

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

**Отчет по лабораторной работе №1**

**Работа со словарями в языке Python**

**По дисциплине «Теории программирования и алгоритмизации»**

Выполнил студент группы ИВТ-б-о-20-1

Бобров Н. В. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р. А. \_\_\_\_\_

(подпись)

Ставрополь 2021

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

Ход работы:

1. Проработал пример из методических рекомендаций.

```
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.
```

Рисунок 1 – Код из примера

```
>>> add  
Фамилия и инициалы? Бобров Н.В.  
Должность? Студент  
Год поступления? 2020  
>>> select  
>>> Известная команда select  
select 1  
1: Бобров Н.В.  
>>>
```

Рисунок 2 – Результат выполнения кода

2. Выполнил первое задание.

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

if __name__ == '__main__':
    # A)
    School = {}
    for i in range(5):
        School.update(
            {input(f'Название класса № {i + 1} : '): int(input(f'Количество учеников класса № {i + 1} : '))})
    print(School)
    # B)
    ClassName = input('Введите класс: ')
    if ClassName in School:
        print(f'Количество учеников в классе {ClassName} : {School[ClassName]} ')
    else:
        print('Такого класса не существует')
    # C)
    for i in range(2):
        School.update({input(f'Название изменяемого класса {i + 1} : '): int(
            input(f'Количество учеников изменяемого класса {i + 1} : '))})
    print(School)
    # D)
    for i in range(1):
        School.update(
            {input(f'Название нового класса {i + 1} : '): int(input(f'Количество учеников нового класса {i + 1} : '))})
    print(School)
```

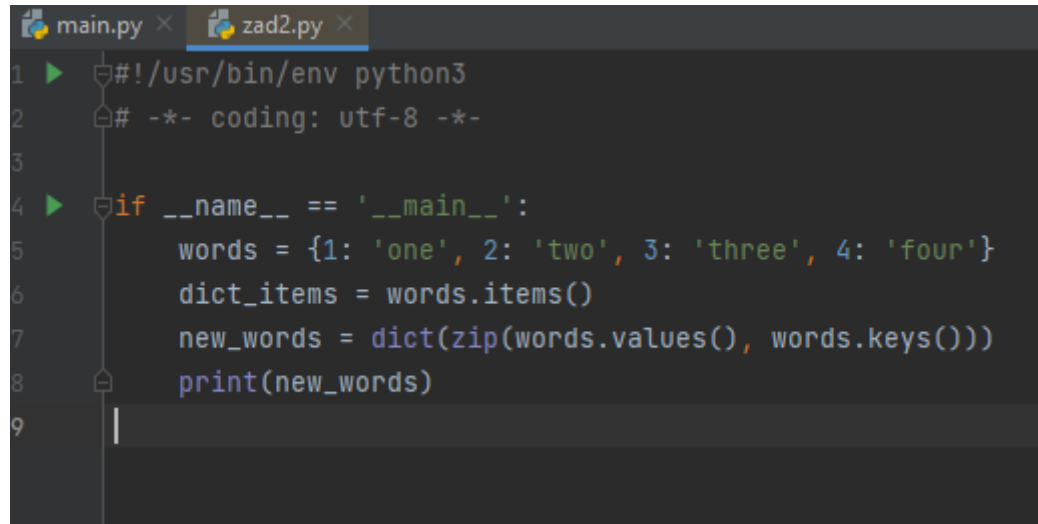
Рисунок 3 – Код для первого задания

```
zad1 x
Название класса № 1 : 1a
Количество учеников класса № 1 : 15
Название класса № 2 : 2г
Количество учеников класса № 2 : 24
Название класса № 3 : 7в
Количество учеников класса № 3 : 12
Название класса № 4 : 5д
Количество учеников класса № 4 : 23
Название класса № 5 : 6а
Количество учеников класса № 5 : 12
{'1a': 15, '2г': 24, '7в': 12, '5д': 23, '6а': 12}
Введите класс: 2г
Количество учеников в классе 2г : 24
Название изменяемого класса 1 : 7в
Количество учеников изменяемого класса 1 : 10
Название изменяемого класса 2 : 6а
Количество учеников изменяемого класса 2 : 14
{'1a': 15, '2г': 24, '7в': 10, '5д': 23, '6а': 14}
Название нового класса 1 : 11а
Количество учеников нового класса 1 : 13
{'1a': 15, '2г': 24, '7в': 10, '5д': 23, '6а': 14, '11а': 13}
Название расформировываемого класса: 5д
{'1a': 15, '2г': 24, '7в': 10, '6а': 14, '11а': 13}

Process finished with exit code 0
```

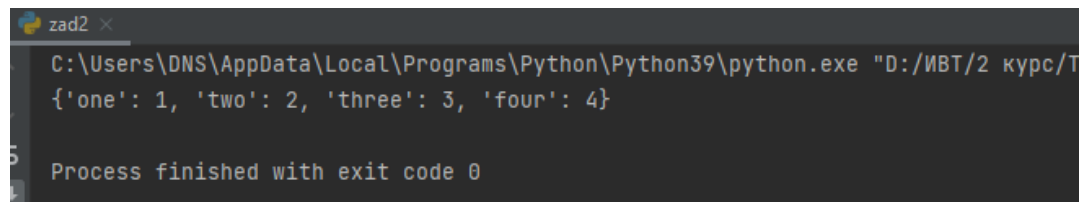
Рисунок 4 – Результат выполнения кода первого задания

### 3. Выполнил второе задание.



```
main.py x zad2.py x
1  > #!/usr/bin/env python3
2  > #- coding: utf-8 -*-
3
4  > if __name__ == '__main__':
5      words = {1: 'one', 2: 'two', 3: 'three', 4: 'four'}
6      dict_items = words.items()
7      new_words = dict(zip(words.values(), words.keys()))
8      print(new_words)
9
```

Рисунок 5 – Код для второго задания



```
zad2 x
C:\Users\DNS\AppData\Local\Programs\Python\Python39\python.exe "D:/ИВТ/2 курс/Т
{'one': 1, 'two': 2, 'three': 3, 'four': 4}
Process finished with exit code 0
```

Рисунок 6 – Результат выполнения кода

### Индивидуальное задание. Вариант 1.

#### Условие:

Использовать словарь, содержащий следующие ключи: фамилия и инициалы; номер группы; успеваемость (список из пяти элементов). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по возрастанию номера группы; вывод на дисплей фамилий и номеров групп для всех студентов, включенных в массив, если средний балл студента больше 4.0; если таких студентов нет, вывести соответствующее сообщение.

1. По аналогу примера составил код для выполнения задания.

```
main.py x
21 # Выполнить действие в соответствии с командой.
22 if command == 'exit':
23     break
24
25 elif command == 'add':
26     # Запросить данные о студенте.
27     name = input("Фамилия и инициалы? ")
28     group = input("Номер группы? ")
29     grade = str(input('Успеваемость: '))
30     # Создать словарь.
31     student = {
32         'name': name,
33         'group': group,
34         'grade': grade,
35     }
36     # Добавить словарь в список.
37     students.append(student)
38     # Отсортировать список в случае необходимости.
39     if len(students) > 1:
40         students.sort(key=lambda item: item.get('group')[:-1])
41
```

Рисунок 7 – Код для задания, часть 1

```
main.py x
65         idx,
66         student.get('name', ''),
67         student.get('group', ''),
68         student.get('grade', 0)
69     )
70 )
71 print(line)
72 elif command.startswith('select '):
73     # Инициализировать счетчик.
74     count = 0
75     # Проверить сведения студентов из списка.
76     for student in students:
77         grade = list(map(int, student.get('grade', '').split()))
78         if sum(grade) / max(len(grade), 1) >= 4.0:
79             print(
80                 '{:>4} {}'.format('*', student.get('name', '')),
81                 '{:>1} {}'.format('группа №', student.get('group', ''))
82             )
83             count += 1
84     if count == 0:
85         print("Студенты с баллом 4.0 и выше не найдены.")
86     else:
87         print(f"Неизвестная команда {command}", file=sys.stderr)
88
```

Рисунок 8 – Код для задания, часть 2

2. Сделал сортировку по номеру группы от меньшей к большей.

```

>>> add
Фамилия и инициалы? Бобров Н.В.
Номер группы? 5
Успеваемость: 5 5 4 4 5
>>> add
Фамилия и инициалы? Плотников Д.В.
Номер группы? 7
Успеваемость: 3 4 3 5 5
>>> add
Фамилия и инициалы? Иванов И.И.
Номер группы? 9
Успеваемость: 2 3 3 5 5
>>> add
Фамилия и инициалы? Пушкин А.С.
Номер группы? 1
Успеваемость: 5 5 5 2 2
>>> list
+-----+-----+-----+-----+
| № |          Ф.И.О.          |      Группа      |  Успеваемость  |
+-----+-----+-----+-----+
|  1 | Пушкин А.С.              | 1                | 5 5 5 2 2 |
|  2 | Бобров Н.В.              | 5                | 5 5 4 4 5 |
|  3 | Плотников Д.В.           | 7                | 3 4 3 5 5 |
|  4 | Иванов И.И.              | 9                | 2 3 3 5 5 |
+-----+-----+-----+-----+

```

Рисунок 9 – Результат сортировки

3. Сделал проверку условий «вывод на дисплей фамилий и номеров групп для всех студентов, включенных в массив, если средний балл студента больше 4.0».

```

+-----+-----+-----+-----+
| № |          Ф.И.О.          |      Группа      |  Успеваемость  |
+-----+-----+-----+-----+
|  1 | Пушкин А.С.              | 1                | 5 5 5 2 2 |
|  2 | Бобров Н.В.              | 5                | 5 5 4 4 5 |
|  3 | Плотников Д.В.           | 7                | 3 4 3 5 5 |
|  4 | Иванов И.И.              | 9                | 2 3 3 5 5 |
+-----+-----+-----+-----+

>>> select
>>> Неизвестная команда select
select 1
    * Бобров Н.В. группа № 5
    * Плотников Д.В. группа № 7

```

Рисунок 10 – Результат проверки условия

### Контрольные вопросы:

1. Что такое словари в языке Python?

Словарь – это структура данных, которая предназначена для хранения произвольных объектов с доступом по ключу.

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

Да, может.

3. Какие методы обхода словарей Вам известны?

- For I in dict
- for key in dict.keys()
- for value in dict.value()
- for key, value in dict.items()

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

Print(dict['Название ключа'])

Print(dict.get('Название ключа'))

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

Dict['Ключ']= значение

6. Что такое словарь исключений?

Исключение в Python – это конструкция, используемая для сигнализации о важном событии, обычно об ошибке, которое происходит при выполнении программы. Исключение может привести к остановке программы, если она не будет должным образом «поймана» (т.е. обработана правильно). Если вы думаете, что ваша программа может вызвать исключение при выполнении.

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

Функция zip объединяет в кортежи элементы из последовательностей переданных в качестве аргументов. Функция прекращает выполнение, как только достигнут конец самого короткого списка.

```
a = [1,2,3]
```

```
b = "xyz"
```

```
c = (None, True)
```

```
res = list(zip(a, b, c))
```

```
print (res)
```

```
[(1, 'x', None), (2, 'y', True)]
```

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

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.

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`

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.strptime(format)` - см. функцию `strptime` 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).

– `datetime.isoformat(sep='T')` - красивая строка вида "YYYY-MM-DDTHH:MM:SS.mmmmmm" или, если `microsecond == 0`, "YYYY-MM-DDTHH:MM:SS"

`datetime.ctime()` - см. `ctime()` из модуля `time`.

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