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

Институт Перспективной инженерии
Департамент цифровых и робототехнических систем и электроники

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

Выполнил: Мендеш Пашкоал Педру
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 А. доцент факультета 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: Цель работы: приобретение навыков по работе со списками и кортежами при написании программ с помощью языка программирования Python версии 3.x.

Теоретическая часть

1. Списки (List)

- Определение: Изменяемые коллекции для хранения элементов разных типов
- Создание: my_list = [1, 2, 3] или list()
- Особенности: Поддерживают индексацию, срезы, изменяемый размер

2. Кортежи (Tuple)

- Определение: Неизменяемые коллекции, аналогичные спискам
- Создание: my_tuple = (1, 2, 3) или tuple()
- Особенности: Экономнее памяти, быстрее списков, для постоянных данных

3. Основные операции

- Доступ: list[индекс], list[срез]
- Поиск: элемент in список, list.index(), list.count()
- Изменение: append(), insert(), pop(), remove(), sort()

4. List Comprehensions

- Определение: Лаконичный синтаксис для создания списков
- Пример: [x**2 for x in range(10)]
- Применение: Замена map() и filter()

5. Ключевые отличия

- Списки: Изменяемые, [], много методов

- Кортежи: Неизменяемые, (), меньше методов, быстрее

Методика выполнения работы

- Изучение теоретического материала
- Создание репозитория GitHub: https://github.com/Pascoalpm/Lab4-python-listas-tuplas
- Клонирование репозитория
- Настройка .gitignore для PyCharm
- Организация ветвления git-flow

1. Запуск примеров и выполнение задачи в PyCharm.

1.1. Примеры

1.1.1. Пример 1

```
#!/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)
```

```
Run    exemplo2 ×
C:\Users\HP\AppData\Local\Programs\Python\Python313\python.exe
1 2 3 4  5 6 7 8 9 10
8

Process finished with exit code 0
```

Рисунок 1. Код (пример 2), выполненный в PyCharm

1.1.1. Пример 2

```
#!/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)
```

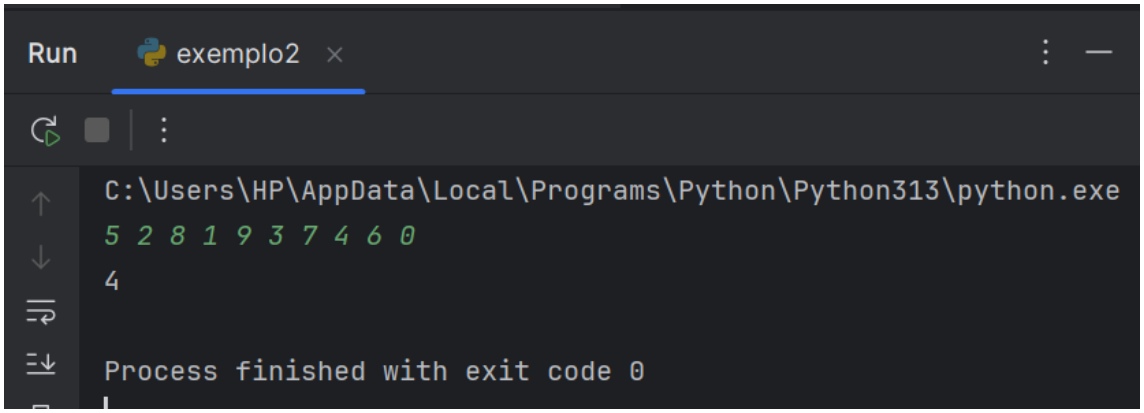

Рисунок 2. Код (пример 2), выполненный в PyCharm

1.2. Индивидуальные задания

1.2.1. Задание 1

Составить программу с использованием одномерных массивов для решения задачи. Номер варианта необходимо получить у преподавателя. Решить индивидуальное задание как с использованием циклов, так и с использованием List Comprehensions.

3. Ввести список A из 10 элементов, найти наименьший элемент и переставить его с последним элементом. Преобразованный список вывести.

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

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")
        exit(1)

    # Найти наименьший элемент и его индекс
    a_min = A[0]
    i_min = 0

    for i, item in enumerate(A):
        if item < a_min:
            i_min, a_min = i, item

    # Переставить наименьший элемент с последним
    A[i_min], A[-1] = A[-1], A[i_min]

    print("Преобразованный список:", A)
```

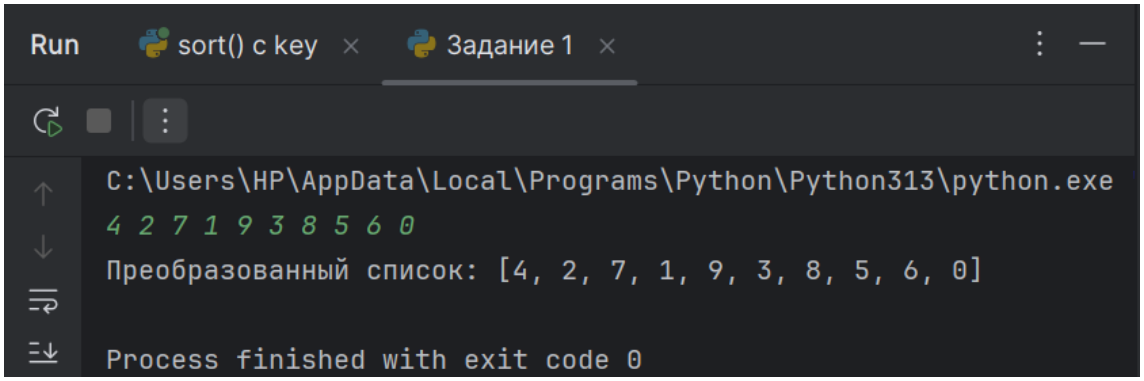

Рисунок 3. Код, выполненный в PyCharm

List Comprehensions

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

if __name__ == '__main__':
    # Ввести список одной строкой
    A = list(map(int, input().split()))

    if len(A) != 10:
        print("Неверный размер списка")
        exit(1)

    # Найти индекс минимального элемента usando List Comprehension +
функции агрегации
    min_value = min(A)  # функция min() из примеров
    i_min = A.index(min_value)  # метод index() из примеров
1
    # Переставить элементы
    A[i_min], A[-1] = A[-1], A[i_min]

    print("Преобразованный список:", A)
```

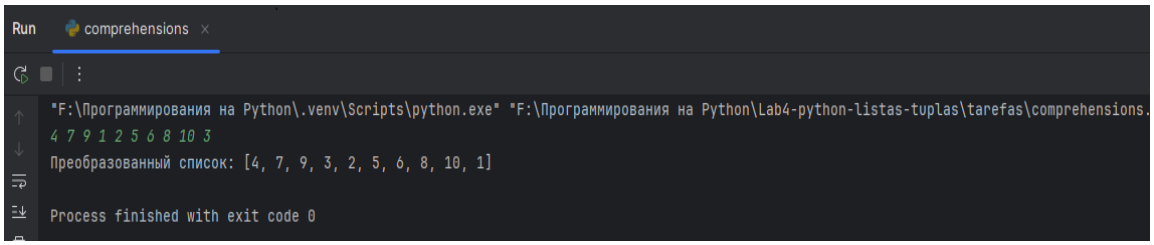

Рисунок 4. Код, выполненный в PyCharm

1.2.2. 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

(https://docs.python.org/3/howto/sorting.html) и объединение нескольких списков.

Номер варианта необходимо получить у преподавателя.

метод sort() с ключом (key)

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

import sys

if __name__ == '__main__':
```

```
    # Ввести список одной строкой
    A = list(map(int, input().split()))

    if not A:
        print("Заданный список пуст", file=sys.stderr)
        exit(1)

    print("Исходный список:", A)

    # 1. Произведение элементов с четными номерами
    product_even = 1
    for i in range(len(A)):
        if i % 2 == 0:
            product_even *= A[i]
    print("1. Произведение элементов с четными номерами:",
product_even)

    # 2. Сумма между нулями
    if 0 in A:
        first_zero = A.index(0)
        last_zero = first_zero
        for i in range(len(A)):
            if A[i] == 0:
                last_zero = i

        if last_zero > first_zero:
            sum_between = 0
            for i in range(first_zero + 1, last_zero):
                sum_between += A[i]
            print("2. Сумма между нулями:", sum_between)
        else:
            print("2. Между нулями нет элементов")
    else:
        print("2. В списке нет нулевых элементов")


    # 3. ПРЕОБРАЗОВАНИЕ - МЕТОД 1: sort() с key
    def sort_key(x):
        return 1 if x < 0 else 0  # отрицательные -> 1, остальные -> 0


    A.sort(key=sort_key)
    print("3. Преобразованный список (sort+key):", A)
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\HP\Documents\2 ano\Программирования на Python\Lab 4\sort() с key.p
4 9 8 23 56
Исходный список: [4, 9, 8, 23, 56]
1. Произведение элементов с четными номерами: 1792
2. В списке нет нулевых элементов
3. Преобразованный список (sort+key): [4, 9, 8, 23, 56]

Process finished with exit code 0
```

Рисунок 5. Код, выполненный в PyCharm

метод объединения списков

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

import sys
```

```
if __name__ == '__main__':
    # Ввести список одной строкой
    A = list(map(int, input().split()))

    if not A:
        print("Заданный список пуст", file=sys.stderr)
        exit(1)

    print("Исходный список:", A)

    # 1. Произведение элементов с четными номерами
    product_even = 1
    for i in range(0, len(A), 2):
        product_even *= A[i]
    print("1. Произведение элементов с четными номерами:",
product_even)

    # 2. Сумма между нулями
    if 0 in A:
        first_zero = A.index(0)
        last_zero = len(A) - 1 - A[::-1].index(0)

        if first_zero < last_zero:
            sum_between = sum(A[first_zero + 1:last_zero])
            print("2. Сумма между нулями:", sum_between)
        else:
            print("2. Между нулями нет элементов")
    else:
        print("2. В списке нет нулевых элементов")

    # 3. ПРЕОБРАЗОВАНИЕ - МЕТОД 2: объединение списков
    positive = []
    negative = []

    for item in A:
        if item >= 0:  # 0 считаем положительным
            positive.append(item)
        else:
            negative.append(item)

    # ОБЪЕДИНЕНИЕ списков операцией +
    A_transformed = positive + negative
    print("3. Преобразованный список (объединение):", A_transformed)
```

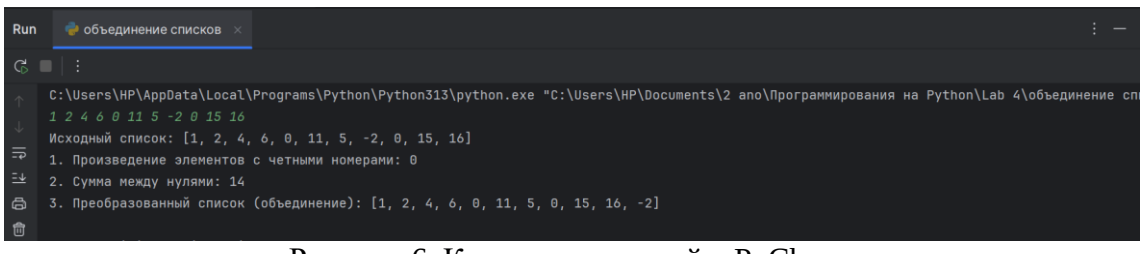

Рисунок 6. Код, выполненный в PyCharm

Задание 3

3. Известны оценки по геометрии каждого из 24 учеников класса. В начале списка перечислены все пятерки, затем все остальные оценки.

Сколько учеников имеет по геометрии оценку «5»? Условный оператор не использовать.

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

if __name__ == '__main__':
    # Ввести оценки
    A = tuple(map(int, input().split()))

    if len(A) != 24:
        print("Неверное количество оценок")
        exit(1)

    # Подсчитать пятерки - метод count() из примеров с кортежами
    count_5 = A.count(5)

    # Создать упорядоченный список (потом преобразуем в кортеж)
    sorted_list = []

    # Сначала добавить все пятерки
    for i in range(count_5):
        sorted_list.append(5)

    # Затем добавить остальные оценки
    for grade in A:
        if grade != 5:
            sorted_list.append(grade)

    # Преобразовать в кортеж (функция tuple() из примеров)
    sorted_grades = tuple(sorted_list)

    print("Количество пятерок:", count_5)
    print("Упорядоченные оценки:", sorted_grades)
```

```
Задание 3
C:\Users\HP\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\HP\Documents\2 ano\Программирования на Python\Задание 3.py"
1 2 5 3 5 6 7 5 9 11 15 13 14 15 65 21 23 22 25 55 1 4 5 7
Количество пятерок: 4
Упорядоченные оценки: (5, 5, 5, 5, 1, 2, 3, 6, 7, 9, 11, 15, 13, 14, 15, 65, 21, 23, 22, 25, 55, 1, 4, 7)

Process finished with exit code 0
```

Рисунок 7. Код, выполненный в PyCharm

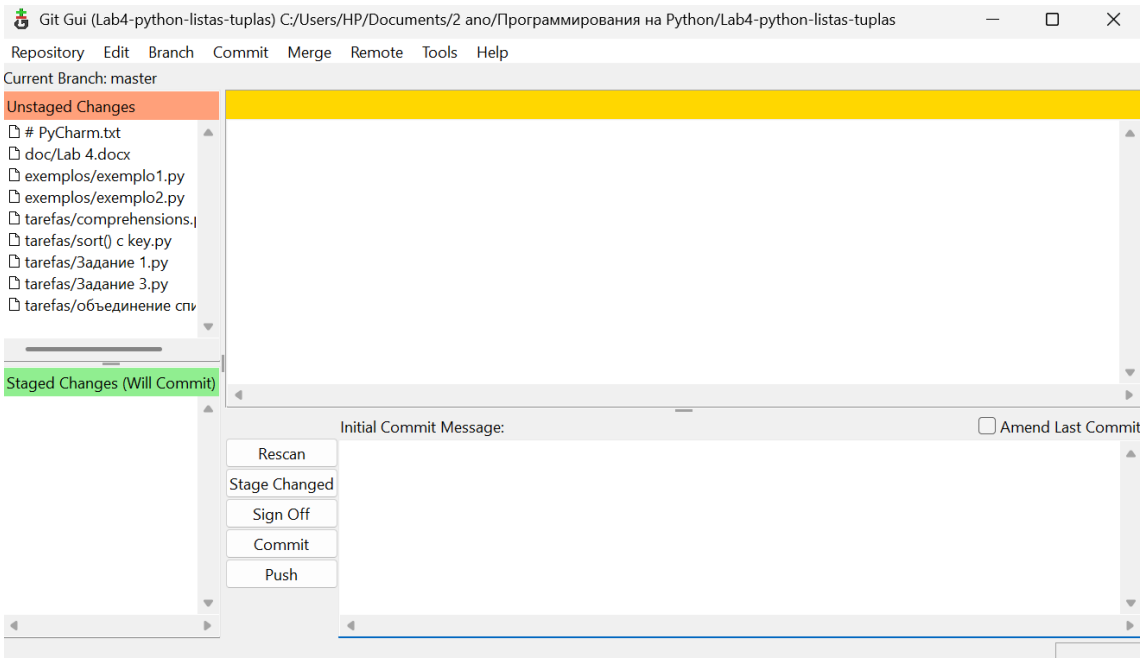

Рисунок 8. Отправка файлов в репозиторий через git gui

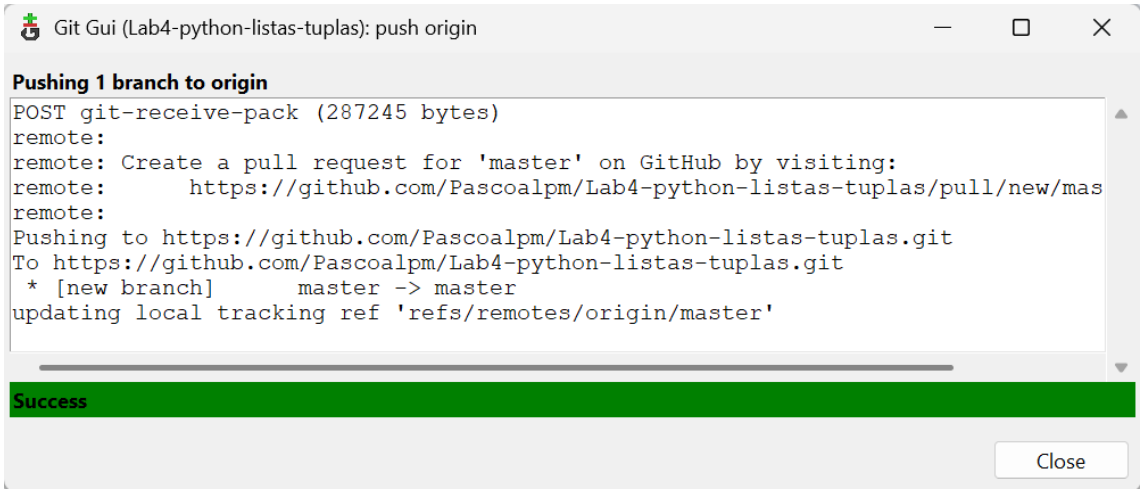

Рисунок 9. Отправка файлов в репозиторий через git gui

Вывод

В ходе лабораторной работы были успешно выполнены все поставленные задачи.

Ответы на контрольные вопросы

1. Что такое списки в языке Python?

Список (list) – это структура данных для хранения объектов различных типов. Список очень похож на массив, но в нем можно хранить объекты различных типов. Размер списка не статичен, его можно изменять.

2. Как осуществляется создание списка в Python?

Для создания списка нужно заключить элементы в квадратные скобки:

```
my_list = [1, 2, 3, 4, 5]
my_list = ['один', 'два', 'три']
my_list = ['один', 10, 2.25, [5, 15]]
```

3. Как организовано хранение списков в оперативной памяти?

При создании списка в памяти резервируется область "контейнера", в котором хранятся ссылки на другие элементы данных в памяти. В отличие от типов данных как число или строка, содержимое "контейнера" списка можно менять.

4. Каким образом можно перебрать все элементы списка?

С помощью цикла for:

```
my_list = ['один', 'два', 'три']
for elem in my_list:
    print(elem)
```

Или с индексами через enumerate():

```
for i, item in enumerate(my_list):
    print(f"{i}, {item}")
```

5. Какие существуют арифметические операции со списками?

Для объединения списков можно использовать оператор сложения +:

```
list_1 = [1, 2, 3]
list_2 = [4, 5, 6]
print(list_1 + list_2)  # [1, 2, 3, 4, 5, 6]
```

6. Как проверить есть ли элемент в списке?

С помощью оператора in:

```
lst = [3, 5, 2, 4, 1]
if 3 in lst:
    print("Список содержит число 3")
```

7. Как определить число вхождений заданного элемента в списке?

С помощью метода count():

```
A = ["ab", "ac", "ab", "ab", "ca"]
d1 = A.count("ab")  # 3
```

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

- append() - добавить в конец
- insert() - вставить по индексу

```
my_list.append('ещё один')
my_list.insert(1, 'Привет')
```

9. Как выполнить сортировку списка?

С помощью метода sort():

```
my_list = ['cde', 'fgh', 'abc']
my_list.sort()  # ['abc', 'cde', 'fgh']
my_list.sort(reverse=True)  # по убыванию
```

10. Как удалить один или несколько элементов из списка?

- pop() - удалить по индексу
- remove() - удалить по значению
- del - удалить срез

```
removed = my_list.pop(2)
my_list.remove('два')
del my_list[1:3]
```

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

List Comprehension - это способ построения списков:

```
# Традиционный способ
b = []
for i in a:
    b.append(i**2)
# List Comprehension
b = [i**2 for i in a]
b = [i for i in a if i % 2 == 0]  # с фильтром
```

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

С помощью синтаксиса [start:stop:step]:

```
a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(a[0:5])    # [0, 1, 2, 3, 4]
print(a[::2])    # [0, 2, 4, 6, 8]
print(a[1:8:2])  # [1, 3, 5, 7]
```

13. Какие существуют функции агрегации для работы со списками?

- len(L) - количество элементов
- min(L) - минимальный элемент
- max(L) - максимальный элемент
- sum(L) - сумма элементов (для числовых списков)

14. Как создать копию списка?

С помощью метода copy() или среза:

```
b = a.copy()
# или
b = a[:]
```

15. Функция sorted() vs метод sort()

- sorted() возвращает новый отсортированный список
- sort() изменяет исходный список и возвращает None

16. Что такое кортежи в языке Python?

Кортеж (tuple) – это неизменяемая структура данных, очень похожая на список.

17. Каково назначение кортежей в языке Python?

- Обезопасить данные от случайного изменения
- Экономия места в памяти
- Прирост производительности
- Можно использовать в качестве ключа у словаря

18. Как осуществляется создание кортежей?

```
a = (1, 2, 3, 4, 5)
b = tuple([1, 2, 3, 4])
c = (42,)  # кортеж из одного элемента
```

19. Как осуществляется доступ к элементам кортежа?

Так же как к элементам списка – через указание индекса:

```
a = (1, 2, 3, 4, 5)
print(a[0])  # 1
```

20. Зачем нужна распаковка (деструктуризация) кортежа?

Для удобного получения значений:

```
name_and_age = ("Bob", 42)
(name, age) = name_and_age
```

21. Какую роль играют кортежи в множественном присваивании?

Позволяют обмен значениями между переменными:

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

22. Как выбрать элементы кортежа с помощью среза?

Так же как со списками:

```
A = (0, 1, 2, 3)
item = A[0:2]  # (0, 1)
```

23. Как выполняется конкатенация и повторение кортежей?

С помощью операторов + и *:

```
A = (1, 2, 3)
B = (4, 5, 6)
C = A + B  # (1, 2, 3, 4, 5, 6)
D = A * 3  # (1, 2, 3, 1, 2, 3, 1, 2, 3)
```

24. Как выполняется обход элементов кортежа?

С помощью циклов for или while:

```
A = ("abc", "abcd", "bcd")
for item in A:
    print(item)
```

25. Как проверить принадлежность элемента кортежу?

С помощью оператора in:

```
A = ("abc", "abcd", "bcd", "cde")
if "abc" in A:
    print("Элемент найден")
```

26. Какие методы работы с кортежами Вам известны?

- index() - поиск индекса элемента
- count() - подсчет количества вхождений

27. Допустимо ли использование функций агрегации с кортежами?

Да, функции len(), min(), max(), sum() работают с кортежами так же как со списками.

28. Как создать кортеж с помощью спискового включения?

С помощью преобразования генератора в кортеж:

```
tpl = tuple(i for i in range(10))
```