МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФГАОУ ВО
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра инфокоммуникаций

Отчет по лабораторной работе №7

По дисциплине основы кроссплатформенного программирования

«Работа с кортежами в языке Python»

Выполнила:

студентк группы ИТС-б-о-21-1

Аллаёров Жамшид Хасан угли

(подпись)

Проверил: Доцент, к.т.н, доцент кафедры

инфокоммуникаций

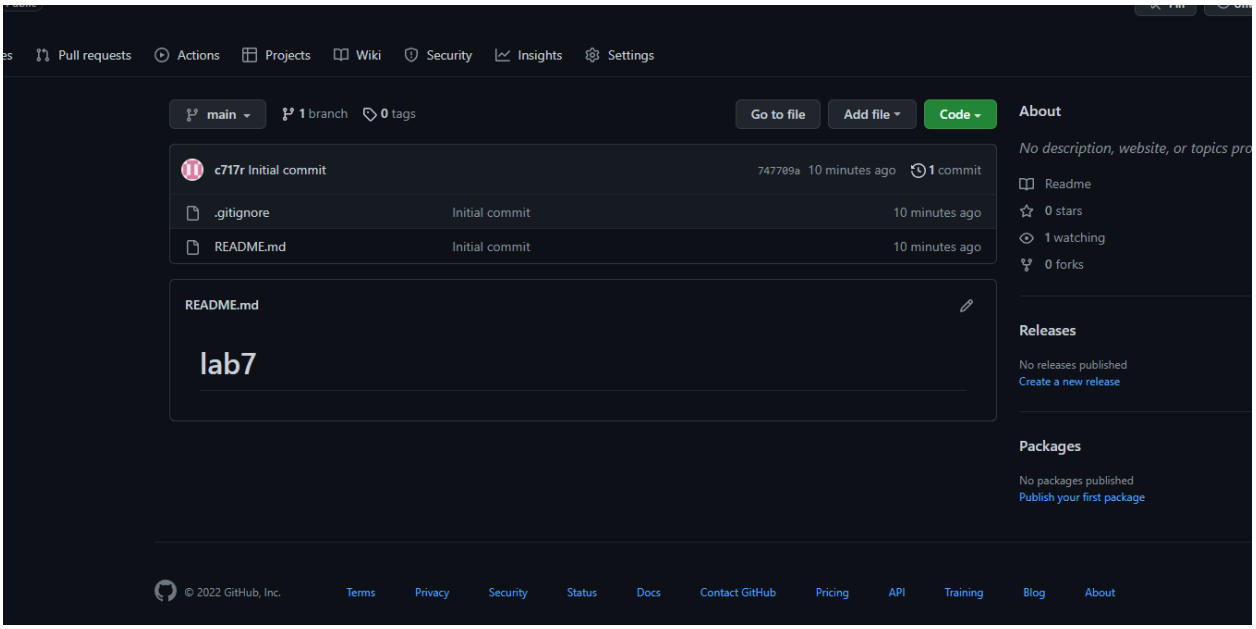
Воронкин Р. А.

Работа защищена с оценкой:

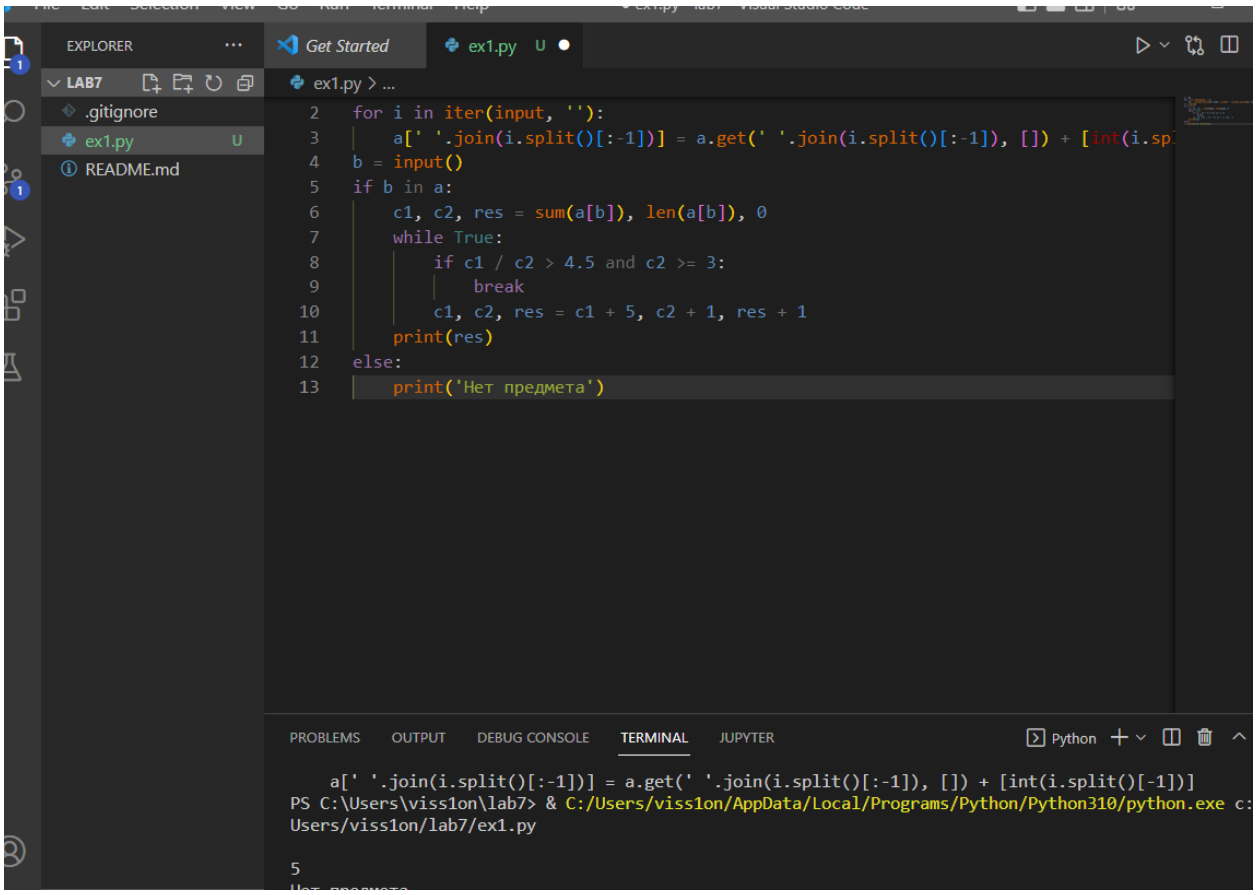
(подпись)

Ставрополь, 2022

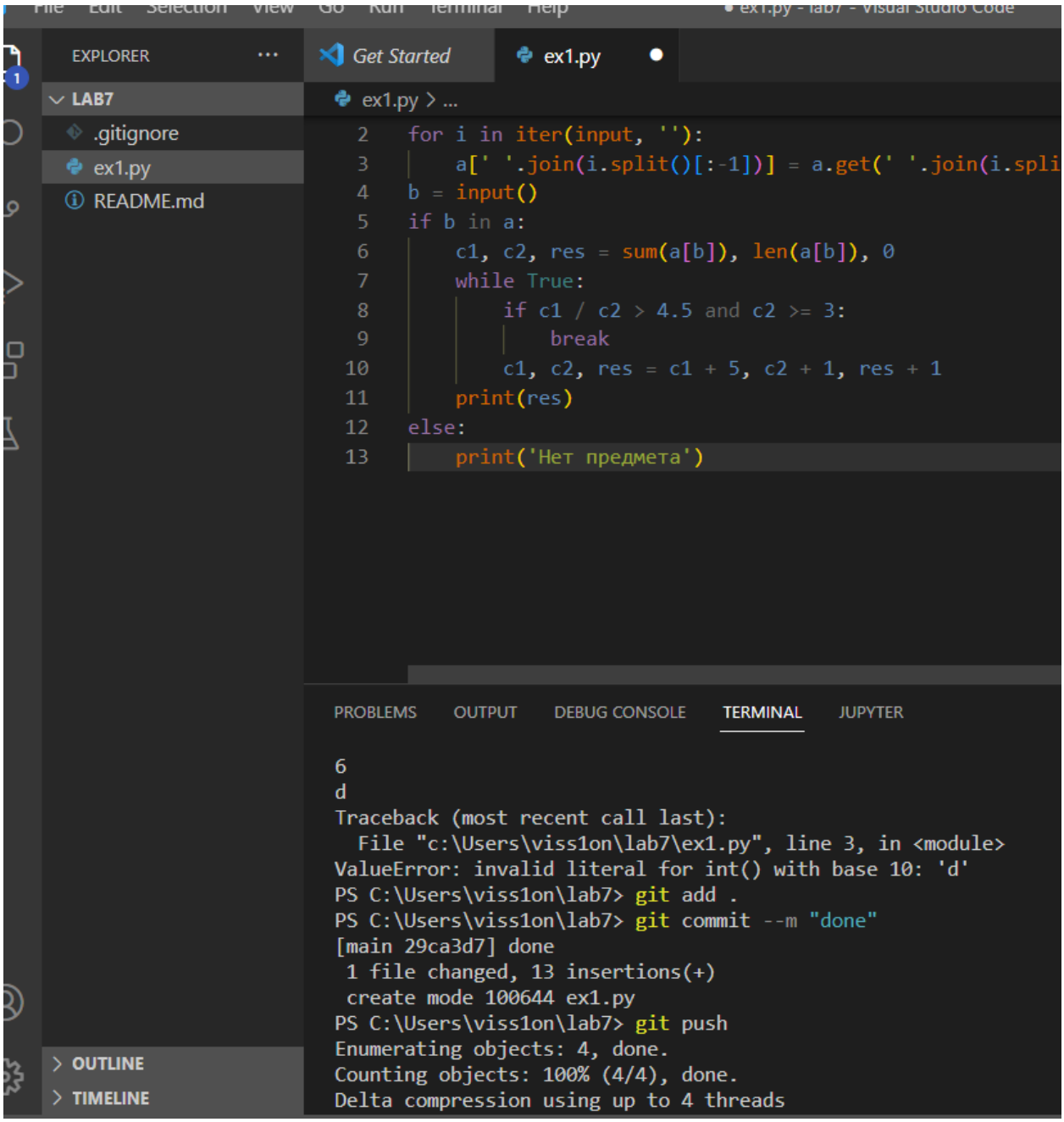
Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия
MIT и выбранный Вами язык программирования (выбор языка программирования будет
доступен после установки флажка **Add .gitignore**)



Известны оценки по геометрии каждого из 24 учеников класса. В начале списка
перечислены все пятерки, затем все остальные оценки. Сколько учеников имеет по
геометрии оценку «5»? Условный оператор не использовать.



Отправьте сделанные изменения на сервер GitHub.



1. Что такое кортежи в языке Python?

Кортежи (tuple) в Python – это те же списки за одним исключением. Кортежи неизменяемые структуры данных. Так же как списки они могут состоять из элементов разных типов, перечисленных через

запятую. Кортежи заключаются в круглые, а не квадратные скобки.

2. Каково назначение кортежей в языке Python?

Кортежи (tuple) в Python – это те же списки за одним исключением. Кортежи неизменяемые структуры данных. Так же как списки они могут состоять из элементов разных типов, перечисленных через запятую. Кортежи заключаются в круглые, а не квадратные скобки.

```
>>> a = (10, 2.13, "square", 89, 'C')
>>> a
(10, 2.13, 'square', 89, 'C')
```

Из кортежа можно извлекать элементы и брать срезы:

```
>>> a[3]
89
>>> a[1:3]
(2.13, 'square')
```

3. Как осуществляется создание кортежей?

Для создания пустого кортежа можно воспользоваться одной из следующих команд.

```
>>> a = ()
>>> print(type(a))
<class 'tuple'>
>>> b = tuple()
>>> print(type(b))
<class 'tuple'>
```

Кортеж с заданным содержанием создается также как список, только вместо квадратных скобок используются круглые.

```
>>> a = (1, 2, 3, 4, 5)
```

```
>>> print(type(a))
<class 'tuple'>
>>> print(a)
(1, 2, 3, 4, 5)
```

При желании можно воспользоваться функцией tuple().

```
>>> a = tuple((1, 2, 3, 4))
>>> print(a)
(1, 2, 3, 4)
```

4. Как осуществляется доступ к элементам кортежа?

Доступ к элементам кортежа осуществляется также как к элементам списка – через указание индекса. Но, как уже было сказано – изменять элементы кортежа нельзя!

5. Зачем нужна распаковка (деструктуризация) кортежа?

Выше мы научились создавать кортежи, а теперь попробуем извлекать значения из них. Для этого достаточно обратиться к элементу кортежа по индексу:

```
name_and_age = ('Bob', 42)

name_and_age[0]  # 'Bob'
name_and_age[1]  # 42
```

Также у кортежа есть длина, которую можно получить с помощью функции len():

```
tuple = (42,)  # (42,)
len(tuple)     # 1
pair = (1, 2)  # (1, 2)
len(pair)      # 2
```

6. Какую роль играют кортежи в множественном присваивании?

Множества — это неупорядоченный список, то есть элементы внутри него располагаются в случайном порядке. Как списки и словари, множество относится к изменяемым типам данных. Но при этом невозможен поиск по индексу, так как элементы в случайном порядке. Зато множества дают высокую скорость поиска по ним. Например, это удобно, когда порядок вам не важен, а нужно просто быстро искать нужные элементы.

Создать пустое множество можно с помощью функции **set()**. А если с элементами, то с помощью их перечисления в фигурных скобках, как у словаря:

Все элементы множества всегда уникальные. Это полезное свойство множеств пригождается, когда вам нужно почистить ваш список и убрать дубли:

Получаем множество с уникальными элементами.

7. Как выбрать элементы кортежа с помощью среза?

Работать с одиночными элементами вы уже умеете. Настало время перейти к очень интересному инструменту, который Python предоставляет для работы с целыми подмножествами элементов списка: к так называемым *срезам (slices)*.

Синтаксис описания срезов

Срезы встроены в язык и снабжены своим собственным синтаксисом — настолько широко они используются. Срез записывается так же, как записывается обращение к элементу списка по индексу:

some_list[START:STOP:STEP]

Всего у среза три параметра:

- START — индекс первого элемента в выборке,
- STOP — индекс элемента списка, перед которым срез должен закончиться (т.е. сам элемент с индексом STOP не будет входить в выборку),
- STEP — шаг прироста выбираемых индексов.

8. Как выполняется конкатенация и повторение кортежей?

6. Конкатенация +

Для кортежей можно выполнять операцию конкатенации, которая обозначается символом +. В простейшем случае для конкатенации двух кортежей общая форма операции следующая

T3 = T1 + T2

где

- *T1*, *T2* – кортежи, для которых нужно выполнить операцию конкатенации. Операнды *T1*, *T2* обязательно должны быть кортежами. При выполнении операции конкатенации для кортежей, использовать в качестве операндов любые другие типы (строки, списки) запрещено;
- *T3* – кортеж, который есть результатом.
- **Пример.**
- # Кортежи. Конкатенация +
- # Конкатенация двух кортежей
- A = (1, 2, 3)
- B = (4, 5, 6)
- C = A + B # C = (1, 2, 3, 4, 5, 6)
-
- # Конкатенация кортежей со сложными объектами
- D = (3, "abc") + (-7.22, ['a', 5]) # D = (3, 'abc', -7.22, ['a', 5])
-
- # Конкатенация трех кортежей
- A = ('a', 'aa', 'aaa')
- B = A + (1, 2) + (True, False) # B = ('a', 'aa', 'aaa', 1, 2, True, False)
-

9. Как выполняется обход элементов кортежа?

Обход кортежа в цикле. Пример

Элементы кортежа можно последовательно просмотреть с помощью операторов цикла while или for.

Пример

```
# Обход кортежа в цикле
# 1. Цикл for
# Заданный кортеж
A = ("abc", "abcd", "bcd", "cde")

# Вывести все элементы кортежа
for item in A:
    print(item)
```

10. Как проверить принадлежность элемента кортежу?

```
>>> 2 in (2, 3, 4)
True
```

in операторе может использоваться для проверки принадлежности какого-либо элемента в последовательности.

И не называйте свой кортеж как tuple. Используйте другое имя.

11. Какие методы работы с кортежами Вам известны?

Защита от дурака. То есть кортеж защищен от изменений, как намеренных (что плохо), так и случайных (что хорошо).

Меньший размер. Дабы не быть голословным:

```
>>>
>>> a = (1, 2, 3, 4, 5, 6)
>>> b = [1, 2, 3, 4, 5, 6]
>>> a.__sizeof__()
```

```
36
>>> b.__sizeof__()
44
```

12. Допустимо ли использование функций агрегации таких как len() , sum() и т. д. при работе
с кортежами?

Давайте разберем, что такое функция sum() в Python и почему это питонический способ суммирования.

Встроенная функция sum() – это эффективный и питонический способ суммирования списка числовых значений. Сложение нескольких чисел является обычным промежуточным шагом во многих вычислениях, поэтому sum() – довольно удобный инструмент для программиста Python.

Еще с помощью sum() можно объединять списки и кортежи. Это интересный дополнительный вариант использования, полезный, когда вам нужно сгладить список списков.

Приведенная ниже информация поможет вам эффективно решать проблемы суммирования в вашем коде с помощью sum() или других альтернативных и специализированных инструментов.

13. Как создать кортеж с помощью спискового включения.

Списковые включения в Python являются краткими синтаксическими конструкциями. Их можно использовать для создания списков из других списков, применяя функции к каждому элементу в списке. В этом разделе объясняется и демонстрируется использование этих выражений.

Примеры

```
# списковое включение, выдаёт [2, 3, 4]
[x + 1 for x in (1, 2, 3)]
```

```
# генераторное выражение, выдаст 2, затем 3, затем 4
(x + 1 for x in (1, 2, 3))

# списковое включение с фильтром выдаёт [2]
[x for x in (1, 2, 3) if x % 2 == 0]

# списковое включение с тройкой
[x + 1 if x % 2 == 0 else x for x in (1, 2, 3)]
```