

**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

Богадунов В.И. « » _____ 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.


[Import a repository.](#)


Owner * Repository name *

 ItsMyLife1337 / ForPrograms2.4 

Great repository names are short and memorable. Need inspiration? How about [psychic-invention?](#)

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.


☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python ▼

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License ▼

 You are creating a public repository in your personal account.

[Create repository](#)

Рисунок 1.1 – Создал репозиторий

```
280 lines (222 sloc) | 5.7 KB
1
2 # Created by https://www.toptal.com/developers/gitignore/api/pycharm,python
3 # Edit at https://www.toptal.com/developers/gitignore?templates=pycharm,python
4
5 ### PyCharm ###
6 # Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio, WebStorm and Rider
7 # Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839
8
9 # User-specific stuff
10 .idea/**/workspace.xml
11 .idea/**/tasks.xml
12 .idea/**/usage.statistics.xml
13 .idea/**/dictionaries
14 .idea/**/shelf
15
16 # AWS User-specific
17 .idea/**/aws.xml
18
19 # Generated files
20 .idea/**/contentModel.xml
```

Рисунок 1.2 – Изменённый файл .gitignore

```
c:\Users\Admin\Desktop\git>git clone https://github.com/ItsMyLife1337/ForPrograms2.4.git
Cloning into 'ForPrograms2.4'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), 4.41 KiB | 563.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.

c:\Users\Admin\Desktop\git>
```

Рисунок 1.3 – Клонирование репозитория

```
c:\Users\Admin\Desktop\git\ForPrograms2.4>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/Admin/Desktop/git/ForPrograms2.4/.git/hooks]

c:\Users\Admin\Desktop\git\ForPrograms2.4>
```

Рисунок 1.4 – Организация репозитория в соответствии с моделью ветвления git-flow

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

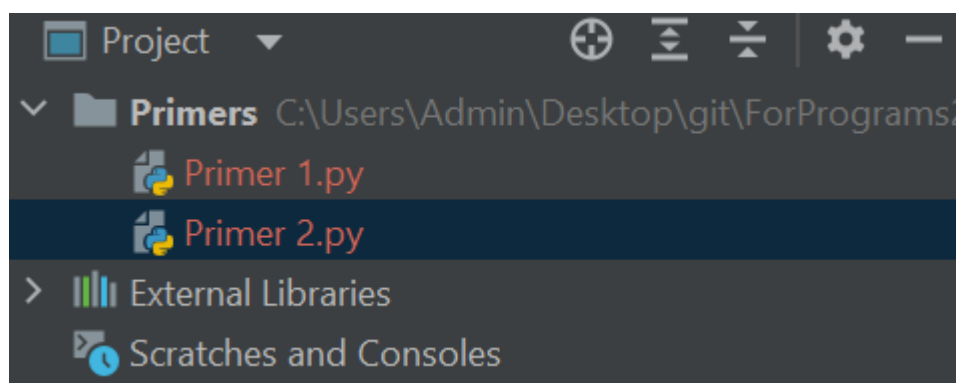
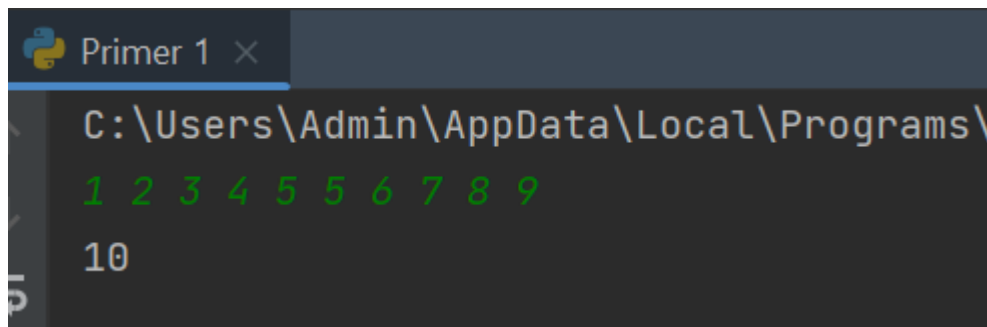
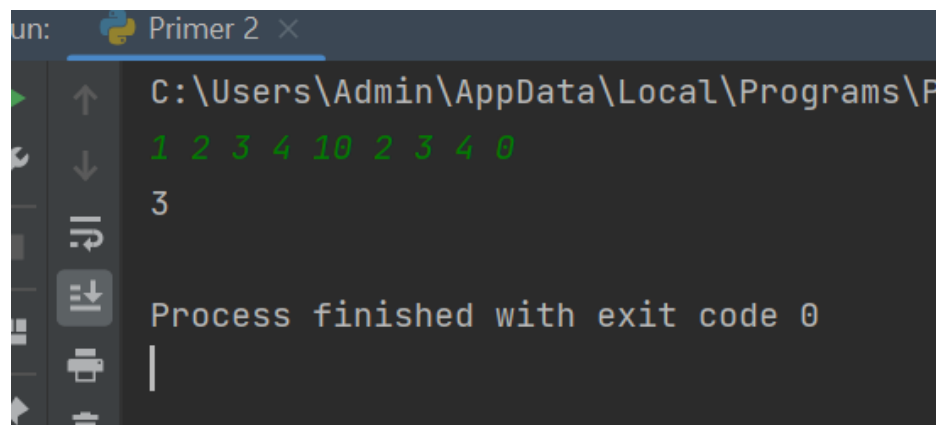


Рисунок 2.1 – Созданные проекты



```
Primer 1 x
C:\Users\Admin\AppData\Local\Programs\
1 2 3 4 5 5 6 7 8 9
10
```

Рисунок 2.2 – Результат выполнения примера №1

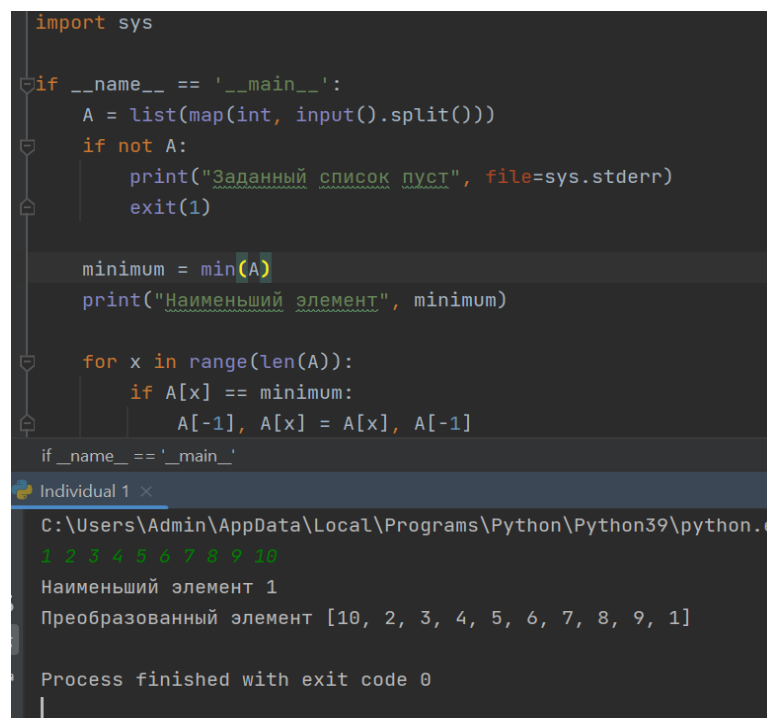


```
Primer 2 x
C:\Users\Admin\AppData\Local\Programs\
1 2 3 4 10 2 3 4 0
3
Process finished with exit code 0
```

Рисунок 2.3 – Результат выполнения примера №2

3. Выполнил 2 индивидуальных задания. Вариант – №3.

Задание №1. Ввести список A из 10 элементов, найти наименьший элемент и переставить его с последним элементом. Преобразованный список вывести.



```
import sys

if __name__ == '__main__':
    A = list(map(int, input().split()))
    if not A:
        print("Заданный список пуст", file=sys.stderr)
        exit(1)

    minimum = min(A)
    print("Наименьший элемент", minimum)

    for x in range(len(A)):
        if A[x] == minimum:
            A[-1], A[x] = A[x], A[-1]

if __name__ == '__main__':
```

```
Individual 1 x
C:\Users\Admin\AppData\Local\Programs\Python\Python39\python.exe
1 2 3 4 5 6 7 8 9 10
Наименьший элемент 1
Преобразованный элемент [10, 2, 3, 4, 5, 6, 7, 8, 9, 1]
Process finished with exit code 0
```

Рисунок 3.1 – Выполненное индивидуальное задание №1

Индивидуальное задание №2. В -3, 3. В списке, состоящем из целых элементов, вычислить:

1. произведение элементов списка с четными номерами;
2. сумму элементов списка, расположенных между первым и последним нулевыми элементами.

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

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == '__main__':
    lis = []
    m, c, k = 1, 0, 0

    n = int(input('Введите количество чисел в списке: '))
    print('Введите числа в список: ')
    for i in range(n):
        lis.append(int(input()))
        if (i+1) % 2 == 0:
            m *= lis[i]

Individual 2 x
Введите количество чисел в списке: 5
Введите числа в список:
2
2
1
3
2
сумма между нулями 0 произведение четных 6
преобразованный список [3, 2, 2, 2, 1]
```

Рисунок 3.2 – Индивидуальное задание №2

```
c:\Users\Admin\Desktop\git\ForPrograms2.4>git commit -m "all but gold"
[develop 58c0fcd] all but gold
5 files changed, 100 insertions(+), 1 deletion(-)
create mode 100644 Primers/Primer 1.py
create mode 100644 Primers/Primer 2.py
create mode 100644 individual/Individual 1.py
create mode 100644 individual/Individual 2.py

c:\Users\Admin\Desktop\git\ForPrograms2.4>
```

Рисунок 4.1 – Сделал коммит всех изменений

```
c:\Users\Admin\Desktop\git\ForPrograms2.4>git merge develop
Updating ab9241e..58c0fcd
Fast-forward
 .gitignore           | 2 +-
 Primers/Primer 1.py  | 20 +++++
 Primers/Primer 2.py  | 34 +++++
 individual/Individual 1.py | 18 +++++
 individual/Individual 2.py | 27 +++++
 5 files changed, 100 insertions(+), 1 deletion(-)
 create mode 100644 Primers/Primer 1.py
 create mode 100644 Primers/Primer 2.py
 create mode 100644 individual/Individual 1.py
 create mode 100644 individual/Individual 2.py

c:\Users\Admin\Desktop\git\ForPrograms2.4>
```

Рисунок 4.2 – Слил ветку develop с веткой main

```
c:\Users\Admin\Desktop\git\ForPrograms2.4>git push --force
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 4 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (16/16), 6.43 KiB | 1.61 MiB/s, done.
Total 16 (delta 2), reused 6 (delta 1), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/ItsMyLife1337/ForPrograms2.4.git
+ b015904...58c0fcd main -> main (forced update)
```

Рисунок 4.3 – Отправил изменения на удалённый репозиторий

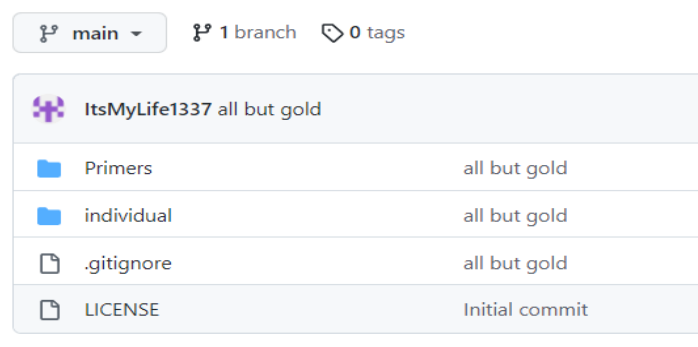


Рисунок 4.4 – Зафиксировал изменения на удалённом репозитории

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.