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

Кафедра инфокоммуникаций

дисциплина «Основы кроссплатформенного программирования»

Отчет по лабораторной работе №6

Работа с кортежами в языке Python

Выполнил: студент группы ИТС-б-о-21-1
Джу Алексей

_____
(подпись)
Проверил: кандидат технических наук,

Доцент кафедры инфокоммуникаций

Роман Александрович Воронкин

_____
(подпись)

Ставрополь, 2022

Лабораторная работа 6. Работа с кортежами в языке Python

Цель работы: приобретение навыков по работе с кортежами при написании программ с помощью языка программирования Python версии 3.x.

Ход работы:

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

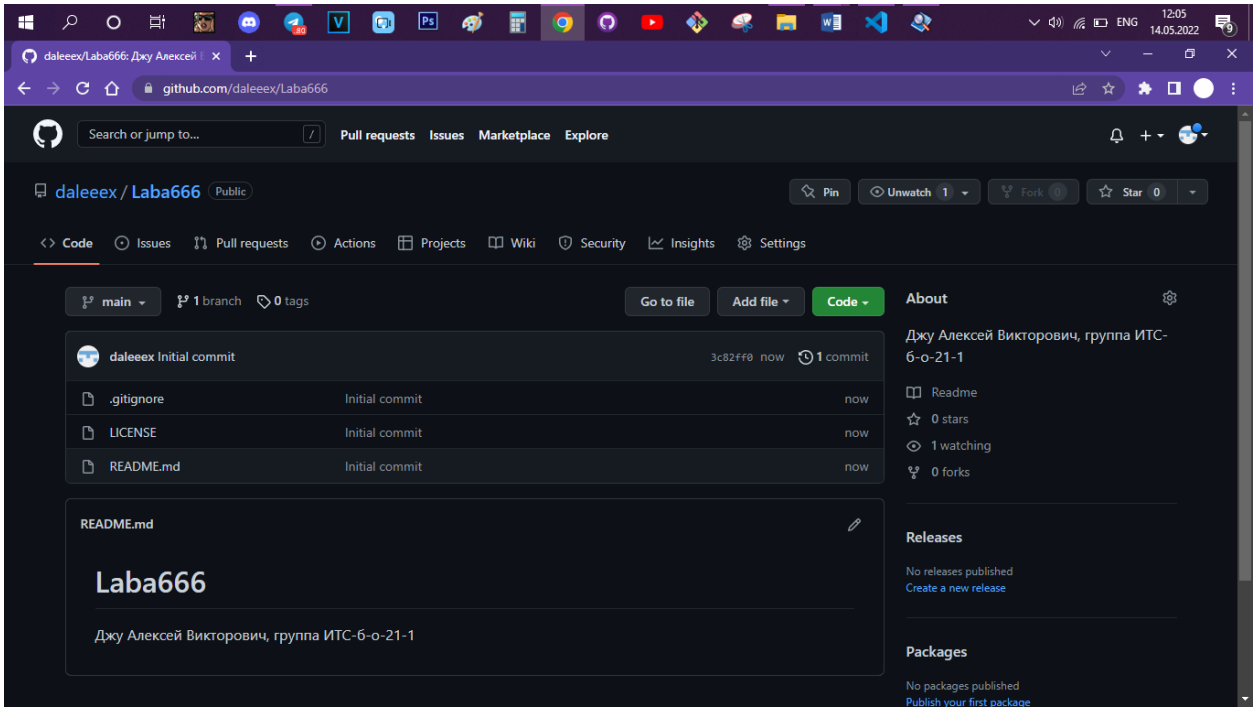

Рисунок 1

Индивидуальное задание

12. В начале кортежа записано несколько равных между собой элементов. Определить количество таких элементов и вывести все элементы, следующие за последним из них. Рассмотреть возможность того, что весь массив заполнен одинаковыми элементами. Условный оператор не использовать.

Рисунок 1.2

Ответы на контрольные вопросы:

1) Кортеж (tuple) – это неизменяемая структура данных, которая по своему подобию очень похожа на список. Как вы наверное знаете, список – это изменяемый тип данных. Т. е. если у нас есть список a = [1, 2, 3] и мы хотим заменить второй элемент с 2 на 15, то мы может это сделать, напрямую обратившись к элементу списка.

2) Существует несколько причин, по которым стоит использовать кортежи вместо списков. Одна из них – это обезопасить данные от случайного изменения. Если мы получили откуда-то массив данных, и у нас есть желание поработать с ним, но при этом непосредственно менять данные мы не собираемся, тогда, это как раз тот случай, когда кортежи придутся как нельзя кстати. Используя их в данной задаче, мы дополнительно получаем сразу несколько бонусов – во-первых, это экономия места. Дело в том, что кортежи в памяти занимают меньший объем по сравнению со списками.

Во-вторых – 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 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)

Определять кортежи очень просто, сложности могут возникнуть только с кортежами, содержащими ровно один элемент. Если мы просто укажем значение в скобках, то Python подумает, что мы хотим посчитать арифметическое выражение со скобками:

not_a_tuple = (42) # 42

Чтобы сказать Python, что мы хотим создать именно кортеж, нужно поставить после элемента кортежа запятую:

tuple = (42,) # (42,)

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

5) Обращение по индексу, это не самый удобный способ работы с кортежами. Дело в том, что кортежи часто содержат значения разных типов, и помнить, по какому индексу что лежит — очень непросто. Но есть способ лучше! Как мы кортеж собираем, так его можно и разобрать:

name_and_age = ('Bob', 42)

(name, age) = name_and_age

name # 'Bob'

age   # 42

Именно таким способом принято получать и сразу разбирать значения, которые возвращает функция (если таковая возвращает несколько значений, конечно): (quotient, modulo) = div_mod(13, 4)

Соответственно кортеж из одного элемента нужно разбирать так:

```
(a,) = (42,)
a  # 42
```

Если же после имени переменной не поставить запятую, то синтаксической ошибки не будет, но в переменную a кортеж запишется целиком, т. е. ничего не распакуется. Всегда помните о запятых!

6) Благодаря тому, что кортежи легко собирать и разбирать, в Python удобно делать такие вещи, как множественное присваивание. Смотрите:

```
(a, b, c) = (1, 2, 3)
a  # 1
b  # 2
c  # 3
```

Используя множественное присваивание, можно провернуть интересный трюк: обмен значениями между двумя переменными. Вот код:

```
a = 100
b = 'foo'
(a, b) = (b, a)
a  # 'foo'
b  # 100
```

Строку (a, b) = (b, a) нужно понимать как "присвоить в a и b значения из кортежа, состоящего из значений переменных b и a ".

7) С помощью операции взятия среза можно получить другой кортеж. Общая форма операции взятия среза для кортежа следующая: T2 = T1[i:j]

T2 – новый кортеж, который получается из кортежа T1;

T1 – исходный кортеж, для которого происходит срез;

i, j – соответственно нижняя и верхняя границы среза. Фактически берутся ко вниманию элементы, лежащие на позициях i, i+1, …, j-1. Значение j определяет позицию за последним элементом среза. Операция взятия среза для кортежа может иметь модификации такие же как и для списков.

8) Для кортежей можно выполнять операцию конкатенации, которая обозначается символом +. В простейшем случае для конкатенации двух кортежей общая форма операции следующая: T3 = T1 + T2

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.

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

10) Проверка вхождения элемента в кортеж

```
# Оператор in
# Заданный кортеж, который содержит строки
A = ("abc", "abcd", "bcd", "cde")
# Ввести элемент
item = str(input("s = "))
if (item in A):
print(item, " in ", A, " = True")
else:
print(item, " in ", A, " = False")
```

11) Метод index(). Поиск позиции элемента в кортеже. Чтобы получить индекс (позицию) элемента в кортеже, нужно использовать метод index(). Общая форма вызова метода следующая: pos = T.index(item)

Метод count(). Количество вхождений элемента в кортеж. Чтобы определить количество вхождений заданного элемента в кортеж используется метод count, общая форма которого следующая: k = T.count(item)

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

Для определения длины кортежа (числа его элементов), используется уже знакомая вам функция: len(a)

13) 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

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