5Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники института перспективной инженерии

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №5
дисциплины «Программирование на Python»
«Работа с множествами и словарями в языке Python»
Вариант 12

Выполнила:
Коробка Валерия Александровна
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.

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

Порядок выполнения работы

Адрес репозитория: https://github.com/13r4lera/Python5.git

Был создан общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python, выполнено клонирование данного репозитория. Файл .gitignore был дополнен правилами, необходимыми для работы с PyCharm (раскомментирована строка с .idea/).

Был проработан пример 1 из лабораторной работы.

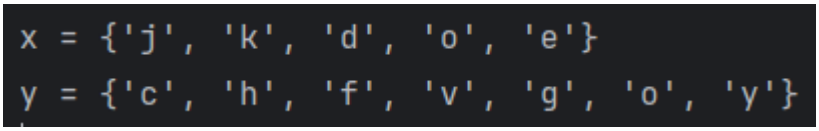

Рисунок 1. Результат работы примера 1

Полный код примера:

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

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

    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(d)).union(cn.difference(bn))

print(f"y = {y}")

Был проработан пример 2 из лабораторной работы.

```
>>> add
Фамилия и инициалы? Korobka V.A.
Должность? student
Год поступления?2024
>>> add
Фамилия и инициалы? Ivamov I.I.
Должность? teacher
Год поступления?2020
>>> list
+------+----------------------------------+----------------------+----------+
|  №   |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Ivamov I.I.                      | teacher              |     2020 |
|    2 | Korobka V.A.                     | student              |     2024 |
+------+----------------------------------+----------------------+----------+
>>> select 3
   1: Ivamov I.I.
>>> help
Список команд:

add - добавить работника;

list - вывести список работников;

select <стаж> - запросить работников со стажем;

help - отобразить справку;
```

Рисунок 2. Результат работы примера 2

Полный код 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', '')
                )
            )

        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 заднным стажем не найдены. ")

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

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

Было выполнено задание 1: Решите задачу: подсчитайте количество гласных в строке, введенной с клавиатуры с использованием множеств. Программа считывает строку, введённую пользователем, и переводит её в нижний регистр для корректного подсчёта символов. Затем с использованием множества гласных букв русского алфавита осуществляется подсчёт количества гласных в строке с помощью метода count(). В результате программа выводит общее число гласных букв во введённой строке.

```
Введите строку:  НАРУТО узумаки
В написанной строке 7 гласных.
```

Рисунок 3. Результат работы программы

Полный код программы:

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



if __name__ == '__main__':
    s = input("Введите строку:  ").lower()
    vowels = {'ё', 'у', 'е', 'э', 'о', 'а', 'ы', 'я', 'и', 'ю'}

    count = 0
    for vowel in vowels:
        count += s.count(vowel)

    print(f"В написанной строке {count} гласных.")
```

Было выполнено задание 2: Решите задачу: определите общие символы в двух строках, введенных с клавиатуры. Программа запрашивает у пользователя две строки и выполняет их обработку с использованием множеств. Символы каждой строки преобразуются в множества, после чего находится их пересечение, содержащее общие символы. Результат работы программы выводится на экран в виде множества общих символов.

```
Введите первую строку:  Я ЛЮБЛЮ гулять в лесу
Введите вторую строку:  я люблю плавать в море
Общие символы в двух строках: {'в', 'я', 'л', 'т', 'ь', ' ', 'е'}
```

Рисунок 4. Результат работы программы

Полный код программы:

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

if __name__ == '__main__':
    text1 = input("Введите первую строку:  ")
```

```
    text2 = input("Введите вторую строку:  ")

    common = set(text1).intersection(text2)
    print(f"Общие символы в двух строках: {common}")
```

Было решено задание 3. Условие задания: Решите задачу: создайте словарь, связав его с переменной school , и наполните данными, которые бы отражали количество учащихся в разных классах (1а, 1б, 2б, 6а, 7в и т. п.). Внесите изменения в словарь согласно следующему: а) в одном из классов изменилось количество учащихся, б) в школе появился новый класс, с) в школе был расформирован (удален) другой класс. Вычислите общее количество учащихся в школе.

Была написана программа, где для хранения информации используется словарь school, где ключом является название класса, а значением количество учеников в данном классе. При добавлении нового класса данные заносятся в словарь school. Вывод информации осуществляется в виде таблицы с названиями классов и количеством учеников. Команда изменения (change) позволяет обновить количество учащихся в выбранном классе. Команда удаления (delete) удаляет класс из словаря. Команда total вычисляет суммарное количество учеников путём суммирования всех значений словаря.

Полный код программы:

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

import sys

if __name__ == '__main__':
    school = {}

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

```
if command == "exit":
    break

elif command == "add":

    name = input("Введите цифру и букву класса без пробела: ")
    amount = int(input("Введите количество учеников: "))
    school[name] = amount

    if len(school) > 1:
        school = dict(sorted(school.items()))

elif command == "list":
    line = '+-{}-+-{}-+'.format(
        '-' * 20,
        '-' * 20
    )
    print(line)
    print(
        '| {:^20} | {:^20} |'.format(
            "Название класса",
            "Количество человек"
        )
    )
    print(line)

    for name, amount in school.items():
        print('| {:^20} | {:^20} |'.format(name, amount))
```

```
            print(line)

        elif command == 'help':
            print("Список команд:\n")
            print("add - добавить класс;\n")
            print("list - вывести список классов;\n")
            print("change класс количество - изменить количество учеников в
классе;\n")
            print("delete класс - расформировать класс;\n")
            print("help - отобразить справку;\n")
            print("exit - завершить работу с программой;\n")

        elif command.startswith("change"):
            parts = command.split(' ', maxsplit=2)
            class_name = parts[1]
            class_amount = int(parts[2])

            if class_name in school:
                school[class_name] = class_amount
            else:
                print("Такого класса нет в школе!", file=sys.stderr)

        elif command.startswith("delete"):
            parts = command.split(' ', maxsplit=1)
            class_name = parts[1]
            if class_name in school:
                school.pop(class_name)

        elif command == 'total':
            total_amount = sum(school.values())
```

```
        print(f"Общее количество учеников: {total_amount}")

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

```
>>> add
Введите цифру и букву класса без пробела: 1a
Введите количество учеников: 10
>>> add
Введите цифру и букву класса без пробела: 2
Введите количество учеников: 20
>>> list
+----------------------+----------------------+
|   Название класса    |  Количество человек  |
+----------------------+----------------------+
|          1a          |          10          |
|          2           |          20          |
+----------------------+----------------------+
>>> change 2 30
>>> delete 1a
>>> list
+----------------------+----------------------+
|   Название класса    |  Количество человек  |
+----------------------+----------------------+
|          2           |          30          |
+----------------------+----------------------+
>>> help
Список команд:

add - добавить класс;

list - вывести список классов;
```

Рисунок 5. Результат работы программы

```
>>> total
Общее количество учеников: 30
```

Рисунок 6. Подсчет общего количества учеников

Было решено задание 4. Условие задания: Решите задачу: 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.

В программе создаётся словарь numbers, где ключами являются числа, а значениями строки. С помощью метода items() извлекаются пары ключ–значение исходного словаря. На их основе формируется новый словарь reversed_numbers, в котором ключами становятся строки, а значениями соответствующие числа. В конце работы выводится на экран новый словарь с обратным отображением ключей и значений.

```
{'one': 1, 'two': 2, 'three': 3, 'four': 4, 'five': 5}

Process finished with exit code 0
```

Рисунок 7. Результат работы программы

Полный код программы:

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

if __name__ == '__main__':
    numbers = {1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}
    reversed_numbers = {}

    for k, v in numbers.items():
        reversed_numbers[v] = k

    print(reversed_numbers)
```

Условие индивидуального задания 1 по варианту 12:

$$A = \{b, k, n, o, q\};$$

$$B = \{a, b, k, u\};$$

$$C = \{o, p\};$$

$$D = \{a, m, n, y, z\};$$

$$X = (A \cup B) \cap D; Y = (\bar{A} \cap D) \cup (C/B).$$

В программе создаются четыре множества A, B, C и D, содержащие отдельные буквы латинского алфавита. Определяется универсальное множество всех букв u для вычисления дополнений. На основе этих множеств вычисляются два новых множества. В результате работы программы выводятся множества X и Y.

```
x = {'a', 'n'}. y = {'o', 'z', 'a', 'p', 'y', 'm'}

Process finished with exit code 0
```

Рисунок 8. Результат работы программы

Полный код программы:

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

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

    a = {'b', 'k', 'n', 'o', 'q'}
    b = {'a', 'b', 'k', 'u'}
    c = {'o', 'p'}
    d = {'a', 'm', 'n', 'y', 'z'}

    x = (a.union(b)).intersection(d)
    y = ((u - a).intersection(d)).union(c - b)

    print(f"x = {x}. y = {y}")
```

Индивидуальное задание 2 по варианту 12: Использовать словарь, содержащий следующие ключи: фамилия, имя; номер телефона; дата рождения (список из трех чисел). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть размещены по алфавиту;

вывод на экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры; если таких нет, выдать на дисплей соответствующее сообщение.

Информация сохраняется в виде списка словарей, где каждый словарь соответствует одному человеку. Для хранения даты рождения используется тип date из модуля datetime. При выполнении команды add осуществляется ввод персональных данных пользователя. Команда list предназначена для вывода списка всех сохранённых записей в табличном виде. Команда select позволяет вывести имена людей, у которых месяц рождения совпадает с указанным пользователем значением. Команда help отображает справочную информацию о доступных командах программы. Команда exit завершает работу приложения.

```
>>> add
Фамилия и имя: korobka vasya
Номер телефона: 000000
Дата рождения (гггг мм дд): 2023 09 19
>>> list
+------+--------------------------------+----------------------+-----------------+
|  №   |          Фамилия Имя           |    Номер телефона    |  Дата рождения  |
+------+--------------------------------+----------------------+-----------------+
|    1 | korobka lera                   | 7625743892           |      2006-09-21 |
|    2 | korobka simba                  | 0000000              |      2023-04-15 |
|    3 | korobka vasya                  | 000000               |      2023-09-19 |
+------+--------------------------------+----------------------+-----------------+
>>>
>>> Неизвестная команда
select 9
korobka lera
korobka vasya
>>> help
Список команд:

add - добавить работника;

list - вывести список работников;

select <месяц> - вывести имена людей, у которых день рождение в этом месяце;

help - отобразить справку;
```

Рисунок 9. Результат работы программы

Полный код программы:

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

import sys
from datetime import date



if __name__ == '__main__':
    people = []

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

        if command == "exit":
            break

        elif command == "add":
            name = input("Фамилия и имя: ")
            phone = input("Номер телефона: ")
            year, month, day = map(int, input("Дата рождения (гггг мм дд): ").split())
            birthday = date(year, month, day)

            person = {
                'name': name,
                'phone': phone,
                'birthday': birthday
            }
```

```
        people.append(person)
        if len(people) > 1:
            people.sort(key=lambda item: item.get('name', ''))

    elif command == "list":
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 15
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
                "№",
                "Фамилия Имя",
                "Номер телефона",
                "Дата рождения"
            )
        )
        print(line)

        for idx, person in enumerate(people, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>15} |'.format(
                    idx,
                    person.get('name', ''),
                    person.get('phone', ''),
                    str(person.get('birthday', ''))
                )
```

```
            )

        print(line)

    elif command.startswith('select'):
        parts = command.split(' ', maxsplit=1)
        birth_month = int(parts[1])

        count = 0
        for person in people:
            if person['birthday'].month == birth_month:
                count += 1
                print(person.get('name', ''))
        if count == 0:
            print("Нет человека с днем рождения в этом месяце")


    elif command == 'help':
        print("Список команд:\n")
        print("add - добавить работника;\n")
        print("list - вывести список работников;\n")
        print("select <месяц> - вывести имена людей, у которых день рождение в этом месяце;\n")
        print("help - отобразить справку;\n")
        print("exit - завершить работу с программой;\n")

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

Ответы на контрольные вопросы

1. Что такое множества в языке Python?

Множеством в языке программирования Python называется неупорядоченная последовательность уникальных значений. В качестве элементов этого набора данных могут выступать любын неизменяемые объекты, такие как числа, символы, строки.

2. Как осуществляется создание множеств в языке Python?

Это можно сделать, просто присвоив переменной последовательность значений, выделив их фигурными скобками. Другой способ создания множеств подразумевает использование вызова set.

3. Как проверить присутствие/отсутствие элемента в множестве?

Для проверки присутствия используется in. Для проверки отсутствия используется not in.

4. Как выполнить перебор всех элементов множества?

for a in {0, 1, 2}

5. Что такое set comprehension?

Для создания множества можно в Python воспользоваться генератором, позволяющим заполнять списки, а также другие наборы данных с учетом неких условий. Например, a = {i for i in [1, 2, 0, 1, 3, 2]}.

6. Как выполнить добавление элемента во множество?

Чтобы внести новые значения, потребуется вызвать метод add. Аргументом в данном случае будет добавляемый элемент последовательности.

7. Как выполнить удаление одного или всех элементов множества?

remove - удаление элемента с генерацией исключения в случае, если такого элемента нет. discard - удаление элемента без генерации исключения,

если элемент отсутствует. pop - удаление первого элемента, генерируется исключение при попытки удаления из пустого множества.

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

Чтобы объединить все элементы двух множеств, стоит воспользоваться методом union на одном из объектов. Чтобы найти общие элементы для двух разных множеств, следует применить функцию intersection, принимающую в качестве аргумента один из наборов данных. Чтобы вычислить разность для двух разных множеств, необходимо воспользоваться методом difference.

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

Чтобы выяснить, является ли множество a подмножеством b, стоит попробовать вывести на экран результат выполнения метода issubset. Чтобы узнать, является ли множество a надмножеством b, необходимо вызвать метод issuperset и вывести результат его работы на экран.

10. Каково назначение множеств frozenset?

Множество, содержимое которого не поддается изменению имеет тип frozenset. Значения из этого набора нельзя удалить, как и добавить новые.

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

Для преобразования множества в строку используется конкатенация текстовых значений, которую обеспечивает функция join. В этом случае ее аргументом является набор данных в виде нескольких строк. Запятая в кавычках выступает в качестве символа, разделяющего значения. Чтобы получить из множества словарь следует передать функции dict набор из нескольких пар значений, в каждом из которых будет находиться ключ. Для преобразования множества в список используется вызов list, получающий в качестве аргумента множество a.

12. Что такое словари в языке Python?

Словарь (dict) представляет собой структуру данных (которая еще называется ассоциативный массив), предназначенную для хранения произвольных объектов с доступом по ключу. Данные в словаре хранятся в формате ключ - значение.

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

Да, функция len() при применении к словарю возвращает количество пар «ключ–значение», содержащихся в словаре.

14. Какие методы обхода словарей Вам известны?

Элементы словаря перебираются в цикле for также, как элементы других сложных объектов. По умолчанию извлекаются только ключи, но по ключам всегда можно получить значения. У словаря как класса есть метод items(), который создает особую структуру, состоящую из кортежей, которые включают ключ и значение.

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

В цикле for можно распаковывать кортежи, таким образом сразу извлекая ключ и значение. Методы словаря keys() и values() позволяют получить отдельно перечни ключей и значений. То же можно сделать с помощью списковых включений.

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

С помщью setdefault() можно добавить элемент в словарь. Равносильно nums[4] = 'four', если элемент с ключом 4 отсутсттвует в словаре. Если он уже есть, то nums[4] = 'four' перезапишет старое значение, setdefault() - нет.

17. Что такое словарь включений?

Словарь включений аналогичен списковым включениям, за исключением того, что он создает объект словаря вместо списка.

18. Самостоятельно изучите возможности функции zip() приведите примеры ее использования.

Функция zip() используется для поэлементного объединения нескольких итерируемых объектов и возвращает итератор кортежей из

соответствующих элементов. Часто применяется для параллельного перебора списков и для создания словарей из двух последовательностей.

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

Модуль datetime предназначен для работы с датой и временем и позволяет создавать, сравнивать и форматировать даты, а также выполнять арифметические операции с временными интервалами. Он предоставляет классы date, time, datetime и timedelta.

Вывод: В ходе практической работы были изучены операции с множествами и словарями в языке Python, а способы их применения при решении задач.