

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ  
ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»  
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

**Кафедра инфокоммуникаций**

**Отчет  
по лабораторной работе №3  
«Работа со списками в языке Python»  
по дисциплине:  
«Введение в системы искусственного интеллекта»**

**Вариант 7**

Выполнил: студент группы ИВТ-б-о-18-1 (2)  
Криворучко Ярослав Евгеньевич

\_\_\_\_\_ (подпись)

Проверил:

Воронкин Роман Александрович

\_\_\_\_\_ (подпись)

Ставрополь, 2022 г.

**Цель работы:** приобретение навыков по работе со списками при написании программ с помощью языка программирования Python версии 3.x.\

## Задание 1

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

```
Ввод [1]: n = 10
sum = 1
a = []
for i in range(n):
    num = float(input('Введите число для a[' + str(i) + '] элемента = '))
    a.append(num)
    if num < 0:
        sum *= num

print('Произведение отрицательных элементов: ', sum)
```

```
Введите число для a[0] элемента = 4
Введите число для a[1] элемента = 6
Введите число для a[2] элемента = -2
Введите число для a[3] элемента = 8
Введите число для a[4] элемента = -4
Введите число для a[5] элемента = -2
Введите число для a[6] элемента = 4
Введите число для a[7] элемента = 6
Введите число для a[8] элемента = 3
Введите число для a[9] элемента = -1
Произведение отрицательных элементов: 16.0
```

Рисунок 1 – Листинг программы

## Задание 2

7. 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.

```

Ввод [8]: from random import randint

n = 10
sum = 1
a = []
max = 10
min = max
t, t2 = False, False

sum = 0
for i in range(n):
    num = randint(-2,max)
    #print(num)
    a.append(num)

    if num <= min:
        min = num
        number = i+1
    if t and num>-1:
        sum+= a[i]
        #print("ss = ", ss)

    if t and t2==False and num<0:
        t = False
        t2 = True
        # print("t2", a[i])

    if num<0 and t2 == False:
        t=True
        #print("t", a[i])

print('Список: ', a )
print('Номер минимального элемента списка: ', number)
print('Сумма элементов находящихся между 1 и 2 отрицательным элементом: ', sum)
a.sort(key=abs)
print('Сортированный список: ', a )

Список: [9, 1, 6, 6, 10, 2, 5, 2, -2, 7]
Номер минимального элемента списка: 9
Сумма элементов находящихся между 1 и 2 отрицательным элементом: 7
Сортированный список: [1, 2, 2, -2, 5, 6, 6, 7, 9, 10]

```

Рисунок 2 – Листинг программы

**Вывод:** в ходе выполнения работы были приобретены навыки по работе со списками при написании программ с помощью языка программирования Python версии 3.x.\

Файл 2.4.ipynb с решением задач находится на **Github:**  
[https://github.com/YaroStavr/LR3\\_Artificial-Intelligence.git](https://github.com/YaroStavr/LR3_Artificial-Intelligence.git)

## Ответ на вопросы

### 1. Что такое списки в языке Python?

Список (list) – это структура данных для хранения объектов различных типов. Если вы использовали другие языки программирования, то вам должно быть знакомо понятие массива. Так вот, список очень похож на массив, только, как было уже сказано выше, в нем можно хранить объекты различных типов. Размер списка не статичен, его можно изменять. Список по своей природе является изменяемым типом данных. Переменная, определяемая как список, содержит ссылку на структуру в памяти, которая в свою очередь хранит ссылки на какие-либо другие объекты или структуры.

### 2. Как осуществляется создание списка в Python?

Для создания списка нужно заключить элементы в квадратные скобки:

```
my_list = [1, 2, 3, 4, 5]
```

Список может выглядеть так:

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']
```

Можно смешивать типы содержимого:

```
my_list = ['один', 10, 2.25, [5, 15], 'пять']
```

Поддерживаются вложенные списки как в примере выше.

Получать доступ к любому элементу списка можно через его индекс. В Python используется система индексации, начиная с нуля.

```
third_elem = my_list[2]
```

Принцип похож на строки.

### 3. Как организовано хранение списков в оперативной памяти?

Как уже было сказано выше, список является изменяемым типом данных. При его создании в памяти резервируется область, которую можно условно назвать некоторым “контейнером”, в котором хранятся ссылки на другие элементы данных в памяти. В отличие от таких типов данных как число или строка, содержимое “контейнера” списка можно менять.

### 4. Каким образом можно перебрать все элементы списка?

Читать элементы списка можно с помощью следующего цикла: `my_list = ['один', 'два', 'три', 'четыре', 'пять']` `for elem in my_list: print(elem)`

Таким образом можно читать элементы списка. А вот что касается их обновления:

```
my_list = [1, 2, 3, 4, 5]
for i in range(len(my_list)):
    my_list[i] += 5
print(my_list)
```

Результат будет следующим:

```
[6, 7, 8, 9, 10]
```

Функция `len()` используется для возврата количества элементов.

Стоит запомнить, что вложенный список — это всегда один элемент вне зависимости от количества его элементов.

```
my_list = ['один', 10, 2.25, [5, 15], 'пять']
print(len(my_list))
```

Для получения в цикле одновременно элемента списка и его индекса необходимо использовать функцию `enumerate`. Встроенная в Python функция `enumerate()` применяется для итерируемых коллекций (строки, списки, словари и т. д.) и создает объект, который генерирует кортежи, состоящие из двух элементов - индекса элемента и самого элемента.

```
>>> a = [10, 20, 30, 40]
>>> for i, item in enumerate(a):
...     print(f"({i}, {item})")
...
(0, 10)
(1, 20)
(2, 30)
(3, 40)
```

## **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)
```

Результат:

```
[1, 2, 3, 4, 5, 6]
```

Список можно повторить с помощью оператора умножения ( \* ):

```
list_1 = [1, 2, 3]
print(list_1 * 2)
```

Результат:

```
[1, 2, 3, 1, 2, 3]
```

## **6. Как проверить есть ли элемент в списке?**

Метод index можно использовать для получения индекса элемента:

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']
print(my_list.index('два'))
```

Результат 1 .

Если в списке больше одного такого же элемента, функция вернет индекс первого.

## **7. Как определить число вхождений заданного элемента в списке?**

Метод count можно использовать для определения числа сколько раз данный элемент

встречается в списке:

```
lst = [1, 2, 2, 3, 3]
print(lst.count(2))
```

Результат 2 .

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

Метод `insert` можно использовать, чтобы вставить элемент в список:

```
my_list = [1, 2, 3, 4, 5]
my_list.insert(1, 'Привет')
print(my_list)
```

Метод `append` можно использовать для добавления элемента в список:

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']
my_list.append('ещё один')
print(my_list)
```

## **9. Как выполнить сортировку списка?**

Для сортировки списка нужно использовать метод `sort`.

```
my_list = ['cde', 'fgh', 'abc', 'klm', 'opq']
list_2 = [3, 5, 2, 4, 1]
my_list.sort()
list_2.sort()
print(my_list)
print(list_2)
```

## **10. Как удалить один или несколько элементов из списка?**

Удалить элемент можно, написав его индекс в методе `pop`:

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']
removed = my_list.pop(2)
print(my_list)
print(removed)
```

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

В языке Python есть две очень мощные функции для работы с коллекциями: `map` и `filter`. Они позволяют использовать функциональный стиль программирования, не прибегая к помощи циклов, для работы с такими типами как `list`, `tuple`, `set`, `dict` и т.п. Списковое включение позволяет обойтись

без этих функций. Приведем несколько примеров для того, чтобы понять о чем идет речь.

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

Слайсы (срезы) являются очень мощной составляющей Python, которая позволяет быстро и лаконично решать задачи выборки элементов из списка. Выше уже был пример использования слайсов, здесь разберем более подробно работу с ними. Создадим список для экспериментов:

```
>>> a = [i for i in range(10)]
```

Слайс задается тройкой чисел, разделенных запятой: start:stop:step. Start – позиция с которой нужно начать выборку, stop – конечная позиция, step – шаг. При этом необходимо помнить, что выборка не включает элемент определяемый stop.

## **13. Какие существуют функции агрегации для работы со списками?**

Для работы со списками Python предоставляет следующие функции:

len(L) - получить число элементов в списке L .

min(L) - получить минимальный элемент списка L .

max(L) - получить максимальный элемент списка L .

sum(L) - получить сумму элементов списка L , если список L содержит только числовые значения.

Для функций min и max элементы списка должны быть сравнимы между собой.

## **14. Как создать копию списка?**

Прежде чем вы действительно можете понять Копировать () Способ в Python, вы должны понимать концепцию «мелкой копии».

На объектно-ориентированных языках, таких как Python, все – это объект. Список – это объект, а элементы в списке тоже объекты. Неглубокая копия списка создает новый объект списка – копия – но она не создает новых элементов списка, но просто копирует ссылки на эти объекты.



Вы можете видеть, что указанный ниже – это только неглубокая копия, указывающая на те же элементы, что и исходный список. В Python, `list.copy()` Способ производит только мелкую копию, которая имеет гораздо более быструю сложность выполнения.

**15. Самостоятельно изучите функцию `sorted` языка Python. В чем ее отличие от метода `sort` списков?**

Для **Python** Встроенные функции **`sorted()`**, Сначала используется, чтобы следовать за функцией-членом в списке **`list.sort()`**. Сделай сравнение. По сути, сортировка списка аналогична сортировке встроенной отсортированной функции, а параметры в основном совпадают. Основное отличие состоит в том, что **`list.sort()`**. Он должен работать с существующим списком, а затем список операций может быть изменен. И встроенная функция **`sorted`** Возвращается новый список, а не операция, основанная на исходном.