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 практической работе №2.5

Дисциплины «Программирование на Python»

Выполнил:

Пустяков Андрей Сергеевич

2 курс, группа ИВТ-б-о-22-1,

09.03.01 «Информатика и вычислительная техника (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____

(подпись)

Руководитель практики:

Воронкин Р. А. кандидат технических наук, доцент, доцент кафедры инфокоммуникаций

_____

(подпись)

Ставрополь, 2023 г.

Тема: Работа с кортежами в языке Python.

Цель: приобрести навыки по работе с кортежами при написании программ с помощью языка программирования Python версии 3.x.

Ход работы:

Создание общедоступного репозитория, клонирование репозитория и организация модели ветвления в нем «git-flow» (рис. 1).

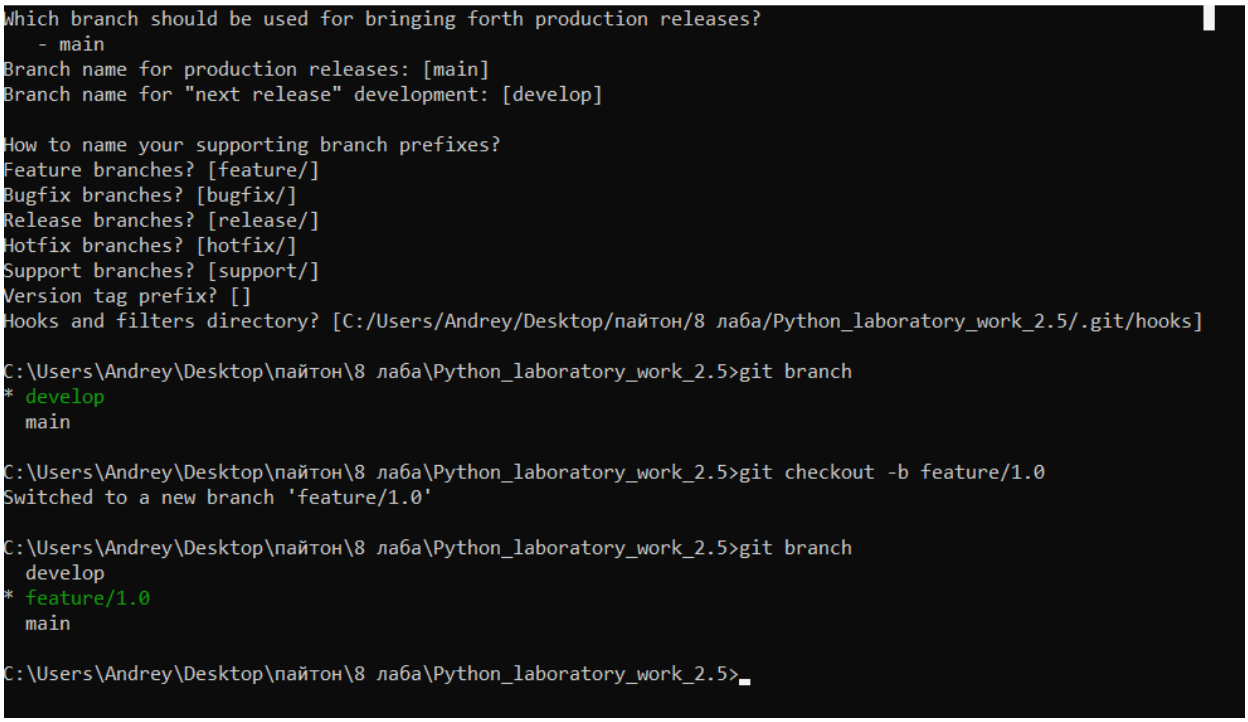

Рисунок 1 – Создание репозитория и модели ветвления «git-flow».

Проработка примеров лабораторной работы:

Пример 1.

Необходимо вести кортеж A из 10 элементов, найти сумму элементов, меньших по модулю 5, и вывести ее на экран. Использовать в программе вместо списков кортежи.

Код программы примера №1 без использования списковых включений и результаты работы программы с различными исходными данными (рис. 2, 3).

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

# Необходимо вести кортеж A из 10 элементов, найти сумму элементов,
# меньших по модулю 5, и вывести ее на экран. Использовать в программе
# вместо списков кортежи.

import sys


if __name__ == '__main__':
    # Ввести кортеж одной строкой.
    A = tuple(map(int, input("Введите кортеж из 10-ти элементов:").split()))
    # Проверить количество элементов кортежа.
    if len(A) != 10:
        print("Неверный размер кортежа", file=sys.stderr)
        exit(1)

    # Найти искомую сумму.
    s = 0
    for item in A:
        if abs(item) < 5:
            s += item

print("Сумма элементов меньше по модуль 5:", s)
```

Рисунок 2 – Код программы примера №1.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python
Введите кортеж из 10-ти элементов:1 2 3 4 5 6 7 8 9 10
Сумма элементов меньше по модуль 5: 10

Process finished with exit code 0
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python
Введите кортеж из 10-ти элементов:1 2 3 4 5 6 7 8 9 10 11
Неверный размер кортежа

Process finished with exit code 1
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.e
Введите кортеж из 10-ти элементов:1 2 3 4 1 2 3 4 1 2
Сумма элементов меньше по модуль 5: 23

Process finished with exit code 0
```

Рисунок 3 – Результаты работы программы примера №1.

Код программы примера №1 с использования списковых включений и результаты работы программы с различными исходными данными (рис. 4, 5).

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

# Необходимо вести кортеж A из 10 элементов, найти сумму элементов,
# меньших по модулю 5, и вывести ее на экран. Использовать в программе
# вместо списков кортежи.

import sys


if __name__ == '__main__':
    # Ввести список одной строкой.
    A = list(map(int, input("Введите кортеж из 10-ти элементов:").split()))
    # Проверить количество элементов списка.
    if len(A) != 10:
        print("Неверный размер списка", file=sys.stderr)
        exit(1)

    # Найти искомую сумму.
    s = sum(a for a in A if abs(a) < 5)
    print("Сумма элементов меньше по модуль 5:", s)
```

Рисунок 4 – Код программы примера №1 с использованием списковых включений.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe "C:
Введите кортеж из 10-ти элементов:1 2 3 4 5 6 7 8 9 10
Сумма элементов меньше по модуль 5: 10

Process finished with exit code 0
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe "C:
Введите кортеж из 10-ти элементов:1 2 3 4 5 6 7 8 9 10 11
Неверный размер списка

Process finished with exit code 1
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe "C:
Введите кортеж из 10-ти элементов:123 1 6 3 28 0 11 5 3 1
Сумма элементов меньше по модуль 5: 8

Process finished with exit code 0
```

Рисунок 5 – Результаты работы программы примера №1 с использованием списковых включений.

Выполнение индивидуальных заданий:

Задание 1.

Даны два кортежа одного размера, в которых нет нулевых элементов. Получить третий кортеж, каждый элемент которого равен 1, если элементы заданных кортежей с тем же номером имеют одинаковый знак, и равен нулю в противном случае (Вариант 26).

Код программы индивидуального задания №1 и результаты работы программы с различными входными данными (рис. 6, 7).

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

# Даны два кортежа одного размера, в которых нет нулевых
# элементов. Получить третий кортеж, каждый элемент которого
# равен 1, если элементы заданных кортежей с тем же номером
# имеют одинаковый знак, и равен нулю в противном случае.

import sys


if __name__ == '__main__':
    # Ввести кортежи одной строкой.
    first_tuple = tuple(map(float, input("Введите кортеж 1"
                                         " без нулевых элементов:
").split()))
    second_tuple = tuple(map(float, input("Введите кортеж 2"
                                          " без нулевых элементов:
").split()))
    # Проверить размер обоих кортежей.
    if len(first_tuple) != len(second_tuple):
        print("Кортежи должны быть одинаковой длины!", file=sys.stderr)
        exit(1)
    # Проверить элементы в кортежах.
    if 0 in first_tuple or 0 in second_tuple:
        print("В кортежах есть нули!", file=sys.stderr)
        exit(1)

    # Составим третий кортеж.
    count = 0  # Счетчик для обращения к элементу второго кортежа
    third_tuple = ()
    for i in first_tuple:
        if ((i > 0 and second_tuple[count] > 0) or
                (i < 0 and second_tuple[count] < 0)):
            third_tuple += (1,)
        else:
            third_tuple += (0,)
        count += 1

    print("Элементы первого кортежа: ", first_tuple)
    print("Элементы второго кортежа: ", second_tuple)
    print("Элементы третьего кортежа: ", third_tuple)
```

Рисунок 6 – Код программы индивидуального задания №1.

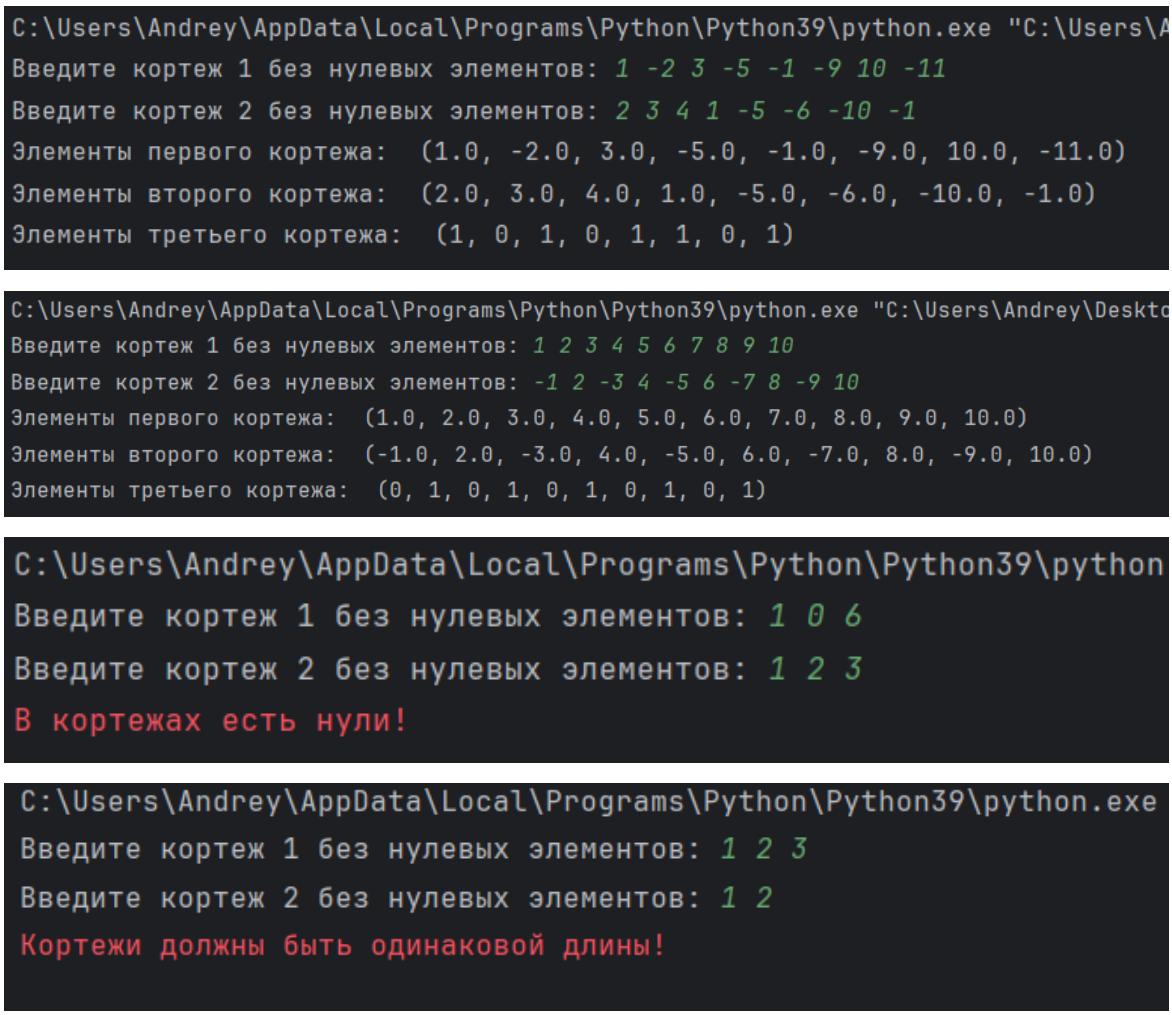

Рисунок 7 – Результаты работы программы индивидуального задания №1.

Ответы на контрольные вопросы:

1. Списки – это структура данных для хранения объектов различных типов. Размер списка не статичен, его можно изменять. Список по своей природе является изменяемым типом данных. Переменная, определяемая как список, содержит ссылку на структуру в памяти, которая в свою очередь хранит ссылки на какие-либо другие объекты или структуры.

2. Кортеж – это неизменяемая структура данных, которая по своему подобию очень похожа на список. Список – это изменяемый тип данных, а кортежи неизменны. Существует несколько причин, по которым стоит использовать кортежи вместо списков. Одна из них – это обезопасить данные от случайного изменения. Кортежи в памяти занимают меньший объем по сравнению со списками. Вторая причина – прирост производительности, который связан с тем, что кортежи работают быстрее, чем списки (т. е. на

операции перебора элементов и т. п. будет тратиться меньше времени). Важно также отметить, что кортежи можно использовать в качестве ключа у словаря.

3. Для создания кортежей можно воспользоваться одним из способов: a = (); a = tuple(). Если необходим кортеж с заранее заданными данными, то их просто необходимо указывать в круглых скобках.

4. Доступ к элементам кортежа осуществляется также как к элементам списка – через указание индекса, а также что специфично для кортежа – деструктуризация, можно обращаться к элементам разбирая кортеж.

5. Кортежи часто содержат значения разных типов, и помнить, по какому индексу что лежит — очень непросто. В связи с этим, кортеж можно не только собрать, но и разобрать: name_and_age = ('Bob', 42) - создание кортежа, (name, age) = name_and_age - присваивание элементов из кортежа переменным name и age.

6. Кортеж очень полезен при обмене значениями: (a, b) = (b, a); и при множественном присваивании: (a, b, c) = (1, "smth", 3.5).

7. Использование среза на кортеже создаёт новый кортеж, например: tuple2 = tuple1[::-1].

8. Кортеж может быть образован путем операции повторения, обозначаемой символом *. При использовании в выражении общая форма операции, следующая: kort2 = kort1 * n. Также, есть возможность выполнять конкатенацию: kort3 = kort1 + kort2 (Важен порядок, в котором идёт суммирование кортежей!).

9. Элементы кортежа можно последовательно просмотреть с помощью операторов цикла while или for (Путём использования оператора in, или по индексам).

10. Оператор in позволяет проверить, есть ли в кортеже требуемый элемент.

11. Для работы с кортежами существуют методы: index() – возвращает индекс найденного в кортеже требуемого элемента; count() – считает количество идентичных запрашиваемому элементу.

12. Существует возможность использования len() и sum() при работе с кортежами.

13. Списковое включение можно использовать и для заполнения кортежа, пример: smth = tuple(i for i in list).

Вывод: в ходе лабораторной работы была изучена работа и использование кортежей в языке программирования Python. Были изучены основные операции при работе с кортежами, способы создания кортежей, рассмотрены способы обращения к элементам кортежа, рассмотрены примеры применения кортежей и их особенности, были изучены основные операторы при работе с кортежами, способы перебора значений в кортежах через цикл или списковые включения, были рассмотрены методы при работе с кортежами, а также отличия кортежа от списка, свойства кортежей.