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

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №4
дисциплины «Программирование на Python»

Вариант 20

Выполнил:
Скоробогатов Виктор Андреевич
2 курс, группа ИВТ-б-о-24-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Роман Александрович,
доцент департамента цифровых, робототехнических систем и электроники института перспективной инженерии

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

Тема: Работа со списками и кортежами в языке Python

Цель: Приобретение навыков по работе со списками и кортежами при написании программ с помощью языка программирования Python версии 3.x.

Ход работы

Ссылка на репозиторий:

https://github.com/LostsSs78/NCFU-Python_03-12-24-13-14

Для выполнения работы был создан новый общедоступный репозиторий на GitHub, использующий лицензию MIT и язык программирования Python, далее он был склонирован на персональный компьютер:

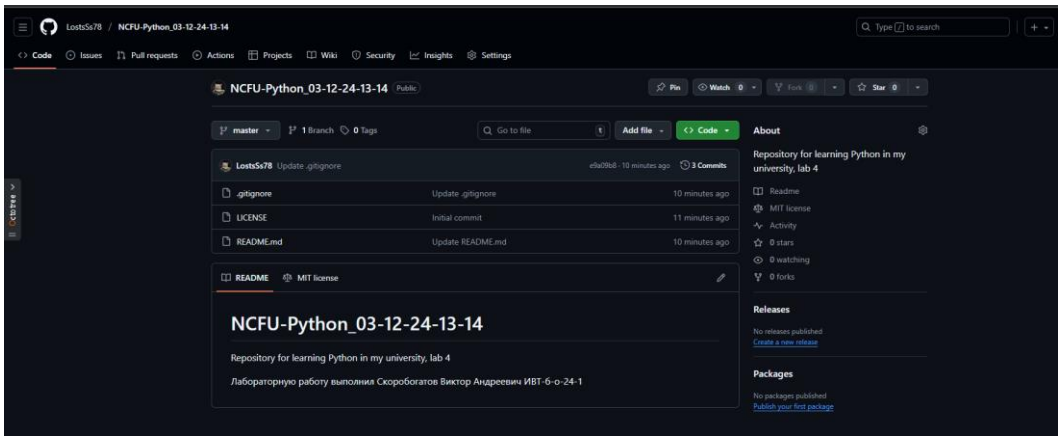

Рисунок 1. Репозиторий на GitHub

Далее был и проработаны примеры, созданы отдельные модули для каждого примера и зафиксированы изменения в репозитории:

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

import sys


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

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

    print(s)
```

Рисунок 2. Пример 1

```
D:\python\4\NCFU-Python_03-12-24-13-14>python Example1.py
1 2 3 4 5 6 7 8 9 10
10

D:\python\4\NCFU-Python_03-12-24-13-14>python Example1.py
8 9 2 4 3 2 1 4 6 4
20
```

Рисунок 3. Результат выполнения кода примера 1 с разными входными данными

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

import sys

if __name__ == '__main__':
    # Ввести список одной строкой.
    a = list(map(int, input().split()))
    # Если список пуст, завершить программу.
    if not a:
        print("Заданный список пуст", file=sys.stderr)
        exit(1)

    # Определить индексы минимального и максимального элементов
    a_min = a_max = a[0]
    i_min = i_max = 0
    for i, item in enumerate(a):
        if item < a_min:
            i_min, a_min = i, item

        if item >= a_max:
            i_max, a_max = i, item

    # Проверить индексы и обменять их местами.
    if i_min > i_max:
        i_min, i_max = i_max, i_min

    # Посчитать количество положительных элементов.
    count = 0
    for item in a[i_min+1:i_max]:
        if item > 0:
            count += 1

    print(count)
```

Рисунок 4. Пример 2

```
D:\python\4\NCFU-Python_03-12-24-13-14>python Example2.py
8 2 1 9 19 2 -3 4 2 1 3 6 7 88 21
6

D:\python\4\NCFU-Python_03-12-24-13-14>python Example2.py
4 7 398 29 39 8 2 -4 12 22 3 5
4
```

Рисунок 5. Результат выполнения кода примера 2 с разными входными данными

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

import sys

if __name__ == '__main__':
    # Ввести кортеж одной строкой.
    A = tuple(map(int, input().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(s)
```

Рисунок 6. Пример 3

```
D:\python\4\NCFU-Python_03-12-24-13-14>python Example3.py
1 2 3 4 5 6 7 8 9 10
10

D:\python\4\NCFU-Python_03-12-24-13-14>python Example3.py
6 4 3 3 3 2 1 7 19 0
16
```

Рисунок 7. Результат выполнения кода примера 3

```
raindance@DESKTOP-33LH9V8 MINGW64 /d/python/4/NCFU-Python_03-12-24-13-14 (master
)
$ git add .

raindance@DESKTOP-33LH9V8 MINGW64 /d/python/4/NCFU-Python_03-12-24-13-14 (master
)
$ git commit -m "Examples done"
[master c7efb41] Examples done
 3 files changed, 75 insertions(+)
 create mode 100644 Example1.py
 create mode 100644 Example2.py
 create mode 100644 Example3.py
```

Рисунок 8. Фиксация изменений в репозитории

Индивидуальное задание 1: в списках U, D, V содержатся значения утренней, дневной и вечерней температуры соответственно за каждый день недели. Подсчитать среднее значение дневной температуры за каждый день.

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

import sys
from random import uniform


if __name__ == '__main__':
    morning = [uniform(0, 10) for i in range(7)]
    day = [uniform(4, 15) for i in range(7)]
    evening = [uniform(0, 7) for i in range(7)]

    average_temp = [round(((morning[i] + day[i] + evening[i]) / 3), 1) for i in range(len(morning))]

    print(average_temp)

```

Рисунок 9. Индивидуальное задание 1

```
D:\python\4\NCFU-Python_03-12-24-13-14>python Individual1.py
[4.4, 5.4, 8.5, 6.2, 5.6, 8.6, 4.9]

D:\python\4\NCFU-Python_03-12-24-13-14>python Individual1.py
[7.6, 7.4, 4.3, 7.1, 5.5, 7.5, 3.6]

D:\python\4\NCFU-Python_03-12-24-13-14>python Individual1.py
[6.9, 3.1, 5.2, 6.4, 5.5, 6.2, 6.5]
```

Рисунок 10. Результаты выполнения программы для индивидуального задания 1

Индивидуальное задание 2: В списке, состоящем из вещественных элементов, вычислить:

– Сумму отрицательных элементов

– Произведение элементов списка, расположенных между максимальным и минимальным элементами

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

from functools import reduce
from random import uniform


if __name__ == '__main__':
    my_list = [round(uniform(-10, 10), 2) for i in range(7)]

    # Отфильтровываем все значения, которые больше нуля, а затем функцией reduce складываем полученные значения
    Sum = reduce(lambda x, y: x + y, filter(lambda x: x <= 0, my_list))

    # Определить индексы минимального и максимального элементов.
    a_min = a_max = my_list[0]
    i_min = i_max = 0
    for i, item in enumerate(my_list):
        if item < a_min:
            i_min, a_min = i, item

        if item >= a_max:
            i_max, a_max = i, item

    # Проверить индексы и обменять их местами.
    if i_min > i_max:
        i_min, i_max = i_max, i_min

    Mult = reduce(lambda x, y: x * y, my_list[i_min:i_max+1])

    print("Исходный список: " + " ".join(map(lambda x: str(x), my_list)))
    print(f"Сумма всех отрицательных элементов списка: {Sum}")
    print(f"Произведение элементов списка между максимальным и минимальным элементами: {Mult}")

```

Рисунок 11. Индивидуальное задание 2

```
D:\python\4\NCFU-Python_03-12-24-13-14>python Individual2.py
Исходный список: 0.35 5.97 -5.74 -9.94 6.35 -3.38 1.13
Сумма всех отрицательных элементов списка: -19.06
Произведение элементов списка между максимальным и минимальным элементами: -63.11899999999999

D:\python\4\NCFU-Python_03-12-24-13-14>python Individual2.py
Исходный список: 6.27 4.55 -3.49 -1.1 -8.91 -0.25 -0.93
Сумма всех отрицательных элементов списка: -14.68
Произведение элементов списка между максимальным и минимальным элементами: -975.8313214650001

D:\python\4\NCFU-Python_03-12-24-13-14>python Individual2.py
Исходный список: -0.79 -8.77 7.94 5.77 -4.47 5.61 9.46
Сумма всех отрицательных элементов списка: -14.029999999999998
Произведение элементов списка между максимальным и минимальным элементами: 95314.16108289913
```

Рисунок 12. Результаты выполнения программы для индивидуального задания 2

Индивидуальное задание 3: имеется информация о количестве осадков, выпавших за каждый день января и за каждый день марта. Определить, в каком из этих месяцев выпало больше осадков.

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

from functools import reduce
from random import uniform


if __name__ == '__main__':
    january = tuple([round(uniform(0, 110), 2) for i in range(31)])
    march = tuple([round(uniform(0, 110), 2) for i in range(31)])
    variants = ["в январе", "в марте"]

    rain_for_january = reduce(lambda x, y: x + y, january)
    rain_for_march = reduce(lambda x, y: x + y, march)

    print(f"Осадков выпало в январе: {rain_for_january}")
    print(f"Осадков выпало в марте: {rain_for_march}")
    print(f"Осадков выпало больше {variants[rain_for_january < rain_for_march]}")

```

Рисунок 13. Индивидуальное задание 3

```
D:\python\4\NCFU-Python_03-12-24-13-14>python Individual3.py
Осадков выпало в январе: 1534.1799999999998
Осадков выпало в марте: 1624.2900000000002
Осадков выпало больше в марте

D:\python\4\NCFU-Python_03-12-24-13-14>python Individual3.py
Осадков выпало в январе: 1972.7900000000002
Осадков выпало в марте: 1960.1100000000004
Осадков выпало больше в январе
```

Рисунок 14. Результаты выполнения программы для индивидуального задания 3

Контрольные вопросы

1. Что такое списки? Это изменяемые упорядоченные коллекции элементов произвольных типов.

2. Создание списка: С помощью квадратных скобок my_list = [] или функции list().

3. Хранение в памяти: Списки хранятся как динамические массивы ссылок (указателей) на объекты, находящиеся в памяти.

4. Перебор элементов: С помощью цикла for (for item in my_list:) или цикла while по индексам.

5. Арифметические операции: Сложение (+) для конкатенации (объединения) и умножение на целое число (*) для повторения элементов.

6. Проверка наличия элемента: С помощью оператора in (например, if x in my_list:).

7. Число вхождений элемента: Метод .count(x).

8. Добавление элемента: .append(x) (в конец), .insert(i, x) (по индексу), .extend(seq) (расширение другим списком).

9. Сортировка: Метод .sort() (сортирует текущий список) или функция sorted() (возвращает новый отсортированный список).

10. Удаление элементов: Оператор del my_list[i], метод .pop(i) (удаляет и возвращает), метод .remove(x) (удаляет по значению), .clear() (очищает весь список).

11. Списковое включение (List Comprehension): Это краткий синтаксис для создания нового списка на основе существующего итерируемого объекта. Пример: [x*2 for x in range(5)].

12. Доступ срезами: Синтаксис список[start:stop:step], где start — начало, stop — конец (не включая), step — шаг.

13. Функции агрегации: len(), sum(), min(), max().

14. Создание копии: Методом .copy(), срезом [:] или конструктором list().

15. sorted vs sort: Метод list.sort() изменяет исходный список и возвращает None. Функция sorted(list) создает новый список, не меняя исходный.

16. Что такое кортежи? Это неизменяемые упорядоченные последовательности элементов.

17. Назначение: Хранение данных, которые не должны меняться (защита от записи), использование в качестве ключей словарей, экономия памяти по сравнению со списками.

18. Создание кортежей: Круглые скобки (1, 2), перечисление через запятую 1, 2 или функция tuple().

19. Доступ к элементам: По индексу в квадратных скобках tup[0].

20. Распаковка: Присвоение значений элементов кортежа сразу нескольким переменным (x, y = point).

21. Роль в множественном присваивании: Python упаковывает значения справа от = в кортеж, а затем распаковывает их в переменные слева (a, b = b, a).

22. Срезы: Так же, как в списках: tup[start:stop]. Возвращается новый кортеж.

23. Конкатенация и повторение: Операторы + (создает новый кортеж) и * (повторяет элементы).

24. Обход элементов: Циклом for item in tup:.

25. Проверка принадлежности: Оператором in.

26. Методы: У кортежей только два метода: .count() (количество вхождений) и .index() (индекс элемента).

27. Функции агрегации: Да, допустимо использовать len(), sum(), min(), max().

28. Кортеж через списковое включение: Напрямую нельзя (скобки () создают генератор). Нужно передать выражение-генератор в конструктор: tuple(x for x in range(5)).