

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

**Кафедра
инфокоммуникаций
Институт цифрового
развития**

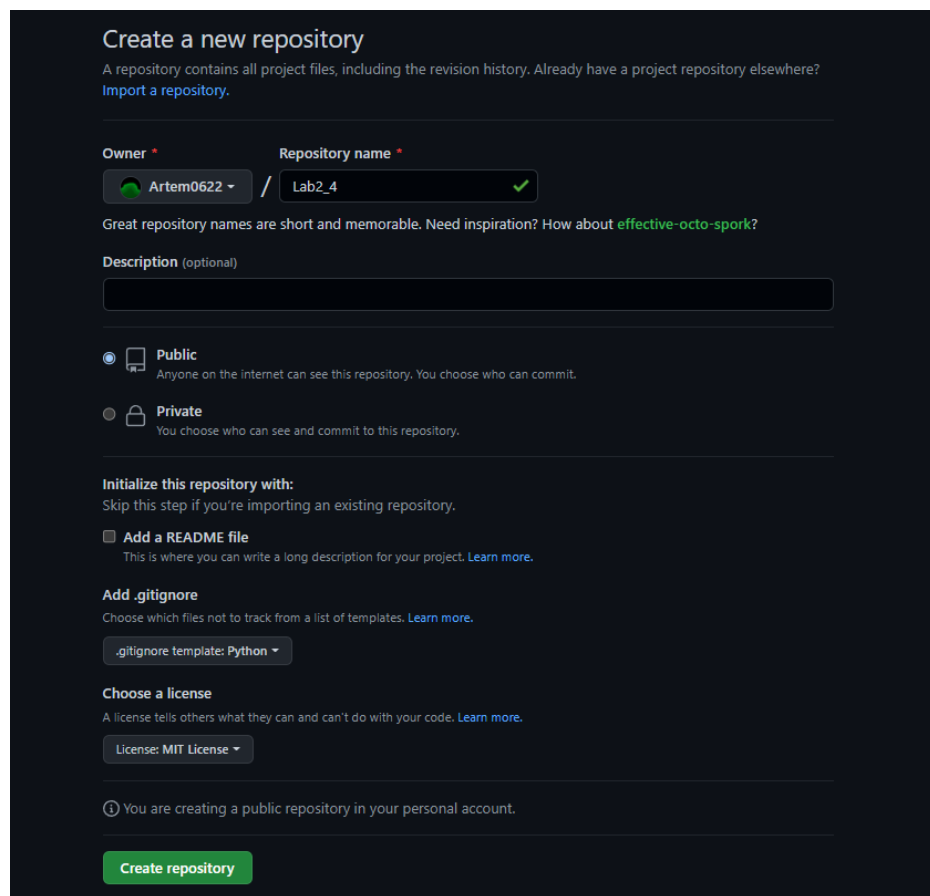
ОТЧЁТ
по лабораторной работе №2.4
Дисциплина: «Основы кроссплатформенного
программирования» Тема: «Работа со списками в языке Python»

Выполнил: студент 1 курса
группы ИВТ-б-о-21-1
Богдашов Артём
Владимирович

Ставрополь 2022

Выполнение работы:

1. Создал репозиторий в GitHub «rep 2.2» в который добавил .gitignore, который дополнил правила для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на лок. сервер и организовал в соответствии с моделью ветвления git-flow.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner ^{*} / Repository name ^{*}

Artem0622 / Lab2_4 ✓

Great repository names are short and memorable. Need inspiration? How about [effective-octo-spork?](#)

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 ▾

ⁱ You are creating a public repository in your personal account.

Create repository

Рисунок 1.1 Создание репозитория

```
C:\Users\Kwil>cd/d C:\Games\Программы\СУ\1.2КС\Программирование\2.4
C:\Games\Программы\СУ\1.2КС\Программирование\2.4>git clone https://github.com/Artem0622/Lab2_4.git
Cloning into 'Lab2_4'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Рисунок 1.2 Клонирование репозитория

```

C:\Games\Программы\СУ\1.2К\Программирование\2.4\Lab2_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:/Games/Программы/СУ/1.2К/Программирование/2.4/Lab2_4/.git/hooks]

```

Рисунок 1.3 Организация репозитория в соответствии с моделью ветвления git-flow

```

*.gitignore - Блокнот
Файл Правка Формат Вид Справка
.idea/
# Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
# Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm

### PyCharm ###
# Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio, WebStorm and Rider
# Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839

# User-specific stuff
.idea/**/workspace.xml
.idea/**/tasks.xml
.idea/**/usage.statistics.xml
.idea/**/dictionaries
.idea/**/shelf

# AWS User-specific
.idea/**/aws.xml

# Generated files
.idea/**/contentModel.xml

# Sensitive or high-churn files

```

Рисунок 1.4 Изменение .gitignore

Пример 1. Ввести список А из 10 элементов, найти сумму элементов, меньших по модулю 5, и вывести ее на экран.

```

IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:/Games/Программы/СУ/1.2К/Программирование/Коды/2_4/Примеры/1.py ==
15 88 9 4 2 7 19 8 11 5
6
>>>

```

Рисунок 2.1 Результат выполнения программы

Пример 2. Написать программу, которая для целочисленного списка определяет, сколько положительных элементов располагается между его максимальным и минимальным элементами.

```
>>> == RESTART: C:/Games/Программы/CV/1.2КС/Программирование/Коды/2_4/Примеры/2.py =  
-14 5 -8 -7 45 -4 77 -1 2  
2  
~ ~ ~
```

Рисунок 2.2 Результат выполнения программы

Индивидуальные задание

Вариант 3

1. Ввести список A из 10 элементов, найти наименьший элемент и переставить его с последним элементом. Преобразованный список вывести.

```
= RESTART: C:/Games/Программы/CV/1.2КС/Программирование/Коды/2_4/Индивидуальные  
задания/1.py  
1 2 3 4 -5 6 7 8 9 -10  
-10  
[1, 2, 3, 4, -5, 6, 7, 8, -10, 9]
```

Рисунок 3.1 Результат выполнения программы

2. 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).

```
IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Games/Программы/CV/1.2KC/Программирование/Коды/2_4/Индивидуальные задания/2.py
Введите элементы списка через пробел:
0 1 2 3 4 5 6 0 -5 8
1. Произведение элементов списка с четными номерами: 0
2. Сумма элементов списка, расположенных между первым и последним нулевыми элементами: 21
3. Преобразованный список: [8, 6, 5, 4, 3, 2, 1, 0, 0, -5]
>>>
= RESTART: C:/Games/Программы/CV/1.2KC/Программирование/Коды/2_4/Индивидуальные задания/2.py
Введите элементы списка :
1 0 2 3 4 5 6 0 -5
1. Произведение элементов списка с четными номерами: 0
2. Сумма элементов списка, расположенных между первым и последним нулевыми элементами: 20
3. Преобразованный список: [6, 5, 4, 3, 2, 1, 0, 0, -5]
>>>
```

Рисунок 3.2 Результат выполнения программы

```
C:\Games\Программы\CV\1.2KC\Программирование\2.4\Lab2_4>git add .
C:\Games\Программы\CV\1.2KC\Программирование\2.4\Lab2_4>git commit -m "saving programs"
[develop b15424e] saving programs
5 files changed, 231 insertions(+), 3 deletions(-)
create mode 100644 "\320\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\321\213\320\265\320\267\320\260\320\264\320\260\320\275\320\270\321\217\1.py"
create mode 100644 "\320\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\321\213\320\265\320\267\320\260\320\264\320\260\320\275\320\270\321\217\2.py"
create mode 100644 "\320\237\321\200\320\270\320\274\320\265\321\200\321\213\1.py"
create mode 100644 "\320\237\321\200\320\270\320\274\320\265\321\200\321\213\2.py"
```

Рисунок 4.1 Коммит изменений

```
C:\Games\Программы\CV\1.2KC\Программирование\2.4\Lab2_4>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Рисунок 4.2 Переход на ветку main

```
C:\Games\Программы\CV\1.2KC\Программирование\2.4\Lab2_4>git merge develop
Updating 2821a92..b15424e
Fast-forward
 .gitignore | 153 ++++++
 .../1.py" | 17 +++
 .../2.py" | 17 +++
 .../1.py" | 16 +++
 .../2.py" | 31 +++++
 5 files changed, 231 insertions(+), 3 deletions(-)
 create mode 100644 "\320\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\321\213\320\265\320\267\320\260\320\264\320\260\320\275\320\270\321\217\1.py"
 create mode 100644 "\320\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\321\213\320\265\320\267\320\260\320\264\320\260\320\275\320\270\321\217\2.py"
 create mode 100644 "\320\237\321\200\320\270\320\274\320\265\321\200\321\213\1.py"
 create mode 100644 "\320\237\321\200\320\270\320\274\320\265\321\200\321\213\2.py"
```

Рисунок 4.3 Слияние ветки main с develop

```
C:\Games\Программы\СУ\1.2КС\Программирование\2.4\Lab2_4>git push
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 12 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 4.49 KiB | 2.25 MiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Artem0622/Lab2_4.git
2821a92..b15424e  main -> main
```

Рисунок 4.4 Пуш изменений

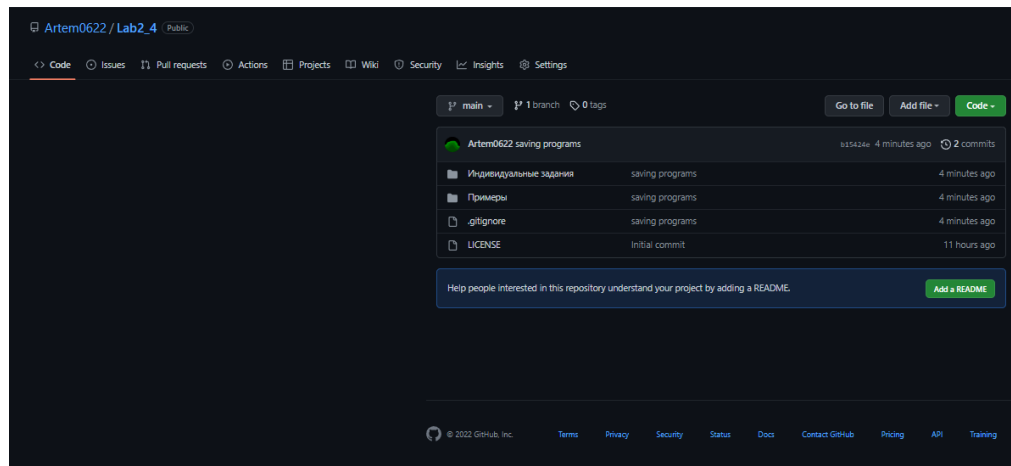


Рисунок 4.5 Изменение на уд сервере

Контрольные вопросы:

1. Что такое списки в языке Python?

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

2. Как осуществляется создание списка в Python?

Для создания списка нужно заключить элементы в квадратные скобки.

3. Как организовано хранение списков в оперативной памяти?

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?

for elem in my_list:

5. Какие существуют арифметические операции со списками?

+, *

6. Как проверить есть ли элемент в списке?

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

7. Как определить число вхождений заданного элемента в списке?

`list.count('элемент')`

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

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

9. Как выполнить сортировку списка?

`list.sort()`

10. Как удалить один или несколько элементов из списка?

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

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.

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

`list[<начало среза>:<конец среза>:<шаг>]`

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

14. Как создать копию списка?

Для создания копии списка необходимо использовать либо метод `copy`,

либо использовать оператор среза

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

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