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 7

Выполнил: студент группы ИВТ-б-о-18-1 (2)
Криворучко Ярослав Евгеньевич
_____(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

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.

Ввод [32]:

```
"""
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.
"""


import sys
from datetime import date, datetime

if __name__ == '__main__':
    # Список пунктов назначения.
    routes = []

    # Организовать бесконечный цикл запроса команд.
    print("Список команд:\n")
    print("add - добавить название пункта назначения;")
    print("list - вывести список пунктов назначения;")
    print("select <пункт назначения> - запросить информацию о поездах, направляющихся в пункт;")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")
    while True:
    # Запросить команду из терминала.
        command = input("\ncomand->>> ").lower()
        # Выполнить действие в соответствие с командой.
        if command == 'exit':
            break
        elif command == 'add':
        # Запросить данные о маршруте.
            destination = input("\nНазвание пункта назначения: ")
            numTrain = input("Номер поезда: ")

            time = input("Время отправления (часы минуты): ")

            t = datetime.strptime(time,"%H %M")
            time = str(datetime.time(t))

            # Создать словарь.
            route = {
                'destination': destination,
                'numTrain': numTrain,
                'time': time,
            }
            # Добавить словарь в список.
            routes.append(route)
            # Отсортировать список в случае необходимости.
            if len(routes) > 1:
                routes.sort(key=lambda item: item.get('time', ''))
```

```
        elif command == 'list':
            # Заголовок таблицы.
            line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 30,
                '-' * 20,
                '-' * 20
            )
            print(line)
            print(
                '| {:^4} | {:^30} | {:^20} | {:^20} |'.format(
                    "№",
                    "Пунк назначения",
                    "Номер поезда",
                    "Время отправления"
                )
            )
            print(line)
            # Вывести данные о всех маршрутах.
            for idx, route in enumerate(routes, 1):
                print(
                    '| {:>4} | {:<30} | {:<20} | {:>20} |'.format(
                        idx,
                        route.get('destination', ''),
                        route.get('numTrain', ''),
                        route.get('time', '')
                    )
                )
            print(line)
        elif command.startswith('select '):


            parts = command.split(' ', maxsplit=1)
            # Получить требуемый стаж.
            station = str(parts[1])
            # Инициализировать счетчик.
            count = 0
            # Проверить сведения работников из списка.
            for route in routes:
                if route.get('destination') == station:
                    count += 1
                    print(
                        ' {:>4}. Номер поезда: {}, Время отправления: {}'.format(count, route.get('numTrain', ''),route.get('time', 0))
                    )
            # Если счетчик равен 0, то работники не найдены.
            if count == 0:
                print("Поездов с данным пунктом назначения не найдено.")
        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - добавить название пункта назначения;")
            print("list - вывести список пунктов назначения;")
            print("select <пункт назначения> - запросить информацию о поездах, направляющихся в пункт;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")
        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)
```

```
Список команд:

add - добавить название пункта назначения;
list - вывести список пунктов назначения;
select <пункт назначения> - запросить информацию о поездах, направляющихся в пункт;
help - отобразить справку;
exit - завершить работу с программой.

comand->>> add

Название пункта назначения: stav
Номер поезда: 1920
Время отправления (часы минуты): 15 00

comand->>> add

Название пункта назначения: stav
Номер поезда: 5594
Время отправления (часы минуты): 10 01

comand->>> add

Название пункта назначения: mos
Номер поезда: 3394
Время отправления (часы минуты): 13 15

comand->>> list
+------+--------------------------------+----------------------+----------------------+
|  №   |        Пунк назначения         |     Номер поезда     |  Время отправления   |
+------+--------------------------------+----------------------+----------------------+
|    1 | stav                           | 5594                 |             10:01:00 |
|    2 | mos                            | 3394                 |             13:15:00 |
|    3 | stav                           | 1920                 |             15:00:00 |
+------+--------------------------------+----------------------+----------------------+

comand->>> select stav
    1. Номер поезда: 5594, Время отправления: 10:01:00
    2. Номер поезда: 1920, Время отправления: 15:00:00

comand->>> help
Список команд:

add - добавить название пункта назначения;
list - вывести список пунктов назначения;
select <пункт назначения> - запросить информацию о поездах, направляющихся в пункт;
help - отобразить справку;
exit - завершить работу с программой.

comand->>> exit
```

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.

Файл 2.6.ipynb с решением задач находится на Github: https://github.com/YaroStavr/LR4_Artificial-Intelligence.git

Ответ 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