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 9

Выполнил: студент группы ИВТ-б-о-18-1 (2)
Полещук Константин Сергеевич
_____(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

Составить программу, выдающую индексы заданного элемента или сообщающую, что такого элемента в списке нет.

Ввод [3]:

```
list = ['a', 'b', 'F', 't']
print(f"Список: {list}")
i = input("Введите элемент, индекс которого хотите получить:")
if i in list:
    print(f"Индекс заданного элемента - {i}:[{list.index(i)}]")
else:
    print("Такого элемента в списке нет")
```

```
Список: ['a', 'b', 'F', 't']
Введите элемент, индекс которого хотите получить:1
Такого элемента в списке нет
```

Рисунок 1 – Листинг программы

Задание №2

В списке, состоящем из целых элементов, вычислить:

1. минимальный по модулю элемент списка;

2. сумму модулей элементов списка, расположенных после первого элемента, равного нулю.

Преобразовать список таким образом, чтобы в первой его половине располагались элементы, стоявшие в четных позициях, а во второй половине - элементы, стоявшие в нечетных позициях.

Ввод [29]:

```python
#Список целых чисел, найти минимальный по модулю элемент, сумма модулей, по
#первого нуля, преобразовать список, что первая половина были четные позици
import math

list_int_number = [4, 3, 7, 0, -4, -1, 1, 14, -9]
min = abs(list_int_number[0])
for i in range(len(list_int_number)-1):
    if min > abs(list_int_number[i+1]):
        min = abs(list_int_number[i+1])

print(f"Минимальный по модулю элемент равен: {min}")

null_index = list_int_number.index(0)
list_after_null = list_int_number[null_index:]
sum = 0
for i in range(len(list_after_null)):
    sum += abs(list_after_null[i])

print(f"Сумма модулей элементов после первого нуля равна: {sum}")

print(f"Исходный список: {list_int_number}")
list_chet = list_int_number[1::2]
list_nechet = list_int_number[::2]
new_list = list_chet + list_nechet
print(f"Преобразованный список:",new_list)
```

```
Минимальный по модулю элемент равен: 0
Сумма модулей элементов после первого нуля равна: 29
Исходный список: [4, 3, 7, 0, -4, -1, 1, 14, -9]
Преобразованный список: [3, 0, -1, 14, 4, 7, -4, 1, -9]
```

Рисунок 2 – Листинг программы

Файл Lab_3 с решением задач находится на **Github**: https://github.com/Scratchykaktus/Python.git

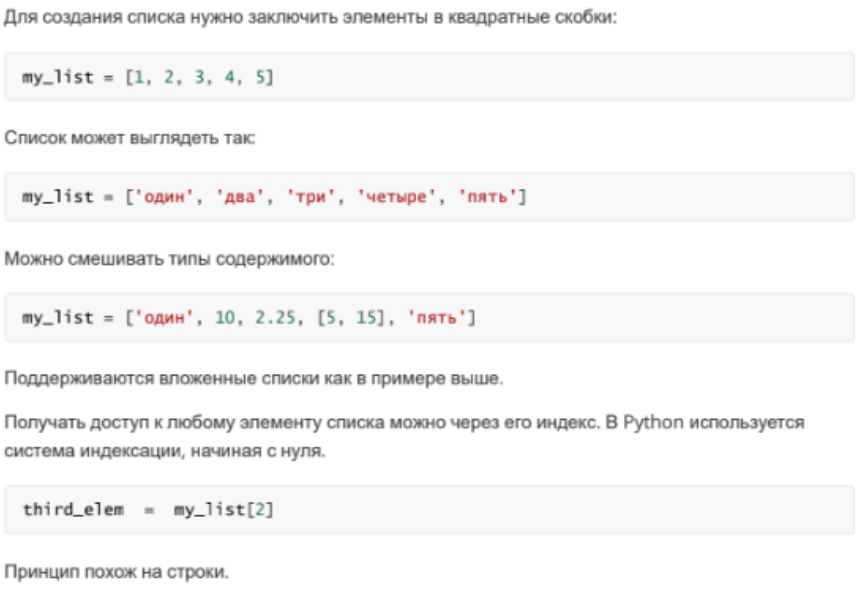
Вывод: в процессе выполнения лабораторной работы, были приобретены навыки по работе со списками при написании программ с помощью языка программирования Python версии 3.x.

Ответы 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?



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 отличии 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?

12. 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.

13. 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.

14. 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.

15. 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.