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

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе №2.6

Дисциплина: «Основы кроссплатформенного программирования»

Тема: «Работа со словарями в языке Python»

Выполнил: студент 2 курса

группы ИВТ-б-о-21-1

Хашиев Адам Мухарбекович

Ставрополь 2022

Выполнение работы:

1. Создал репозиторий в GitHub «rep 2.6» в который добавил .gitignore, который дополнил правила для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на лок. сервер и организовал в соответствии с моделью ветвления git-flow.

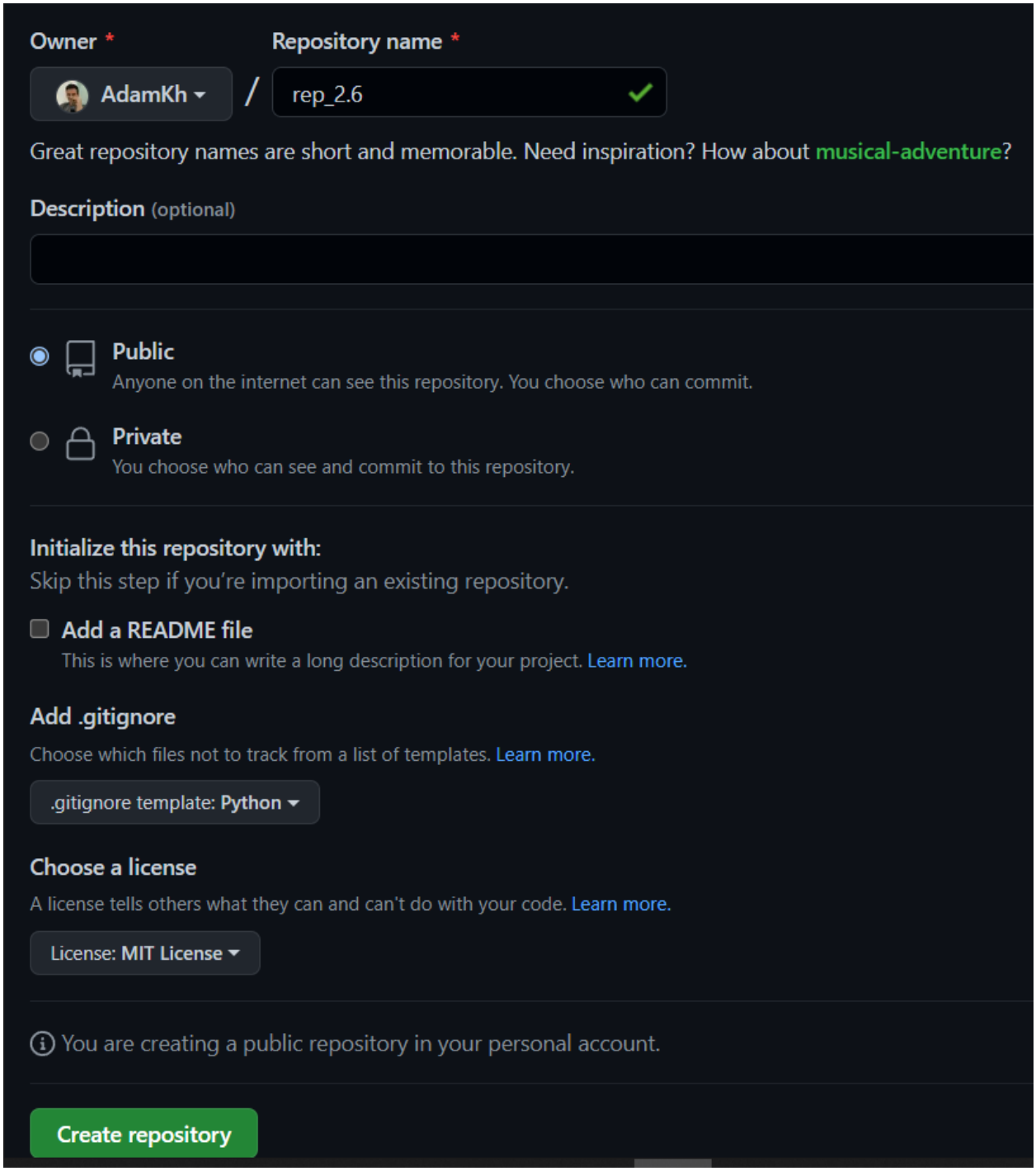

Рисунок 1.1 Создание репозитория

```
C:\>git clone https://github.com/AdamKh/rep_2.6.git
Cloning into 'rep_2.6'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

C:\>
```

Рисунок 1.2 Клонирование репозитория

```
C:\rep_2.6>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:/rep_2.6/.git/hooks]

C:\rep_2.6>
```

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

.gitignore – Блокнот

Файл Правка Формат Вид Справка

```
.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,
# 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
```

Рисунок 1.4 Изменение .gitignore

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

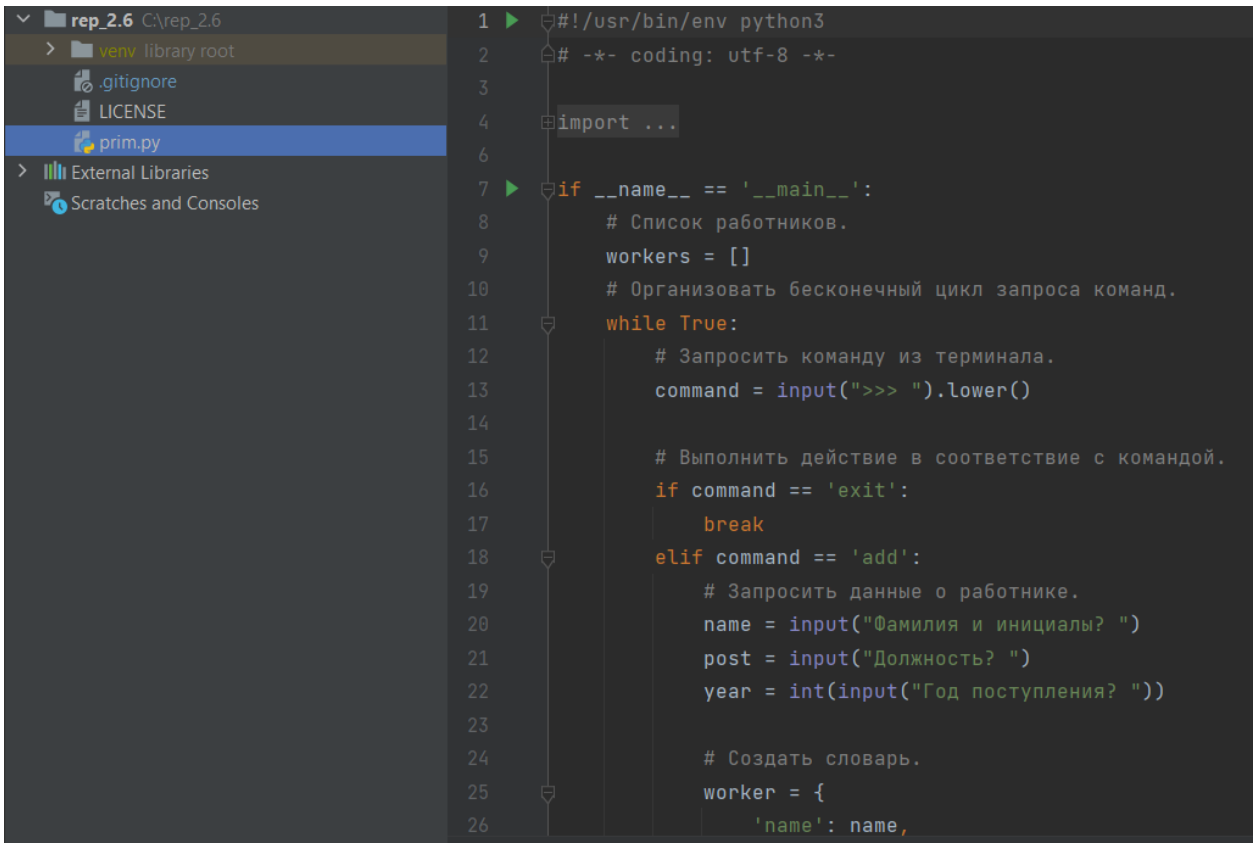

Рисунок 2.1 Создание проекта в PyCharm

```
C:\rep_2.6\venv\Scripts\python.exe C:/rep_2.6/prim.py
>>> help
Список команд:

add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку;
exit - завершить работу с программой.
>>> add
Фамилия и инициалы? Zolo I S
Должность? Owner
Год поступления? 2004
>>> list
+------+----------------------------------+----------------------+----------+
|  №   |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Zolo I S                         | Owner                |     2004 |
+------+----------------------------------+----------------------+----------+
>>> 200
>>> Неизвестная команда 200
select 200
Работники с заданным стажем не найдены.
>>> select 10
   1: Zolo I S
>>>
```

Рисунок 2.2 Рез-т выполнения программы

3. Выполнил задания.

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.

Решите задачу: создайте словарь, где ключами являются числа, а значениями – строки. Примените к нему метод items(), с с помощью полученного объекта dict_items создайте новый словарь, "обратный" исходному, т. е. ключами являются строки, а значениями – числа.

prim.py zadaniya.py

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


if __name__ == '__main__':
    # 1
    school = {'1a': 20, '2b': 24, '11a': 15, '11b': 10}
    school['1a'] = 59
    school['2b'] = 127

    del school['2b']
    S = sum(school.values())
    print(school, "\nКоличество учеников: ", S, '\n')

    # 2
    a = {1: 'one', 2: 'two', 3: 'three'}
    inverse_a = {}
    for k, v in a.items():
        inverse_a[v] = k
    print(a)
    print(inverse_a)
```

```
C:\rep_2.6\venv\Scripts\python.exe C:/rep_2.6/zadaniya.py
{'1a': 59, '11a': 15, '11b': 10}
Количество учеников:  84

{1: 'one', 2: 'two', 3: 'three'}
{'one': 1, 'two': 2, 'three': 3}

Process finished with exit code 0
```

Рисунок 3.1 Вывод программы задания

4. (15 вариант). Выполнил индивидуальное задание.

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

import sys

if __name__ == '__main__':
    print("help - список всех команд")
    # Список людей.
    humans = []
    # 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.
            surname = input("Фамилия: ")
            name = input("Имя: ")
            Zodiak = input("Знак Зодиака: ")
            date = [
                int(input("Год рождения: ")),
                int(input("Месяц рождения: ")),
                int(input("День рождения: "))
                ]

            # Создать словарь.
            human = {
                'surname': surname,
                'name': name,
                'Zodiak': Zodiak,
                'date': date
            }
            # Добавить словарь в список.
            humans.append(human)

            # Отсортировать список в случае необходимости.
            if len(humans) > 1:
                humans.sort(key=lambda item: item.get('date', ''))

        elif command == 'list':
            # Заголовок таблицы.
            line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 30,
                '-' * 20,
                '-' * 15
            )
            print(line)
            print(
                '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
                    "№",
                    "Фамилия и имя",
                    "Знак Зодиака",
                    "Дата рождения"
                )
            )
            print(line)

            # Вывести данные о всех сотрудниках.
            for idx, worker in enumerate(humans, 1):
                date = worker.get('date', '')
```

```
                print(
                    '| {:^4} | {:<14} {:<15} | {:<20} |
{}.{}.{:<7}'|'.format(
                        idx,
                        worker.get('surname', ''),
                        worker.get('name', ''),
                        worker.get('Zodiak', ''),
                        date[0],
                        date[1],
                        date[2]
                    )
                )
            print(line)

        elif command.startswith('select '):
            addedZZ = command[7:]
            # Инициализировать счетчик.
            count = 0
            # Проверить сведения работников из списка.
            for human in humans:
                if human.get('Zodiak', '') == addedZZ:
                    count += 1
                    print(
                        '{:>4}: {} {}'.format(
                            count, human.get('surname', ''),
                            human.get('name', '')
                        )
                    )

            # Если счетчик равен 0, то работники не найдены.
            if count == 0:
                print("Люди с таким ЗЗ не найдены.")

        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - добавить человека;")
            print("list - вывести список людей;")
            print("help - список всех команд;")
            print("exit - завершить работу с программой.")
        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)
```

```
C:\rep_2.6\venv\Scripts\python.exe C:/rep_2.6/ind.py
help - список всех команд
>>> help
Список команд:

add - добавить человека;
list - вывести список людей;
help - список всех команд;
exit - завершить работу с программой.
>>> add
Фамилия: Technik
Имя: Pasha
Знак Зодиака: rak
Год рождения: 1983
Месяц рождения: 6
День рождения: 4
>>> list
+------+--------------------------------+----------------------+-----------------+
|  №   |         Фамилия и имя          |     Знак Зодиака     |  Дата рождения  |
+------+--------------------------------+----------------------+-----------------+
|  1   | Technik        Pasha           | rak                  | 1983.6.4        |
+------+--------------------------------+----------------------+-----------------+
>>> select oven
Люди с таким 33 не найдены.
>>> select rak
   1: Technik Pasha
>>>
```

Рисунок 4.1 Вывод программы индивидуального задания

5. Сделал коммит, выполнил слияние с веткой main, и запушил изменения в уд. репозиторий.

```
C:\rep_2.6>git add .

C:\rep_2.6>git commit -m "added programs + modidied .gitignore"
[develop 2582c62] added programs + modidied .gitignore
 4 files changed, 379 insertions(+), 3 deletions(-)
 create mode 100644 ind.py
 create mode 100644 prim.py
 create mode 100644 zadaniya.py

C:\rep_2.6>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\rep_2.6>
```

```
C:\rep_2.6>git push
Everything up-to-date
```

Рисунок 4.1 коммит и пуш изменений и переход на ветку main

```
C:\rep_2.6>git merge develop
Updating 5d4b8d1..2582c62
Fast-forward
 .gitignore   | 157 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--
 ind.py       | 105 ++++++++++++++++++++++++++++++++++++++++
 prim.py      |  99 ++++++++++++++++++++++++++++++++++++++
 zadaniya.py  |  21 ++++++++
 4 files changed, 379 insertions(+), 3 deletions(-)
 create mode 100644 ind.py
 create mode 100644 prim.py
 create mode 100644 zadaniya.py

C:\rep_2.6>_
```

Рисунок 4.2 Слияние ветки main с develop

```
C:\rep_2.6>git push
info: please complete authentication in your browser...
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 5.16 KiB | 2.58 MiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/AdamKh/rep_2.6.git
   5d4b8d1..2582c62  main -> main
```

Рисунок 4.3 Пуш изменений на удаленный сервер

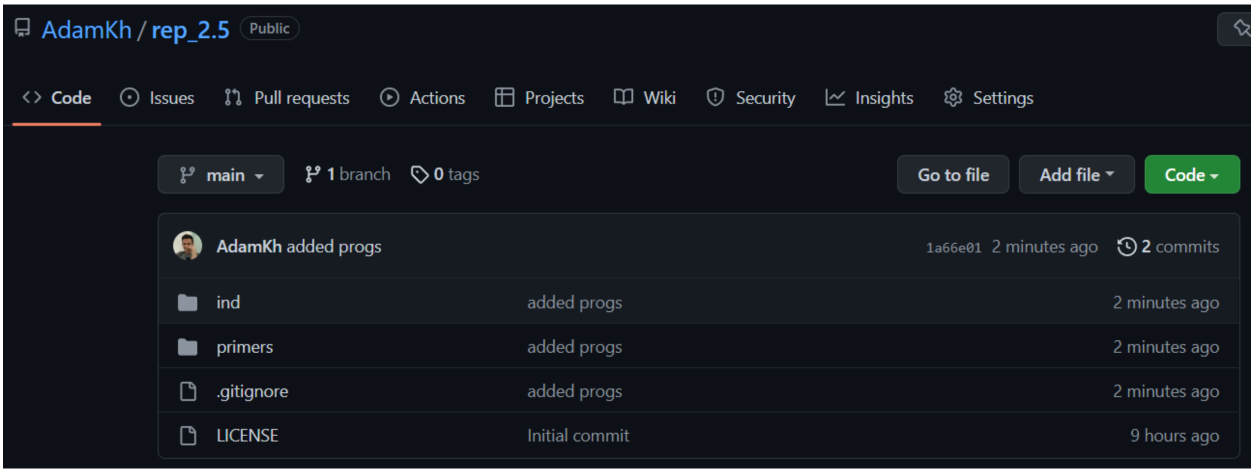

Рисунок 4.4 Изменения на удаленном сервере

Контр. вопросы и ответы на них:

1. Что такое словари в языке Python?

Словари в Python – это изменяемые отображения ссылок на объекты, доступные по ключу.

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

Функция len() возвращает длину (количество элементов) в объекте. Аргумент может быть последовательностью, такой как строка, байты, кортеж, список или диапазон или коллекцией (такой как словарь, множество или неизменяемое множество).

3. Какие методы обхода словарей Вам известны?

Самый очевидный вариант обхода словаря — это попытаться напрямую запустить цикл for по объекту словаря, так же как мы делаем это со списками, кортежами, строками и любыми другими итерируемыми объектами.

```
for something in currencies:
 print(something)
```

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

С помощью метода .get()

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

С помощью функции dict.update()

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() 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.

Предположим, что есть список имен и номером сотрудников, и их нужно объединить в массив кортежей. Для этого можно использовать функцию zip(). Вот пример программы, которая делает именно это:

```
employee_numbers = [2, 9, 18, 28]
employee_names = ["Дима", "Марина", "Андрей", "Никита"]
zipped_values = zip(employee_names, employee_numbers)
```

```
zipped_list = list(zipped_values)
print(zipped_list)
```

Функция zip возвращает следующее:

```
[('Дима', 2), ('Марина', 9), ('Андрей', 18), ('Никита', 28)]
```

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. Так, он состоит из объектов следующих типов:

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

Как получить текущие дату и время?

```
import datetime
dt_now = datetime.datetime.now()
print(dt_now)
```

Результат:

```
2022-09-11 15:43:32.249588
```

Получить текущую дату:

```
from datetime import date
current_date = date.today()
print(current_date)
```

Результат:

```
2022-09-11
```

Получить текущее время:

```
import datetime
current_date_time = datetime.datetime.now()
```

```
current_time = current_date_time.time()
print(current_time)
```

Результат:

```
15:51:05.627643
```