МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Кафедра инфокоммуникаций

Отчет

по лабораторной работе №4

«Работа со словарями в языке Python»

по дисциплине:

«Введение в системы искусственного интеллекта»

Вариант 8

Выполнил: студент группы ИВТ-б-о-18-1 (2)
Михайличенко Руслан Михайлович
_____(подпись)
Проверил:
Воронкин Роман Александрович
_____(подпись)

Ставрополь, 2022 г.

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

Задание 1

8. 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 номерам поездов; вывод на экран информации о поезде, номер которого введен с клавиатуры; если таких поездов нет, выдать на дисплей соответствующее сообщение.

Ввод [*]:

```
import sys

if __name__ == '__main__':
    # Список .
    train = []

    # 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 .
            race = input("Название пункта назначения рейса ")
            number = input("Номер рейса ")
            type = float(input("Время "))

            # Создать словарь.
            trains = {
                'race': race,
                'number': number,
                'type': type,
            }

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

        elif command == 'list':
            # Заголовок таблицы.
            line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 30,
                '-' * 20,
                '-' * 20
            )
            print(line)
            print(
                '| {:^4} | {:^30} | {:^20} | {:^20} |'.format(
```

```
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^20} |'.format(
                "No",
                "Пункт",
                "Номер",
                "Тип самолёта."
            )
        )
        print(line)

        # Вывести данные о всех рейсах.
        for idx, trains in enumerate(train, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>20} |'.format(
                    idx,
                    trains.get('race', ''),
                    trains.get('number', ''),
                    trains.get('type', 0)
                )
            )

        print(line)

    elif command.startswith('select '):
        parts = command.split(' ', maxsplit=2)
        sel = (parts[1])

        count = 0
        for trains in train:
            if trains.get('race') == sel:

                count += 1
                print(
                    '{:>4}: {}'.format(count, trains.get('race', ''))
                )
                print('Номер рейса:', trains.get('number', ''))
                print('Тип поезда:', trains.get('type', ''))

        # Если счетчик равен 0, то рейсы не найдены.
        if count == 0:
            print("Рейс не найден.")

    elif command == 'help':
        # Вывести справку о работе с программой.
```



```
        # Если счетчик равен 0, то рейсы не найдены.
        if count == 0:
            print("Рейс не найден.")

    elif command == 'help':
        # Вывести справку о работе с программой.
        print("Список команд:\n")
        print("add - добавить рейс;")
        print("list - вывести список рейсов;")
        print("select <товар> - информация о рейсе;")
        print("help - отобразить справку;")
        print("exit - завершить работу с программой.")

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

```
>>> add
Название пункта назначения рейса 2
Номер рейса 3
Тип самолёта 13.20

>>>
```

Рисунок 1 – Листинг программы

Вывод: в ходе выполнения работы были приобретены навыки по работе со словарями при написании программ с помощью языка программирования Python версии 3.x.

Файлы с решением данных задач находится на **Github** :

https://github.com/Mihayilichenko/lab

Ответы на вопросы

1. Что такое словари в языке Python?

Словарь – это изменяемый тип данных. Следовательно, как и список он передается в функцию по ссылке. Поэтому иногда, чтобы избежать нежелательного изменения глобального словаря его копируют. Это делают и с другими целями.

Метод fromkeys() позволяет создать словарь из списка, элементы которого становятся ключами.

Применять метод можно как классу dict, так и к его объектам:

```
>>> a = [1, 2, 3]
>>> c = dict.fromkeys(a)
>>> c
{1: None, 2: None, 3: None}
>>> d = dict.fromkeys(a, 10)
>>> d
{1: 10, 2: 10, 3: 10}
>>> c
{1: None, 2: None, 3: None}
```

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

Словарь включений аналогичен списковым включениям, за исключением того, что он создаёт объект словаря вместо списка. Как и в случае со списком, мы можем использовать условный оператор внутри словаря включения, чтобы получить только элементы словаря, удовлетворяющие заданному критерию.

```
>>> {name: len(name) for name in ('Stack', 'Overflow', 'Exchange') if len(name)
> 6}
{'Exchange': 8, 'Overflow': 8}
```

Или переписать с помощью генераторного выражения.

>>> dict((name, len(name)) for name in ('Stack', 'Overflow', 'Exchange') if len(name) > 6)

{'Exchange': 8, 'Overflow': 8}

3. Какие методы обхода словарей Вам известны?

1) Если в цикле используются и ключи, и значения словаря, то нужно использовать метод .items();

2)Если в цикле используются только значения словаря, а ключи не важны, то нужно использовать метод .values();

3) Если в цикле нужны ключи словаря и ничего больше, то нужно использовать метод .keys().

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

Стандартный способ доступа к значению словаря – через квадратные скобки. Как видим, если ключ представлен в виде числа, то его пишем без кавычек. Если обратимся к несуществующему ключу, то получим в ответ ошибку KeyError.

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

В Python есть много встроенных **структур данных**, используемых для хранения разных типов информации. Словарь (dict) — одна из таких структур, которая хранит данные в формате пар ключ-значение. Получить доступ к значениям словаря Python можно с помощью ключей. Этот материал посвящен подробному обсуждению словаря.

Для создания словаря в Python необходимо передать последовательность элементов внутри фигурных скобок `{}`, разделив их запятыми (`,`). Каждый элемент имеет ключ и значение, выраженное парой «ключ: значение».

Значения могут быть представлять собой любые типы данных и повторяться, но ключи обязаны быть уникальными.

6. Что такое словарь включений?

Списковые включения в Python являются краткими синтаксическими конструкциями. Их можно использовать для создания списков из других списков, применяя функции к каждому элементу в списке.

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

Функция zip() в Python создает итератор, который объединяет элементы из нескольких источников данных. Эта функция работает со списками, кортежами, множествами и словарями для создания списков или кортежей, включающих все эти данные.

В Python есть несколько встроенных функций, которые позволяют перебирать данные. Одна из них — zip. Функция `zip()` в Python создает итератор, который объединяет элементы из нескольких источников данных.

У функции `zip()` множество сценариев применения. Например, она пригодится, если нужно создать набор словарей из двух массивов, каждый из которых содержит имя и номер сотрудника.

Предположим, что есть список имен и номером сотрудников, и их нужно объединить в массив кортежей. Для этого можно использовать функцию zip(). Вот пример программы, которая делает именно это:

```
employee_numbers = [2, 9, 18, 28]
employee_names = ["Дима", "Марина", "Андрей", "Никита"]

zipped_values = zip(employee_names, employee_numbers)
zipped_list = list(zipped_values)

print(zipped_list)
```

Функция zip возвращает следующее:

```
[('Дима', 2), ('Марина', 9), ('Андрей', 18), ('Никита', 28)]
```

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

Datetime — важный элемент любой программы, написанной на Python. Этот модуль позволяет управлять датами и временем, представляя их в таком виде, в котором пользователи смогут их понимать.

datetime включает различные компоненты. Так, он состоит из объектов следующих типов:

- date — хранит дату
- time — хранит время
- datetime — хранит дату и время