

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙ-
СКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Основы кроссплатформенного программирования

Отчет по лабораторной работе №2.6

Тема: «Работа со словарями в языке Python»

Выполнил студент группы

ИВТ-б-о-21-1

Богадунов В.И. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил доцент

Кафедры инфокоммуникаций, старший
преподаватель

Воронкин Р.А.

(подпись)

Ставрополь 2022

Ход работы:

1. Создал репозиторий в **GitHub**, дополнил правила в `.gitignore` для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на компьютер и организовал в соответствии с моделью ветвления `git-flow`.

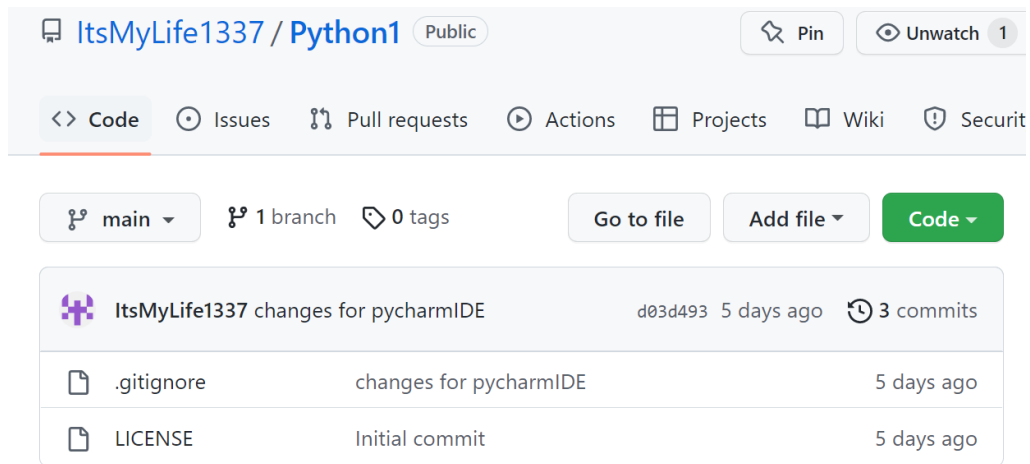


Рисунок 1.1 – Созданный репозиторий

```
.idea/  
# Created by https://www.toptal.com/developers/gitignore/api/python,pycharm  
# Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm  
  
### PyCharm ###  
# Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio, WebStorm and Rider  
# Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839  
  
# User-specific stuff  
.idea/**/workspace.xml  
.idea/**/tasks.xml  
.idea/**/usage.statistics.xml  
.idea/**/dictionaries  
.idea/**/shelf  
  
# AWS User-specific  
.idea/**/aws.xml  
  
# Generated files  
.idea/**/contentModel.xml  
  
# Sensitive or high-churn files  
.idea/**/dataSources/
```

Рисунок 1.2 – Дополнил правила в `.gitignore`

```

Microsoft Windows [Version 10.0.19044.1889]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Admin>cd /d c:\users\admin\desktop\git\python1

c:\Users\Admin\Desktop\git\Python1>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/Admin/Desktop/git/Python1/.git/hooks]

c:\Users\Admin\Desktop\git\Python1>

```

Рисунок 1.3 – Организация репозитория в соответствии с моделью ветвления git-flow

2. Создал проект Pycharm в папке репозитория, проработал примеры ЛР

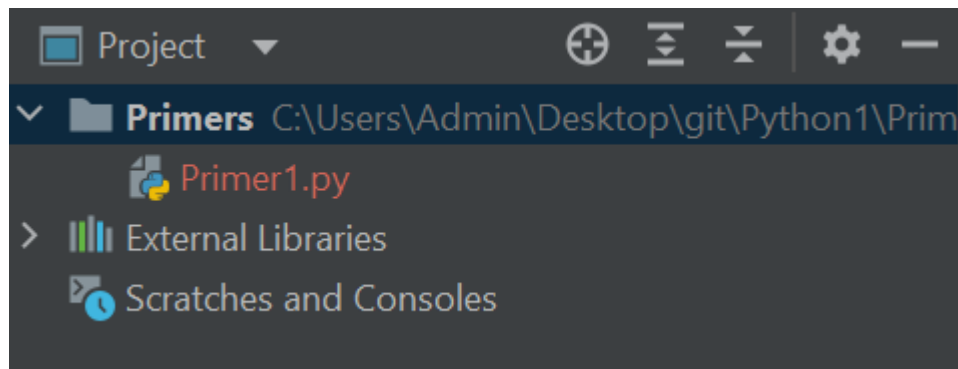


Рисунок 2.1 – Созданные проекты

```

Primer1 x
| 2 | Иванов ИИ | Сварщик | 2019 |
| 3 | Иващенко | Столяр | 2021 |
+-----+-----+-----+-----+
>>> select
>>> Неизвестная команда select
select 2
1: Богданенко БВ
2: Иванов ИИ
>>>

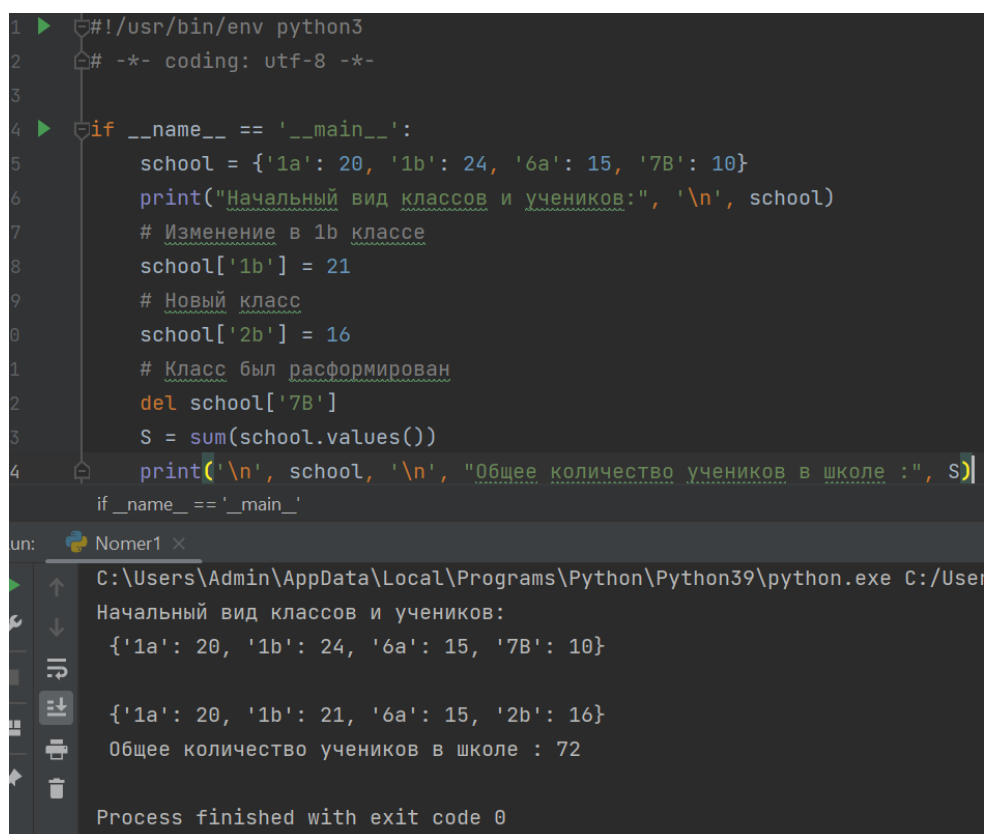
```

Рисунок 2.2 – Результат выполнения Примера №1

3. Выполнил 2 задачи:

№1. Решите задачу: создайте словарь, связав его с переменной school, и наполните данными, которые бы отражали количество учащихся в разных классах (1а, 1б, 2б, 6а, 7в и т. п.). Внесите изменения в словарь согласно следующему: а) в одном из классов изменилось количество учащихся, б) в школе появился новый класс, с) в школе был расформирован (удален) другой класс. Вычислите общее количество учащихся в школе.

№2. Решите задачу: создайте словарь, где ключами являются числа, а значениями – строки. Примените к нему метод items(), с помощью полученного объекта dict_items создайте новый словарь, "обратный" исходному, т. е. ключами являются строки, а значениями – числа.



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     school = {'1a': 20, '1b': 24, '6a': 15, '7B': 10}
6     print("Начальный вид классов и учеников:", '\n', school)
7     # Изменение в 1b классе
8     school['1b'] = 21
9     # Новый класс
10    school['2b'] = 16
11    # Класс был расформирован
12    del school['7B']
13    S = sum(school.values())
14    print('\n', school, '\n', "Общее количество учеников в школе :", S)
15
16 if __name__ == '__main__':
```

Output:

```
Начальный вид классов и учеников:
{'1a': 20, '1b': 24, '6a': 15, '7B': 10}

{'1a': 20, '1b': 21, '6a': 15, '2b': 16}
Общее количество учеников в школе : 72

Process finished with exit code 0
```

Рисунок 3.1 – Выполненная задача №1

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

if __name__ == '__main__':
    numb = {
        1: 'a',
        2: 'b',
        3: 'c'
    }
    print(numb)
    print({v: k for k, v in numb.items()})
```

Nomer2 x

C:\Users\Admin\AppData\Local\Programs\Python\Pyt

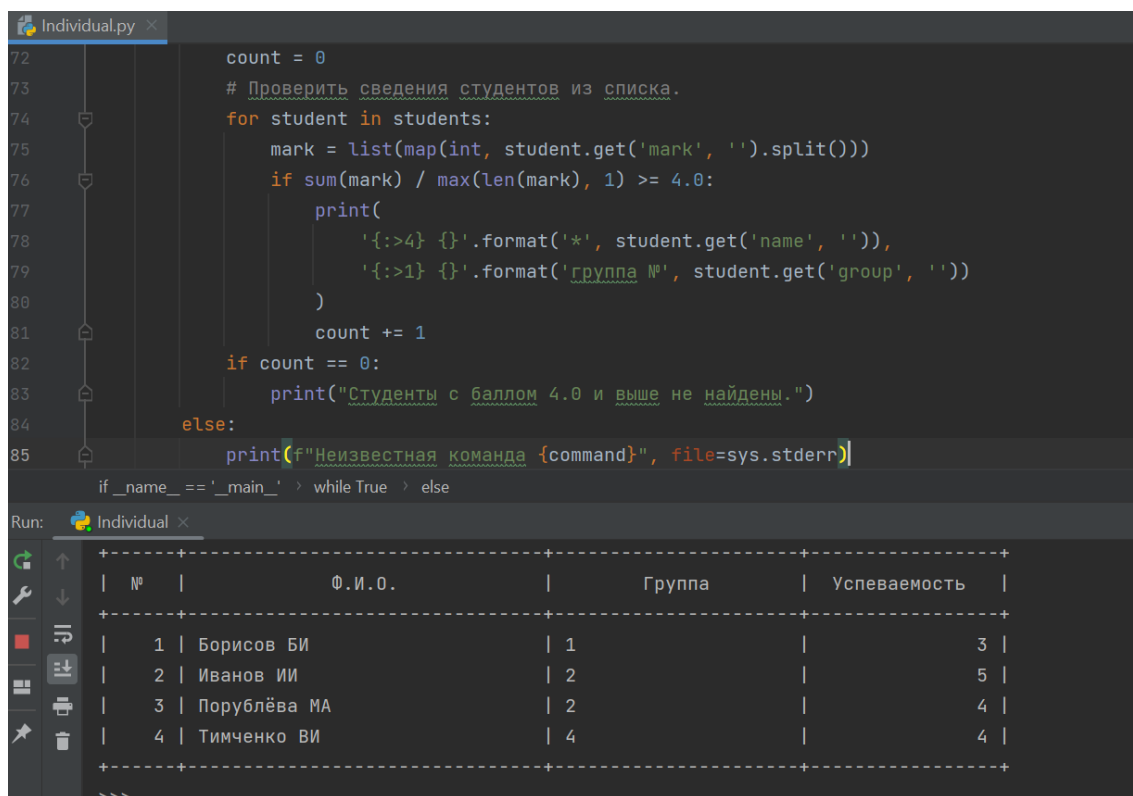
{1: 'a', 2: 'b', 3: 'c'}

{'a': 1, 'b': 2, 'c': 3}

Process finished with exit code 0

Рисунок 3.2 – Выполненная задача №2

№4. Индивидуальное задание. В – 1. 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.



```

72 count = 0
73 # Проверить сведения студентов из списка.
74 for student in students:
75     mark = list(map(int, student.get('mark', '').split()))
76     if sum(mark) / max(len(mark), 1) >= 4.0:
77         print(
78             '{:>4} {}'.format('*', student.get('name', '')),
79             '{:>1} {}'.format('группа №', student.get('group', ''))
80         )
81         count += 1
82     if count == 0:
83         print("Студенты с баллом 4.0 и выше не найдены.")
84 else:
85     print(f"Неизвестная команда {command}", file=sys.stderr)

if __name__ == '__main__': > while True > else

```

Run: Individual ×

№	Ф.И.О.	Группа	Успеваемость
1	Борисов БИ	1	3
2	Иванов ИИ	2	5
3	Порублёва МА	2	4
4	Тимченко ВИ	4	4

Рисунок 4.1 – Индивидуальное задание

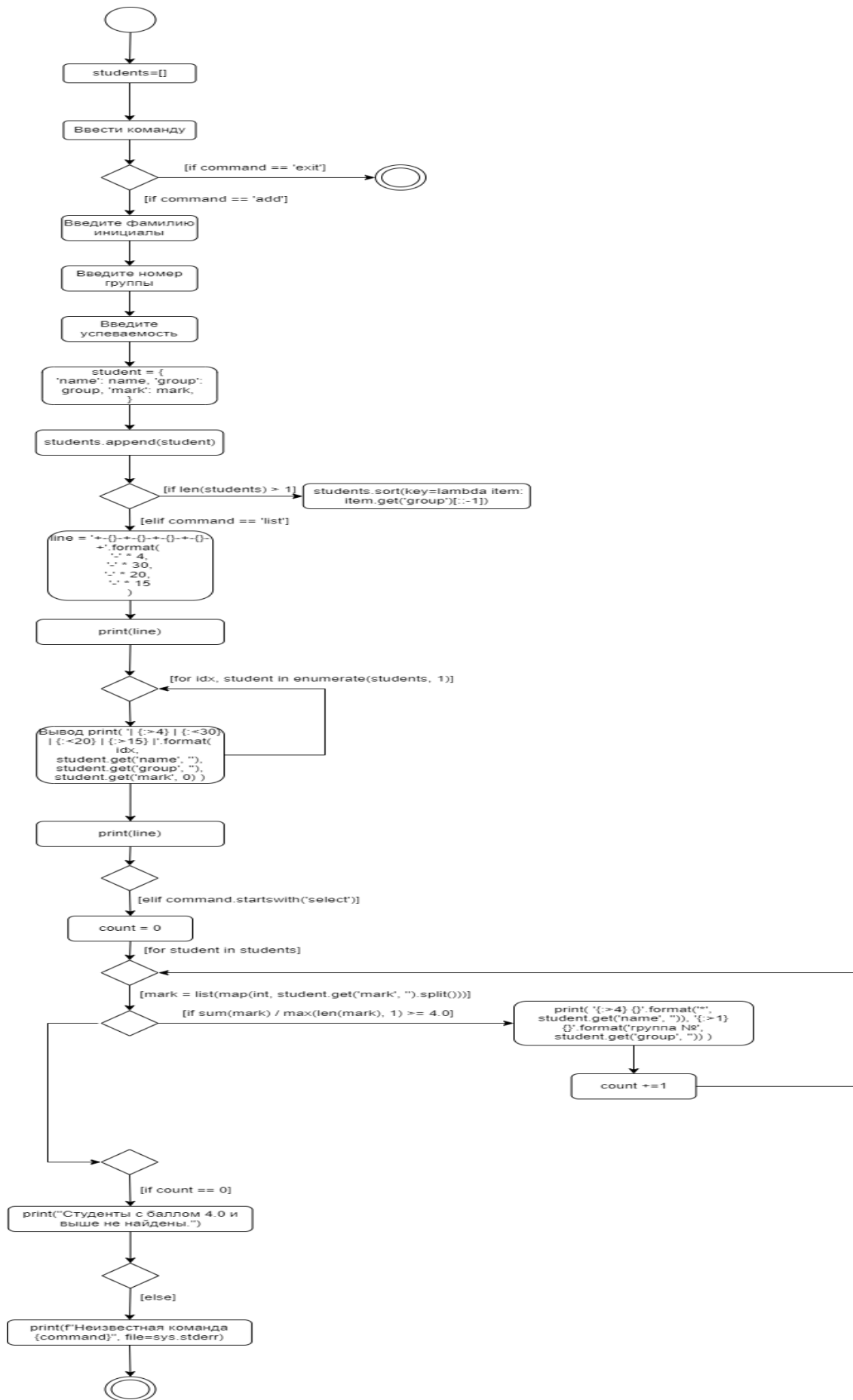


Рисунок 4.2 – UML диаграмма программы

Ответы на контрольные вопросы:

1. Что такое словари?

Словарь – структура данных (которая ещё называется ассоциативный массив), предназначенная для хранения произвольных объектов с доступом по ключу.

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

Да. Она возвращает размер словаря.

3. Какие методы обхода словарей Вам известны?

items(), keys() и values(), а также методы clear(), copy(), fromkeys(), get(), pop(), popitem(), setdefault(), update().

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

С помощью указания ключа в квадратных скобках: a["key"] или с помощью метода get().

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

Можно привычным способом присвоить значение элементу словаря по ключу: a["key"] = value. Есть метод setdefault(), который перезапишет старое значение элемента.

6. Что такое словарь включений?

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

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

Функция `zip()` берёт на вход несколько списков и создаёт из них список кортежей, такой, что первый элемент полученного списка содержит кортеж из первых элементов всех списков-аргументов, второй элемент - кортеж из вторых элементов и так далее.

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.

Вывод: в результате выполнения лабораторной работы были приобретены теоретические знания и практические навыки для работы со словарями при написании программ с помощью языка Python версии 3.x.