МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №9 по дисциплине: «Основы программной инженерии»

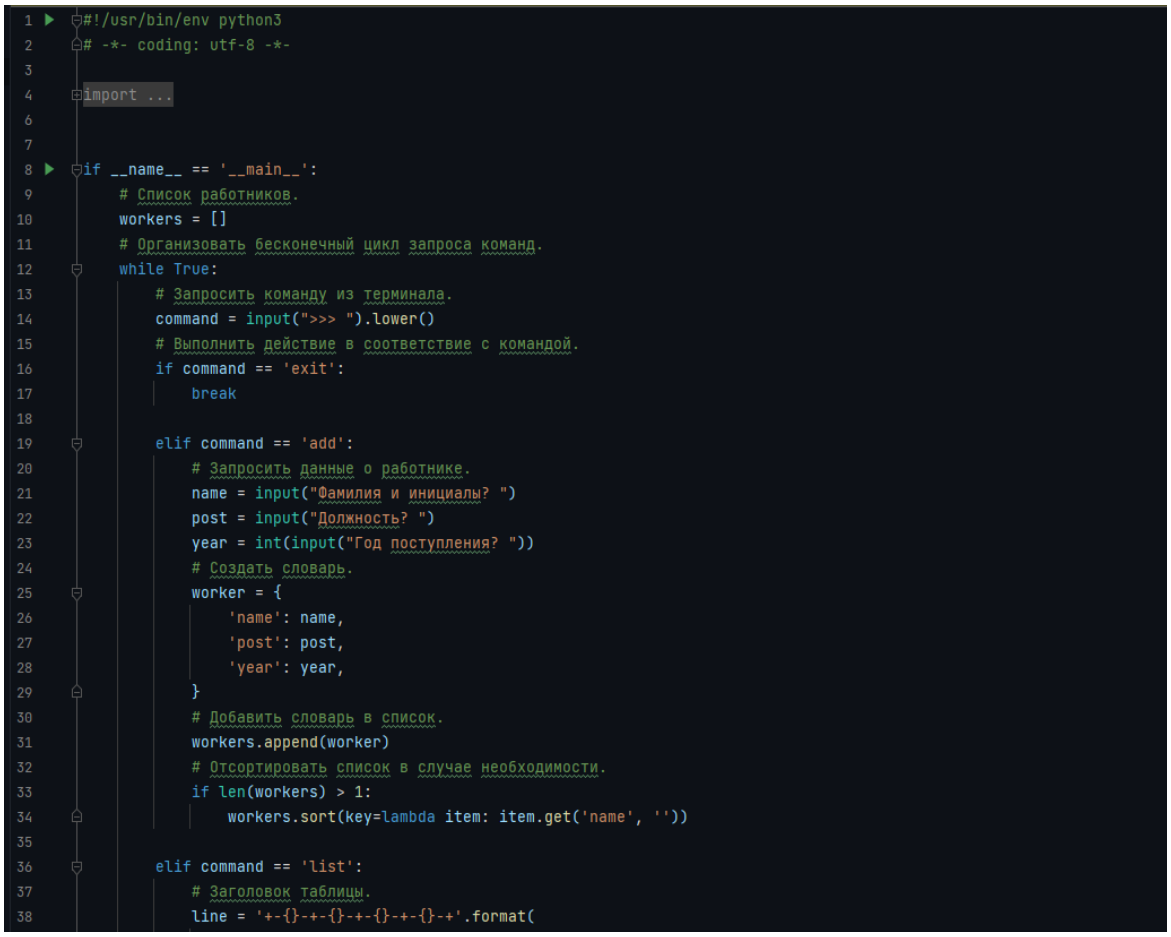
Выполнил:
Гребе Владислав
Александрович,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры
прикладной математики и
компьютерной безопасности,
Воронкин Р.А.

Отчет защищен с оценкой_____Дата защиты_____

Ставрополь, 2021 г.

ВЫПОЛНЕНИЕ:



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

import ...


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(
```

Рисунок 9.1 – Код первого примера.

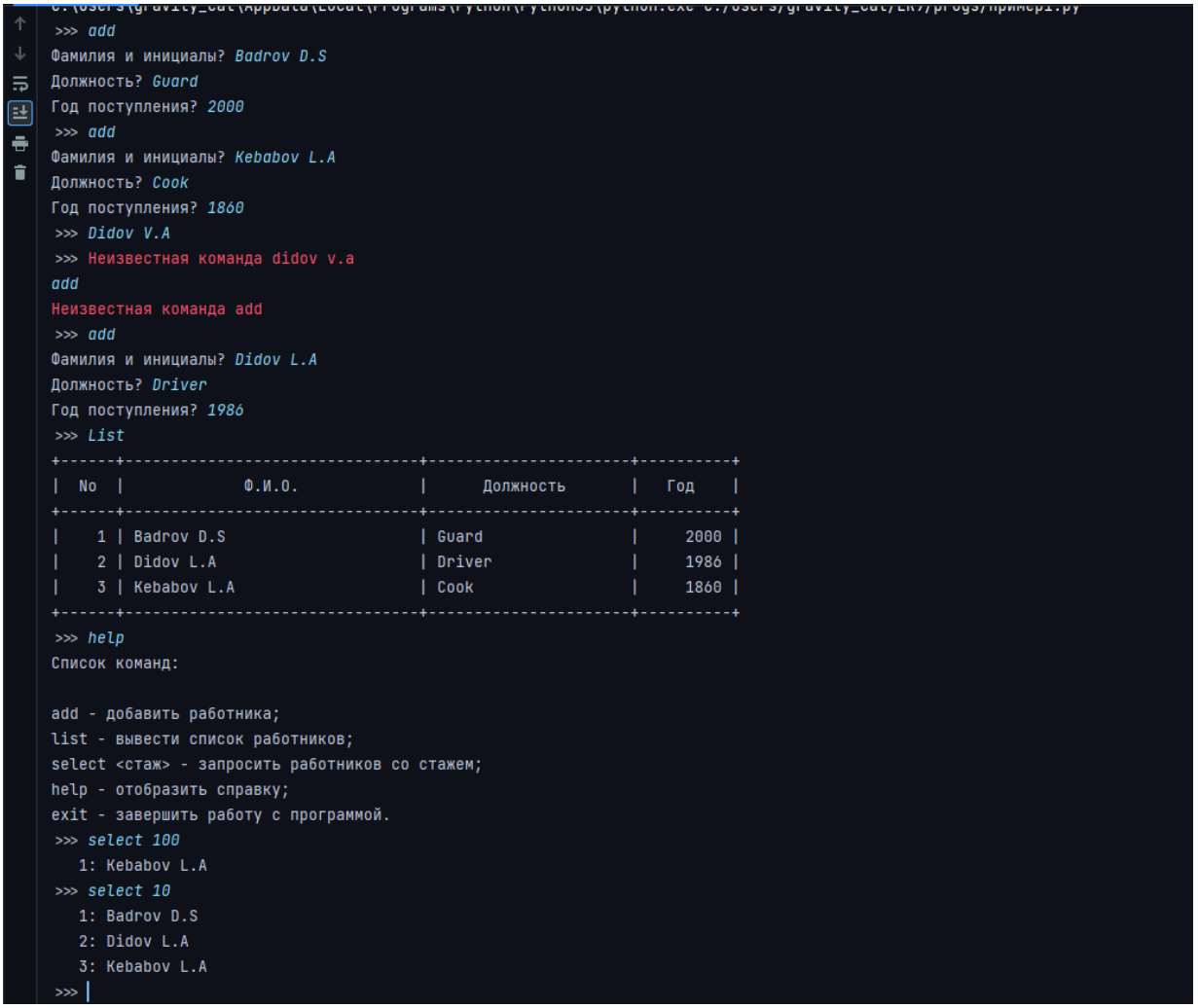
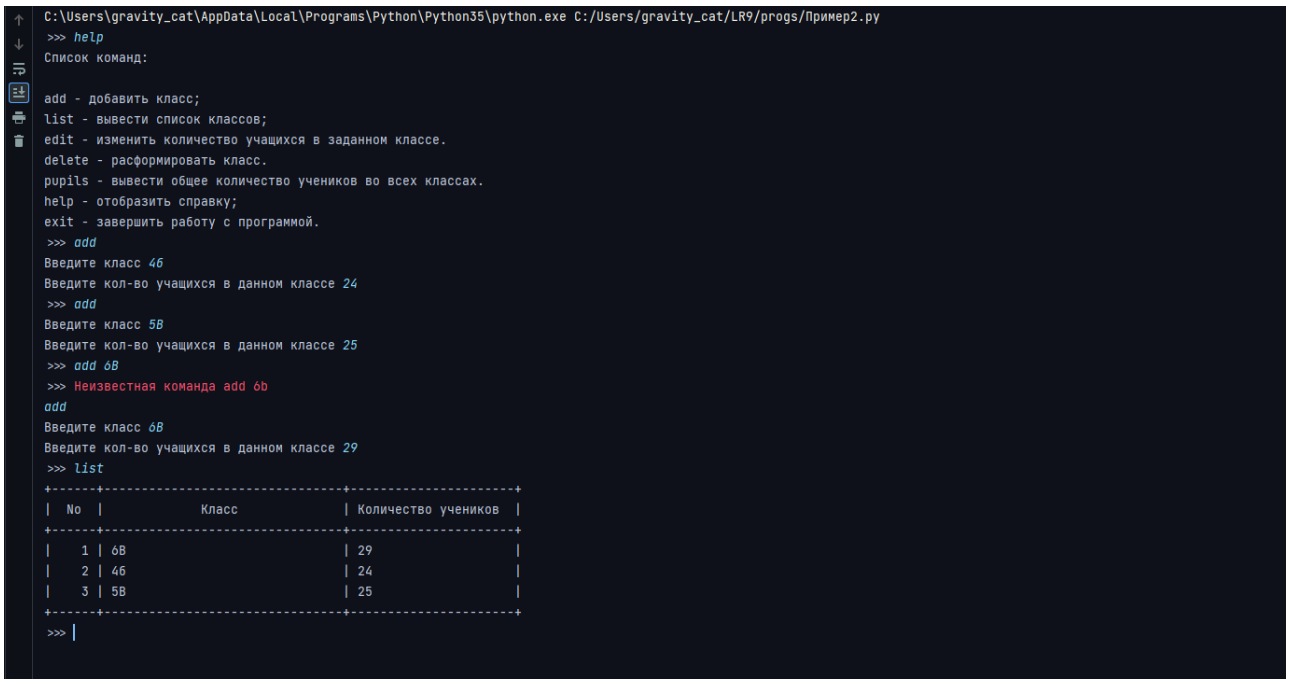


Рисунок 9.2 – Результат выполнения первого примера

```
            )
        print(line)

    elif command == 'edit':
        class_name = input("Введите класс, в котором нужно внести "
                           "изменения ")
        pupils = input("Введите новое количество учащихся в классе "
                       +class_name )
        school[class_name] = pupils
        print("Количество учащихся было успешно изменено!")

    elif command == 'delete':
        class_name = input("Введите класс, который нужно расформировать ")
        del school[class_name]
        print("Класс был успешно расформирован")

    elif command == 'pupils':
        sum_of_pupils = 0
        for class_name in school:
            sum_of_pupils += int(school[class_name])
        print("Количество учеников в школе: " +str(sum_of_pupils))

    elif command == 'help':
        # Вывести справку о работе с программой.
        print("Список команд:\n")
        print("add - добавить класс;")
        print("list - вывести список классов;")
        print("edit - изменить количество учащихся в заданном классе.")
        print("delete - расформировать класс.")
        print("pupils - вывести общее количество учеников во всех "
              "классах.")
        print("help - отобразить справку;")
        print("exit - завершить работу с программой.")
    else:
        print("Неизвестная команда " +str(command), file=sys.stderr)
```

Рисунок 9.3 – Код второго примера



```
C:\Users\gravity_cat\AppData\Local\Programs\Python\Python35\python.exe C:/Users/gravity_cat/LR9/progs/Пример2.py
>>> help
Список команд:

add - добавить класс;
list - вывести список классов;
edit - изменить количество учащихся в заданном классе.
delete - расформировать класс.
pupils - вывести общее количество учеников во всех классах.
help - отобразить справку;
exit - завершить работу с программой.
>>> add
Введите класс 4б
Введите кол-во учащихся в данном классе 24
>>> add
Введите класс 5В
Введите кол-во учащихся в данном классе 25
>>> add 6В
>>> Неизвестная команда add 6b
add
Введите класс 6В
Введите кол-во учащихся в данном классе 29
>>> list
+------+--------------------------------+----------------------+
|  No  |             Класс              | Количество учеников  |
+------+--------------------------------+----------------------+
|    1 | 6В                             | 29                   |
|    2 | 4б                             | 24                   |
|    3 | 5В                             | 25                   |
+------+--------------------------------+----------------------+
>>>
```

Рисунок 9.4 - Результат выполнения второго примера

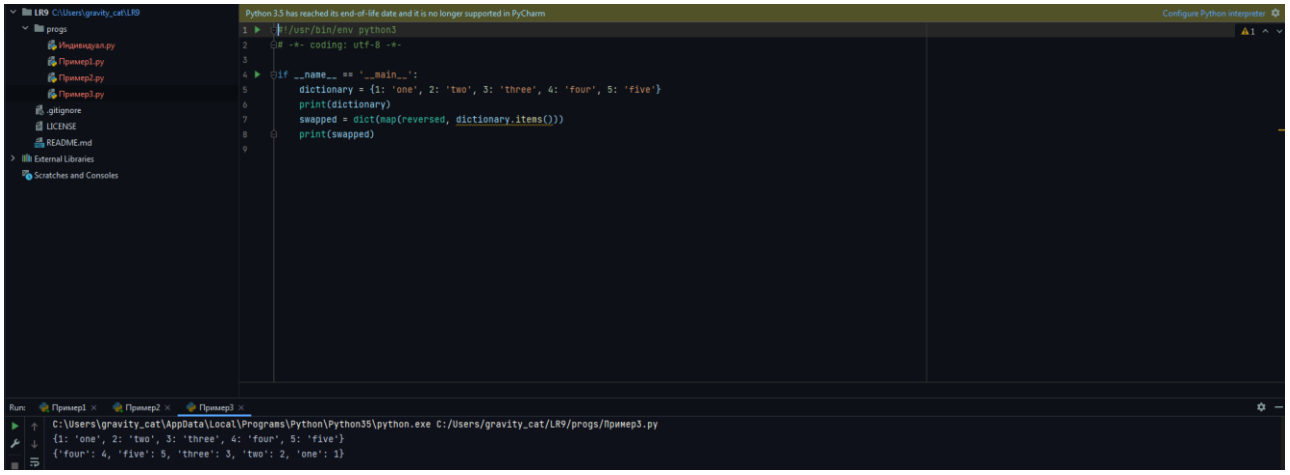


Рисунок 9.5 – Третий пример

ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

Вариант 5

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

import sys


if __name__ == '__main__':
    flights = []
    while True:
        command = input(">>> ").lower()
        if command == 'exit':
            break

        elif command == 'add':
            flight_destination = input("Введите название пункта назначения ")
            flight_number = input("Введите номер рейса ")
            airplane_type = input("Введите тип самолета ")
            flight = {
                'flight_destination': flight_destination,
                'flight_number': flight_number,
                'airplane_type': airplane_type,
            }
            flights.append(flight)
            if len(flights) > 1:
                flights.sort(
                    key=lambda item:
                    item.get('flight_destination', ''))

        elif command == 'list':
            line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 30,
                '-' * 20,
                '-' * 15
            )
            print(line)
            print(
```

Рисунок 9.6 – Код индивидуального занятия

```
C:\Users\gravity_cat\AppData\Local\Programs\Python\Python35\python.exe C:/Users/gravity_cat/LR9/progs/Пример3.py
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}
{'four': 4, 'five': 5, 'three': 3, 'two': 2, 'one': 1}
```

Рисунок 9.7 – Результат выполнения задания

ОТВЕТЫ НА ВОПРОСЫ

1. Словарь представляет собой структуру данных (которая ещё называется ассоциативный массив), предназначенную для хранения произвольных объектов с доступом по ключу.

2. Да, len() может быть использован – он выводит количество элементов (пар типа «ключ: элемент»).

3. Перебор ключей в цикле for, перебор элементов в цикле for, одновременный перебор ключей и их значений в цикле for.

4. С помощью метода get(), при обходе в цикле for, используя переменную в качестве счетчика ключей.

5. С помощью метода setdefault(), при непосредственном обращении к ключу словаря.

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

7. 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. 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. Функция zip() принимает итерируемый объект, например, список, кортеж, множество или словарь в качестве аргумента. Затем она генерирует список кортежей, которые содержат элементы из каждого объекта, переданного в функцию. Предположим, что есть список имен и номером сотрудников, и их нужно объединить в массив кортежей. Для этого можно использовать функцию zip().

8. Модуль datetime предоставляет классы для обработки времени и даты разными способами. Поддерживается и стандартный способ представления времени, однако больший упор сделан на простоту манипулирования датой, временем и их частями. Классы, предоставляемые модулем datetime:

• Класс datetime.date(year, month, day) - стандартная дата. Атрибуты: year, month, day. Неизменяемый объект.

• Класс datetime.time(hour=0, minute=0, second=0, microsecond=0, tzinfo=None) - стандартное время, не зависит от даты. Атрибуты: hour, minute, second, microsecond, tzinfo.

• Класс datetime.timedelta - разница между двумя моментами времени, с точностью до микросекунд.

• Класс datetime.tzinfo - абстрактный базовый класс для информации о временной зоне (например, для учета часового пояса и / или летнего времени).

• Класс datetime.datetime(year, month, day, hour=0, minute=0, second=0, microsecond=0, tzinfo=None) - комбинация даты и времени. Обязательные

аргументы:

• datetime.MINYEAR (1) ≤ year ≤ datetime.MAXYEAR (9999)

• 1 ≤ month ≤ 12

• 1 ≤ day ≤ количество дней в данном месяце и году Необязательные:

• 0 ≤ minute < 60

• 0 ≤ second < 60

• 0 ≤ microsecond < 1000000 Методы класса datetime:

• datetime.today() - объект datetime из текущей даты и времени. Работает также, как и datetime.now() со значением tz=None.

• datetime.fromtimestamp(timestamp) - дата из стандартного представления времени.

• datetime.fromordinal(ordinal) - дата из числа, представляющего собой количество дней, прошедших с 01.01.1970.

• datetime.now(tz=None) - объект datetime из текущей даты и времени.

• datetime.combine(date, time) - объект datetime из комбинации объектов date и time.

• datetime.strptime(date_string, format) - преобразует строку в datetime (так же, как и функция strptime из модуля time).

• datetime.strftime(format) - см. функцию strftime из модуля time.

• datetime.date() - объект даты (с отсечением времени).

• datetime.time() - объект времени (с отсечением даты).

• datetime.replace([year[, month[, day[, hour[, minute[, second[, microsecond[, tzinfo]]]]]]]]) - возвращает новый объект datetime с изменёнными атрибутами.

• datetime.timetuple() - возвращает struct_time из datetime.

• datetime.toordinal() - количество дней, прошедших с 01.01.1970.

• datetime.timestamp() - возвращает время в секундах с начала эпохи.

• datetime.weekday() - день недели в виде числа, понедельник - 0, воскресенье - 6.

• datetime.isoweekday() - день недели в виде числа, понедельник - 1, воскресенье - 7.

• datetime.isocalendar() - кортеж (год в формате ISO, ISO номер недели, ISO день недели).

• d a t e t i m e . i s o f o r m a t (s e p = ' T ') - к р а с и в а я с т р о к а в и д а " Y Y Y YMMDDTHH:MM:SS.mmmmmm" или, если microsecond == 0, "YYYY-MM-DDTHH:MM:SS"

• datetime.ctime() - преобразует время, выраженное в секундах с начала эпохи в строку вида "Thu Sep 27 16:42:37 2012".