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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.4

Выполнила:
Кузнецова Алена Валерьевна
1 курс, группа ИВТ-б-о-21-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2022 г.

Цель работы: приобретение навыков по работе со списками при написании программ с помощью языка программирования Python.

Выполнение работы:

1. Создали общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

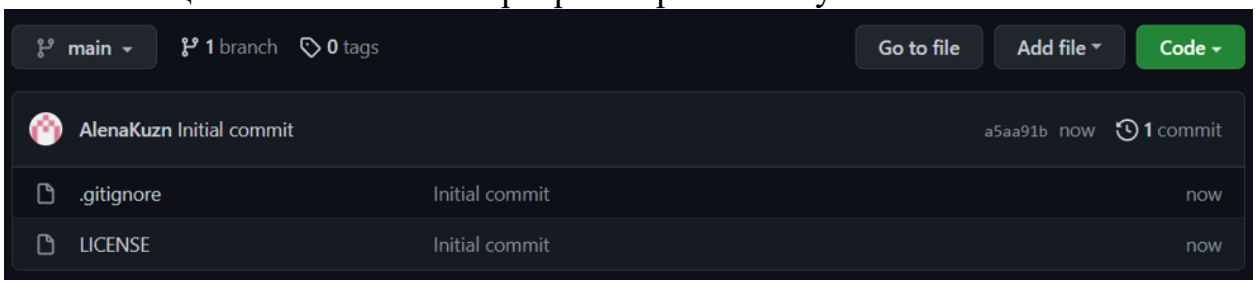

Рисунок 1 – Созданный репозиторий

2. Выполнили клонирование созданного репозитория.

```
C:\Users\akuzn>cd /d C:\Users\akuzn

C:\Users\akuzn>git clone https://github.com/AlenaKuzn/2.4.git
Cloning into '2.4'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

Рисунок 2 – Клонирование репозитория

3. Дополнили файл .gitignore необходимыми правилами для работы с IDE PyCharm.

```
267 lines (212 sloc)  5.25 KB
# 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/
.idea/**/dataSources.ids
.idea/**/dataSources.local.xml
.idea/**/sqlDataSources.xml
.idea/**/dynamic.xml
.idea/**/uiDesigner.xml
.idea/**/dbnavigator.xml

# Gradle
.idea/**/gradle.xml
.idea/**/libraries
```

Рисунок 3 – Дополнение файла .gitignore

4. Организовали свой репозиторий в соответствие с моделью ветвления git-flow.

```
C:\Users\akuzn\2.4\individ>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/akuzn/2.4/.git/hooks]
```

Рисунок 4 – Организация по модели ветвления git-flow

5. Проработайте примеры лабораторной работы. Создайте для каждого примера отдельный модуль языка Python. Зафиксируйте изменения в репозитории.

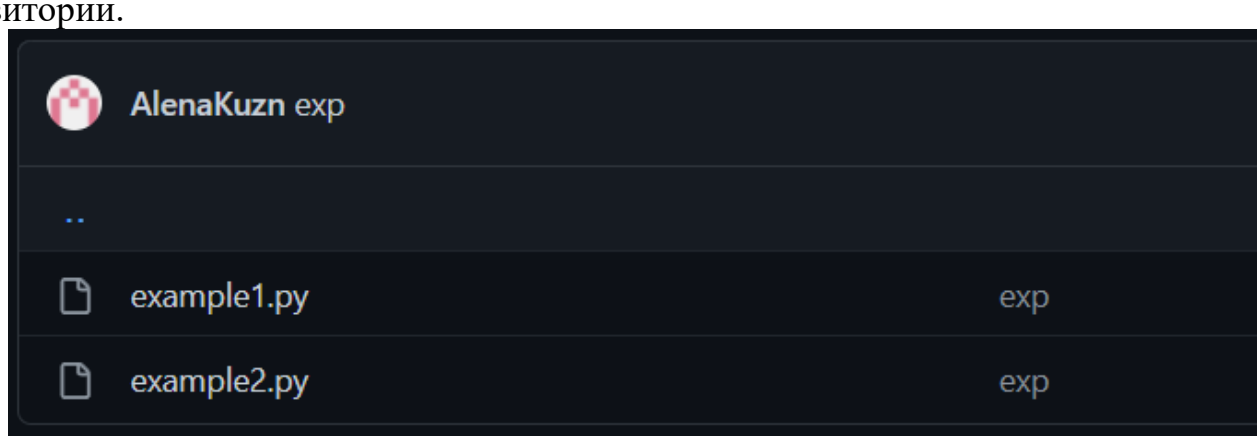

Рисунок 5 – Зафиксированные примеры в репозитории

6. Выполните индивидуальные задания, согласно своего варианта.

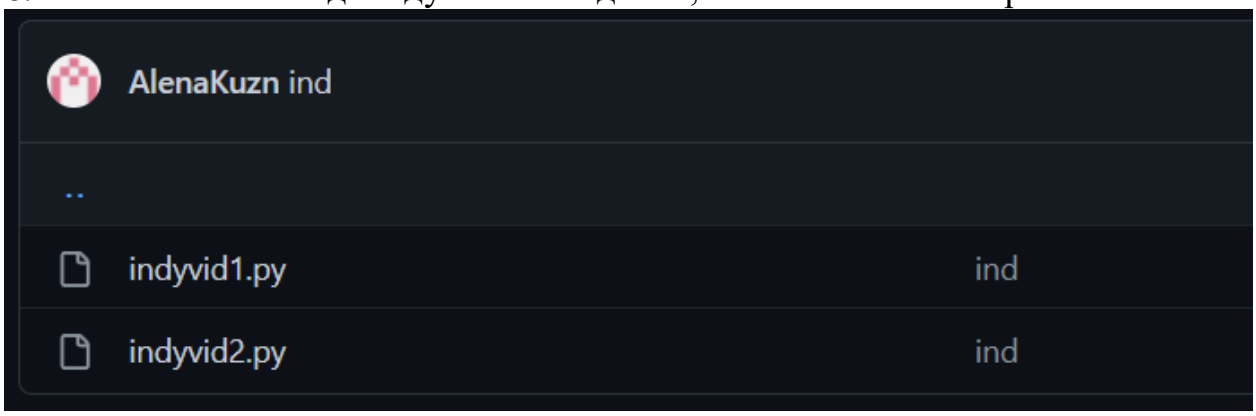

Рисунок 6 – Индивидуальные задания

7. Зафиксируйте сделанные изменения в репозитории.

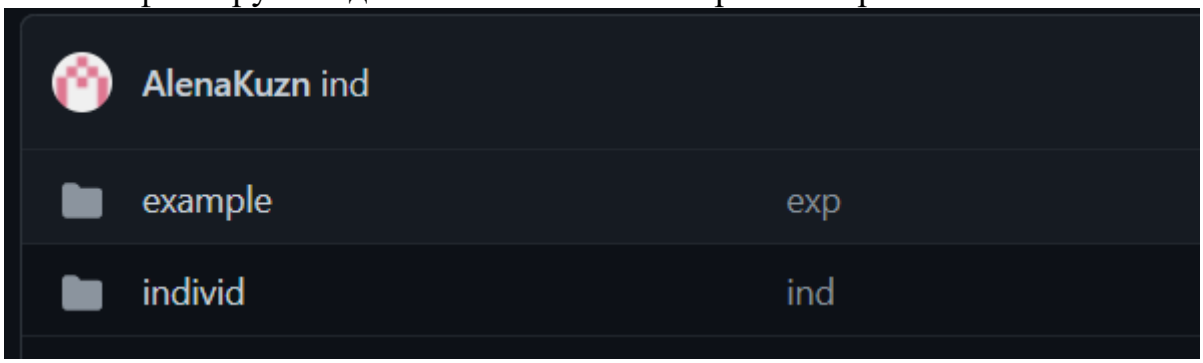

Рисунок 7 – Все изменения зафиксированные в репозитории

8. Выполните слияние ветки для разработки с веткой main / master.

```
C:\Users\akuzn\2.4>git push --force origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:      https://github.com/AlenaKuzn/2.4/pull/new/develop
remote:
To https://github.com/AlenaKuzn/2.4.git
 * [new branch]      develop -> develop
```

Рисунок 8 – Отправка изменения на репозиторий

Контрольные вопросы:

1. Что такое списки в языке Python?

Список – это структура данных для хранения объектов различных типов.

2. Как осуществляется создание списка в Python?

- Для создания списка нужно заключить элементы в квадратные скобки: my_list = [1, 2, 3, 4, 5]
- Список может выглядеть так: my_list = ['один', 'два', 'три', 'четыре', 'пять']
- Можно смешивать типы содержимого: my_list = ['один', 10, 2.25, [5, 15], 'пять']

3. Как организовано хранение списков в оперативной памяти?

При его создании в памяти резервируется область, которую можно условно назвать некоторым “контейнером”, в котором хранятся ссылки на другие элементы данных в памяти. В отличии от таких типов данных как число или строка, содержимое “контейнера” списка можно менять.

4. Каким образом можно перебрать все элементы списка?

С помощью циклы:

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']
for elem in my_list:
print(elem)
```

5. Какие существуют арифметические операции со списками?

Сложение, умножение

6. Как проверить есть ли элемент в списке?

Для того, чтобы проверить, есть ли заданный элемент в списке Python необходимо использовать оператор in.

7. Как определить число вхождений заданного элемента в списке?

Метод count можно использовать для определения числа сколько раз данный элемент встречается в списке.

8. Как осуществляется добавление (вставка) элемента в список?

Метод insert можно использовать, чтобы вставить элемент в список:

```
my_list = [1, 2, 3, 4, 5]
my_list.insert(1,'Привет')
print(my_list)
```

Метод append можно использовать для добавления элемента в список:

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']
my_list.append('ещё один')
print(my_list)
```

9. Как выполнить сортировку списка?

Для сортировки списка нужно использовать метод sort.

Для сортировки списка в порядке убывания необходимо вызвать метод sort с аргументом reverse=True.

10. Как удалить один или несколько элементов из списка?

Удалить элемент можно, написав его индекс в методе pop.

Элемент можно удалить с помощью метода remove.

Оператор del можно использовать для тех же целей.

Можно удалить несколько элементов с помощью оператора среза.

Можно удалить все элементы из списка с помощью метода clear.

11. Что такое списковое включение и как с его помощью осуществлять обработку списков?

List Comprehensions чаще всего на русский язык переводят как абстракция списков или списковое включение, является частью синтаксиса языка, которая предоставляет простой способ

построения списков. Проще всего работу list comprehensions показать на примере. Допустим вам необходимо создать список целых чисел от 0 до n, где n предварительно задается. Классический способ решения данной задачи выглядел бы так.

12. Как осуществляется доступ к элементам списков с помощью срезов?

Слайсы (срезы) являются очень мощной составляющей Python, которая позволяет быстро иьлаконично решать задачи выборки элементов из списка. Выше уже был пример использования

слайсов, здесь разберем более подробно работу с ними.

13. Какие существуют функции агрегации для работы со списками?

Для работы со списками Python предоставляет следующие функции:

- len(L) - получить число элементов в списке L .
- min(L) - получить минимальный элемент списка L .
- max(L) - получить максимальный элемент списка L .
- sum(L) - получить сумму элементов списка L , если список L содержит только числовые значения.

14. Как создать копию списка?

Для создания копии списка необходимо использовать метод copy.

15. **Самостоятельно изучите функцию sorted языка Python. В чем ее отличие от метода sort списков?**

Для сортировки по возрастанию достаточно вызвать функцию сортировки Python sorted(), которая вернёт новый отсортированный список. Также можно использовать метод списков list.sort(), который изменяет исходный список (и возвращает None во избежание путаницы). Обычно это не так удобно, как использование sorted(), но если вам не нужен исходный список, то так будет немного эффективнее. Ещё одно отличие заключается в том, что метод list.sort() определён только для списков, в то время как sorted() работает со всеми итерируемыми объектами.