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

Институт перспективной инженерии
Департамент цифровых, роботехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №4
дисциплины «Программирование на Python»
Вариант 7

Выполнила:
Еремина Татьяна Евгеньевна
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)

Проверил:
Воронкин Р.А., 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.

Практическая часть:

Исходный репозиторий: https://github.com/TL-hash/labs4

Пример 1. Вывести список A из 10 элементов, найти сумму элементов, меньших по модулю 5, и вывести её на экран.

Решение (1 способ):

1. Используя конструкцию list(map(int, input().split())), вводится целочисленный массив одной строкой.

2. Перебираются элементы и суммируются в случае, если они по модулю меньше 5.

Листинг программы примера:

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

import sys


if __name__ == '__main__':
    a = list(map(int, input().split()))
    if len(A) != 10:
        print("Неверный размер списка", file=sys.stderr)
        exit(1)

    s = 0
    for item in A:
        if abs(item) < 5:
            s += item

    print(s)
```

Результат выполнения:

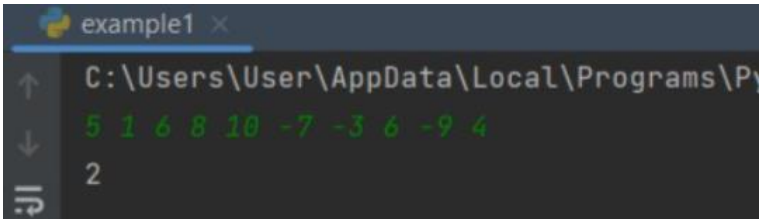

Рисунок 1 - Результат выполнения примера 1 (1 способ)

Решение (2 способ):

1. Используем списковые включения для поиска суммы элементов, которые по модулю меньше 5.

2. Для нахождения абсолютного значения целого числа используем функцию abs (для вещественных чисел лучше использовать функцию fabs пакета math)

Листинг программы примера:

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

import sys


if __name__ == '__main__':
    A = list(map(int, input().split()))
    if len(A) != 10:
        print("Неверный размер списка", file=sys.stderr)
        exit(1)

    s = sum([a for a in A if abs(a) < 5])
    print(s)
```

Результат выполнения:

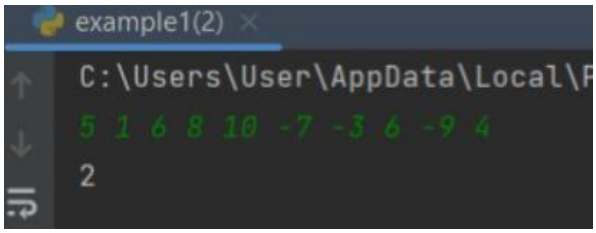

Рисунок 2 - Результат выполнения примера 1 (2 способ)

Пример 2. Написать программу, которая для целочисленного списка определяет, сколько положительных элементов располагается между его максимальным и минимальными элементами.

Решение:

1. Если индекс минимума больше индекса максимума необходимо поменять элементы местами, чтобы корректно выбрать диапазон между ними.

2. Если список пуст – выводится сообщение об ошибке и программа завершается.

Листинг программы примера:

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

import sys


if __name__ == "__main__":
    a = list(map(int, input().split()))
    if not a:
        print("Заданный список пуст", file=sys.stderr)
        exit(1)

    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

    if i_min > i_max:
        i_min, i_max = i_max, i_min

    count = 0
    for item in a[i_min+1:i_max]:
        if item > 0:
            count += 1

    print(count)
```

Результат выполнения:

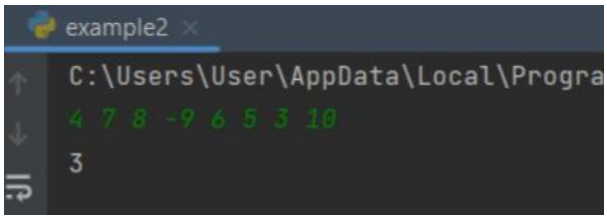

Рисунок 3 - Результат выполнения примера 2

Пример 3. Ввести кортеж A из 10 элементов, найти сумму элементов, меньших по модулю 5, и вывести её на экран. Использовать в программе вместо списков кортежи.

Решение (1 способ): для ввода целочисленного массива одной строкой используется конструкция tuple(map(int, input().split())).

Листинг программы:

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

import sys


if __name__ == "__main__":
    A = tuple(map(int, input().split()))
    if len(A) != 10:
        print("Неверный размер кортежа", file=sys.stderr)
        exit(1)

    s = 0
    for item in A:
        if abs(item) < 5:
            s += item

    print(s)
```

Результат выполнения:

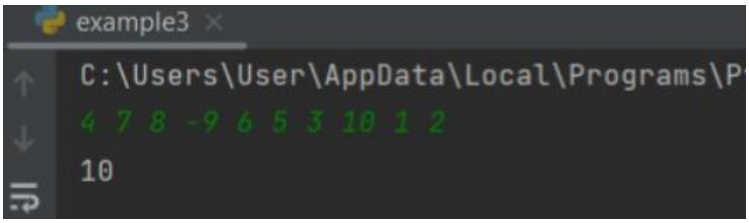

Рисунок 4 - Результат выполнения примера 3 (1 способ)

Решение 2 способом, используя списковые включения.

Листинг программы:

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

import sys

if __name__ == "__main__":
    A = tuple(map(int, input().split()))
    if len(A) != 10:
        print("Неверный размер кортежа", file=sys.stderr)
        exit(1)

    s = sum(a for a in A if abs(a) < 5)
    print(s)
```

Результат выполнения:

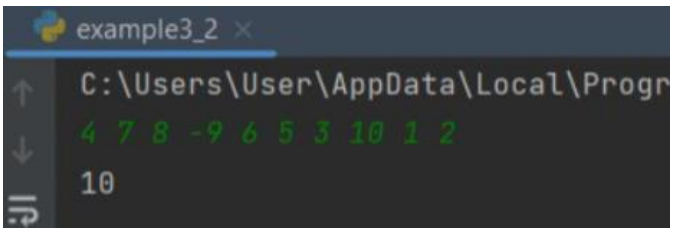

Рисунок 5 - Результат выполнения примера 3 (2 способ)

Задание 1. Составить программу с использованием одномерных массивов для решения задачи. Решить индивидуальное задание как с использованием циклов, так и с использованием List Comprehensions.

Вывести список A из 10 элементов, найти произведение отрицательных элементов и вывести его на экран.

Решение (1 способ):

1. Используя конструкцию list(map(int, input().split())), введём целочисленный массив одной строкой.
2. Проходим по каждому элементу списка A циклом for.
3. Если элемент меньше нуля (т.е. отрицательный), умножаем его на p.

Листинг программы:

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

import sys


if __name__ == '__main__':
    a = list(map(int, input().split()))
    if len(a) != 10:
        print("Неверный размер списка", file=sys.stderr)
        exit(1)

    p = 1
    for item in a:
        if item < 0:
            p *= item

    print(p)
```

Результат выполнения:

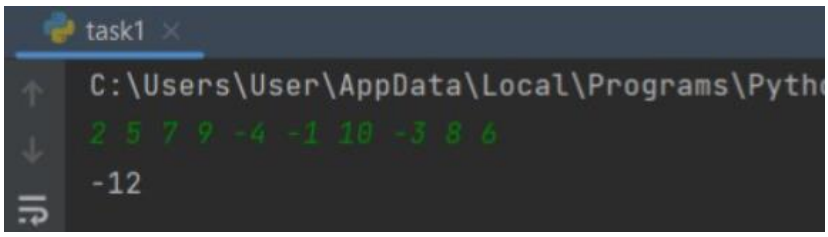

Рисунок 6 - Результат выполнения задания 1 (1способ)

Решение (2 способ):

1. Используя конструкцию list(map(int, input().split())), введём целочисленный массив одной строкой.

2. Используем списковые включения [item for item in A if item < 0], которые проходят по всем элементам списка A, берут только те, которые меньше нуля (т.е. отрицательные), создают новый список из этих элементов.

Листинг программы:

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

import sys
import math


if __name__ == '__main__':
    a = list(map(int, input().split()))
    if len(a) != 10:
        print("Неверный размер списка", file=sys.stderr)
        exit(1)

    p = math.prod([item for item in a if item < 0])
    print(p)
```

Результат выполнения:

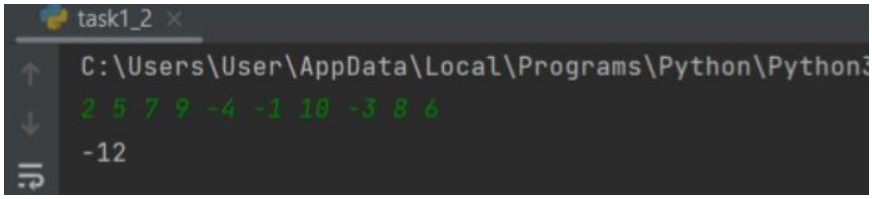

Рисунок 7 - Результат выполнения задания 1 (2 способ)

Задание 2. Составить программу с использованием одномерных массивов для решения задачи на переупорядочивание элементов списка. Для сортировки допускается использовать метод sort с заданным параметром key и объединение нескольких списков.

В списке, состоящем из вещественных элементов, вычислить:

1. номер минимального элемента списка;

2. сумму элементов списка, расположенных между первым и вторым отрицательными элементами.

Преобразовать список таким образом, чтобы сначала располагались все элементы, модуль которых не превышает 1, а потом – все остальные.

Решение:

1. Используя конструкцию list(map(int, input().split())), введём целочисленный массив одной строкой.

2. Поиск номера минимального элемента производится с помощью i_min = A.index(min(A)), где min(A) – находит наименьшее значение в списке, A.index(...) – возвращает индекс первого вхождения этого значения.

3. List comprehension для поиска индексов отрицательных элементов собирает все индексы, где элемент меньше нуля, enumerate(A) – даёт пары (индекс, значение).

4. Если менее двух отрицательных чисел, то сумма = 0, выводится сообщение. Если два или более, то берётся срез списка между первым и вторым отрицательными (не включая их).

5. Преобразование списка: low_abs – все элементы, у которых модуль ≤ 1, high_abs – все элементы с модулем > 1.

6. Альтернативный способ преобразования списка: A.sort(key=lambda x: abs(x) > 1) – сортирует список на месте. Элементы с abs(x) <= 1 – ключ False, иначе True.

Листинг программы:

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

import sys


if __name__ == '__main__':
    a = list(map(float, input().split()))
    if not a:
        print("Заданный список пуст", file=sys.stderr)
```

```
        exit(1)

    i_min = a.index(min(A))
    print(f"Номер минимального элемента: {i_min}")

    i_neg = [i for i, item in enumerate(A) if item < 0]

    if len(i_neg) < 2:
        sum_between = 0
        print(f"Недостаточно отрицательных элементов для вычисления суммы")
    else:
        first_neg = i_neg[0]
        second_neg = i_neg[1]
        sum_between = sum(a[first_neg+1:second_neg])
        print(f"Сумма между первым и вторым отрицательными элементами: {sum_between}")

    low_abs = [item for item in a if abs(item) <= 1]
    high_abs = [item for item in a if abs(item) > 1]
    sorted_list = low_abs + high_abs
    #a.sort(key=lambda x: abs(x) > 1)

    print(f"Преобразованный список: {sorted_list}")
    #print(f"Преобразованный список: {a}")
```

Результат выполнения:

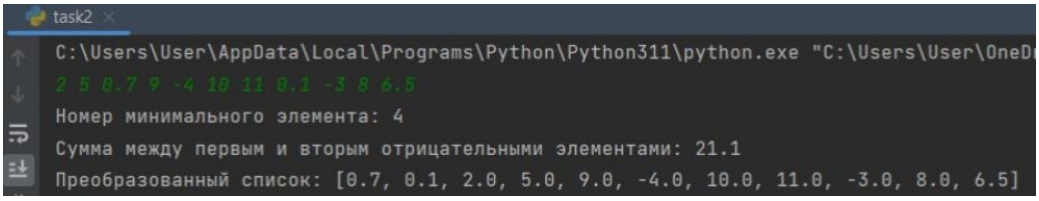

Рисунок 8 - Результат выполнения задания 2

Задание 3. Использовать кортежи для решения задачи.

Дан кортеж целых чисел. Если в нем есть хотя бы одна пара соседних четных чисел, то напечатать все элементы, предшествующие элементам последней из таких пар.

Решение:

1. Для ввода целочисленного массива одной строкой используется конструкция tuple(map(int, input().split())).

2. Перебираем все соседние пары, проверяем на чётность. Если оба числа чётные – запоминаем индекс первого из них (i).

3. Если last_pos остался -1, значит пар не найдено, выводим сообщение. Иначе: A[:last_pos] – срез от начала до индекса last_pos (не включая его) это все элементы до первого числа последней пары.

Листинг программы:

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

import sys

if __name__ == "__main__":
    a = tuple(map(int, input().split()))
    if len(a) < 2:
        print("Кортеж должен содержать хотя бы 2 элемента", file=sys.stderr)
        exit(1)

    last_pos = -1

    for i, (x, y) in enumerate(zip(a, a[1:])):
        if x % 2 == 0 and y % 2 == 0:
            last_pos = i

    if last_pos == -1:
        print("В кортеже нет пар соседних четных элементов")
    else:
        print("Элементы перед последней парой четных:")
        for elem in a[:last_pos]:
            print(elem, end=" ")
```

Результат выполнения:

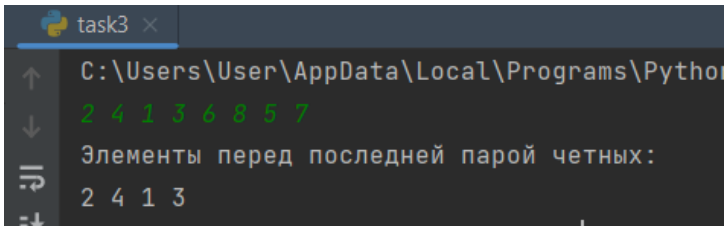

Рисунок 9 - Результат выполнения задания 3

Контрольные вопросы:

1. Что такое списки в языке Python?

Список – это изменяемая упорядоченная коллекция объектов произвольных типов, поддерживающая индексацию и срезы.

2. Как осуществляется создание списка в Python?

Список создаётся:

- с помощью квадратных скобок: lst = [1, 2, 3],

- функцией list(): lst = list((1, 2, 3)),
- генератором списков: lst = [x for x in range(5)].

3. Как организовано хранение списков в оперативной памяти?

Список хранится как массив ссылок на объекты. Сам список содержит указатели (адреса) на реальные объекты, размещённые в куче (heap).

4. Каким образом можно перебрать все элементы списка?

- Через for по элементам: for x in lst: ...
- Через for по индексам: for i in range(len(lst)): ...
- С помощью enumerate(): for i, x in enumerate(lst): ...

5. Какие существуют арифметические операции со списками?

- Конкатенация: lst1 + lst2
- Повторение: lst * n

6. Как проверить, есть ли элемент в списке?

С помощью оператора in: if x in lst:… Возвращает True, если элемент присутствует, иначе False.

7. Как определить число вхождений заданного элемента в списке?

С помощью метода .count(): n = lst.count(x)

8. Как осуществляется добавление (вставка) элемента в список?

- В конец: lst.append(x)
- По индексу: lst.insert(i, x)
- Несколько элементов: lst.extend(iterable)

9. Как выполнить сортировку списка?

- На месте: lst.sort() (можно с key, reverse)
- Создать новый: new_lst = sorted(lst)

10. Как удалить один или несколько элементов из списка?

- По значению: lst.remove(x) (первое вхождение)
- По индексу: lst.pop(i) (возвращает удалённый элемент)
- По срезу: del lst[i:j]
- Очистить: lst.clear()

11. Что такое списковое включение и как с его помощью осуществлять обработку списков?

Списковое включение (list comprehension) – краткий синтаксис для создания нового списка: new = [выражение for элемент in последовательность if условие]. Пример: [x**2 for x in range(5) if x % 2 == 0].

12. Как осуществляется доступ к элементам списков с помощью срезов?

Срез: lst[start:stop:step]

- start – начало (включительно),
- stop – конец (не включается),
- step – шаг (по умолчанию 1).

Пример: lst[1:4], lst[::-1] (реверс).

13. Какие существуют функции агрегации для работы со списками?

len(), sum(), min(), max(), all(), any() – работают с любыми итерируемыми объектами, включая списки.

14. Как создать копию списка?

Поверхностная: lst.copy() или lst[:]

Глубокая (при вложенных объектах): copy.deepcopy(lst)

15. В чём отличие sorted() от метода sort()?

- list.sort() – изменяет исходный список, возвращает None.
- sorted() – возвращает новый отсортированный список, исходный не меняет.

Обе принимают key и reverse.

16. Что такое кортежи в языке Python?

Кортеж – это неизменяемая упорядоченная коллекция объектов произвольных типов.

17. Каково назначение кортежей?

Для хранения неизменяемых данных, защиты от случайного изменения, использования в качестве ключей словаря, возврата нескольких значений из функции.

18. Как осуществляется создание кортежей?

- Через запятые: t = 1, 2, 3
- В скобках: t = (1, 2, 3)
- С одним элементом: t = (x,) (обязательна запятая)
- Через tuple(): t = tuple([1, 2, 3])

19. Как осуществляется доступ к элементам кортежа?

Так же, как и у списков: по индексу t[i], через срезы t[i:j], итерацией.

20. Зачем нужна распаковка кортежа?

Для присвоения значений элементов кортежа отдельным переменным: a, b = (1, 2).

21. Какую роль играют кортежи в множественном присваивании?

Python автоматически упаковывает правую часть в кортеж, а затем распаковывает его в переменные слева: x, y = 10, 20 # эквивалентно x, y = (10, 20)

22. Как выбрать элементы кортежа с помощью среза?

Точно так же, как у списков: t[1:4], t[::-1] и т.д., результат – новый кортеж.

23. Как выполняется конкатенация и повторение кортежей?

- Конкатенация: t1 + t2
- Повторение: t * n

Результат – новый кортеж.

24. Как выполняется обход элементов кортежа?

Через цикл for: for item in t: .../ for i, item in enumerate(t): ...

25. Как проверить принадлежность элемента кортежу?

Оператором in: if x in t: ...

26. Какие методы работы с кортежами Вам известны?

Только два:

- .count(x) – число вхождений,
- .index(x) – индекс первого вхождения.

(Кортежи неизменяемы, поэтому методов мало).

27. Допустимо ли использование функций агрегации (len(), sum() и т.д.) с кортежами?

Да, кортежи – итерируемые объекты, поэтому len(), sum(), min(), max(), all(), any() работают с ними так же, как со списками.

28. Как создать кортеж с помощью спискового включения?

В Python списковое включение (list comprehension) создаёт список, а не кортеж. Чтобы получить кортеж, нужно обернуть результат в tuple(): t = tuple(x for x in range(5) if x % 2 == 0).

Вывод: были приобретены навыки по работе со списками и кортежами при написании программ с помощью языка программирования Python версии 3.x, реализованы алгоритмы обработки одномерных массивов с использованием циклов и списковых включений, выполнены поиск, фильтрация, суммирование и переупорядочивание элементов.