

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 №9
дисциплины «Основы программной инженерии»

Выполнила:
Ламская Ксения Вячеславовна
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и
сопровождение программного
обеспечения», очная форма обучения

(подпись)

Доцент кафедры инфокоммуникаций
Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: Лабораторная работа 2.6 Работа со словарями в языке Python.

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

Порядок выполнения работы


1. Создание репозитория GitHub.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

 ksenia-lamskaya ▾

Repository name *

/ 9laba

✓ 9laba is available.

Great repository names are short and memorable. Need inspiration? How about [literate-potato](#) ?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  main as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создания репозитория

2. Проработала примеры из лабораторной работы.

```
#!/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)

```

Рисунок 2.1 – Код из примера 1

```

>>> add
Фамилия и инициалы? Ламская К.В.
Должность? Красотка
Год поступления? 2004
>>> list
+-----+-----+-----+-----+
| № | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Ламская К.В. | Красотка | 2004 |
+-----+-----+-----+-----+
>>> add
Фамилия и инициалы? Светалкова С.А.
Должность? Гадалка
Год поступления? 2004
>>> add
Фамилия и инициалы? Горчаков Р.Р.
Должность? Умник
Год поступления? 2005
>>> list
+-----+-----+-----+-----+
| № | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Горчаков Р.Р. | Умник | 2005 |
| 2 | Ламская К.В. | Красотка | 2004 |
| 3 | Светалкова С.А. | Гадалка | 2004 |
+-----+-----+-----+-----+
>>> select 10
1: Горчаков Р.Р.
2: Ламская К.В.
3: Светалкова С.А.

```

Рисунок 2.2 – Вывод программы из примера 1

3. 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.

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

import sys

if __name__ == '__main__':
    school = {'1a': 26, '1b': 27, '2b': 30, '6a': 30, '7v': 28}
    school['1a'] = 30
    school['3g'] = 27
    del school['7v']

    sum_of_students = sum(school.values())

    print(school)
    print(f'Общее количество обучающихся в школе: {sum_of_students}')
```

Рисунок 3.1 – Код программы

```
{'1a': 30, '1b': 27, '2b': 30, '6a': 30, '3g': 27}
Общее количество обучающихся в школе: 144
```

Рисунок 3.2 – Вывод программы

4. 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.

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

import sys

if __name__ == '__main__':
    dictt = {1: 'A', 2: 'B', 3: 'C', 4: 'D'}
    dict_items = {v: k for k, v in dictt.items()}

    print(dict_items)
```

Рисунок 4.1 – Код программы

```
{'A': 1, 'B': 2, 'C': 3, 'D': 4}
```

Рисунок 4.2 – Вывод программы

5. Использовать словарь, содержащий следующие ключи: название начального пункта маршрута; название конечного пункта маршрута; номер маршрута. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по номерам маршрутов; вывод на экран информации о маршруте, номер которого введен с клавиатуры; если таких маршрутов нет, выдать на дисплей соответствующее сообщение.

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

import sys

if __name__ == '__main__':
    print("Список команд:\n")
    print("add - добавить маршрут;")
    print("list - вывести список маршрутов;")
    print("select <тип> - вывод на экран пунктов маршрута, используя номер маршрута;")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")

    # Список маршрутов.
    point = []

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

        # Выполнить действие в соответствие с командой.
        match command:
            case 'exit':
                break

            case 'add':
                # Запросить данные о маршруте.
```

```
        name = input("Название начального пункта маршрута: ")
        name2 = input("Название конечного пункта маршрута: ")
        number = int(input("Номер маршрута: "))

        # Создать словарь.
        i = {'name': name, 'name2': name2, 'number': number}

        # Добавить словарь в список.
        point.append(i)

        # Отсортировать список в случае необходимости.
        if len(point) > 1:
            point.sort(key=lambda item: item.get('number', ''))

    case 'list':
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "№",
                "Начальный пункт.",
                "Конечный пункт",

```

```

        "№ маршрута"
    )
)
print(line)

# Вывести данные о всех маршрутах.
for idx, i in enumerate(point, 1):
    print(
        '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
            idx,
            i.get('name', ''),
            i.get('name2', ''),
            i.get('number', '')
        )
    )
print(line)

case 'select':
    # Разбить команду на части для выделения номера маршрута.
    parts = input("Введите значение: ")
    # Проверить сведения.
    count = 0
    for i in point:
        for k, v in i.items():
            if v == int(parts):
                print("Начальный пункт маршрута - ", i["name"])
                print("Конечный пункт маршрута - ", i["name2"])

                print("Конечный пункт маршрута - ", i["name2"])
                count += 1

    # Если счетчик равен 0, то маршруты не найдены.
    if count == 0:
        print("Маршрут с таким номером не найден")

case 'help':
    # Вывести справку о работе с программой.
    print("Список команд:\n")
    print("add - добавить маршрут;")
    print("list - вывести список маршрутов;")
    print("select <тип> - вывод на экран пунктов маршрута, используя номер маршрута;")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")

case _:
    print(f"Неизвестная команда {command}")

```

Рисунок 5.1 – Код программы


```

>>> add
Название начального пункта маршрута: A
Название конечного пункта маршрута: B
Номер маршрута: 2
>>> add
Название начального пункта маршрута: C
Название конечного пункта маршрута: D
Номер маршрута: 4
>>> add
Название начального пункта маршрута: G
Название конечного пункта маршрута: H
Номер маршрута: 7
>>> list
+-----+-----+-----+-----+
| № | Начальный пункт. | Конечный пункт | № маршрута |
+-----+-----+-----+-----+
| 1 | A | B | 2 |
| 2 | C | D | 4 |
| 3 | G | H | 7 |
+-----+-----+-----+-----+
>>> select 4
Неизвестная команда select 4
>>> select
Введите значение: 4
Начальный пункт маршрута - C
Конечный пункт маршрута - D

```

Рисунок 5.2 – Вывод программы

Контрольные вопросы

1. Что такое словари в языке Python?

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

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

Да, функция len() может быть использована для работы со словарями в Python. Она возвращает количество элементов (пар ключ-значение) в словаре.

3. Какие методы обхода словарей Вам известны?

- Цикл for по ключам
- Использование метода items(), который возвращает пары ключ-значение
- Обход только ключей с использованием метода keys()
- Обход только значений с использованием метода values()

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

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
for value in my_dict.values():
    print(value)
```

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

```
my_dict = {}
my_dict['ключ'] = 'значение'
```

6. Что такое словарь включений?

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

```
{x: x * x for x in (1, 2, 3, 4)}
```

```
{1: 1, 2: 4, 3: 9, 4: 16}
```

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

Функция `zip()` в Python используется для объединения двух или более итерируемых объектов (списков, кортежей, и т. д.) в один объект, создавая пары значений. Это может быть полезно, когда вам нужно объединить данные из нескольких источников. Вот примеры использования функции `zip()`.

```
names = ['Анна', 'Петр', 'Мария']
```

```
scores = [85, 92, 78]
```

```
student_data = list(zip(names, scores))
```

```
print(student_data)
```

Результат:

```
[('Анна', 85), ('Петр', 92), ('Мария', 78)]
```

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

Модуль `datetime` в Python предоставляет обширный функционал для работы с датой и временем. Вот некоторые из его основных возможностей

1. Создание объектов даты и времени:

- `datetime.date`: Представляет дату (год, месяц, день).

- `datetime.time`: Представляет время (час, минута, секунда, микросекунда).

- `datetime.datetime`: Представляет комбинацию даты и времени.

2. Получение текущей даты и времени:

- `datetime.datetime.now()`: Возвращает текущую дату и время.

3.Разбор и форматирование даты и времени:

- `datetime.datetime.strptime()`: Разбор строки в объект `datetime`.
- `datetime.datetime.strftime()`: Преобразование объекта `datetime` в строку с заданным форматом.

4.Арифметика с датой и временем:

- Можно выполнять операции сложения и вычитания времени и даты, а также вычислять разницу между двумя моментами времени.

5.Работа с таймзонами:

- Модуль `datetime` поддерживает работу с часовыми поясами и таймзонами.

6.Извлечение информации:

- Можно получать год, месяц, день, часы, минуты, секунды и другую информацию о дате и времени.

7.Выполнение сравнений:

- Можно сравнивать даты и времена на предмет того, какой из них раньше или позже.

8.Работа с интервалами времени:

- Модуль `datetime` поддерживает интервалы времени, которые позволяют выразить продолжительность времени.