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

Институт перспективной инженерии
Департамент цифровых, роботехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №5
дисциплины «Программирование на Python»
Вариант 7

Выполнила:
Еремина Татьяна Евгеньевна
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)

Проверил:
Воронкин Р.А., доцент департамента 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 при работе с множествами словарями при написании программ с помощью языка программирования Python версии 3.x.

Практическая часть:

Исходный репозиторий: https://github.com/TL-hash/labs5

Пример 1. Определить результаты выполнения операций над множествами. Считать элементы множества строками.

Решение:

1. u = set("abcdefghijklmnopqrstuvwxyz") создаёт универсальное множество U.
2. Заданы множества a, b, c, d.
3. x = (a.intersection(b)).union(c): a.intersection(b) общие элементы A и B, .union(c) объединяет с C.
4. bn = u.difference(b) все буквы, кроме тех, что в B.
5. cn = u.difference(c) все буквы, кроме тех, что в C.
6. y = (a.difference(b)).union(cn.difference(bn)) вычисляет $Y = (A \setminus B) \cup (\bar{C} \setminus \bar{B})$.

Листинг примера:

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

if __name__ == "__main__":
    u = set("abcdefghijklmnopqrstuvwxyz")

    a = {"b", "c", "h", "o"}
    b = {"d", "f", "g", "o", "v", "y"}
    c = {"d", "e", "j", "k"}
    d = {"a", "b", "f", "g"}

    x = (a.intersection(b)).union(c)
    print(f"x = {x}")

    bn = u.difference(b)
    cn = u.difference(c)

    y = (a.difference(b)).union(cn.difference(bn))
    print(f"y = {y}")
```

Результат выполнения:

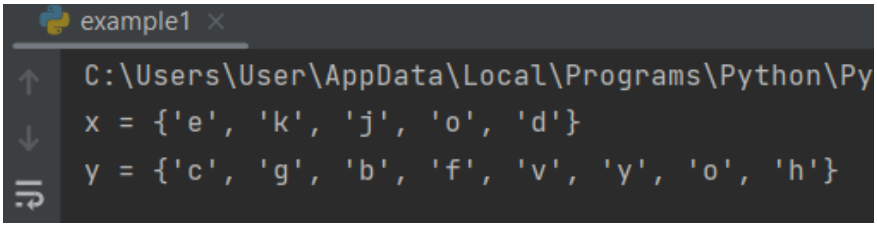

Рисунок 1 - Результат выполнения примера 1

Пример 2. Использовать словарь, содержащий следующие ключи: фамилия и инициалы работника, название занимаемой должности, год поступления на работу. Написать программу выполняющую следующие действия:

- Ввод с клавиатуры данных в список, состоящий из заданных словарей
- Записи должны быть размещены по алфавиту
- Вывод на дисплей фамилий работников, чей стаж работы в организаии превышает значение, введённое с клавиатуры
- Если таких работников нет, вывести на дисплей соответствующее сообщение.

Решение:

1. workers — список для хранения данных о работниках.
2. Команда exit — завершение работы.
3. Команда add — добавление работника. Считывает данные и создаёт словарь с ключами: 'name', 'post', 'year', добавляет его в список workers. После добавления (если сотрудников > 1) — сортирует список по Ф.И.О. с помощью sort(key=lambda ...). item.get('name', '') — безопасное получение значения (если ключа нет — вернёт пустую строку).
4. Команда list — вывод таблицы всех работников, формирует таблицу в виде ASCII-арта. enumerate(..., 1) — нумерация с 1. Выравнивание: {:>4} — по правому краю (номер), {:<30} — по левому (Ф.И.О., должность), {:>8} — по правому (год).

5. Команда select <стаж> — поиск по стажу. Извлекает число стажа из команды. Вычисляет стаж как текущий год - год приёма. Если стаж ≥ заданного — выводит Ф.И.О. Если никто не найден — выводит сообщение.

6. Команда help — справка по командам.

Листинг примера:

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

import sys
from datetime import date

if __name__ == "__main__":
    workers = []

    while True:
        command = input(">>> ").lower()

        if command == 'exit':
            break

        elif command == 'add':
            name = input("Фамилия и инициалы? ")
            post = input("Должность? ")
            year = int(input("Год поступления? "))

            worker = {
                'name': name,
                'post': post,
                'year': year,
            }

            workers.append(worker)
            if len(workers) > 1:
                workers.sort(key=lambda item: item.get('name', ''))

        elif command == 'list':
            line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 30,
                '-' * 20,
                '-' * 8
            )
            print(line)
            print(
                '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                    "№",
                    "Ф.И.О.",
                    "Должность",
                    "Год"
```

```
            )
        )
        print(line)

        for idx, worker in enumerate(workers, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                    idx,
                    worker.get('name', ''),
                    worker.get('post', ''),
                    worker.get('year', 0)
                )
            )

        print(line)

    elif command.startswith('select '):
        today = date.today()

        parts = command.split(' ', maxsplit=1)
        period = int(parts[1])

        count = 0
        for worker in workers:
            if today.year - worker.get('year', today.year) >= period:
                count += 1
                print(
                    '{:>4}: {}'.format(count, worker.get('name', ''))
                )


        if count == 0:
            print("Работники с заданным стажем не найдены.")

    elif command == 'help':
        print("Список команд:\n")
        print("add - добавить работника;")
        print("list - вывести список работников;")
        print("select <стаж> - запросить работников со стажем;")
        print("help - отобразить справку;")
        print("exit - завершить работу с программой.")

    else:
        print(f"Неизвестная команда {command}", file=sys.stderr)
```

Результат выполнения:

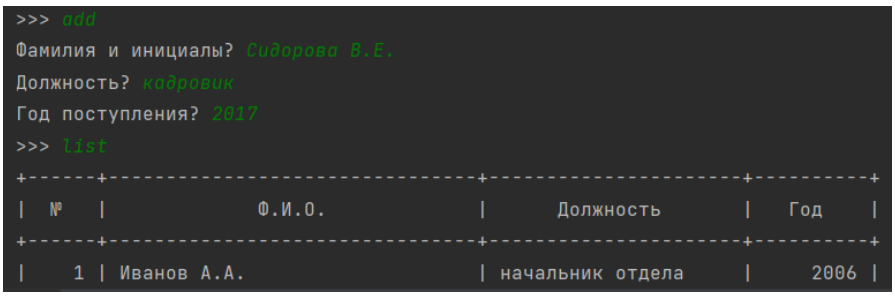

Рисунок 2 - Результат выполнения примера 2

Задание 1. Подсчитайте количество гласных в строке, введенной с клавиатуры, с использованием множеств.

Решение:

1. vowels = {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'} создаёт множество гласных букв латинского алфавита (как строчные, так и заглавные).

2. Считываем строку и инициализируем счётчик.

3. Проходим по каждому символу строки с помощью цикла, проверяем, является ли текущий символ гласной буквой (есть ли он во множестве vowels).

4. Если символ – гласная, увеличиваем счётчик.

5. Выводим итоговое число гласных.

Листинг задания 1:

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

if __name__ == "__main__":
    vowels = {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'}
    s = input("Введите строку: ")

    count = 0
    for char in s:
        if char in vowels:
            count += 1

    print(f"Количество гласных: {count}")
```

Результат выполнения:

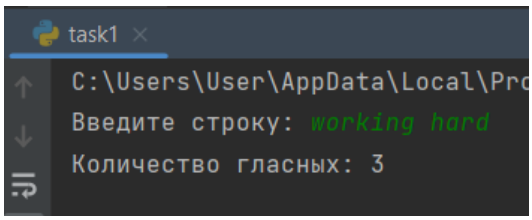

Рисунок 3 - Результат выполнения задания 1

Задание 2. Определите общие символы в двух строках, введённых с клавиатуры.

Решение:

1. set(s1) — преобразует строку в множество уникальных символов (дубликаты удаляются).
2. .intersection() — находит общие элементы между двумя множествами.
3. ".join(common) — объединяет символы из множества в одну строку для вывода.
4. Если общих символов нет, выводится соответствующее сообщение.

Листинг задания 2:

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

if __name__ == "__main__":
    s1 = input("Введите первую строку: ")
    s2 = input("Введите вторую строку: ")

    set1 = set(s1)
    set2 = set(s2)

    common = set1.intersection(set2)

    if common:
        print(f"Общие символы: {".join(common)}")
    else:
        print("Общих символов нет.")
```

Результат выполнения:

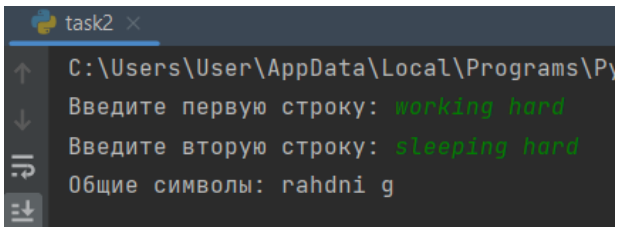

Рисунок 4 - Результат выполнения задания 2

Задание 3. Создайте словарь school, связав его с переменной. Наполните данными, отражающими количество учащихся в разных классах (например: '1а': 25, '2б': 30, ...).

Внесите изменения согласно следующему:

а) В одном из классов изменилось количество учащихся.

б) В школе появился новый класс.

в) В школе был расформирован (удалён) другой класс.

Вычислите общее количество учащихся в школе.

Решение:

1. Создаем словарь school и наполняем его данными.
2. school['2а'] = 32 обновляет количество учеников в классе '2а'.
3. school['6а'] = 26 создание нового ключа.
4. del school['1б'] удаление пары ключ-значение.
5. Подсчёт суммы total_students = sum(school.values()), school.values() — возвращает все значения (количество учеников).

Листинг задания 3:

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

if __name__ == "__main__":
    school = {
        '1а': 25,
        '1б': 28,
        '2а': 30,
        '2б': 27,
        '3а': 29,
        '4а': 31,
        '5а': 30,
    }
```

```
print("Исходное состояние школы:")
for klass, count in school.items():
    print(f"{klass}: {count} учеников")

school['2а'] = 32
print("\nа) Количество учеников в классе '2а' изменено на 32.")

school['6а'] = 26
print("б) Добавлен новый класс '6а' с 26 учениками.")

del school['1б']
print("в) Класс '1б' расформирован.")

total_students = sum(school.values())

print(f"\nОбщее количество учащихся в школе: {total_students}")
```

Результат выполнения:

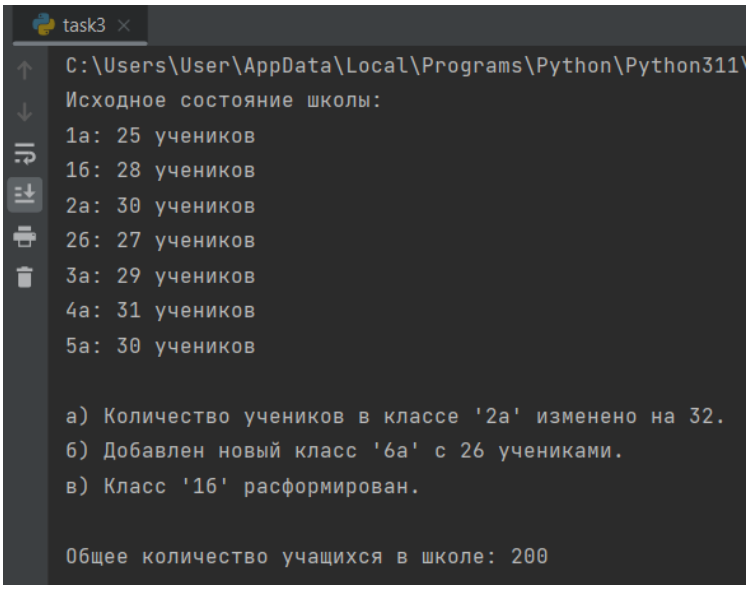

Рисунок 5 - Результат выполнения задания 3

Задание 4. Создайте словарь, где ключами являются числа, а значениями — строки.

Примените к нему метод items(), с помощью полученного объекта dict_items создайте новый словарь, «обратный» исходному: т.е. ключами являются строки, а значениями — числа.

Решение:

1. Создадим исходный словарь, где ключи – числа, значения – строки.

2. items() возвращает объект типа dict_items, содержащий пары (ключ, значение).

3. Создадим обратный словарь через списковое включение: for k, v in dict_items — перебирает каждую пару (ключ, значение) из исходного словаря, {v: k ...} — создаёт новый словарь, где: ключом становится значение исходного (v), значением становится ключ исходного (k).

Листинг задания 4:

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

if __name__ == "__main__":
    nums = {
        2: 'ab',
        4: 'cd',
        6: 'ef',
        8: 'gh',
       10: 'ij',
       12: 'kl',
       13: 'mo',
    }

    print(f"Исходный словарь: {nums}")

    dict_items = nums.items()

    swapped = {v: k for k, v in dict_items}

    print(f"\nСловарь, 'обратный' исходному: {swapped}")
```

Результат выполнения:

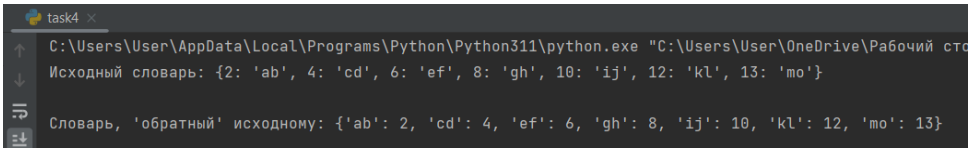

Рисунок 6 - Результат выполнения задания 4

Задание 1 (индивидуальное, вариант 7): определить результат выполнения операций над множествами. Считать элементы множества строками. Проверить результаты вручную.

$$A = \{b, f, g, m, o\};$$
$$B = \{b, g, h, l, u\};$$
$$C = \{e, f, m\};$$
$$D = \{e, g, l, p, q, u, v\};$$
$$X = (A \cap C) \cup B;$$
$$Y = (A \cap \bar{B}) \cup (C / D).$$

Решение:

1. Создаётся множество u, содержащее все буквы латинского алфавита.

2. Определяем исходные множества a, b, c, d.

3. x = (a.intersection(c).union(b)) пересечение A и C объеддиняем с B.

4. y = (a.intersection(bn).union(c.difference(d))) пересечение A с не B в объединении с C без B.

Листинг задания 1:

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

if __name__ == "__main__":
    u = set("abcdefghijklmnopqrstuvwxyz")

    a = {"b", "f", "g", "m", "o"}
    b = {"b", "g", "h", "l", "u"}
    c = {"e", "f", "m"}
    d = {"e", "g", "l", "p", "q", "u", "v"}

    x = (a.intersection(c).union(b))
    print(f"x = {x}")

    bn = u.difference(b)

    y = (a.intersection(bn).union(c.difference(d)))
    print(f"y = {y}")
```

Результат выполнения:

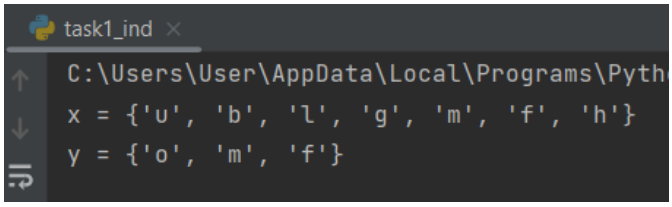

Рисунок 7 Результата выполнения индивидуального задания

Задание 2 (индивидуальное, вариант 7): Использовать словарь, содержащий следующие ключи: название пункта назначения; номер поезда; время отправления. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по времени отправления поезда; вывод на экран информации о поездах, направляющихся

в пункт, название которого введено с клавиатуры; если таких поездов нет, выдать на дисплей соответствующее сообщение.

Решение:

1. command = input(">>> ").strip().lower() считывает команду, удаляет лишние пробелы и приводит к нижнему регистру.

2. Команда exit завершает цикл и программу.

3. departure_time = datetime.strptime(time_str, "%H:%M").time() преобразует строку времени в объект time; при ошибке — выводит сообщение и возвращается к вводу команды.

4. Создаётся словарь train с тремя ключами: 'destination', 'number', 'departure_time'.

5. Сортируется список по времени отправления с помощью trains.sort(key=lambda item: item['departure_time'])

6. Формируется таблица в формате ASCII с заголовками при вызове команды list.

7. Команда select <пункт> извлекает название пункта из команды. Если его нет — выводит ошибку и продолжает цикл.

8. found_trains = [train for train in trains if train['destination'].lower() == target_destination.lower()] использует списковое включение для фильтрации поездов по пункту назначения без учёта регистра.

9. Команда help выводит список всех доступных команд с кратким описанием.

Листинг задания 2:

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

import sys
from datetime import datetime

if __name__ == "__main__":
    trains = []

    while True:
        command = input(">>> ").strip().lower()
```

```
if command == 'exit':
    break

elif command == 'add':
    destination = input("Название пункта назначения? ")
    number = input("Номер поезда? ")
    time_str = input("Время отправления (ЧЧ:ММ)? ")

    try:
        departure_time = datetime.strptime(time_str, "%H:%M").time()
    except ValueError:
        print("Неверный формат времени. Используйте ЧЧ:ММ.", file=sys.stderr)
        continue

    train = {
        'destination': destination,
        'number': number,
        'departure_time': departure_time
    }

    trains.append(train)

    trains.sort(key=lambda item: item['departure_time'])

elif command == 'list':
    line = '+-{}-+-{}-+-{}-+'.format(
        '-' * 20,
        '-' * 15,
        '-' * 10
    )
    print(line)
    print(
        '| {:^20} | {:^15} | {:^10} |'.format(
            "Пункт назначения",
            "Номер поезда",
            "Время"
        )
    )
    print(line)

    for train in trains:
        print(
            '| {:<20} | {:<15} | {:>10} |'.format(
                train['destination'],
                train['number'],
                train['departure_time'].strftime("%H:%M")
            )
        )

    print(line)

elif command.startswith('select '):
```

```
            parts = command.split(' ', maxsplit=1)
            if len(parts) < 2:
                print("Не указано название пункта назначения.", file=sys.stderr)
                continue

            target_destination = parts[1].strip()

            found_trains = [
                train for train in trains
                if train['destination'].lower() == target_destination.lower()
            ]

            if found_trains:
                print(f"\nПоезда, направляющиеся в {target_destination}:")
                for train in found_trains:
                    print(
                        f"  Поезд №{train['number']}, отправление в {train['departure_time'].strftime('%H:%M')}"
                    )
            else:
                print(f"Поездов в пункт '{target_destination}' не найдено.")

        elif command == 'help':
            print("Список команд:\n")
            print("add - добавить информацию о поезде;")
            print("list - вывести список всех поездов;")
            print("select <пункт> - показать поезда, идущие в указанный пункт;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")

        else:
            print(f"Неизвестная команда: {command}", file=sys.stderr)
```

Результат выполнения:

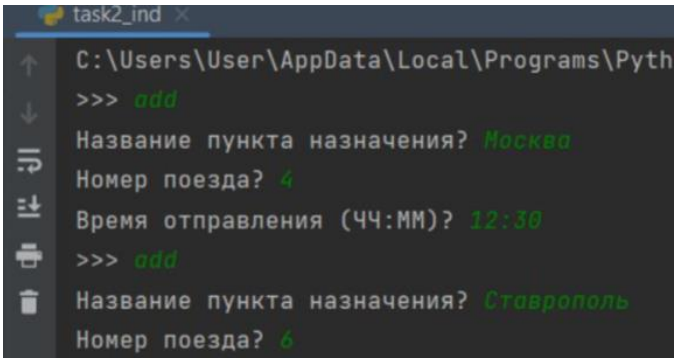

Рисунок 8 - Результат выполнения индивидуального задания (1)

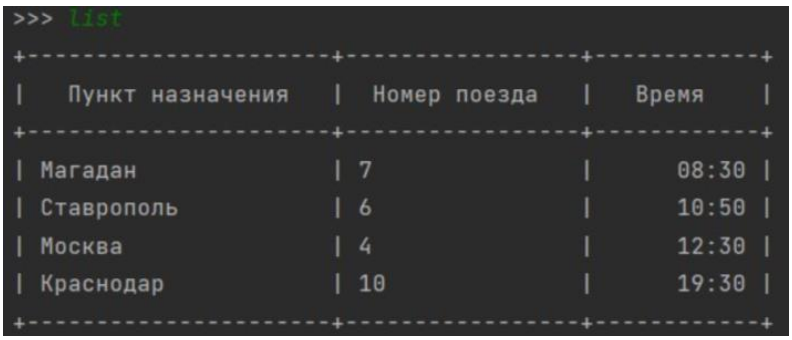

Рисунок 9 - Результат выполнения индивидуального задания (2)

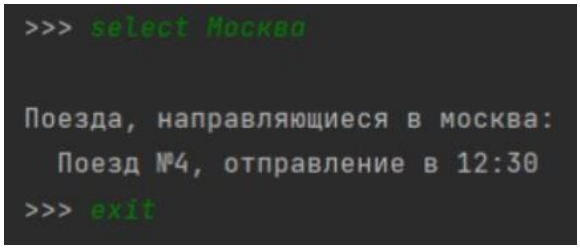

Рисунок 10 - Результат выполнения индивидуального задания (3)

Контрольные вопросы:

1. Что такое множества в языке Python?

Множество — это неупорядоченная коллекция уникальных элементов. Оно поддерживает операции теории множеств: объединение, пересечение, разность и др. Элементы множества должны быть неизменяемыми (например, числа, строки, кортежи).

2. Как осуществляется создание множеств в Python?

Множество можно создать:

- с помощью фигурных скобок: s = {1, 2, 3},
- с помощью функции set(): s = set([1, 2, 3]),
- с помощью генератора множеств: s = {x for x in range(5)}.

3. Как проверить присутствие/отсутствие элемента в множестве?

С помощью оператора in или not in: if element in my_set:
print("Элемент есть").

4. Как выполнить перебор элементов множества?

Используя цикл for: for item in my_set:
print(item)

5. Что такое set comprehension?

Set comprehension — это краткий способ создания множества с помощью выражения. Синтаксис: {выражение for элемент in последовательность if условие}.

Пример: {x**2 for x in range(5)}, вывод: {0, 1, 4, 9, 16}.

6. Как выполнить добавление элемента во множество?

С помощью метода .add(element): my_set.add(5)

7. Как выполнить удаление одного или всех элементов множества?

Удаление одного элемента: .remove(element) (выдаёт ошибку, если элемента нет) или .discard(element) (без ошибки).

Удаление всех элементов: .clear().

8. Как выполняются основные операции над множествами: объединение, пересечение, разность?

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

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

- Надмножество: A >= B или A.issuperset(B)
- Подмножество: A <= B или A.issubset(B)
- Строгое подмножество: A < B

10. Каково назначение множества frozenset?

frozenset — это неизменяемая версия множества. Его нельзя изменять после создания, поэтому его можно использовать в качестве ключа словаря или элемента другого множества.

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

В строку: str(my_set)

В список: list(my_set)

В словарь: невозможно напрямую, но можно создать словарь из пар (ключ, значение), используя dict(zip(keys, values)).

12. Что такое словари в языке Python?

Словарь — это неупорядоченная коллекция пар «ключ: значение». Ключи должны быть уникальными и неизменяемыми. Словари позволяют быстро находить значение по ключу.

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

Да, len(dict) возвращает количество пар «ключ-значение» в словаре.

14. Какие методы обхода словарей Вам известны?

По ключам: for key in my_dict: ...

По значениям: for value in my_dict.values(): ...

По парам: for key, value in my_dict.items(): ...

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

Прямой доступ: my_dict[key] (выдаёт ошибку, если ключа нет),

Метод .get(key, default): my_dict.get('key', 'default_value') (без ошибки).

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

Простое присваивание: my_dict[key] = value

Метод .update({key: value})

17. Что такое словарь включений?

Dictionary comprehension — это краткий способ создания словаря: {ключ: значение for элемент in последовательность if условие}.

Пример: {x: x**2 for x in range(5)}, вывод {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}.

18. Самостоятельно изучите возможности функции zip(). Приведите примеры её использования.

Функция zip() объединяет несколько итерируемых объектов в итератор кортежей.

Пример:

```
keys = ['a', 'b', 'c']
values = [1, 2, 3]
d = dict(zip(keys, values))  # {'a': 1, 'b': 2, 'c': 3}
```

19. Самостоятельно изучите возможности модуля datetime. Каким функционалом по работе с датой и временем обладает этот модуль?

Модуль datetime предоставляет классы для работы с датами и временем:

- datetime.datetime — дата и время,
- datetime.date — только дата,
- datetime.time — только время,
- datetime.timedelta — разница между датами/временем.

Примеры:

```
from datetime import datetime
now = datetime.now()
today = datetime.today()
delta = datetime(2025, 12, 17) - datetime.now()
```

Вывод: приобретены навыки при работе с множествами словарями при написании программ с помощью языка программирования Python версии 3.x. В ходе выполнения лабораторной работы были освоены ключевые операции с множествами и словарями в Python. Реализованы задачи на выполнение теоретико-множественных операций (объединение, пересечение, разность, дополнение), подсчёт и фильтрацию с использованием множеств, а также моделирование структурированных данных через словари.