**Лабораторная работа №6  
Тема: Работа с кортежами и списками**

### **Задача 1. Условие.**

Программа для вычисления произведения отрицательных элементов кортежа, имеющих нечетные индексы.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | tuple\_nums | Кортеж целых чисел | кортеж целых |  |
| Промежуточные данные | product | Произведение отрицательных элементов с нечетными индексами | целое |  |
| Выходные данные | product | Итоговое произведение | целое |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | tuple\_nums | (1, -2, 3, -4, 5) |
| Выходные данные | product | 8 |

**Листинг программы:**

tuple\_nums = tuple(map(int, input("Введите целые числа через пробел: ").split()))  
product = 1

for i in range(1, len(tuple\_nums), 2):  
if tuple\_nums[i] < 0:  
product \*= tuple\_nums[i]

if product == 1:  
print("Нет отрицательных элементов с нечетными индексами")  
else:  
print(f"Произведение отрицательных элементов с нечетными индексами: {product}")

### **Задача 2. Условие.**

Программа для формирования двух списков: списка четных чисел и списка нечетных чисел из исходного списка.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | list\_nums | Исходный список целых чисел | список целых |  |
| Выходные данные | even\_nums | Список четных чисел | список целых |  |
| Выходные данные | odd\_nums | Список нечетных чисел | список целых |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | list\_nums | [1, 2, 3, 4, 5, 6] |
| Выходные данные | even\_nums | [2, 4, 6] |
| Выходные данные | odd\_nums | [1, 3, 5] |

**Листинг программы:**

list\_nums = list(map(int, input("Введите целые числа через пробел: ").split()))  
even\_nums = [num for num in list\_nums if num % 2 == 0]  
odd\_nums = [num for num in list\_nums if num % 2 != 0]

print(f"Список четных чисел: {even\_nums}")  
print(f"Список нечетных чисел: {odd\_nums}")

### **Задача 3. Условие.**

Программа для вычисления среднего арифметического элементов кортежа, удовлетворяющих условию abs(korteg[i]) > C.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | tuple\_nums | Кортеж целых чисел | кортеж целых |  |
| Входные данные | C | Пороговое значение | вещественное |  |
| Выходные данные | avg | Среднее арифметическое | вещественное |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | tuple\_nums | (10, -3, 20, 5, -15) |
| Входные данные | C | 5 |
| Выходные данные | avg | 15.00 |

**Листинг программы:**

tuple\_nums = tuple(map(int, input("Введите элементы кортежа через пробел: ").split()))  
C = float(input("Введите значение C: "))  
filtered\_elements = [num for num in tuple\_nums if abs(num) > C]

if filtered\_elements:  
avg = sum(filtered\_elements) / len(filtered\_elements)  
print(f"Среднее арифметическое элементов: {avg:.2f}")  
else:  
print("Нет элементов, удовлетворяющих условию.")

### **Задача 4. Условие.**

Программа для нахождения максимального и минимального элементов среди положительных нечетных элементов целочисленного кортежа A(10).

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | tuple\_nums | Кортеж целых чисел | кортеж целых |  |
| Выходные данные | max\_val | Максимальный положительный нечетный элемент | целое |  |
| Выходные данные | min\_val | Минимальный положительный нечетный элемент | целое |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | tuple\_nums | (1, -2, 3, -4, 5, 7, -8, 9, -10, 11) |
| Выходные данные | max\_val | 11 |
| Выходные данные | min\_val | 1 |

**Листинг программы:**

import random

tuple\_nums = tuple(random.randint(-20, 50) for \_ in range(10))  
odd\_positive = [num for num in tuple\_nums if num > 0 and num % 2 != 0]

if odd\_positive:  
max\_val = max(odd\_positive)  
min\_val = min(odd\_positive)  
print(f"Максимальный положительный нечетный элемент: {max\_val}")  
print(f"Минимальный положительный нечетный элемент: {min\_val}")  
else:  
print("Нет положительных нечетных элементов.")

### **Задача 5. Условие.**

Программа для создания списка из N элементов случайными целыми числами в интервале от -20 до 40 и изменения элементов списка.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | N | Количество элементов списка | целое | N > 0 |
| Промежуточные данные | list\_nums | Список случайных чисел | список целых |  |
| Выходные данные | list\_nums | Измененный список | список целых |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | N | 5 |
| Входные данные | list\_nums | [10, -5, 20, -15, 30] |
| Выходные данные | list\_nums | [5.0, 1, 10.0, 3, 15.0] |

**Листинг программы:**

import random

N = int(input("Введите количество элементов списка: "))  
list\_nums = [random.randint(-20, 40) for \_ in range(N)]

print(f"Исходный список: {list\_nums}")

for i in range(len(list\_nums)):  
if list\_nums[i] > 0:  
list\_nums[i] /= 2  
else:  
list\_nums[i] = i

print(f"Измененный список: {list\_nums}")

### **Задача 6. Условие.**

Программа для определения процента нечетных элементов в списке.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | N | Количество элементов списка | целое | N > 0 |
| Промежуточные данные | list\_nums | Список случайных чисел | список целых |  |
| Выходные данные | percent\_odd | Процент нечетных элементов | вещественное |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | N | 6 |
| Входные данные | list\_nums | [2, 5, 7, 8, 9, 10] |
| Выходные данные | percent\_odd | 50.00% |

**Листинг программы:**

import random

N = int(input("Введите количество элементов списка: "))  
list\_nums = [random.randint(1, 40) for \_ in range(N)]

odd\_count = len([num for num in list\_nums if num % 2 != 0])  
percent\_odd = (odd\_count / N) \* 100

print(f"Список: {list\_nums}")  
print(f"Процент нечетных элементов: {percent\_odd:.2f}%")

### **Задача 7. Условие.**

Программа для сортировки списка из N элементов случайными целыми числами в интервале от 1 до 50 по возрастанию.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | N | Количество элементов списка | целое | N > 0 |
| Промежуточные данные | list\_nums | Список случайных чисел | список целых |  |
| Выходные данные | list\_nums | Отсортированный список | список целых |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | N | 5 |
| Входные данные | list\_nums | [5, 3, 1, 4, 2] |
| Выходные данные | list\_nums | [1, 2, 3, 4, 5] |

**Листинг программы:**

import random

N = int(input("Введите количество элементов списка: "))  
list\_nums = [random.randint(1, 50) for \_ in range(N)]

print(f"Исходный список: {list\_nums}")  
list\_nums.sort()  
print(f"Отсортированный список: {list\_nums}")

### **Задача 8. Условие.**

Программа для формирования списка: сначала отрицательные, затем положительные, и, наконец, нулевые элементы.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | list\_A | Исходный список целых чисел | список целых | длина списка = 10 |
| Промежуточные данные | negatives | Список отрицательных элементов | список целых |  |
| Промежуточные данные | positives | Список положительных элементов | список целых |  |
| Промежуточные данные | zeros | Список нулевых элементов | список целых |  |
| Выходные данные | result\_list | Итоговый список | список целых |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | list\_A | [-5, 0, 3, -2, 4, 0, -1, 2, 0, 1] |
| Выходные данные | result\_list | [-5, -2, -1, 3, 4, 2, 1, 0, 0, 0] |

**Листинг программы:**

list\_A = list(map(int, input("Введите 10 чисел через пробел: ").split()))  
negatives = [x for x in list\_A if x < 0]  
positives = [x for x in list\_A if x > 0]  
zeros = [x for x in list\_A if x == 0]  
result\_list = negatives + positives + zeros

print(f"Преобразованный список: {result\_list}")

### **Задача 9. Условие.**

Программа для нахождения максимального элемента списка, его номера и замены с первым элементом.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | list\_nums | Исходный список целых чисел | список целых | длина списка = N |
| Промежуточные данные | max\_index | Индекс максимального элемента | целое |  |
| Выходные данные | list\_nums | Измененный список | список целых |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | list\_nums | [10, 15, 5, 20, 7] |
| Выходные данные | list\_nums | [20, 15, 5, 10, 7] |

**Листинг программы:**

import random

N = int(input("Введите количество элементов списка: "))  
list\_nums = [random.randint(1, 30) for \_ in range(N)]

print(f"Исходный список: {list\_nums}")  
max\_index = list\_nums.index(max(list\_nums))  
list\_nums[0], list\_nums[max\_index] = list\_nums[max\_index], list\_nums[0]  
print(f"Измененный список: {list\_nums}")

### **Задача 10. Условие.**

Программа для включения числа D в отсортированный список A[10] с сохранением упорядоченности.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | D | Число для включения в список | целое |  |
| Входные данные | list\_A | Исходный отсортированный список | список целых | длина списка = 10 |
| Выходные данные | list\_A | Обновленный отсортированный список | список целых |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | D | 8 |
| Входные данные | list\_A | [1, 3, 5, 7, 9, 10, 12, 15, 18, 20] |
| Выходные данные | list\_A | [1, 3, 5, 7, 8, 9, 10, 12, 15, 18, 20] |

**Листинг программы:**

import bisect

D = int(input("Введите число D: "))  
list\_A = sorted(map(int, input("Введите 10 чисел через пробел: ").split()))  
bisect.insort(list\_A, D)

print(f"Обновленный список: {list\_A}")

### **Задача 11. Условие.**

Программа для удаления первых четырех нулевых элементов из списка.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | list\_A | Исходный список целых чисел | список целых | длина списка = N |
| Промежуточные данные | zeros\_removed | Счетчик удаленных нулей | целое |  |
| Выходные данные | list\_A | Обновленный список | список целых |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | list\_A | [0, 1, 0, 2, 0, 3, 4, 0, 5] |
| Выходные данные | list\_A | [1, 2, 3, 4, 5] |

**Листинг программы:**

list\_A = list(map(int, input("Введите элементы списка через пробел: ").split()))  
zeros\_removed = 0

for i in range(len(list\_A)):  
if list\_A[i] == 0:  
list\_A[i] = None  
zeros\_removed += 1  
if zeros\_removed == 4:  
break

list\_A = [x for x in list\_A if x is not None]  
print(f"Обновленный список: {list\_A}")

### **Задача 12. Условие.**

Программа для определения наличия серий знакочередующихся чисел в списке.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | N | Количество элементов списка | целое | N > 0 |
| Промежуточные данные | list\_nums | Список случайных чисел | список целых |  |
| Промежуточные данные | series\_count | Количество серий знакочередующихся чисел | целое |  |
| Выходные данные | series\_count | Итоговое количество серий | целое |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | N | 6 |
| Входные данные | list\_nums | [-1, 2, -3, 4, -5, 6] |
| Выходные данные | series\_count | 5 |

**Листинг программы:**

import random

N = int(input("Введите количество элементов списка: "))  
list\_nums = [random.randint(-30, 50) for \_ in range(N)]  
series\_count = 0  
is\_in\_series = False

print(f"Список: {list\_nums}")

for i in range(1, len(list\_nums)):  
if list\_nums[i] \* list\_nums[i - 1] < 0:  
if not is\_in\_series:  
series\_count += 1  
is\_in\_series = True  
else:  
is\_in\_series = False

print(f"Количество серий знакочередующихся чисел: {series\_count}")

### **Задача 13. Условие.**

Программа для отбора шин с диаметрами, отличающимися не более чем на D см и весом не более W кг.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | diameters | Кортеж диаметров шин | кортеж целых | длина кортежа = 10 |
| Входные данные | weights | Кортеж весов шин | кортеж целых | длина кортежа = 10 |
| Входные данные | D | Максимальное различие в диаметре | целое | D > 0 |
| Входные данные | W | Максимальное различие в весе | целое | W > 0 |
| Выходные данные | pairs | Список пар индексов шин, удовлетворяющих условиям | список пар |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | diameters | (20, 22, 21, 19, 25, 18, 24, 23, 22, 21) |
| Входные данные | weights | (50, 45, 55, 48, 53, 47, 52, 50, 54, 49) |
| Входные данные | D | 3 |
| Входные данные | W | 5 |
| Выходные данные | pairs | [(0, 1), (1, 2), (1, 8), (3, 5), (7, 9)] |

**Листинг программы:**

import random

diameters = tuple(random.randint(10, 50) for \_ in range(10))  
weights = tuple(random.randint(20, 100) for \_ in range(10))  
D = int(input("Введите максимальное различие в диаметре: "))  
W = int(input("Введите максимальное различие в весе: "))

print(f"Диаметры шин: {diameters}")  
print(f"Веса шин: {weights}")

pairs = [(i, j) for i in range(10) for j in range(i+1, 10)  
if abs(diameters[i] - diameters[j]) <= D and abs(weights[i] - weights[j]) <= W]

if pairs:  
print(f"Подходящие пары шин: {pairs}")  
else:  
print("Нет подходящих пар шин")

### **Задача 14. Условие.**

Программа для формирования двух списков: номеров положительных и отрицательных элементов.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | list\_A | Исходный список целых чисел | список целых | длина списка = 10 |
| Промежуточные данные | positive\_indices | Список индексов положительных элементов | список целых |  |
| Промежуточные данные | negative\_indices | Список индексов отрицательных элементов | список целых |  |
| Выходные данные | positive\_indices | Номера положительных элементов | список целых |  |
| Выходные данные | negative\_indices | Номера отрицательных элементов | список целых |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | list\_A | [-3, 2, -5, 4, -1, 6, 0, -7, 8, 9] |
| Выходные данные | positive\_indices | [1, 3, 5, 8, 9] |
| Выходные данные | negative\_indices | [0, 2, 4, 7] |

**Листинг программы:**

list\_A = list(map(int, input("Введите 10 чисел через пробел: ").split()))  
positive\_indices = [i for i in range(len(list\_A)) if list\_A[i] > 0]  
negative\_indices = [i for i in range(len(list\_A)) if list\_A[i] < 0]

print(f"Номера положительных элементов: {positive\_indices}")  
print(f"Номера отрицательных элементов: {negative\_indices}")

### **Задача 15. Условие.**

Программа для вычисления среднего арифметического квадратов положительных элементов списка.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | N | Количество элементов списка | целое | N > 0 |
| Промежуточные данные | list\_nums | Список случайных чисел | список целых |  |
| Промежуточные данные | positive\_squares | Список квадратов положительных элементов | список вещественных |  |
| Выходные данные | average\_square | Среднее арифметическое квадратов положительных элементов | вещественное |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | N | 5 |
| Входные данные | list\_nums | [1, -2, 3, -4, 5] |
| Выходные данные | average\_square | 11.67 |

**Листинг программы:**

import random

N = int(input("Введите количество элементов списка: "))  
list\_nums = [random.randint(-20, 40) for \_ in range(N)]

positive\_squares = [num \*\* 2 for num in list\_nums if num > 0]  
if positive\_squares:  
average\_square = sum(positive\_squares) / len(positive\_squares)  
print(f"Среднее арифметическое квадратов положительных элементов: {average\_square:.2f}")  
else:  
print("Нет положительных элементов.")

### **Задача 16. Условие.**

Программа для обмена соседних элементов, стоящих на четных и нечетных местах.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | tuple\_nums | Исходный кортеж целых чисел | кортеж целых | длина кортежа > 1 |
| Промежуточные данные | new\_tuple | Измененный кортеж | кортеж целых |  |
| Выходные данные | new\_tuple | Кортеж после обмена элементов | кортеж целых |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | tuple\_nums | (1, 2, 3, 4, 5, 6) |
| Выходные данные | new\_tuple | (2, 1, 4, 3, 6, 5) |

**Листинг программы:**

tuple\_nums = tuple(map(int, input("Введите целые числа через пробел: ").split()))  
new\_tuple = list(tuple\_nums)

for i in range(1, len(new\_tuple), 2):  
new\_tuple[i - 1], new\_tuple[i] = new\_tuple[i], new\_tuple[i - 1]

new\_tuple = tuple(new\_tuple)  
print(f"Измененный кортеж: {new\_tuple}")

### **Задача 17. Условие.**

Программа для умножения элементов списка на максимальный элемент.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | N | Количество элементов списка | целое | N > 0 |
| Промежуточные данные | list\_nums | Список случайных вещественных чисел | список вещественных |  |
| Промежуточные данные | max\_value | Максимальный элемент списка | вещественное |  |
| Выходные данные | list\_nums | Обновленный список | список вещественных |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | N | 4 |
| Входные данные | list\_nums | [2.5, 1.2, 3.6, 2.9] |
| Выходные данные | list\_nums | [9.0, 1.2, 3.6, 2.9] |

**Листинг программы:**

import random

N = int(input("Введите количество элементов списка: "))  
list\_nums = [random.uniform(1, 30) for \_ in range(N)]  
max\_value = max(list\_nums)

for i in range(len(list\_nums)):  
if i % 2 == 0 and list\_nums[i] < max\_value:  
list\_nums[i] \*= max\_value

print(f"Обновленный список: {list\_nums}")

### **Задача 18. Условие.**

Программа для нахождения наибольшего отрицательного элемента в списке.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | N | Количество элементов списка | целое | N > 0 |
| Промежуточные данные | list\_nums | Список случайных чисел | список целых |  |
| Промежуточные данные | negative\_nums | Список отрицательных элементов | список целых |  |
| Выходные данные | max\_negative | Наибольший отрицательный элемент | целое |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | N | 5 |
| Входные данные | list\_nums | [-10, 5, -3, 7, -1] |
| Выходные данные | max\_negative | -1 |

**Листинг программы:**

import random

N = int(input("Введите количество элементов списка: "))  
list\_nums = [random.randint(-15, 20) for \_ in range(N)]  
negative\_nums = [num for num in list\_nums if num < 0]

if negative\_nums:  
max\_negative = max(negative\_nums)  
print(f"Наибольший отрицательный элемент: {max\_negative}")  
else:  
print("Отрицательных элементов нет.")

### **Задача 19. Условие.**

Программа для подсчета элементов, значения которых находятся в диапазоне от A до B.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | N | Количество элементов списка | целое | N > 0 |
| Входные данные | A | Начало диапазона | целое | A < B |
| Входные данные | B | Конец диапазона | целое | B > A |
| Промежуточные данные | list\_nums | Список случайных чисел | список целых |  |
| Выходные данные | count\_in\_range | Количество элементов в диапазоне | целое |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | N | 6 |
| Входные данные | A | 10 |
| Входные данные | B | 20 |
| Входные данные | list\_nums | [5, 12, 15, 25, 18, 30] |
| Выходные данные | count\_in\_range | 3 |

**Листинг программы:**

import random

N = int(input("Введите количество элементов списка: "))  
A = int(input("Введите начало диапазона A: "))  
B = int(input("Введите конец диапазона B: "))  
list\_nums = [random.randint(1, 50) for \_ in range(N)]

count\_in\_range = len([num for num in list\_nums if A <= num <= B])

print(f"Список: {list\_nums}")  
print(f"Количество элементов в диапазоне от {A} до {B}: {count\_in\_range}")

### **Задача 20. Условие.**

Программа для проверки, является ли последовательность монотонно убывающей.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | list\_A | Список целых чисел | список целых | длина списка = N |
| Выходные данные | is\_decreasing | Логическое значение, является ли список монотонно убывающим | логическое |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | list\_A | [10, 8, 6, 4, 2] |
| Выходные данные | is\_decreasing | True |

**Листинг программы:**

list\_A = list(map(int, input("Введите элементы списка через пробел: ").split()))  
is\_decreasing = all(list\_A[i] > list\_A[i + 1] for i in range(len(list\_A) - 1))

if is\_decreasing:  
print("Последовательность монотонно убывающая")  
else:  
print("Последовательность не является монотонно убывающей")

### **Задача 21. Условие.**

Программа для замены нулями элементов между максимальным и минимальным значениями.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | N | Количество элементов списка | целое | N > 0 |
| Промежуточные данные | list\_nums | Список случайных чисел | список целых |  |
| Промежуточные данные | max\_index | Индекс максимального элемента | целое |  |
| Промежуточные данные | min\_index | Индекс минимального элемента | целое |  |
| Выходные данные | list\_nums | Обновленный список | список целых |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | N | 5 |
| Входные данные | list\_nums | [3, 10, 2, 1, 15] |
| Выходные данные | list\_nums | [3, 10, 0, 0, 15] |

**Листинг программы:**

import random

N = int(input("Введите количество элементов списка: "))  
list\_nums = [random.randint(1, 30) for \_ in range(N)]  
max\_index = list\_nums.index(max(list\_nums))  
min\_index = list\_nums.index(min(list\_nums))

if min\_index < max\_index:  
list\_nums[min\_index + 1

] = [0] \* (max\_index - min\_index - 1)  
else:  
list\_nums[max\_index + 1

] = [0] \* (min\_index - max\_index - 1)

print(f"Обновленный список: {list\_nums}")

### **Задача 22. Условие.**

Программа для нахождения индекса последнего отрицательного элемента в кортеже.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | tuple\_nums | Исходный кортеж целых чисел | кортеж целых |  |
| Выходные данные | last\_negative\_index | Индекс последнего отрицательного элемента | целое |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | tuple\_nums | (1, -2, 3, -4, 5) |
| Выходные данные | last\_negative\_index | 3 |

**Листинг программы:**

tuple\_nums = tuple(map(int, input("Введите целые числа через пробел: ").split()))  
last\_negative\_index = len(tuple\_nums) - 1 - tuple\_nums[::-1].index(next(filter(lambda x: x < 0, tuple\_nums[::-1])))

print(f"Индекс последнего отрицательного элемента: {last\_negative\_index}")

### **Задача 23. Условие.**

Программа для проверки наличия хотя бы одной пары совпадающих чисел в кортеже.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | tuple\_A | Исходный кортеж целых чисел | кортеж целых | длина кортежа = 10 |
| Выходные данные | has\_duplicate | Логическое значение, есть ли хотя бы одна пара совпадающих чисел | логическое |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | tuple\_A | (1, 2, 3, 4, 5, 6, 7, 8, 9, 1) |
| Выходные данные | has\_duplicate | True |

**Листинг программы:**

tuple\_A = tuple(map(int, input("Введите 10 целых чисел через пробел: ").split()))  
has\_duplicate = len(tuple\_A) != len(set(tuple\_A))

if has\_duplicate:  
print("Есть хотя бы одна пара совпадающих чисел")  
else:  
print("Совпадающих чисел нет")

### **Задача 24. Условие.**

Программа для определения, имеются ли игроки одинакового роста в двух командах.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | team1\_heights | Кортеж ростов игроков первой команды | кортеж целых | длина кортежа = 10 |
| Входные данные | team2\_heights | Кортеж ростов игроков второй команды | кортеж целых | длина кортежа = 10 |
| Выходные данные | common\_heights | Набор одинаковых ростов игроков | множество целых |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | team1\_heights | (180, 190, 175, 165, 185, 170, 180, 185, 200, 195) |
| Входные данные | team2\_heights | (160, 170, 180, 185, 190, 175, 200, 165, 180, 190) |
| Выходные данные | common\_heights | {170, 180, 185, 190, 200} |

**Листинг программы:**

team1\_heights = tuple(map(int, input("Введите росты игроков первой команды через пробел: ").split()))  
team2\_heights = tuple(map(int, input("Введите росты игроков второй команды через пробел: ").split()))

common\_heights = set(team1\_heights) & set(team2\_heights)

if common\_heights:  
print(f"Игроки одинакового роста: {common\_heights}")  
else:  
print("Игроков одинакового роста нет")

### **Задача 25. Условие.**

Программа для нахождения максимального и минимального элементов списка и вычисления их разности.

**Таблица данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс** | **Имя** | **Смысл** | **Тип, структура** | **Ограничения на значения** |
| Входные данные | N | Количество элементов списка | целое | N > 0 |
| Промежуточные данные | list\_nums | Список случайных чисел | список целых |  |
| Выходные данные | max\_value | Максимальный элемент списка | целое |  |
| Выходные данные | min\_value | Минимальный элемент списка | целое |  |
| Выходные данные | difference | Разность между максимальным и минимальным элементами | целое |  |

**Тесты.**

|  |  |  |
| --- | --- | --- |
| **Класс** | **Имя** | **Значение** |
| Входные данные | N | 5 |
| Входные данные | list\_nums | [3, 10, 2, 15, 7] |
| Выходные данные | max\_value | 15 |
| Выходные данные | min\_value | 2 |
| Выходные данные | difference | 13 |

**Листинг программы:**

import random

N = int(input("Введите количество элементов списка: "))  
list\_nums = [random.randint(1, 50) for \_ in range(N)]

max\_value = max(list\_nums)  
min\_value = min(list\_nums)  
difference = max\_value - min\_value

print(f"Список: {list\_nums}")  
print(f"Максимальный элемент: {max\_value}, Минимальный элемент: {min\_value}")  
print(f"Разность между максимальным и минимальным элементами: {difference}")