Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Перспективной инженерии
Департамент цифровых и робототехнических систем и электроники

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 5
дисциплины «Программирование на Python»

Выполнил: Мендеш Пашкоал Педру
2 курс, группа ИВТ-б-о-24-1,
09.03.01 «Информатика и Вычислительная техника», направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Роман А. доцент факультета цифровых, робототехнических систем и электроники института перспективной инженерии.

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

Тема: Работа с множествами и словарями в языке Python

Цель: приобретение навыков по работе с множествами и словарями при написании программ с помощью языка программирования Python версии 3.x.

Теоретическая часть

Множества (Set)

Множество - неупорядоченная коллекция уникальных элементов. Используются для хранения неповторяющихся данных и выполнения математических операций.

```
A = {1, 2, 3}
B = set([4, 5, 6])
```

Основные операции:

- Объединение: A.union(B) или A | B
- Пересечение: A.intersection(B) или A & B
- Разность: A.difference(B) или A – B

Словари (Dict)

Словарь - структура данных "ключ-значение". Позволяет хранить данные с быстрым доступом по ключу.

Создание:

```
student = {'name': 'Иванов И.И.', 'group': '101', 'grades': [5, 4, 3]}
```

Основные методы:

- keys() - все ключи
- values() - все значения
- items() - пары ключ-значение
- get() - получение значения по ключу

Методика выполнения работы

- Изучение теоретического материала
- Создание кода для задания в PyCharm.
- Создание репозитория GitHub: https://github.com/Pascoalpm/Sets-and-Dictionaries-in-Python
- Клонирование репозитория
- Настройка .gitignore для PyCharm

Индивидуальное задание

Задание 1

Определить результат выполнения операций над множествами. Считать элементы множества строками. Проверить результаты вручную. Номер варианта задания необходимо получить у преподавателя.

3.

$$A = \{a, h, m, o, r\};$$
$$B = \{j, k, o, u, y\};$$
$$C = \{g, h, j\};$$
$$D = \{g, j, q\};$$
$$X = (A \cap C) \cup (D \cap B);$$
$$Y = (A \cap B) \cup (D / C).$$

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == "__main__":
    u = set("abcdefghijklmnopqrstuvwxyz")  # CORREÇÃO: estava
"uvnxyz", correto é "uvwxyz"
    # Создание множеств
    a = {"a", "h", "m", "o", "r"}
    b = {"j", "k", "o", "u", "y"}
    c = {"g", "h", "j"}
    d = {"g", "j", "q"}

    print("Исходные множества:")
    print(f"a = {sorted(a)}")
    print(f"b = {sorted(b)}")
    print(f"c = {sorted(c)}")
    print(f"d = {sorted(d)}")
    print()

    # Вычисление X = (A ∩ C) ∪ (D ∩ B)
```

```
    print("1. Вычисление X = (A ∩ C) ∪ (D ∩ B)")

    A_intersection_C = a.intersection(c)
    print(f"   A ∩ C = {sorted(A_intersection_C)}")

    D_intersection_B = d.intersection(b)
    print(f"   D ∩ B = {sorted(D_intersection_B)}")

    x = A_intersection_C.union(D_intersection_B)
    print(f"   X = (A ∩ C) ∪ (D ∩ B) = {sorted(x)}")

    # Проверка вручную
    print("\n   Проверка вручную:")
    print("   A ∩ C = {'a', 'h', 'm', 'o', 'r'} ∩ {'g', 'h', 'j'} =
{'h'}")
    print("   D ∩ B = {'g', 'j', 'q'} ∩ {'j', 'k', 'o', 'u', 'y'} =
{'j'}")
    print("   X = {'h'} ∪ {'j'} = {'h', 'j'}")

    # Вычисление Y = (A ∩ B) ∪ (D/C)
    print("\n2. Вычисление Y = (A ∩ B) ∪ (D/C)")
    print("   (где D/C означает D - C)")

    A_intersection_B = a.intersection(b)
    print(f"   A ∩ B = {sorted(A_intersection_B)}")

    D_difference_C = d.difference(c)
    print(f"   D/C = D - C = {sorted(D_difference_C)}")

    y = A_intersection_B.union(D_difference_C)
    print(f"   Y = (A ∩ B) ∪ (D/C) = {sorted(y)}")

    # Проверка вручную
    print("\n   Проверка вручную:")
    print("   A ∩ B = {'a', 'h', 'm', 'o', 'r'} ∩ {'j', 'k', 'o', 'u',
'y'} = {'o'}")
    print("   D - C = {'g', 'j', 'q'} - {'g', 'h', 'j'} = {'q'}")
    print("   Y = {'o'} ∪ {'q'} = {'o', 'q'}")

    print("\n3. Итоговые результаты:")
    print(f"   x = {sorted(x)}")
    print(f"   y = {sorted(y)}")
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\HP\Documents\2 ano\Программирования на Python\Lab 5\Множества.py"
Исходные множества:
a = ['a', 'h', 'm', 'o', 'r']
b = ['j', 'k', 'o', 'u', 'y']
c = ['g', 'h', 'j']
d = ['g', 'j', 'q']

1. Вычисление X = (A ∩ C) ∪ (D ∩ B)
   A ∩ C = ['h']
   D ∩ B = ['j']
   X = (A ∩ C) ∪ (D ∩ B) = ['h', 'j']

   Проверка вручную:
   A ∩ C = {'a', 'h', 'm', 'o', 'r'} ∩ {'g', 'h', 'j'} = {'h'}
   D ∩ B = {'g', 'j', 'q'} ∩ {'j', 'k', 'o', 'u', 'y'} = {'j'}
   X = {'h'} ∪ {'j'} = {'h', 'j'}

2. Вычисление Y = (A ∩ B) ∪ (D/C)
   (где D/C означает D - C)
```

Рисунок 1. Код выполненный в PyCharm

```
2. Вычисление Y = (A ∩ B) U (D/C)
   (где D/C означает D - C)
   A ∩ B = ['o']
   D/C = D - C = ['q']
   Y = (A ∩ B) U (D/C) = ['o', 'q']

   Проверка вручную:
   A ∩ B = {'a', 'h', 'm', 'o', 'r'} ∩ {'j', 'k', 'o', 'u', 'y'} = {'o'}
   D - C = {'g', 'j', 'q'} - {'g', 'h', 'j'} = {'q'}
   Y = {'o'} U {'q'} = {'o', 'q'}

3. Итоговые результаты:
   x = ['h', 'j']
   y = ['o', 'q']
```

Рисунок 2. Код выполненный в PyCharm

Задание 2

Составить программу с использованием списков и словарей для решения задачи. Номер варианта определяется по согласованию с преподавателем.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == '__main__':
    # Список студентов
    students = []

    print("Система управления студентами")
    print("Введите команду (add, list, select, help, exit)")

    # Организовать бесконечный цикл запроса команд
    while True:
        # Запросить команду из терминала
        command = input(">>> ").lower()

        # Выполнить действие в соответствие с командой
        if command == 'exit':
            break

        elif command == 'add':
            # Запросить данные о студенте
            name = input("Фамилия и инициалы? ")
            group = input("Номер группы? ")

            # Успеваемость (5 элементов)
            print("Успеваемость (5 оценок через пробел, например: 5 4
3 2 4):")
            grades_input = input("Оценки: ")
            grades_list = grades_input.split()
```

```
            grades = []
            for i in range(5):
                if i < len(grades_list):
                    try:
                        grade = int(grades_list[i])
                        grades.append(grade)
                    except ValueError:
                        grades.append(0)
                else:
                    grades.append(0)

            # Создать словарь
            student = {
                'name': name,
                'group': group,
                'grades': grades
            }

            # Добавить словарь в список
            students.append(student)

            # Упорядочить по алфавиту
            if len(students) > 1:
                students.sort(key=lambda item: item.get('name', ''))

            print(f"Студент {name} добавлен.")

        elif command == 'list':
            print("\nСписок студентов:")
            print("-" * 40)

            for idx, student in enumerate(students, 1):
                print(f"{idx}. {student.get('name', '')} -
{student.get('group', '')}")
                print(f"   Оценки: {student.get('grades', [])}")

            if not students:
                print("Студентов нет")

            print("-" * 40)

        elif command == 'select':
            print("\nСтуденты с оценкой 2:")
            print("-" * 40)

            count = 0
            for student in students:
                if 2 in student.get('grades', []):
                    count += 1
                    print(f"{count}. {student.get('name', '')}
({student.get('group', '')})")

            if count == 0:
                print("Студентов с оценкой 2 не найдено.")

            print("-" * 40)

        elif command == 'help':
            print("\nСписок команд:")
            print("add - добавить студента")
```

```
            print("list - вывести список студентов")
            print("select - запросить студентов с оценкой 2")
            print("help - отобразить справку")
            print("exit - завершить работу с программой")

        else:
            print(f"Неизвестная команда '{command}'", file=sys.stderr)
            print("Введите 'help' для списка команд")
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\HP\Documents\2 ano\Программирования на Python\Lab 5\Система управлени
Система управления студентами
Введите команду (add, list, select, help, exit)
>>> add
Фамилия и инициалы? Мендеш П.П
Номер группы? 2
Успеваемость (5 оценок через пробел, например: 5 4 3 2 4):
Оценки: 2 3 5 6 4
Студент Мендеш П.П добавлен.
>>> list

Список студентов:
----------------------------------------
1. Мендеш П.П - 2
   Оценки: [2, 3, 5, 6, 4]
----------------------------------------
>>> select

Студенты с оценкой 2:
----------------------------------------
1. Мендеш П.П (2)
----------------------------------------
```

Рисунок 3. Код выполненный в PyCharm

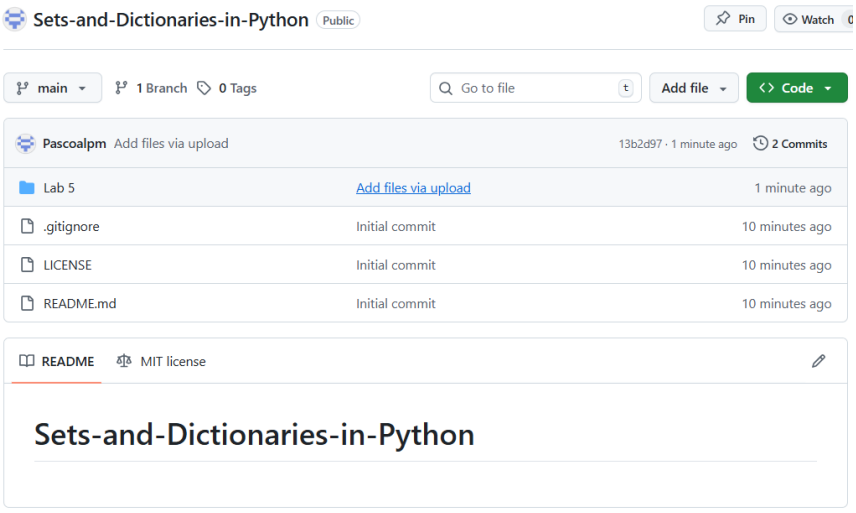

Рисунок 4. Файлы загружены в репозиторий Github.

Вывод

В ходе лабораторной работы были изучены основные принципы работы с множествами и словарями в Python. Полученные знания применены на практике при решении индивидуальных заданий.

Ответы на вопросы для защиты работы

1. Что такое множества в языке Python?

Множества в Python — это неупорядоченные коллекции уникальных элементов. Они используются для хранения неповторяющихся данных и выполнения математических операций над множествами.

2. Как осуществляется создание множеств в Python?

Множества можно создать двумя способами:

- Используя фигурные скобки: {1, 2, 3}
- Используя функцию set(): set([1, 2, 3])

Пустое множество создается только с помощью set(), так как {} создает пустой словарь.

3. Как проверить присутствие/отсутствие элемента в множестве?

Для проверки присутствия используется оператор in: элемент in множество

Для проверки отсутствия используется оператор not in: элемент not in множество

4. Как выполнить перебор элементов множества?

Перебор выполняется с помощью цикла for:

```
for элемент in множество:
    print(элемент)
```

Порядок перебора не гарантирован, так как множества неупорядочены.

5. Что такое set comprehension?

Set comprehension — это способ создания множеств с использованием компактного синтаксиса, аналогичного списковым включениям:

{выражение for элемент in итерируемый_объект if условие}

6. Как выполнить добавление элемента во множество?

- Добавление одного элемента: множество.add(элемент)
- Добавление нескольких элементов: множество.update([элемент1, элемент2])

7. Как выполнить удаление одного или всех элементов множества?

- remove(элемент) — удаляет элемент, вызывает ошибку если элемента нет
- discard(элемент) — удаляет элемент, не вызывает ошибку если элемента нет
- pop() — удаляет и возвращает случайный элемент
- clear() — удаляет все элементы

8. Как выполняются основные операции над множествами?

- Объединение: union() или оператор |
- Пересечение: intersection() или оператор &
- Разность: difference() или оператор -
- Симметрическая разность: symmetric_difference() или оператор ^

9. Как определить, что некоторое множество является надмножеством или подмножеством другого множества?

- Проверка подмножества: A.issubset(B) или A <= B
- Проверка надмножества: A.issuperset(B) или A >= B
- Строгое подмножество: A < B
- Строгое надмножество: A > B

10. Каково назначение множеств frozenset?

frozenset — это неизменяемая версия множества. Используется когда нужно:

- Использовать множество как ключ словаря
- Хранить множество в другом множестве

- Гарантировать, что множество не будет изменено

11. Как осуществляется преобразование множеств в строку, список, словарь?

- В строку: ','.join(множество) (только для строковых элементов)
- В список: list(множество)
- В словарь: dict([(ключ, значение), ...]) (каждый элемент должен быть кортежем из двух элементов)

12. Что такое словари в языке Python?

Словари — это изменяемые неупорядоченные коллекции пар "ключ-значение". Ключи должны быть неизменяемыми типами, значения могут быть любыми.

13. Может ли функция len() быть использована при работе со словарями?

Да, len(словарь) возвращает количество пар ключ-значение в словаре.

14. Какие методы обхода словарей Вам известны?

- keys() — возвращает все ключи
- values() — возвращает все значения
- items() — возвращает пары (ключ, значение) в виде кортежей

15. Какими способами можно получить значения из словаря по ключу?

- Через квадратные скобки: словарь[ключ] (вызывает ошибку если ключа нет)
- Метод get(): словарь.get(ключ, значение_по_умолчанию)
- Метод setdefault(): словарь.setdefault(ключ, значение_по_умолчанию)

16. Какими способами можно установить значение в словаре по ключу?

- Прямое присваивание: словарь[ключ] = значение
- Метод update(): словарь.update({ключ: значение})
- Метод setdefault(): словарь.setdefault(ключ, значение)

17. Что такое словарь включений?

Словарь включений — это компактный способ создания словарей:

{ключ: значение for элемент in итерируемый_объект if условие}

18. Самостоятельно изучите возможности функции zip()

Функция zip() объединяет несколько итерируемых объектов:

```
ключи = ['a', 'b', 'c']
значения = [1, 2, 3]
словарь = dict(zip(ключи, значения))  # {'a': 1, 'b': 2, 'c': 3}
```

19. Самостоятельно изучите возможности модуля datetime

Модуль datetime предоставляет:

- date — работа с датами
- time — работа со временем
- datetime — комбинация даты и времени
- timedelta — разница между датами/временем
- Методы: today(), now(), strftime() для форматирования

Инструкции по выполнению лабораторной работы

Шаги выполнения:

1. Создать репозиторий на GitHub с лицензией MIT
2. Клонировать репозиторий на локальный компьютер
3. Настроить .gitignore для PyCharm/VSCode