

Министерство науки и высшего образования Российской  
Федерации Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра инфо коммуникаций

**ОТЧЕТ**

**ПОЛАБОРАТОРНОЙ РАБОТЫ №2.4**

**Дисциплины «Основы кроссплатформенного  
программирования»**

Выполнил:  
Волошин Алексей Вадимович  
1 курс, группа ИТС-б-о-22-1,  
11.03.02 «Инфокоммуникационные  
технологии и системы связи»,  
направленность (профиль)  
«Инфокоммуникационные системы и  
сети», очная форма обучения

\_\_\_\_\_  
(подпись)

Руководитель практики: Воронкин Р. А.  
, канд. техн. наук, доцент кафедры инфо  
коммуникаций

\_\_\_\_\_  
(подпись)

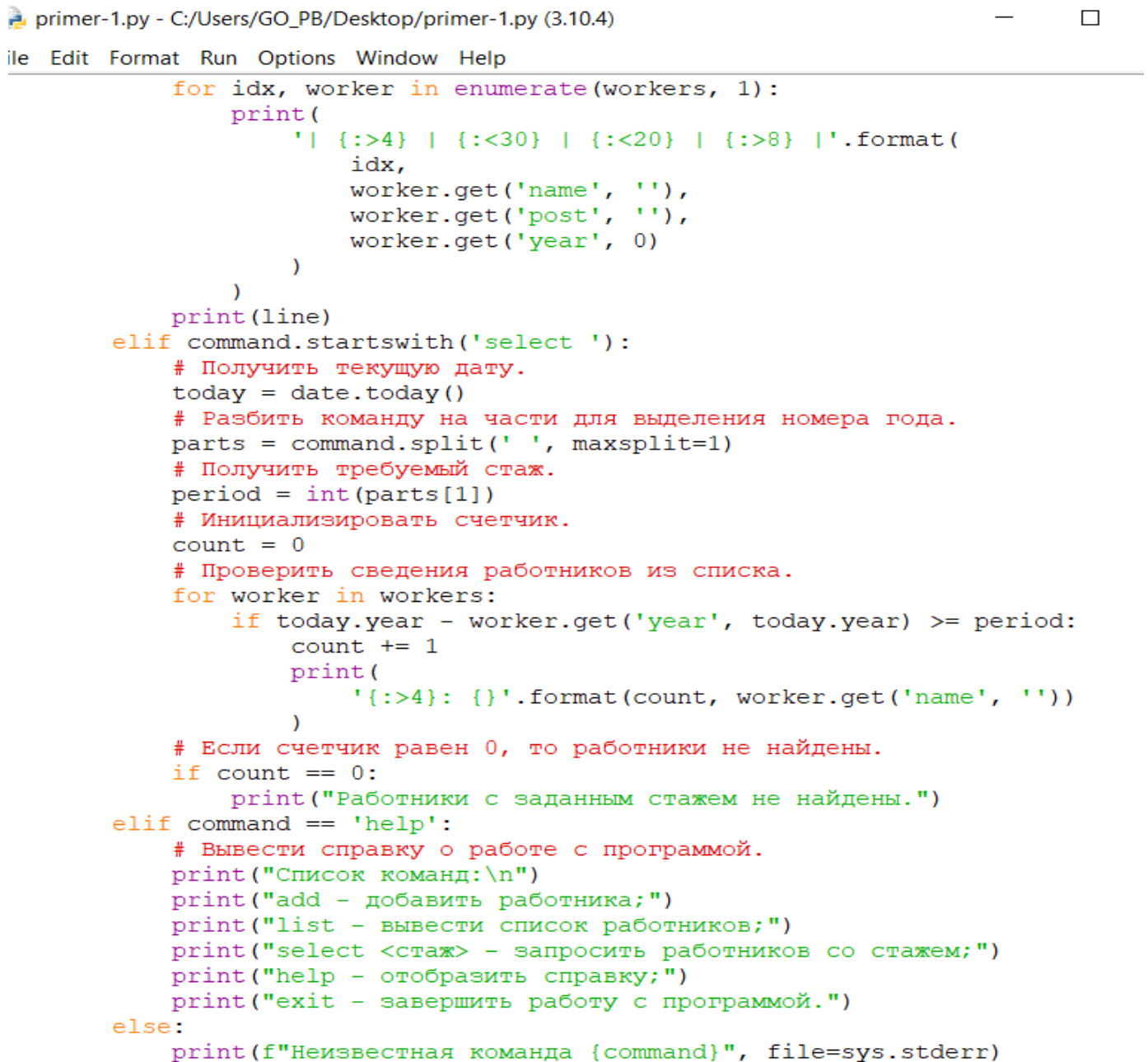
Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

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

**Ход работы:**

**Пример**

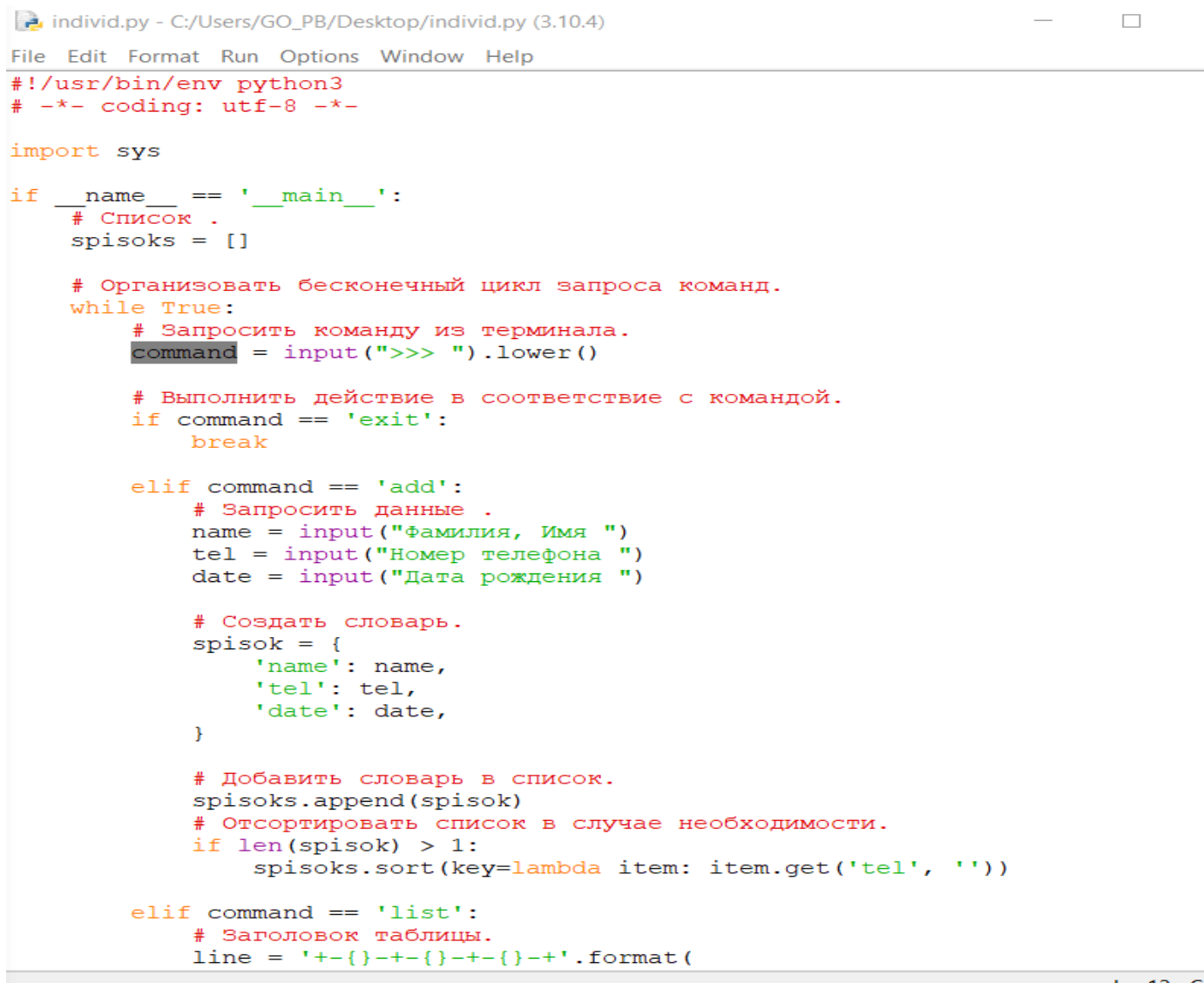


```
primer-1.py - C:/Users/GO_PB/Desktop/primer-1.py (3.10.4)
File Edit Format Run Options Window Help
for idx, worker in enumerate(workers, 1):
    print(
        '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
            idx,
            worker.get('name', ''),
            worker.get('post', ''),
            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. Работа программы «Пример»

**Задание**

Использовать словарь, содержащий следующие ключи: фамилия, имя; номер телефона; дата рождения (список из трех чисел). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по датам рождения; вывод на экран информации о человеке, номер телефона которого введен с клавиатуры; если такого нет, выдать на дисплей соответствующее сообщение

The image shows a screenshot of a text editor window titled 'individ.py - C:/Users/GO\_PB/Desktop/individ.py (3.10.4)'. The editor has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code is written in Python 3 and uses UTF-8 encoding. It defines a list 'spisoks' and enters a 'while True' loop to process commands. The commands include 'exit' (breaks the loop), 'add' (adds a new record with name, phone number, and date of birth), and 'list' (prints a table header). The 'add' command uses 'input' to get user data and creates a dictionary 'spisok' to be appended to 'spisoks'. The 'list' command starts with a table header line.

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

import sys

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

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

        # Выполнить действие в соответствии с командой.
        if command == 'exit':
            break

        elif command == 'add':
            # Запросить данные .
            name = input("Фамилия, Имя ")
            tel = input("Номер телефона ")
            date = input("Дата рождения ")

            # Создать словарь.
            spisok = {
                'name': name,
                'tel': tel,
                'date': date,
            }

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

        elif command == 'list':
            # Заголовок таблицы.
            line = '+-{}-+-{}-+-{}-+'.format(
```

Рисунок 2. Работа программы «Индивидуальное задание»

**Вывод:** Я приобрёл навыки по работе со словарями при написании программ с помощью языка программирования Python.

### Контрольные вопросы:

1. Что такое словари в языке Python?

Ответ: 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?

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

3. Какие методы обхода словарей Вам известны?

Ответ: 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 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?

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

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

Ответ: 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`.

2. Что такое словарь включений?

Ответ: 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.

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

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

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

Ответ: `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



