

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

Институт цифрового развития  
Кафедра инфокоммуникаций

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.6**  
**дисциплины «Программирование на Python»**  
**Вариант \_\_\_\_**

Выполнил:  
Иващенко Олег Андреевич  
2 курс, группа ИВТ-б-о-22-1,  
09.03.02 «Информационные и  
вычислительные машины»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной техники и  
автоматизированных систем»

---

(подпись)

Руководитель практики:  
Воронкин Роман Александрович,  
доцент кафедры инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

**Тема:** «Работа со словарями в языке Python»

**Цель:** Приобретение навыков по работе со словарями при написании программ с помощью языка программирования Python версии 3.x.

### Порядок выполнения работы

Таблица 1 – Код программы example.py

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

import sys
from datetime import date

if __name__ == '__main__':
    # Список работников.
    workers = []

    # 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 работнике.
            name = input("Фамилия и инициалы? ")
            post = input("Должность? ")
            year = int(input("Год поступления? "))

            # Создать словарь.
            worker = {
                'name': name,
                'post': post,
                'year': year,
            }

            # Добавить словарь в список.
            workers.append(worker)
            # Отсортировать список в случае необходимости.
            if len(workers) > 1:
                workers.sort(key=lambda item: item.get('name', ''))

        elif command == 'list':
            # Заголовок таблицы.
            line = '+-{}-+-{}-+-{}-+-{}-+'.format(
```

```

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

# Вывести данные о всех сотрудниках.
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 '):
    # Получить текущую дату.
    today = date.today()

    # Разбить команду на части для выделения номера года.
    parts = command.split(' ', maxsplit=1)
    # Получить требуемый стаж.
    period = int(parts[1])

    # Инициализировать счетчик.
    count = 0
    # Проверить сведения работников из списка.
    for worker in workers:
        if today.year - worker.get('year', today.year) >= period:
            count += 1
            print(
                '{:>4}: {}'.format(count, worker.get('name', ""))
            )

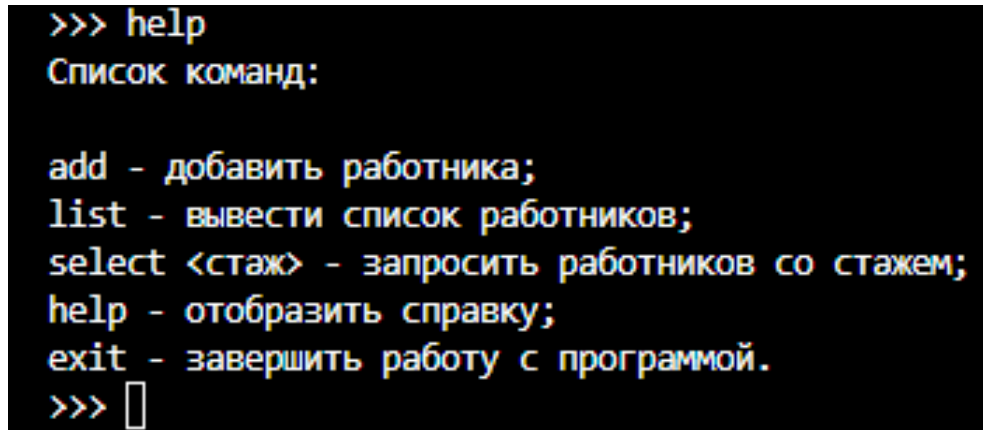
    # Если счетчик равен 0, то работники не найдены.
    if count == 0:
        print("Работники с заданным стажем не найдены.")

elif command == 'help':

```

```
# Вывести справку о работе с программой.
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.")

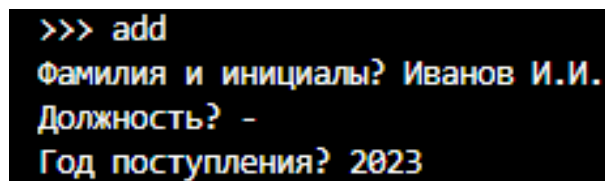
else:
    print(f"Неизвестная команда {command}", file=sys.stderr)
```



```
>>> help
Список команд:

add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку;
exit - завершить работу с программой.
>>> █
```

Рисунок 1.1 – Вывод команды help программы example.py



```
>>> add
Фамилия и инициалы? Иванов И.И.
Должность? -
Год поступления? 2023
█
```

Рисунок 1.2 – Вывод команды add и добавление нового сотрудника в программе example.py

Задание 1. 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 (удалён) другой класс. Вычислите общее количество учащихся в школе.

Таблица 2 – Код программы individual\_1:

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

```
if __name__ == "__main__":
    school = []

    while True:
        cmd = input(">>> ")

        if cmd == "exit":
            print("Завершение работы программы...")
            break

        elif cmd == "help":
            print("add - добавление нового класса")
            print("list - вывод список текущих классов")
            print("remove - удаление класса")
            print("change - изменение существующего класса")
            print("getsum - вывод общего количества учащихся")
            print("exit - выход из программы")

        elif cmd == "add":
            class_name = input("Название класса: ")
            class_students = input("Количество учеников: ")

            new_class = {
                'name': class_name,
                'students': class_students
            }
            school.append(new_class)

        elif cmd == "list":
            if len(school) > 0:
                for object_class in school:
                    print(f"Класс {object_class['name']}. "
                          f"Количество учеников: {object_class['students']}")
            else:
                print("Нет созданных классов")

        elif cmd == "remove":
            remove_class_name = input("Введите удаляемый класс: ")

            for object_class in school:
                removed = False

                if object_class['name'] == remove_class_name:
                    school.remove(object_class)
                    removed = True
                    break

            if removed == False:
                print("Класс не был найден")
            else:
```

```

        print("Класс успешно удалён")

    elif cmd == "change":
        changed_class = input("Введите название класса для изменения: ")
        changed = False

        for object_class in school:
            if object_class['name'] == changed_class:
                object_class['name'] = input("Введите новое имя класса: ")
                object_class['students'] = input("Введите новое "
                                                "количество учеников: ")

                changed = True
                break

        if changed == False:
            print("Класс не найден")
        else:
            print("Класс изменён")

    elif cmd == "getsum":
        class_sum = 0

        for object_class in school:
            class_sum += int(object_class['students'])

        print(f"Общее количество учащихся: {class_sum}")

    else:
        print("Введённая команда не существует. Введите help")

```

```

>>> help
add - добавление нового класса
list - вывод список текущих классов
remove - удаление класса
change - изменение существующего класса
getsum - вывод общего количества учащихся
exit - выход из программы

```

Рисунок 2.1 – Вывод команды help программы individual\_1.py

```

>>> list
Нет созданных классов
>>> add
Название класса: 7a
Количество учеников: 30
>>> add
Название класса: 7b
Количество учеников: 29
>>> list
Класс 7a. Количество учеников: 30
Класс 7b. Количество учеников: 29

```

Рисунок 2.2 – Вывод всех классов и добавление новых в программе

individual\_1.py

```
>>> list
Класс 7a. Количество учеников: 30
Класс 7b. Количество учеников: 29
>>> remove
Введите удаляемый класс: 7a
Класс успешно удалён
>>> list
Класс 7b. Количество учеников: 29
```

Рисунок 2.3 – Удаление классов в программе individual\_1.py

```
>>> list
Класс 7b. Количество учеников: 29
>>> change
Введите название класса для изменения: 7b
Введите новое имя класса: 7c
Введите новое количество учеников: 25
Класс изменён
>>> list
Класс 7c. Количество учеников: 25
```

Рисунок 2.4 – Изменение существующего класса в программе individual\_1.py

```
>>> list
Класс 7c. Количество учеников: 25
Класс 7a. Количество учеников: 30
Класс 10a. Количество учеников: 19
>>> getsum
Общее количество учащихся: 74
```

Рисунок 2.5 – Вывод общего количества учащихся в программе

individual\_1.py

Задание 2. 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.

Таблица 3 – Код программы individual\_2.py

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

if __name__ == "__main__":
```

```

first_dict = {3: 'tree',
              2: 'two',
              1: 'one'}
second_dict = dict(map(reversed, first_dict.items()))
print(first_dict.items())
print(second_dict.items())

```

```

dict_items([(3, 'tree'), (2, 'two'), (1, 'one')])
dict_items([('tree', 3), ('two', 2), ('one', 1)])

```

Рисунок 3 – Вывод программы individual\_3.py

Индивидуальное задание. Использовать словарь, содержащий следующие ключи: фамилия, имя, номер телефона, дата рождения (список из трёх чисел). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по трём первым цифрам номера телефона; вывод на экран информации о человеке, чья фамилия введена с клавиатуры; если такого нет, выдать на дисплей соответствующее сообщение.

Таблица 4 – Код программы individual\_3.py

```

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

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

    while True:
        cmd = input(">>> ")

        if cmd == "help":
            print("add - добавление новых записей")
            print("find - найти запись по фамилии")

        elif cmd == "add":
            surname = input("Введите фамилию: ")
            name = input("Введите имя: ")
            phone = input("Введите номер телефона: ")
            date = tuple(map(int, input("Введите дату рождения: ").split('.')))

            new_member = {'surname': surname,

```



```

        'name': name,
        'phone': phone,
        'date': date
    }
    member_list.append(new_member)
    member_list.sort(key=lambda item: item.get('phone')[:3])

elif cmd == "list":
    for member in member_list:
        print(f"{member['surname']} {member['name']}, "
              f"{member['phone']}, {member['date']}")

elif cmd == "find":
    surname = input("Введите фамилию: ")
    count = 0

    for member in member_list:
        if member['surname'] == surname:
            print(f"{member['surname']} {member['name']}, "
                  f"{member['phone']}, {member['date']}")
            count += 1

    if count == 0:
        print("Записи не найдены")

elif cmd == "exit":
    print("Завершение работы программы...")
    break

else:
    print(f"Команды {cmd} не существует")

```

```

>>> add
Введите фамилию: Иванов
Введите имя: Иван
Введите номер телефона: 78889995623
Введите дату рождения: 01.01.2001
>>> list
Иванов Иван, 78889995623, (1, 1, 2001)
>>> add
Введите фамилию: Иванов
Введите имя: Петр
Введите номер телефона: 77889995623
Введите дату рождения: 02.02.2002
>>> list
Иванов Петр, 77889995623, (2, 2, 2002)
Иванов Иван, 78889995623, (1, 1, 2001)

```

Рисунок 4.1 – Добавление новых записей и сортировка по номеру телефона в программе individual\_3.py

```
>>> list
Петров Андрей, 75559994623, (3, 3, 2003)
Иванов Петр, 77889995623, (2, 2, 2002)
Иванов Иван, 78889995623, (1, 1, 2001)
>>> find
Введите фамилию: Иванов
Иванов Петр, 77889995623, (2, 2, 2002)
Иванов Иван, 78889995623, (1, 1, 2001)
```

Рисунок 4.2 – Поиск записей по фамилии в программе individual\_3.py

### Контрольные вопросы

1. Что такое словари в языке Python?

Словарь в Python – это изменяемая структура данных, представляющая собой набор пар ключ-значение. Ключи уникальны в пределах словаря, и они используются для доступа к соответствующим значениям.

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

Да, функция len() может быть использована для получения количества элементов (пар ключ-значение) в словаре.

3. Какие методы обхода словарей Вам известны?

Наиболее распространённые методы обхода словарей включают использование цикла for для перебора ключей и пар ключ-значение. Методы keys(), values() и items() также могут использоваться для получения ключей, значений или пар ключ-значение соответственно.

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

Значения из словаря можно получить, используя квадратные скобки и указывая ключ, например:

```
value = new_dict['ключ']
```

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

Значение в словаре можно установить или изменить, также используя квадратные скобки, например:

```
new_dict['ключ'] = значение
```

6. Что такое словарь включений?

Словарь включений в Python – это компактный способ создания словаря в одной строке кода. Он позволяет создавать словари, используя цикл for для определения ключей и значений. Пример:

```
new_dict = {k: k**2 for k in range(1, 6)}
```

7. Самостоятельно изучите возможности функции zip() приведите примеры её использования.

Функция zip() в Python используется для объединения элементов из нескольких итерируемых объектов (например, два кортежа можно объединить в словарь с помощью метода zip([ключ], [значение])).

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

Модуль datetime предоставляет классы для работы с датой и временем. Он включает в себя классы datetime, date, time, timedelta и другие. С его помощью можно выполнять операции с датами, вычислять разницу между двумя датами, форматировать даты для вывода и многое другое.

**Выводы:** В процессе выполнения лабораторной работы были приобретены навыки по работе со списками при написании программ с помощью языка программирования Python версии 3.x, были написаны 4 программы: пример из лабораторной работы, 2 задания и индивидуальная задача.