Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

ОТЧЕТ

по лабораторной работе № 7

Название: Язык программирования Python

Дисциплина: Языки интернет программирования

Студент ИУ6-64Б (Группа) _____ (Подпись, дата) Д. О. Романов (И.О. Фамилия)

Преподаватель _____ (Подпись, дата) В. И. Кузнецов (И.О. Фамилия)

Москва, 2026

1. ЦЕЛЬ РАБОТЫ

Цель работы - знакомство с синтаксисом и базовыми возможностями языка программирования Python. Изучение встроенных структур данных, работы со строками, файловым вводом-выводом, а также освоение принципов создания контекстных менеджеров и использование сторонних библиотек.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1. Особенности языка Python

Python - высокоуровневый интерпретируемый язык программирования с динамической типизацией. Он поддерживает различные парадигмы программирования, включая объектно-ориентированную, функциональную и процедурную. Ключевой особенностью синтаксиса является использование отступов для выделения блоков кода.

2.2. Структуры данных

- Список (List): изменяемая упорядоченная последовательность элементов.
- Словарь (Dict): неупорядоченная коллекция пар "ключ-значение". Доступ к элементам осуществляется по ключу.
- Кортеж (Tuple): неизменяемая упорядоченная последовательность.

2.3. Контекстные менеджеры

Конструкция with позволяет управлять ресурсами, гарантируя выполнение определенных действий при входе в блок кода и выходе из него (например, автоматическое закрытие файлов). Для создания собственного контекстного менеджера необходимо реализовать методы __enter__ и __exit__ в классе.

3. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

3.1. Задание 1: Топ-N слов из файла

Разработана функция top_n_words, которая считывает текстовый файл, выполняет приведение текста к нижнему регистру, очистку от пунктуации с использованием регулярных выражений и фильтрацию слов длиной менее 3 символов. Для подсчета частоты слов использован класс collections.Counter.

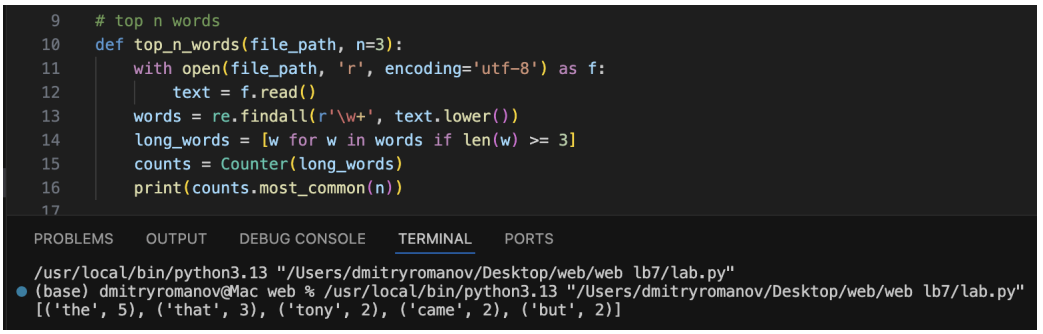

Рисунок 1 - Реализация и результат Задания 1

3.2. Задание 2: Слияние списка словарей

Реализована функция merge_dicts для объединения данных из списка словарей. Использован collections.defaultdict(int) для автоматического суммирования значений по ключам. Добавлена обработка ошибок для пропуска некорректных данных (отсутствующие ключи, нечисловые значения).

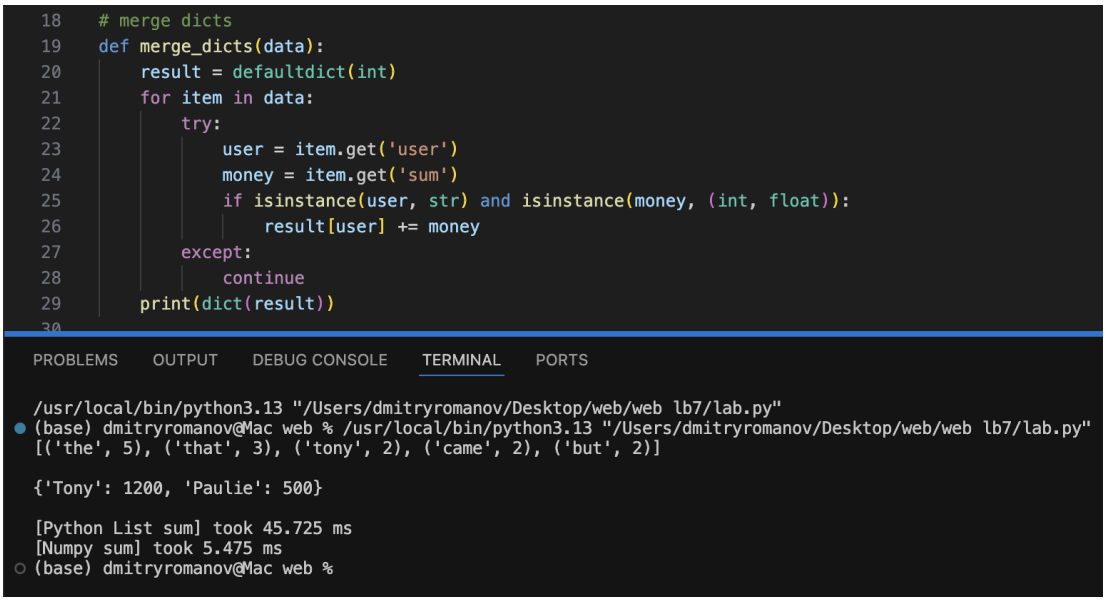

Рисунок 2 - Реализация и результат Задания 2

3.3. Задание 3: Контекст-менеджер и сравнение производительности

Разработан класс Timer, реализующий протокол контекстного менеджера для измерения времени выполнения блока кода с использованием time.perf_counter(). Проведено сравнение скорости суммирования массива из 10 миллионов элементов, выполненное стандартными средствами Python и библиотекой NumPy. Результаты показали, что NumPy выполняет операцию значительно быстрее, что подтверждает эффективность использования векторизованных операций для больших объемов данных.

```python
class Timer:
    def __init__(self, name, stream=sys.stdout):
        self.name = name
        self.stream = stream
    def __enter__(self):
        self.start = time.perf_counter()
        return self
    def __exit__(self, exc_type, exc_val, exc_tb):
        elapsed = (time.perf_counter() - self.start) * 1000
        print(f"[{self.name}] took {round(elapsed,3)} ms", file=self.stream)
```

Рисунок 3 - Реализация таймера и код сравнения

```python
    size = 10_000_000
    py_list = list(range(size))
    np_array = np.arange(size)
    with Timer("Python List sum"):
        sum(py_list)
    with Timer("Numpy sum"):
        np_array.sum()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

/usr/local/bin/python3.13 "/Users/dmitryromanov/Desktop/web/web lb7/lab.py"
(base) dmitryromanov@Mac web % /usr/local/bin/python3.13 "/Users/dmitryroman
[('the', 5), ('that', 3), ('tony', 2), ('came', 2), ('but', 2)]

{'Tony': 1200, 'Paulie': 500}

[Python List sum] took 45.725 ms
[Numpy sum] took 5.475 ms
(base) dmitryromanov@Mac web %
```

Рисунок 4 - Результаты сравнения производительности Python list и NumPy array

4. ВЫВОДЫ

1. В ходе работы изучены основные конструкции языка Python: функции, циклы, обработка исключений и работа со структурами данных.

2. Реализована обработка текстовых файлов и анализ частотности слов с использованием регулярных выражений.

3. Получены навыки работы со словарями и обработки некорректных данных при агрегации информации.

4. Освоено создание собственных контекстных менеджеров для профилирования кода.

5. Экспериментально подтверждено преимущество использования библиотеки NumPy для численных вычислений по сравнению со встроенными средствами языка.

5. ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое список (list) в Python, и чем он отличается от кортежа (tuple)?

   Список - изменяемая упорядоченная коллекция элементов. Кортеж - неизменяемая упорядоченная коллекция. Главное отличие: список можно редактировать после создания, а кортеж - нет, что делает его быстрее и позволяет использовать как ключ словаря.

2. Что такое функции высшего порядка и как они используются в Python?

   Это функции, которые принимают другие функции в качестве аргументов или возвращают их. В Python функции являются объектами первого класса, что позволяет использовать их в `map`, `filter` и декораторах.

3. Объясните разницу между операторами == и is в Python.

   Оператор == сравнивает значения объектов (эквивалентность данных). Оператор `is` сравнивает идентичность объектов (ссылаются ли переменные на одну и ту же область памяти).

4. Что такое декораторы и как они работают?

   Декоратор - это функция, которая принимает другую функцию и расширяет её поведение без изменения исходного кода. Работает на основе механизма замыканий и применяется с помощью синтаксиса `@`.

5. Как работает механизм обработки исключений, и зачем использовать try / except / else / finally?

   Механизм перехватывает ошибки выполнения. `try` содержит код, где возможна ошибка. `except` обрабатывает возникшее исключение. `else` выполняется, если ошибок не было. `finally` выполняется всегда, независимо от наличия ошибок.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Лутц М. Изучаем Python. Том 1. – 5-е изд. – СПб.: Диалектика, 2019.
2. Кузнецов В.И. Язык программирования Python: методические указания к выполнению лабораторной работы №7. – М.: МГТУ им. Н.Э. Баумана, 2025.
3. Официальная документация Python 3. – URL: https://docs.python.org/3/
4. Рамальо Л. Python. К вершинам мастерства. – М.: ДМК Пресс, 2022.