

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Основы кроссплатформенного программирования

Отчет по лабораторной работе №2.4

Тема: «Работа со списками в языке Python»

Выполнил студент группы

ИВТ-б-о-21-1

Артемьев А.В « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил доцент

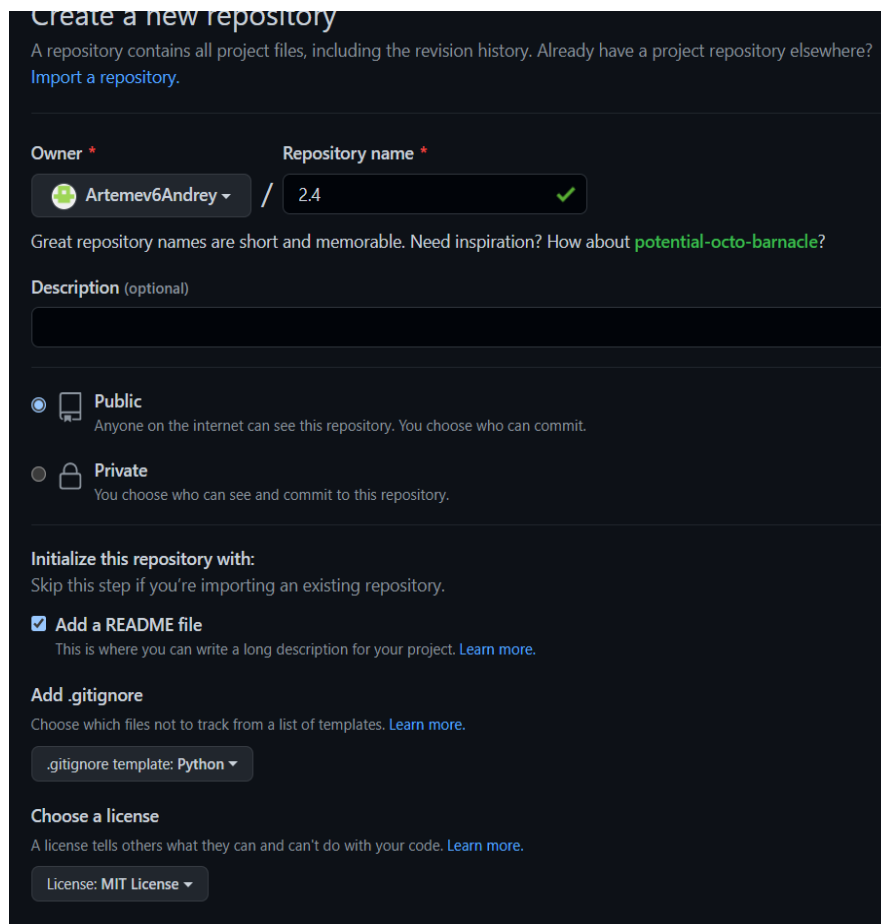
Кафедры инфокоммуникаций, старший
преподаватель

Воронкин Р.А.

(подпись)

Ставрополь 2022

1. Создал репозиторий в GitHub, дополнил правила в .gitignore для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на компьютер и организовал в соответствии с моделью ветвления git-flow.



The screenshot shows the 'Create a new repository' page on GitHub. At the top, it says 'Create a new repository' and 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. Below this, there are two input fields: 'Owner' with a dropdown menu showing 'Artemev6Andrey' and 'Repository name' with a text input '2.4' and a green checkmark. A hint text says 'Great repository names are short and memorable. Need inspiration? How about [potential-octo-barnacle?](#)'. There is an optional 'Description' text area. Below that, there are two radio button options: 'Public' (selected) and 'Private'. The 'Public' option says 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option says 'You choose who can see and commit to this repository.' Under 'Initialize this repository with:', it says 'Skip this step if you're importing an existing repository.' There is a checked checkbox 'Add a README file' with a link 'Learn more.' Below that, there is a section 'Add .gitignore' with a link 'Learn more.' and a dropdown menu '.gitignore template: Python'. At the bottom, there is a section 'Choose a license' with a link 'Learn more.' and a dropdown menu 'License: MIT License'.

Рисунок 1.1 – Репозиторий

```
aa715@DESKTOP-70QJC9F MINGW64 ~/Desktop/2.4/2.4 (main)
$ git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/] Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/aa715/Desktop/2.4/2.4/.git/hooks]

aa715@DESKTOP-70QJC9F MINGW64 ~/Desktop/2.4/2.4 (develop)
$
```

Рисунок 1.2 – Организация репозитория в соответствии с моделью ветвления git-flow

2. Создал проект PyCharm в папке репозитория, проработал примеры ЛР.

```
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)

if __name__ == '__main__':
    for item in A:
        if abs(item) < 5:
```

1пример ×

C:\Users\aa715\Desktop\venv\Scripts\python.exe "C:/Users/aa715/Desktop/1пример.py"

9 8 7 7 6 5 4 3 2 1

10

Process finished with exit code 0

Рисунок 2.1 – Результат выполнения примера №1

```
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)

if __name__ == '__main__':
```

2пример ×

C:\Users\aa715\Desktop\venv\Scripts\python.exe "C:/Users/aa715/Desktop/2пример.py"

5 4 9 8 7 5 6 5 2 1

6

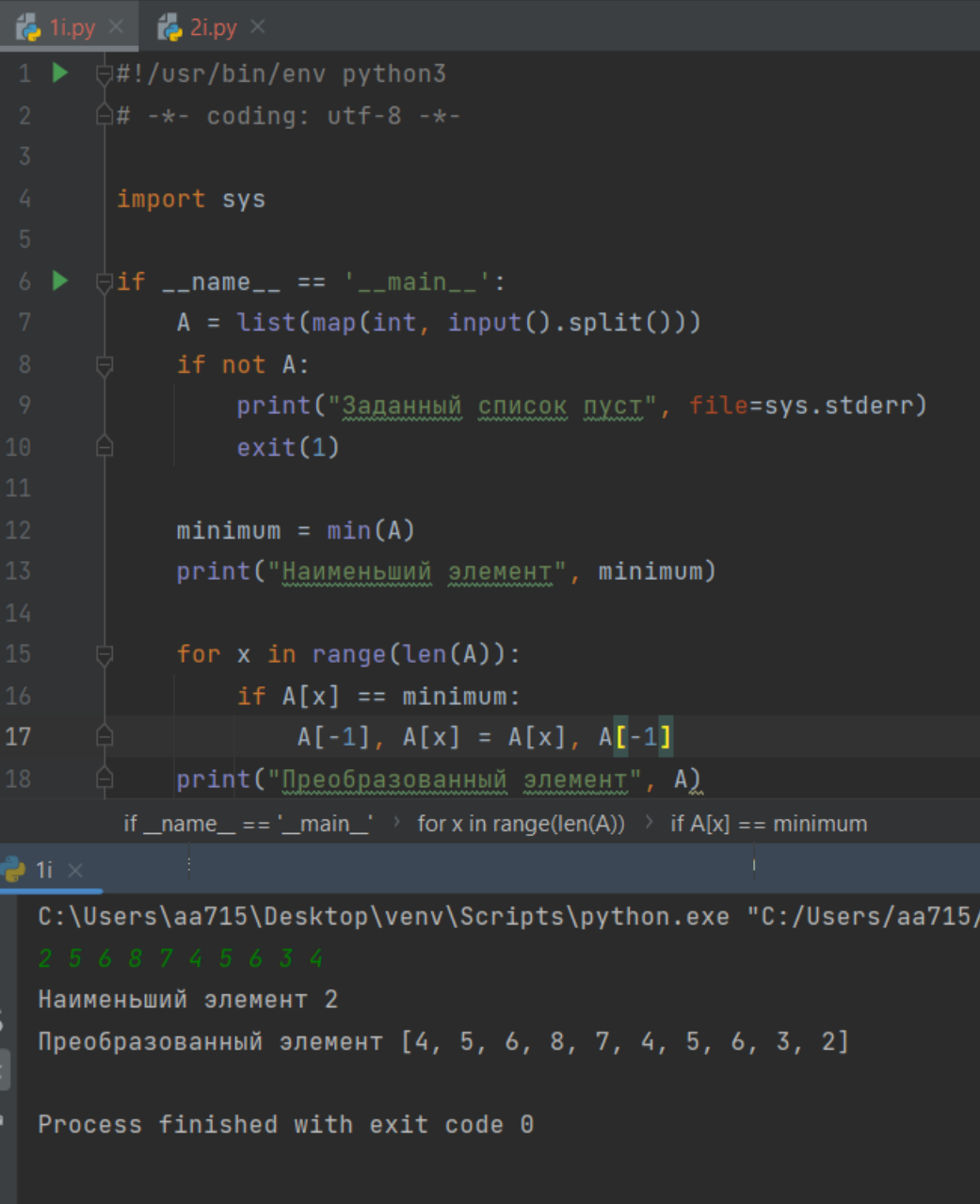
Process finished with exit code 0

Рисунок 2.2 – Результат выполнения примера №2

3. Выполнил 2 индивидуальных задания.

Задание №1. Ввести список A из 10 элементов, найти наименьший

элемент и переставить его с последним элементом. Преобразованный список вывести.



The image shows a Python IDE with two tabs: '1i.py' and '2i.py'. The '1i.py' tab is active and contains the following code:

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6 if __name__ == '__main__':
7     A = list(map(int, input().split()))
8     if not A:
9         print("Заданный список пуст", file=sys.stderr)
10        exit(1)
11
12    minimum = min(A)
13    print("Наименьший элемент", minimum)
14
15    for x in range(len(A)):
16        if A[x] == minimum:
17            A[-1], A[x] = A[x], A[-1]
18    print("Преобразованный элемент", A)
```

The output window at the bottom shows the execution results:

```
C:\Users\aa715\Desktop\venv\Scripts\python.exe "C:/Users/aa715/
2 5 6 8 7 4 5 6 3 4
Наименьший элемент 2
Преобразованный элемент [4, 5, 6, 8, 7, 4, 5, 6, 3, 2]

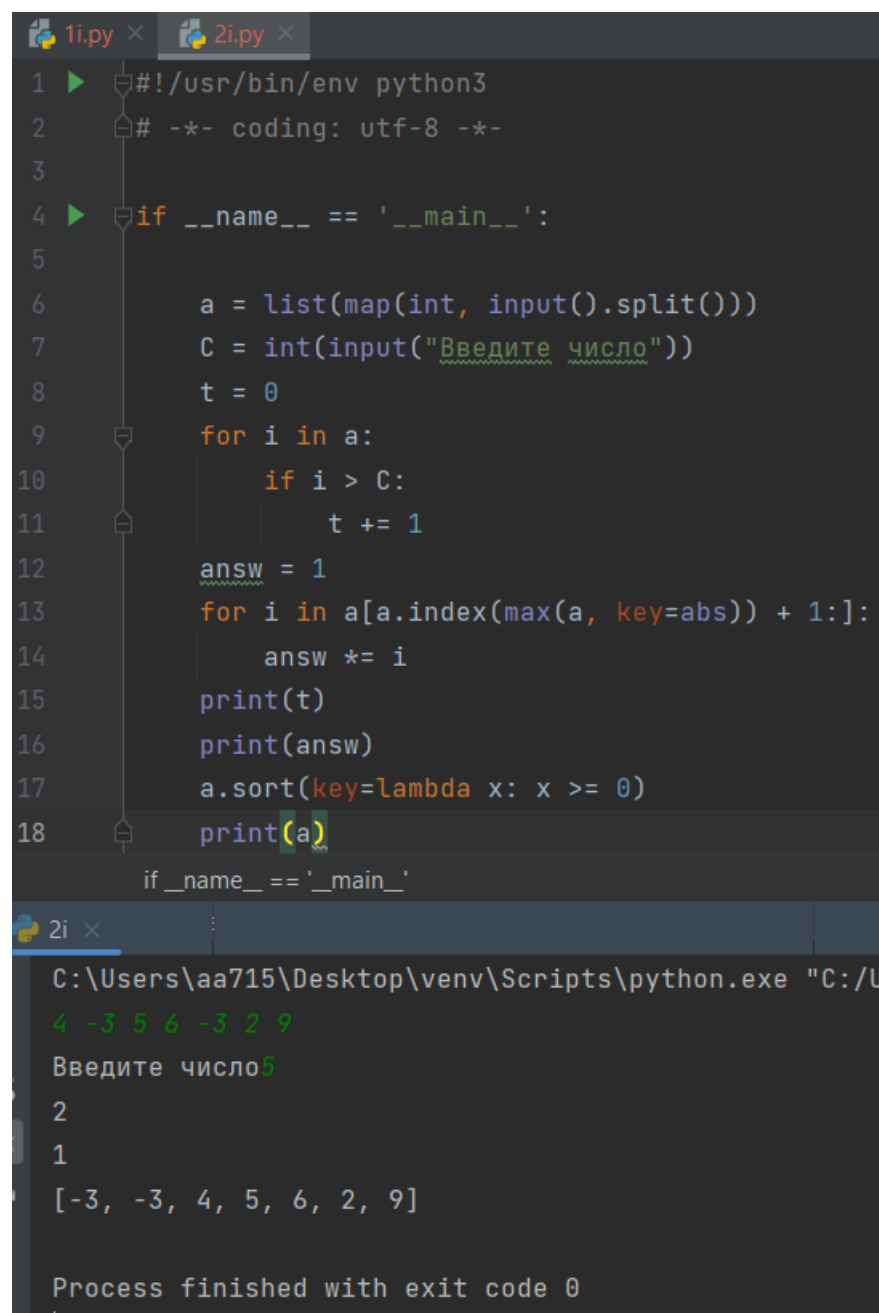
Process finished with exit code 0
```

Рисунок 3.1 – Выполненное индивидуальное задание №1

Индивидуальное задание №2. В списке, состоящем из вещественных элементов, вычислить:

- 1) количество элементов списка, больших C ;
- 2) произведение элементов списка, расположенных после максимального по модулю элемента.

Преобразовать список таким образом, чтобы сначала располагались все отрицательные элементы, а потом - все положительные (элементы, равные 0, считать положительными).



```
1  > #!/usr/bin/env python3
2  > # -*- coding: utf-8 -*-
3
4  > if __name__ == '__main__':
5
6      a = list(map(int, input().split()))
7      C = int(input("Введите число"))
8      t = 0
9      for i in a:
10         if i > C:
11             t += 1
12     answ = 1
13     for i in a[a.index(max(a, key=abs)) + 1:]:
14         answ *= i
15     print(t)
16     print(answ)
17     a.sort(key=lambda x: x >= 0)
18     print(a)
if __name__ == '__main__':

2i x
C:\Users\aa715\Desktop\venv\Scripts\python.exe "C:/U
4 -3 5 6 -3 2 9
Введите число5
2
1
[-3, -3, 4, 5, 6, 2, 9]
Process finished with exit code 0
```

Рисунок 3.2 – Индивидуальное задание №2

```

aa715@DESKTOP-70QJC9F MINGW64 ~/Desktop/2.4/2.4 (main)
$ git merge develop
Updating b04ec33..9a8b906
Fast-forward
 .../1i.py" | 18 ++++++++
 .../2i.py" | 18 ++++++++
 ...277\321\200\320\270\320\274\320\265\321\200.py" | 20 ++++++++
 ...277\321\200\320\270\320\274\320\265\321\200.py" | 34 ++++++++
 4 files changed, 90 insertions(+)
 create mode 100644 "...277\321\200\320\270\320\274\320\265\321\200.py"
 create mode 100644 "...277\321\200\320\270\320\274\320\265\321\200.py"
 create mode 100644 "...277\321\200\320\270\320\274\320\265\321\200.py"
 create mode 100644 "...277\321\200\320\270\320\274\320\265\321\200.py"

```

Рисунок 4.2 – Слил ветку develop с веткой main

Контрольные вопросы:

1. Что такое списки в языке Python?

Список (list) – это структура данных для хранения объектов различных типов. 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.

Списки в Python - упорядоченные изменяемые коллекции объектов произвольных типов (почти как массив, но типы могут отличаться).

2. Как осуществляется создание списка в Python?

Для создания списка нужно заключить элементы в квадратные скобки.

3. Как организовано хранение списков в оперативной памяти?

При создании списка 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?

Перебор элементов списка состоит в том, что мы в цикле просматриваем все элементы этого списка

Читать элементы списка можно с помощью следующего цикла:

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

```
for elem in my_list:
```

```
    print(elem)
```


Перебор элементов списка состоит в том, что мы в цикле просматриваем все элементы списка и, если нужно, выполняем с каждым из них некоторую операцию. Переменная цикла изменяется от 0 до N-1, где N – количество элементов списка, то есть в диапазоне range(N):

```
for i in range(N):
```

```
    A[i] += 1
```

в этом примере все элементы списка A увеличиваются на 1.

Если список изменять не нужно, для перебора его элементов удобнее всего использовать такой цикл:

```
for x in A:
```

5. Какие существуют арифметические операции со списками?

Для объединения списков можно использовать оператор сложения (+). Список можно повторить с помощью оператора умножения (*).

6. Как проверить есть ли элемент в списке?

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

```
lst = ['test', 'twest', 'twest', 'treast'] 'test' in lst
```

```
# Вывод: True 'toast' in lst # Вывод: False
```

7. Как определить число вхождений заданного элемента в списке?

Метод count можно использовать для определения числа сколько раз данный элемент встречается в списке.

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

Метод append можно использовать для добавления элемента в список. Метод insert можно использовать, чтобы вставить элемент в список.

9. Как выполнить сортировку списка?

Для сортировки списка нужно использовать метод `sort()`, в порядке возрастания будет(`list1.sort()`). Для сортировки списка в порядке убывания необходимо вызвать метод `sort` с аргументом `reverse()`=`True`(`list1.reverse()`).

10.Как удалить один или несколько элементов из списка?

Удалить элемент можно, написав его индекс (на основе его индекса) в методе `pop`(`list.pop(index)`). Если не указывать индекс, то функция удалит последний элемент.

Элемент можно удалить с помощью метода `remove` (пишется `my_list.remove(100)`).

Оператор `del` можно использовать для тех же целей `del list[index]`.

Можно удалить несколько элементов с помощью оператора среза.

Можно удалить все элементы из списка с помощью метода `clear` (`list.clear()`).

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

List Comprehensions чаще всего на русский язык переводят как абстракция списков или списковое включение, является частью синтаксиса языка, которая предоставляет простой способ построения 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.

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

Со списками, так же как и со строками, можно делать срезы. А именно:

$A[i:j]$ срез из $j-i$ элементов $A[i], A[i+1], \dots, A[j-1]$. $A[i:j:-1]$

срез из $i-j$ элементов $A[i], A[i-1], \dots, A[j+1]$ (то есть меняется порядок элементов).

$A[i:j:k]$ срез с шагом k : $A[i], A[i+k], A[i+2*k], \dots$. Если значение $k < 0$, то элементы идут в противоположном порядке. Каждое из чисел i или j может отсутствовать, что означает “начало строки” или “конец строки”. Списки, в отличие от строк, являются изменяемыми объектами: можно отдельному элементу списка присвоить новое значение. Но можно менять и целиком срезы.

13. Какие существуют функции агрегации для работы со списками?

Для работы со списками Python предоставляет следующие функции:

- $\text{len}(L)$ - получить число элементов в списке L
- $\text{min}(L)$ - получить минимальный элемент списка L
- $\text{max}(L)$ - получить максимальный элемент списка L
- $\text{sum}(L)$ - получить сумму элементов списка L , если список L содержит только числовые значения.

14. Как создать копию списка?

Операция присваивания не копирует объект, он лишь создаёт ссылку на объект. Для изменяемых коллекций, или для коллекций, содержащих изменяемые элементы, часто необходима такая копия, чтобы её можно было изменить, не изменяя оригинал. Данный модуль предоставляет общие (поверхностная и глубокая) операции копирования.

`Spisok = copy.copy(oldspisok)`

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

Функция `sorted()` в Python возвращает отсортированный список из элементов в итерируемом объекте. `list.sort()` на 13% быстрее, чем `sorted()`.

Ещё одно отличие заключается в том, что метод `list.sort()` определён только для списков, в то время как `sorted()` работает со всеми итерируемыми объектами.

Вывод: приобрёл навыки для работы со списками, при написании программ с помощью языка программирования Python.