иМинистерство 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 №8
дисциплины «Программирование на Python»
Вариант 31

Выполнил:
Репкин Александр Павлович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной
техники и автоматизированных
систем», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р.А., канд. техн. наук,
доцент кафедры инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: Работа с кортежами в языке Python.

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

Порядок выполнения работы:

1. Выполнен первый пример. В нём анализировались 10 введённых чисел. Если по модулю они меньше 5, то они суммировались, после чего сумма выводится на экран.

```
Добрый день, введите элементы через пробел: -3 -2 -1 1 2 3 9 5 -2 0
Сумма подходящих элементов =  -2
```

Рисунок 1. Полученный результат примера №1.

2. Выполнено индивидуальное задание – необходимо напечатать все элементы, следующие за первой парой одинаковых соседних элементов.

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

# Вариант по списку - №31. Взят вариант №5. Первый вариант выполнения - элементы после пары.
if __name__ == '__main__':
    # Ввести кортеж одной строкой. Не обязательно должны быть только числа, поэтому map не нужен.
    A = tuple(input(
        "Good day! Please, enter elements.\nDon't forget putting spaces between them - ").split())
    # Введение переменных для проверки найденного элемента.
    first_pair_found = False
    previous_element = " "
    for i in A:
        if first_pair_found:
            print(i, end=" ")
        elif i == previous_element:
            print("Great news! We found a pair of equal elements -",
                  previous_element, "! List of next elements:")
            first_pair_found = True
        else:
            previous_element = i
    if not first_pair_found:
        print("Excuse us, but we couldn't find any pairs of equal elements.")
```

Рисунок 2. Полученный код индивидуального задания, последующие элементы.

```
Good day! Please, enter elements.
Don't forget putting spaces between them - 1 2 3 4 5 r r a b c d e e 4 3 2 1
Great news! We found a pair of equal elements - r ! List of next elements:
a b c d e e 4 3 2 1
```

Рисунок 3. Полученный результат индивидуального задания, последующие элементы.

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

# Вариант по списку - №31. Взят вариант №5. Второй вариант выполнения - элементы перед парой.
if __name__ == '__main__':
    # Ввести кортеж одной строкой. Не обязательно должны быть только числа, поэтому map не нужен.
    A = tuple(input(
        "Good day! Please, enter elements.\nDon't forget putting spaces between them - ").split())
    # Введение переменных для проверки найденного элемента.
    first_pair_found = False
    previous_element = " "
    for i in A:
        if i == previous_element:
            first_pair_found = True
            break
        else:
            previous_element = i
    if first_pair_found:
        print("Great news! We found a pair of equal elements -",
              previous_element, "! List of previous elements:")
        for i in A:
            if i != previous_element:
                print(i, end=" ")
            else:
                break
    else:
        print("Excuse us, but we couldn't find any pairs of equal elements.")

```

Рисунок 4. Полученный код индивидуального задания, предыдущие элементы.

```
Good day! Please, enter elements.
Don't forget putting spaces between them - 1 2 3 4 5 e e d c b a r r 4 3 2 1
Great news! We found a pair of equal elements - e ! List of previous elements:
1 2 3 4 5
```

Рисунок 5. Полученный результат индивидуального задания, предыдущие элементы.

3. Ответы на вопросы.

1) Что такое списки в языке Python?

Ответ: списки – 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) Каково назначение кортежей в языке Python?

Ответ: кортеж (tuple) – 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) Как осуществляется создание кортежей?

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

4) Как осуществляется доступ к элементам кортежа?

Ответ: доступ к элементам кортежа осуществляется также как к элементам списка – через указание индекса.

5) Зачем нужна распаковка (деструктизация) кортежа?

Ответ: 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) Какую роль играют кортежи в множественном присваивании?

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

7) Как выбрать элементы кортежа с помощью среза?

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

8) Как выполняется конкатенация и повторение кортежей?

Ответ: 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) Как выполняется обход элементов кортежа?

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

10) Как проверить принадлежность элемента к кортежу?

Ответ: оператор in позволяет проверить, есть ли в кортеже требуемый элемент.

11) Какие методы работы с кортежами существуют?

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

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

Ответ: присутствует возможность использования len() и sum() при работе с кортежами.

13) Как создать кортеж с помощью спискового включения?

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

Вывод: в ходе выполнения лабораторной работы были приобретены навыки работы с кортежами при написании программ с помощью языка программирования Python версии 3.x.