МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФГАОУ ВО
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт цифрового развития

Кафедра инфокоммуникаций

дисциплина «Основы кроссплатформенного программирования»

Отчет по лабораторной работе №4

Работа со списками в языке Python

Выполнил: студент группы ИТС-б-о-21-1
Джу Алексей

_____
(подпись)
Проверил: кандидат технических наук,

Доцент кафедры инфокоммуникаций

Роман Александрович Воронкин

_____
(подпись)

Ставрополь, 2022

Лабораторная работа 2.4 Работа со списками в языке Python

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

Теоретический конспект:

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.

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. Для того, чтобы лучше визуально представлять себе этот процесс взгляните на картинку ниже. Изначально был создан список содержащий ссылки на объекты 1 и 2, после операции a[1] = 3, вторая ссылка в списке стала указывать на объект 3.

Как списки хранятся в 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. Для того, чтобы лучше визуально представлять себе этот процесс взгляните на картинку ниже. Изначально был создан

Ход работы:

Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python. Выполнил клонирование созданного репозитория. (https://github.com/Chek228/Laboratornaya-Rab.4)

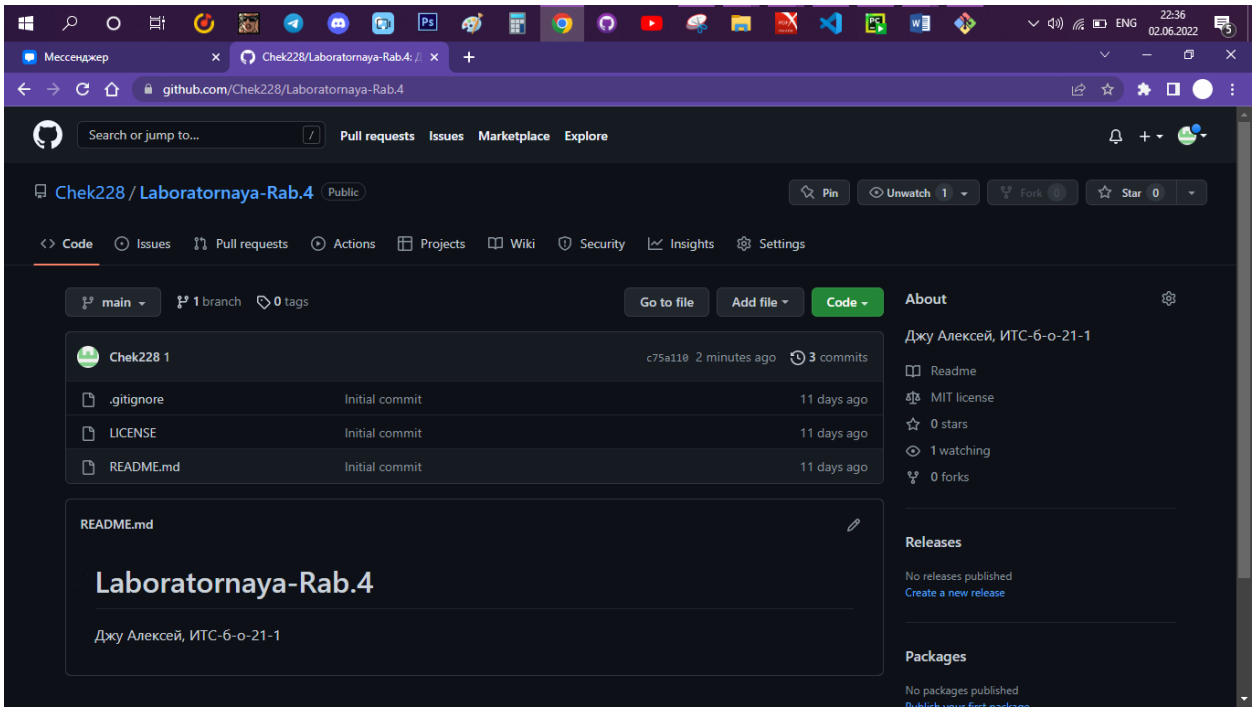

Рисунок 1

Задание 1.

```
1 2 3 4 8 16 17 18 19 20
Количество необходимых элементов =  2
Полученный элемент =  [8, 16]
PS C:\Users\Lenovo\Desktop>
```

Рисунок 1.1

Задание 2.

```
6 7 8 9 10 -6 -7 -8 -9 -10
Произведение отрицательных элементов списка = - -30240
Сумма положительных элементов списка =  40
Обратный список:  [-10, -9, -8, -7, -6, 10, 9, 8, 7, 6]
```

Рисунок 1.3

Ответы на контрольные вопросы:

1) 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 храни ссылки на какие-либо другие объекты или структуры

2) 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 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. Для того, чтобы лучше визуально представлять себе этот процесс взгляните на картинку ниже. Изначально был создал список содержащий ссылки на объекты 1 и 2, после операции *a[1] = 3*, вторая ссылка в списке стала указывать на объект 3.

4) 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 отколичества его элементов.

5) 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) Для того, чтобы проверить, есть ли заданный элемент в списке Python необходимо использовать оператор in :

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

Результат:

Список содержит число 3

Если требуется, чтобы элемент отсутствовал в списке, необходимо использовать оператор not in :

```
lst = [3, 5, 2, 4, 1]
    if 0 not in lst:
       print("Список не содержит нулей")
```

Результат:

Список не содержит нулей

7) 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) Метод append можно использовать для добавления элемента в список:

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

Результат: ['один', 'два', 'три', 'четыре', 'пять', 'ещё один']

Можно добавить и больше одного элемента таким способом

```
    my_list = ['один', 'два', 'три', 'четыре', 'пять']
    list_2  =  ['шесть',  'семь']
    my_list.extend(list_2)
    print(my_list)
```

Результат: ['один', 'два', 'три', 'четыре', 'пять', 'шесть', 'семь']

9) 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)
```

Вывод: ['abc', 'cde', 'fgh', 'klm', 'opq']

[1, 2, 3, 4, 5]

Для сортировки списка в порядке убывания необходимо вызвать метод sort с аргументом

reverse=True.

```
list_2 = [3, 5, 2, 4, 1]
list_2.sort(reverse=True)
```

Вывод: [5, 4, 3, 2, 1]

10) 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)
```

Результат: ['один', 'два', 'четыре', 'пять'] три

Если не указывать индекс, то функция удалит последний элемент.

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

Результат:

['один', 'два', 'три', 'четыре']пять

Элемент можно удалить с помощью метода remove.

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

Оператор del можно использовать для тех же целей:

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

Можно удалить несколько элементов с помощью оператора среза

```
    my_list = ['один', 'два', 'три', 'четыре', 'пять']
    del my_list[1:3]
    print(my_list)
```

Можно удалить все элементы из списка с помощью метода clear:

```
    a = [1, 2, 3, 4, 5]
    print(a)
    a.clear()
    print(a)
```

11) 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. Пример с заменой функции map. Пусть у нас есть список и нужно получить на базе него новый, который содержит элементы первого, возведенные в квадрат.

12) 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) 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) Чтобы создать неглубокую копию списка списков, используйте list.copy () Способ, который создает новый внешний список с копированными ссылками на одинаковые внутренние списки.

Чтобы создать глубокую копию с копированными внутренними списками, импортируйте скопировать Библиотека и звонок copy.deepcopy (список) скопировать как внутренние, так и внешние списки.

15) Sorted() возвращает новый отсортированный список, оставляя исходный список незатронутым. list.sort() сортирует список на месте , изменяет индексы списка и возвращает None (как и все операции на месте).