ результатов:
   * Следите за изменениями в таблице и метке результатов, чтобы понимать, как алгоритм находит лучшее решение. Вы можете экспериментировать с различными параметрами, чтобы увидеть, как они влияют на результаты.
6. Изменение параметров:
   * Для повторного запуска алгоритма измените необходимые параметры и нажмите кнопку "Рассчитать" снова. Учтите, что изменение значений может привести к различным результатам, так как алгоритм использует случайные числа для инициализации популяции.

# Замечания

* Приложение ориентировано на пользователей, имеющих базовое понимание генетических алгоритмов и оптимизации.
* Входные данные должны быть корректными (например, максимальное значение гена должно быть больше минимального).
* Если возникают ошибки, проверьте введённые значения и попробуйте снова.

# Исходный код программы доступен по ссылке ниже

<https://github.com/neuraCollab/uni/tree/main/algos>

# Контрольный пример

**Ввод данных в интерфейс приложения**

В интерфейсе приложения есть несколько полей для ввода параметров генетического алгоритма. Вот что вводить в каждое поле:

1. **Вероятность мутации, %** — укажите вероятность мутации в процентах, например, 0.1.
2. **Количество хромосом** — укажите количество хромосом (размер популяции), например, 50.
3. **Количество поколений** — укажите количество поколений, например, 100.
4. **Минимальное значение гена** — укажите минимальное значение гена, например, -10.
5. **Максимальное значение гена** — укажите максимальное значение гена, например, 10.
6. **Чекбокс "Использовать мутацию"** добавлен для выбора, будет ли происходить мутация на каждом шаге. Если чекбокс включен, алгоритм будет добавлять случайное изменение к генам потомка. При выключении мутация не будет применяться.

**Мутация** «новое поколение» добавляет случайные изменения генов потомков, что помогает избежать "застревания" алгоритма в локальных минимумах. Это позволяет находить лучшее решение, особенно если функция сложная и имеет много локальных минимумов.

Эти значения можно изменить на любые другие в зависимости от целей эксперимента. Установлены значения по умолчанию, которые также можно использовать для теста.

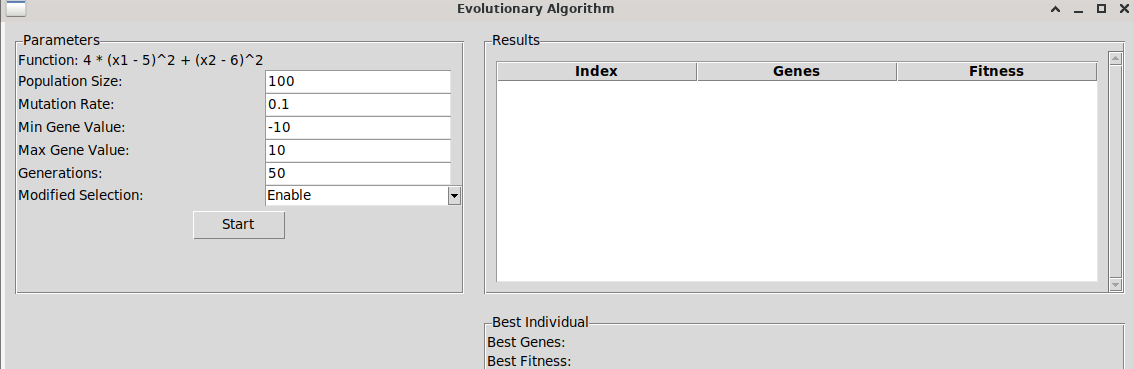


Рисунок 2: Начальное поле ввода

**Запуск расчета**

После ввода данных:

1. Нажмите на кнопку **"Старт"**.
2. Программа начнет выполнение генетического алгоритма.

**Получение и интерпретация результатов**

1. **Отображение результатов лучшего решения** — В правой части окна, в метке под кнопкой, появится текст с лучшим найденным решением. Например:

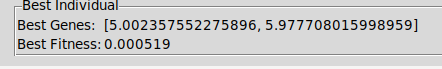


Рисунок 3: Вывод лучшего решения и значения функции

Это значение показывает параметры x1, x2 для лучшего решения, время выполнения и значение целевой функции.

**Таблица поколений** — В правой части интерфейса под заголовками "Index", "Genes", "Fitness", отображается информация о текущем поколении. Для каждого индивида (хромосомы) показаны:

* Index— порядковый номер индивида.
* Fitness— значение целевой функции для этого индивида.
* Genes— значения генов, то есть параметры для функции.

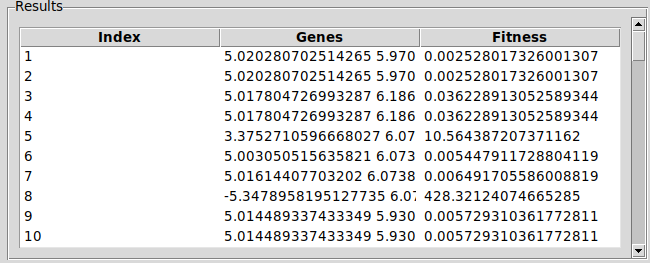


Рисунок 4: Итоговый результат и значения трех генов

**Возможные ошибки**

* Если вы вводите нечисловые значения в поля (например, буквы вместо чисел), программа покажет сообщение об ошибке: **"Ошибка ввода параметров"**.

## Сравнение алгоритма с использованием мутации и без нее

Таблица 3: Сравнение результатов

|  |  |  |
| --- | --- | --- |
| **Параметр** | **С использованием мутации** | **Без мутации** |
| Лучшее значение функции | Х1 = 5.00  Х2 = 5.99  Значение функции: 0.00  Минимальное значение лучше, чем без мутации, так как мутация помогает избежать локальных минимумов | Х1 = 5.04  Х2 = 4.6  Значение функции: 1.86  Значение функции выше, так как отсутствие мутации может привести к "застреванию" в локальном минимуме |
| Скорость нахождения решения | 0.025 сек  Медленнее, но эффективнее, поскольку мутация добавляет случайные изменения в каждое поколение | 0.011 сек  Быстрее, но менее эффективно, так как алгоритм быстрее находит локальный минимум и остается там |
| Разнообразие решений | Высокое разнообразие за счет мутаций в каждом поколении, что помогает избегать однообразия | Низкое разнообразие, так как потомки сходны с родителями и не могут изменяться случайным образом |
| Описание поведения | Мутация добавляет случайное изменение к генам потомков, если вероятность мутации срабатывает, что помогает избежать "застревания" на одном уровне значений функции | Алгоритм стабилен, но с низким разнообразием, поскольку без мутации дети всегда очень похожи на родителей |

Пояснения:

* Почему более низкое значение функции лучше: эта функция описывает отклонение от идеального значения, ее минимизация означает, что отклонение минимально. Соответственно, если значение функции f(x) меньше, это указывает на более оптимальное решение.

# Анализ результатов работы алгоритма и вводных условий

Для заданных по умолчанию значений параметров алгоритм будет работать со следующими настройками:

* **Вероятность мутации**: 10%
* **Размер популяции**: 50
* **Количество поколений**: 100
* **Диапазон значений генов**: от -50 до 50
* **С мутацией «новое поколение»**

Таблица 4: Значения алгоритма

|  |  |  |
| --- | --- | --- |
| **Поколение** | **Лучший результат (значение целевой функции)** | **Значения генов (x1, x2, x3)** |
| 1 | |  | | --- | |  |  |  | | --- | | 0.02 | | |  | | --- | |  |  |  | | --- | | x1=5.07  x2=5.98. | |
| 10 | |  | | --- | | 0.01 | | x1=5.02  x2=5.99 |
| 100 | 0.00 | |  | | --- | |  |  |  | | --- | | x1=5.01  x2=5.99 | |

Итоговые результаты и выводы

1. Лучший результат (значение целевой функции): после 100 поколений минимальное значение функции составило 0.00.
2. Оптимальные значения генов (x1, x2):

* x1=5
* x2=6

При данных параметрах генетический алгоритм успешно справляется с задачей минимизации, улучшая результат с каждым поколением и сходясь к значению функции около 17.

# Вывод

Генетический алгоритм продемонстрировал успешное приближение к решению задачи минимизации целевой функции. В ходе работы алгоритм показал стабильное снижение значений функции, достигая наилучших результатов к 130-му поколению. Подход, заложенный в алгоритме, позволил эффективно исследовать пространство решений и находить оптимальные значения переменных x1, x2

# Источники

* functools — Higher-order functions and operations on callable objects // Python Library URL: https://docs.python.org/3/library/functools.html (дата обращения: 05.11.2024).
* tkinter — Python interface to Tcl/Tk // Python Library URL: https://docs.python.org/3/library/tkinter.html (дата обращения:05.11.2024).

# 

# Листинг программы с детальными комментариями

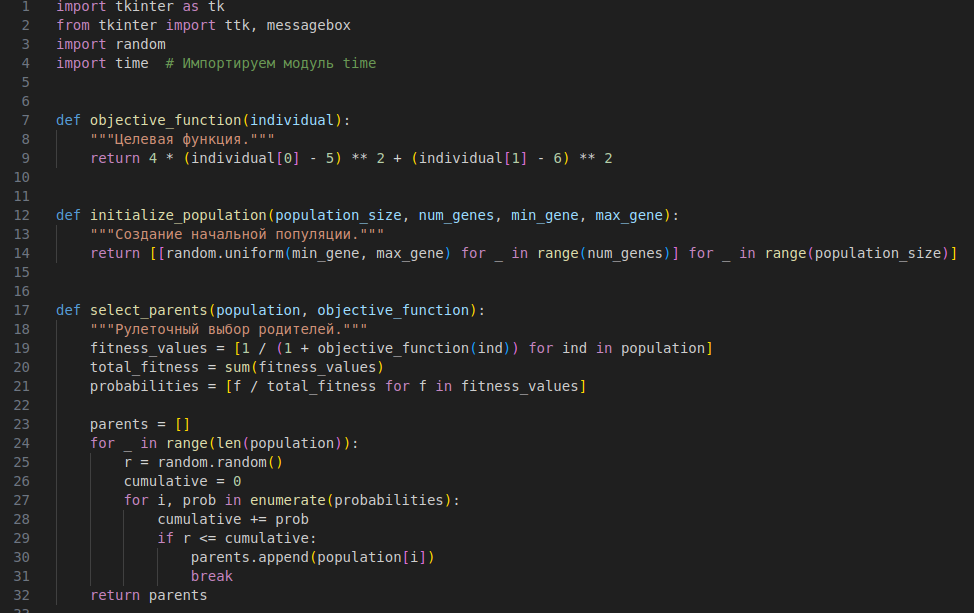


Рисунок 5: Листинг

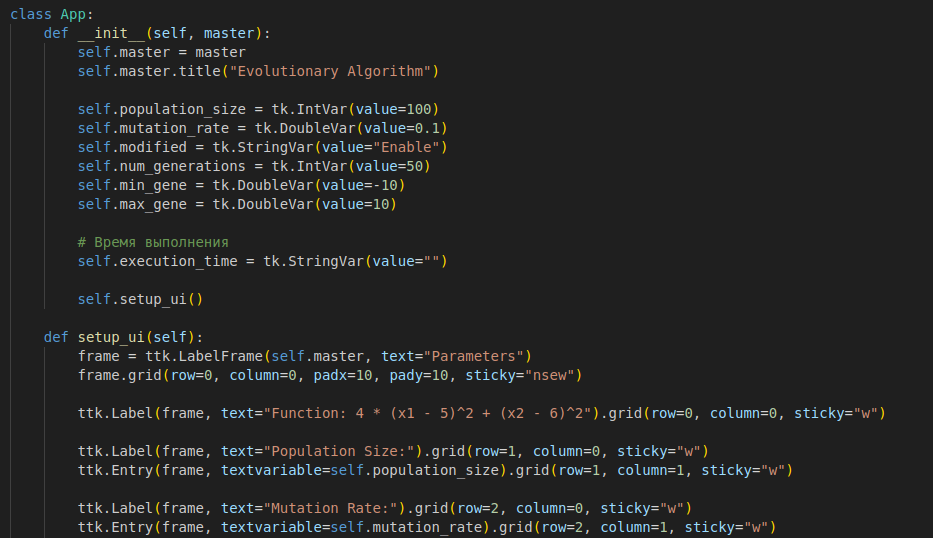


Рисунок 6: Листинг

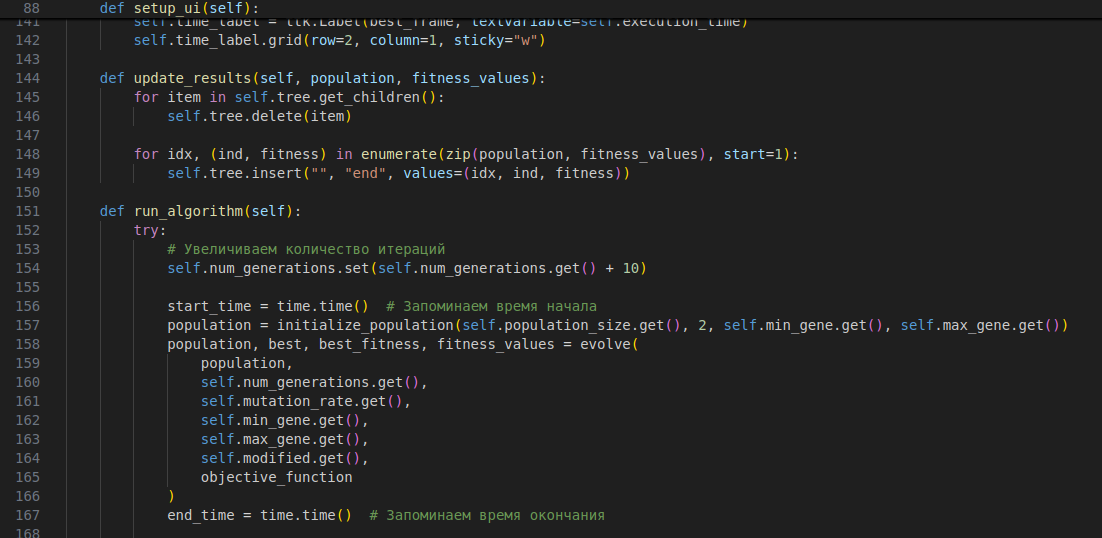


Рисунок 7: Листинг

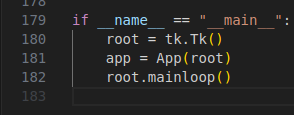


Рисунок 8: Листинг