МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Основы кроссплатформенного программирования

Отчет по лабораторной работе №7

Работа со словарями в языке Python

Выполнил студент группы ИТС-б-з-22-1

Дубинин Олег Александрович

« »_____2023г.

Подпись студента_____

Работа защищена « »_____2023г.

Проверил доцент, кандидат технических наук, доцент кафедры инфокоммуникаций

Воронкин Роман Александрович

_____

(подпись)

Ставрополь, 2023 г.

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

Ход работы:

https://github.com/OLEG1232155/7 - репозиторий

Пример 1. Использовать словарь, содержащий следующие ключи: фамилия и инициалы работника; название занимаемой должности; год поступления на работу. Написать программу, выполняющую следующие действия:

- *ввод с клавиатуры данных в список, состоящий из заданных словарей;*
- *записи должны быть размещены по алфавиту;*
- *вывод на дисплей фамилий работников, чей стаж работы в организации превышает значение, введенное с клавиатуры;*
- *если таких работников нет, вывести на дисплей соответствующее сообщение.*

```python
#!/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(
                    "No",
                    "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 1 – Окно программы примера

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.

individ.py

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

import sys

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

    # 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 человеке.
            point = input("Пункт назначения: ")
            number = input("Номер поезда: ")
            time = input("Время отправления: ")

            # Создать словарь.
            train = {
                'point': point,
                'number': number,
                'time': time,
            }

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

        elif command == 'list':
            # Заголовок таблицы.
            line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 6,
                '-' * 20,
                '-' * 30,
                '-' * 20
            )
            print(line)
            print(
                '| {:^6} | {:^20} | {:^30} | {:^20} |'.format(
                    "№",
                    "Пункт назначения",
                    "Номер поезда",
                    "Время отправления"
                )
            )
```

```
            print(line)

            # Вывести данные о всех людях.
            for idx, train in enumerate(trains, 1):
                print(
                    '| {:>6} | {:<20} | {:<30} | {:<20} |'.format(
                        idx,
                        train.get('point', ''),
                        train.get('number', ''),
                        train.get('time', 0)
                    )
                )

            print(line)

        elif command.startswith('select '):
            # Разбить команду на части для выделения Нормера поезда
            parts = command.split(' ', maxsplit=1)
            # Получить требуемый номер поезда
            number_train = parts[1]
            search_train = []

            # Проверить сведения о поездах
            for train in trains:
                if train["number"] == number_train:
                    search_train.append(train)

            if len(search_train) > 0:
                line_new = '+-{}-+-{}-+-{}-+-{}-+'.format(
                    '-' * 6,
                    '-' * 20,
                    '-' * 30,
                    '-' * 20
                )
                print(line_new)

                print(
                    '| {:^6} | {:^20} | {:^30} | {:^20} | '.format(
                        "№",
                        "Пункт назначения",
                        "Номер поезда",
                        "Время отправления"
                    )
                )
                print(line_new)

                for idx_new, spisok_new in enumerate(search_train, 1):
                    print(
                        '| {:>6} | {:<20} | {:<30} | {:<20} | '.format(
                            idx_new,
                            spisok_new.get('point', ''),
                            spisok_new.get('number', ''),
                            spisok_new.get('time', '')
                        )
                    )
```

```
111
112                 print(line_new)
113
114             else:
115                 print("Поезда с заданным номером не найдены.")
116
117         elif command == 'help':
118             # Вывести справку о работе с программой.
119             print("Список команд:\n")
120             print("add - добавить новый поезд;")
121             print("list - вывести список поездов;")
122             print("select <номер поезда> - запросить данные о поезде;")
123             print("help - отобразить справку;")
124             print("exit - завершить работу с программой.")
125         else:
126             print(f"Неизвестная команда {command}", file=sys.stderr)
127
```

```
C:\Users\n1có\AppData\Local\Programs\Python\Python39\python.exe "C:\Users\n1có\Desktop\Олег(кросс)\pycha
>>> add
Пункт назначения: Stavropol
Номер поезда: 2134
Время отправления: 20:20
>>> Moscow
>>> Неизвестная команда moscow
add
Пункт назначения: Moscow
Номер поезда: 1003
Время отправления: 3:50
>>> Novosibirsk
>>> Неизвестная команда novosibirsk
add
Пункт назначения: Novosibirsk
Номер поезда: 3454
Время отправления: 4:00
>>> list
+--------+----------------------+------------------------------+----------------------+
|   №    |   Пункт назначения   |         Номер поезда         |  Время отправления   |
+--------+----------------------+------------------------------+----------------------+
|      1 | Moscow               | 1003                         | 3:50                 |
|      2 | Stavropol            | 2134                         | 20:20                |
|      3 | Novosibirsk          | 3454                         | 4:00                 |
+--------+----------------------+------------------------------+----------------------+
>>> select 1003
+--------+----------------------+------------------------------+----------------------+
|   №    |   Пункт назначения   |         Номер поезда         |  Время отправления   |
+--------+----------------------+------------------------------+----------------------+
|      1 | Moscow               | 1003                         | 3:50                 |
+--------+----------------------+------------------------------+----------------------+
>>> select 1111
Поезда с заданным номером не найдены.
>>> help
Список команд:

add - добавить новый поезд;
list - вывести список поездов;
select <номер поезда> - запросить данные о поезде;
help - отобразить справку;
exit - завершить работу с программой.
>>> exit

Process finished with exit code 0
```

Рисунок 2 – Окно программы индивидуального задания

Ответы на 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.

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

Да может! Функция len() возвращает длину (количество элементов) в объекте.

3. Какие методы обхода словарей Вам известны?

У словаря как класса есть метод *items()*, который создает особую структуру, состоящую из кортежей. Каждый кортеж включает ключ и значение:

```
>>> n = nums.items()
>>> n
dict_items([(1, 'one'), (2, 'two'), (3, 'three')])
```

Методы словаря *keys()* и *values()* позволяют получить отдельно перечни ключей и значений. Так что если, например, надо перебрать только значения или только ключи, лучше воспользоваться одним из этих методов:

```
>>> v_nums = []
>>> for v in nums.values():
...     v_nums.append(v)
...
>>> v_nums
['one', 'two', 'three']
```

Так же существуют методы *clear()*, *copy()*, *fromkeys()*, *get()*, *pop()*, *popitem()*, *setdefault()*, *update()*.

Метод *clear()* удаляет все элементы словаря, но не удаляет сам словарь. В итоге остается пустой Словарь. Метод *fromkeys()* позволяет создать словарь из списка, элементы которого становятся ключами. Применять метод можно как классу *dict*, так и к его объектам. Метод *get()* позволяет получить элемент по его ключу. Метод *pop()* удаляет из словаря элемент по указанному ключу и возвращает значение удаленной пары. Метод *popitem()* не принимает аргументов, удаляет и возвращает произвольный элемент. С помощью setdefault() можно добавить элемент в словарь. С помощью *update()* можно добавить в словарь

другой словарь.

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

Операция dict[key] вернет элемент словаря dict с ключом key. Операция вызывает исключение KeyError, если ключ key отсутствует в словаре.

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

Операция d[key] = value добавит в словарь dict новый элемент - пару ключ-значение.

Если в словаре существует ключ key то эта операция присвоит ключу key новое значение value.

6. Что такое словарь включений?

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.

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

Функция zip() создает итератор кортежей, который объединяет элементы каждой из переданных последовательностей *iterables.

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:

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

Вывод: приобрел навыки по работе со словарями при написании программ с помощью языка программирования Python версии 3.x.