**РЕСПУБЛИКА БЕЛАРУСЬ**

**КОМИТЕТ ПО ОБРАЗОВАНИЮ МИНГОРИСПОЛКОМА**

**учреждение образования**

**МИНСКИЙ ГОСУДАРСТВЕННЫЙ КОЛЛЕДЖ ЦИФРОВЫХ**

**ТЕХНОЛОГИИ**

**Группа 81ТП**

**Отчет**

**по лабораторной работе № 6**

**“**Разработка и отладка алгоритмов работы с списками, кортежами, множествами**”**

Учебный предмет

«Инструментальное программное обеспечение»

**Исполнитель: Житкевич М.Д.**

**Руководитель: Бровка Д.С.**

Минск, 2024

Цели:

1. Ознакомление с основами работы с коллекциями данных в Python: списками, кортежами и множествами.
2. Изучение методов и операций, применяемых к данным структурам, для эффективного управления данными.
3. Развитие практических навыков разработки и оптимизации алгоритмов, работающих с коллекциями.

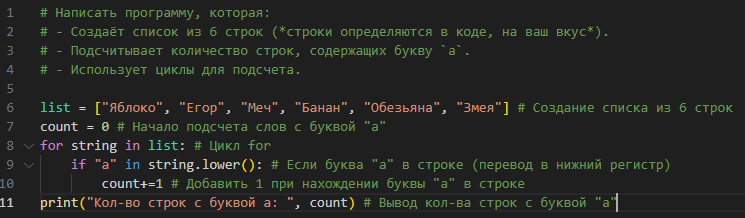
Задачи:

1. Изучить основные методы и операции, доступные для списков, кортежей и множеств.
2. Реализовать алгоритмы для добавления, удаления, сортировки и поиска элементов в структурах данных.
3. Научиться оптимизировать код для повышения производительности работы с коллекциями.
4. Выполнить практические задачи, выполняя операции над списками, кортежами и множествами для закрепления теоретических знаний.

Вариант 1

1. Написать программу, которая:

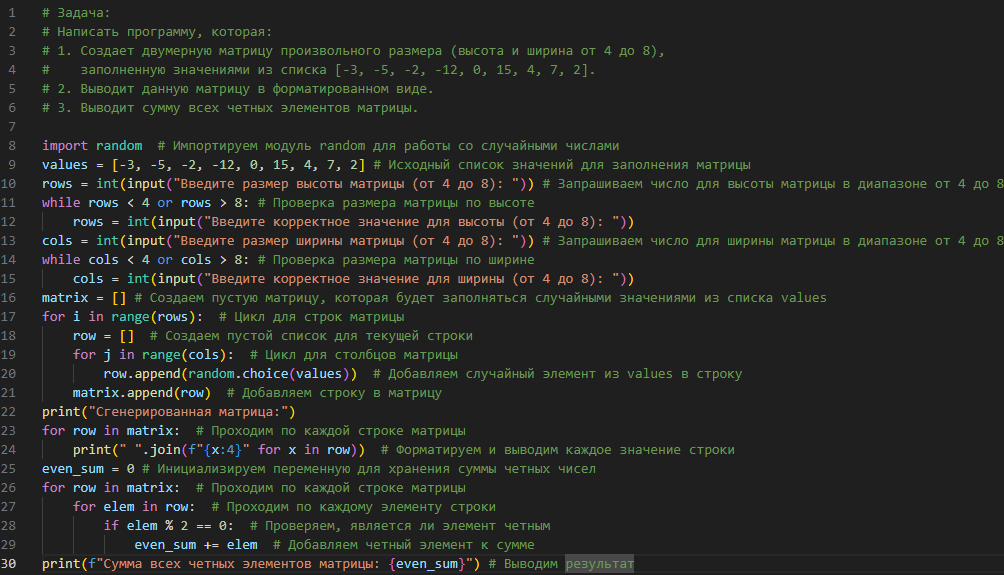
* Создаёт список из 6 строк (*строки определяются в коде, на ваш вкус*).
* Подсчитывает количество строк, содержащих букву a.
* Использует циклы для подсчета.

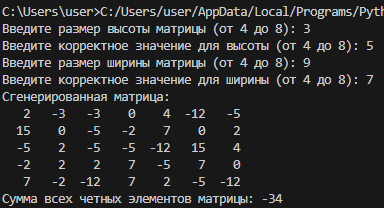




1. Написать программу, которая:

* Создает двумерную матрицу произвольного размера от 4 до 8 во высоте и ширине, заполненную значениями из списка **[-3, -5, -2, -12, 0, 15, 4, 7, 2]**
* Выводит данную матрицу в форматированном виде.
* Выводит сумму всех четных элементов

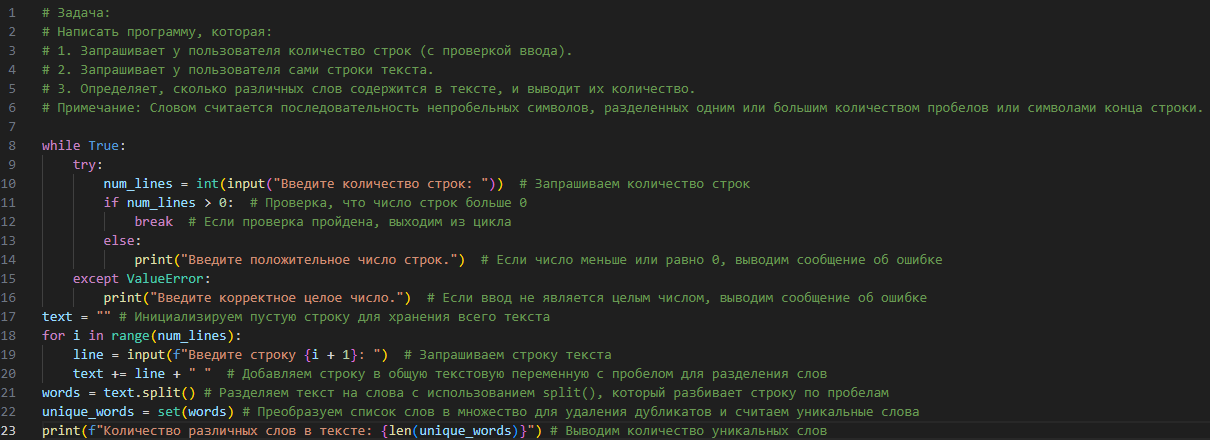


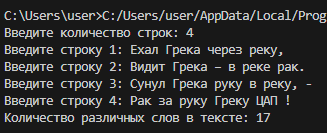


1. Написать программу, которая:

* Запрашивает у пользователя количество строк. (*должно быть проверка*)
* Затем сами строки.
* Определяет, сколько различных слов содержится в этом тексте, и выводит из количество

*Словом считается последовательность непробельных символов идущих подряд, слова разделены одним или большим числом пробелов или символами конца строки.*



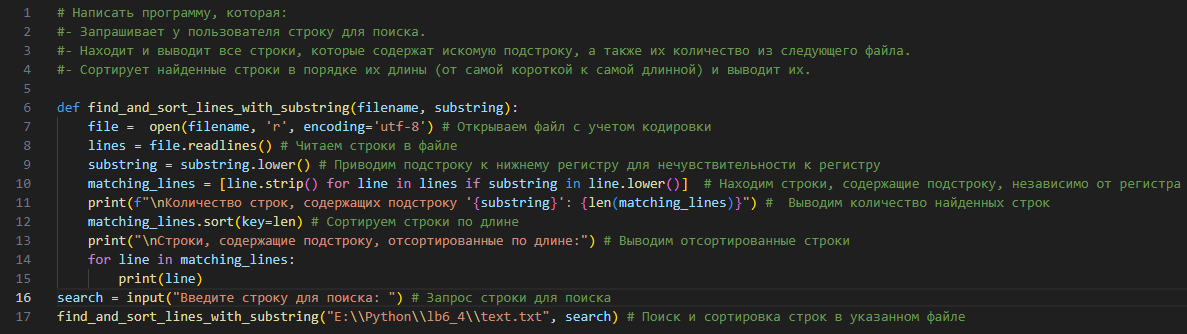


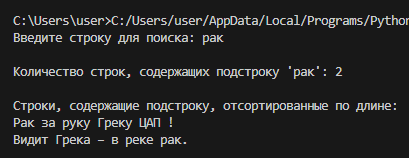
1. Написать программу, которая:

* Запрашивает у пользователя строку для поиска.
* Находит и выводит все строки, которые содержат искомую подстроку, а также их количество из следующего файла.

[text.txt](https://prod-files-secure.s3.us-west-2.amazonaws.com/d9fc6719-e1f9-49a0-8e26-8e1860bb2010/5284659f-3402-4df2-94b5-ae272f5b7c15/text.txt)

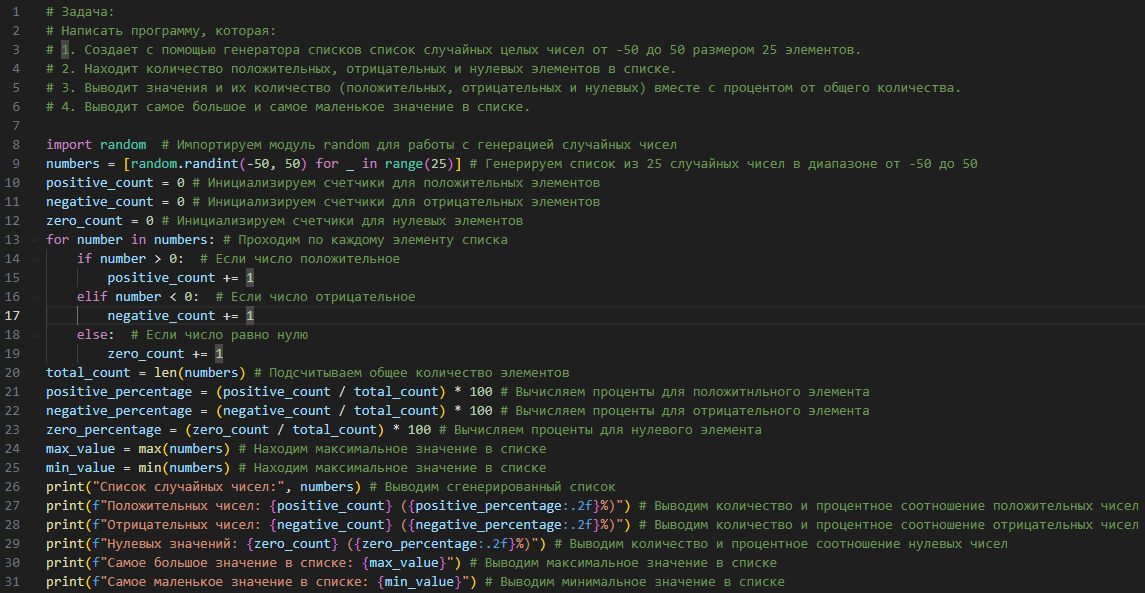
* Сортирует найденные строки в порядке их длины (от самой короткой к самой длинной) и выводит их.

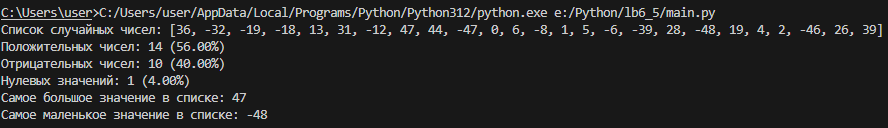




1. Написать программу, которая:

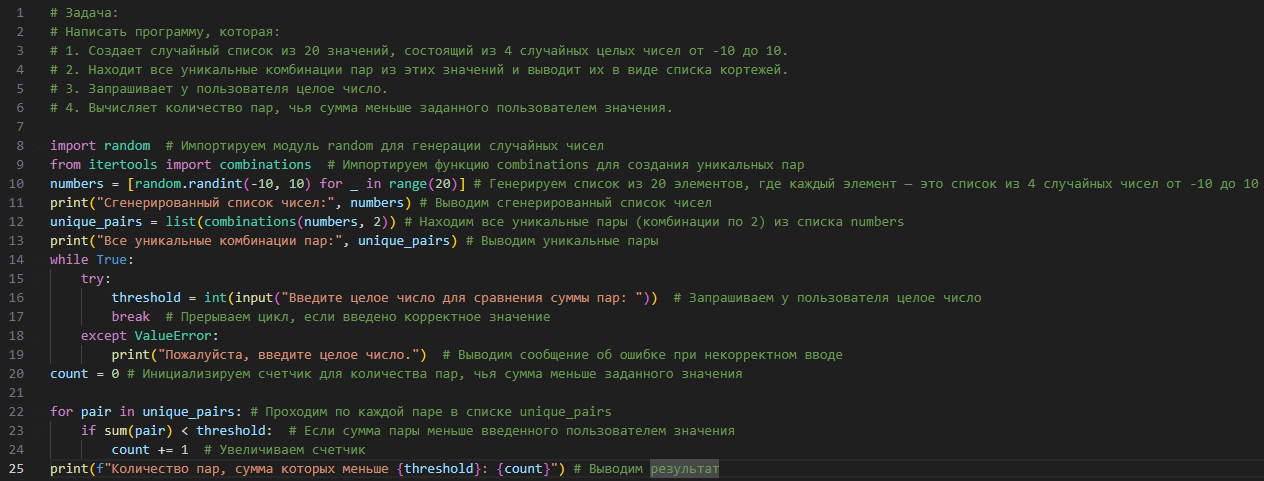
* Создаёт с помощью генератора списков, список случайных целых чисел от **-50** до **50** размером **25** элементов.
* Находит количество положительных, отрицательных и нулевых элементов в списке.
* Выводит значения и их (*положительных, отрицательных и нулевых*) количество вместе с процентом от общего количества.
* Выводит самое большое и самое маленькое значение

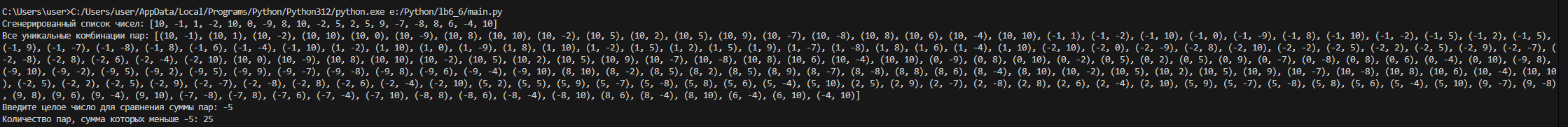




1. Написать программу, которая:

* Создаёт случайный список из **20** значений по **4** случайных целых чисел от **-10** до **10**.
* Находит все уникальные комбинации пар из этих значений и выводит их в виде списка кортежей.
* Пользователь вводит целое число.
* Вычисляет количество пар, чья сумма меньше заданного пользователем значения.





Ответы на контрольные вопросы:

* 1. **Список** — это изменяемая коллекция элементов в Python, которая может содержать элементы разных типов (например, числа, строки и другие списки). Список создается с помощью квадратных скобок [], а элементы разделяются запятыми.
  2. **Изменяемость:** Списки изменяемы (можно добавлять, удалять, изменять элементы), а кортежи неизменяемы (после создания элементы изменить нельзя).

**Синтаксис:** Списки создаются с помощью квадратных скобок [], а кортежи — с помощью круглых скобок ().

**Производительность:** Кортежи обычно работают быстрее, чем списки, так как их элементы неизменяемы.

**Использование:** Кортежи часто используются для хранения постоянных наборов данных, а списки — для данных, которые могут изменяться.

* 1. **Преимущества множеств:**

**Уникальность элементов:** Множества автоматически удаляют дубликаты, что удобно для хранения уникальных значений.

**Быстрые операции:** Операции поиска и удаления элементов в множествах происходят быстрее, чем в списках.

**Недостатки множеств:**

**Неупорядоченность:** Множества не хранят порядок элементов, поэтому нельзя обращаться к элементу по индексу.

**Неизменяемость элементов:** Множества могут содержать только изменяемые элементы, такие как строки, числа и кортежи.

* 1. Элементы списка можно получить по индексу. Индексация начинается с 0 для первого элемента.

list = [10, 20, 30, 40, 50]

print(list[0]) # Вывод: 10 (первый элемент)

print(list[-1]) # Вывод: 50 (последний элемент)

* 1. **append(item)** — добавляет элемент в конец списка.

**insert(index, item)** — добавляет элемент в определенную позицию.

**extend(iterable)** — добавляет несколько элементов из итерируемого объекта (например, списка) в конец списка.

* 1. **remove(value)** — удаляет первый найденный элемент с указанным значением.

**pop(index)** — удаляет элемент по указанному индексу и возвращает его. Если индекс не указан, удаляет последний элемент.

**del** — удаляет элемент или срез по индексу.

**clear()** — очищает весь список.

* 1. **Индекс** — это числовое значение, которое указывает на позицию элемента в списке. Индексация начинается с 0 для первого элемента и идет до n-1 для списка длиной n. Индексы также могут быть отрицательными: -1 соответствует последнему элементу, -2 — предпоследнему и так далее.
  2. **Срезы** позволяют получить подсписок, определяя начальный, конечный индекс и шаг. Синтаксис: list[start:end:step]. Начальный индекс включается в срез, а конечный — нет.

Пример:

list = [1, 2, 3, 4, 5, 6]

print(list[1:4]) # [2, 3, 4] - от индекса 1 до 4 (не включая 4)

print(list[:3]) # [1, 2, 3] - от начала до индекса 3 (не включая 3)

print(list[::2]) # [1, 3, 5] - каждый второй элемент

print(list[::-1]) # [6, 5, 4, 3, 2, 1] - список в обратном порядке

* 1. **Оператором +** — создает новый список.

**Методом extend()** — добавляет элементы второго списка в первый.

* 1. **sort()** — сортирует список на месте, изменяя его. Принимает параметр reverse=True для сортировки по убыванию.

**sorted()** — возвращает новый отсортированный список, не изменяя оригинал.

* 1. **Генераторы списков** (или **list comprehensions**) — это компактный способ создания списков на основе существующих списков или других итерируемых объектов, используя синтаксис, который позволяет комбинировать выражения, циклы и условия. Они обеспечивают краткий и читабельный способ построения списка.
  2. Генераторы списков создают новый список, выполняя операцию на каждом элементе из итерируемого объекта.
  3. Фильтрация данных в генераторах списков достигается добавлением условия в конце выражения. Только элементы, соответствующие этому условию, включаются в новый список.
  4. Проверить наличие элемента можно с помощью оператора in.
  5. Метод **index()** возвращает индекс первого вхождения указанного элемента. Если элемент не найден, возникает ошибка ValueError.
  6. **Кортеж** — это неизменяемая коллекция элементов, аналогичная списку. Кортежи быстрее обрабатываются, занимают меньше памяти и защищены от изменений, что делает их подходящими для хранения данных, которые не должны изменяться.
  7. Преобразовать список в кортеж можно с помощью функции tuple(), а кортеж в список — с помощью list().
  8. Кортежи поддерживают только методы count() и index() из-за своей неизменяемости:

**count(value)** — возвращает количество вхождений указанного значения.

**index(value)** — возвращает индекс первого вхождения указанного значения.

* 1. Основное различие заключается в том, что **списки изменяемы**, то есть можно добавлять, удалять и изменять их элементы. **Кортежи неизменяемы**: после создания изменить их элементы нельзя.
  2. **Множество (set)** создается с помощью фигурных скобок {} или функции set(). Множества содержат только уникальные элементы и не поддерживают порядок.
  3. Множества поддерживают различные операции, такие как:

**add(element)** — добавляет элемент в множество.

**remove(element)** — удаляет элемент, если он присутствует, иначе вызывает ошибку.

**discard(element)** — удаляет элемент, если он присутствует, но не вызывает ошибку, если его нет.

**pop()** — удаляет и возвращает случайный элемент.

**clear()** — очищает множество.

**union(other\_set)** — объединяет два множества.

**intersection(other\_set)** — находит общие элементы двух множеств.

**difference(other\_set)** — возвращает элементы, которые есть в одном множестве, но отсутствуют в другом.

**symmetric\_difference(other\_set)** — возвращает элементы, которые есть в одном множестве или другом, но не в обоих.

* 1. Метод **issubset(other\_set)** проверяет, является ли одно множество подмножеством другого.
  2. Для объединения множеств используется метод **union()** или оператор |, для пересечения — метод **intersection()** или оператор &.
  3. Преобразовав список во множество, можно проверить, равны ли длины множества и списка. Если они совпадают, то все элементы списка уникальны.
  4. Цикл for позволяет перебрать каждый элемент списка.
  5. Цикл **while** выполняется, пока условие истинно, и его используют, когда заранее неизвестно количество итераций. В **for** цикле заранее известен диапазон значений или итерируемый объект.
  6. Для изменения элементов можно использовать for цикл с индексом, чтобы изменять элементы по их позициям.
  7. Чтобы избежать дублирования, перед добавлением элемента можно проверить его наличие или использовать множество, так как в нем автоматически удаляются повторяющиеся значения.
  8. Для обработки ошибок, возникающих при неправильном индексе, можно использовать конструкцию **try-except**.
  9. **map(function, iterable)** применяет функцию к каждому элементу итерируемого объекта.

**filter(function, iterable)** возвращает элементы, которые удовлетворяют условию.

* 1. Вложенные списки — это списки внутри других списков. Доступ к элементам осуществляется с помощью двойного индекса.
  2. Функция len возвращает количество элементов.
  3. Синтаксис генератора множества аналогичен генератору списка, но использует фигурные скобки.
  4. Используйте условие if not для проверки на пустоту.
  5. Списки сравниваются на равенство, а для множеств доступны операции подмножества, пересечения, объединения и разности.
  6. Используйте метод .copy() или функции list() и set().
  7. Возникнет ошибка IndexError, которую можно обработать с помощью try-except.
  8. Добавьте условие в конце выражения генератора списка.
  9. Используйте метод .reverse() или sorted() с параметром reverse=True.
  10. Используйте функции max() и min().

Вывод: я ознакомился с основами работы с коллекциями данных в Python: списками, кортежами и множествами. Изучил методы и операции, применяемые к данным структурам, для эффективного управления данными. Развил практические навыки разработки и оптимизации алгоритмов, работающих с коллекциями.