

**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

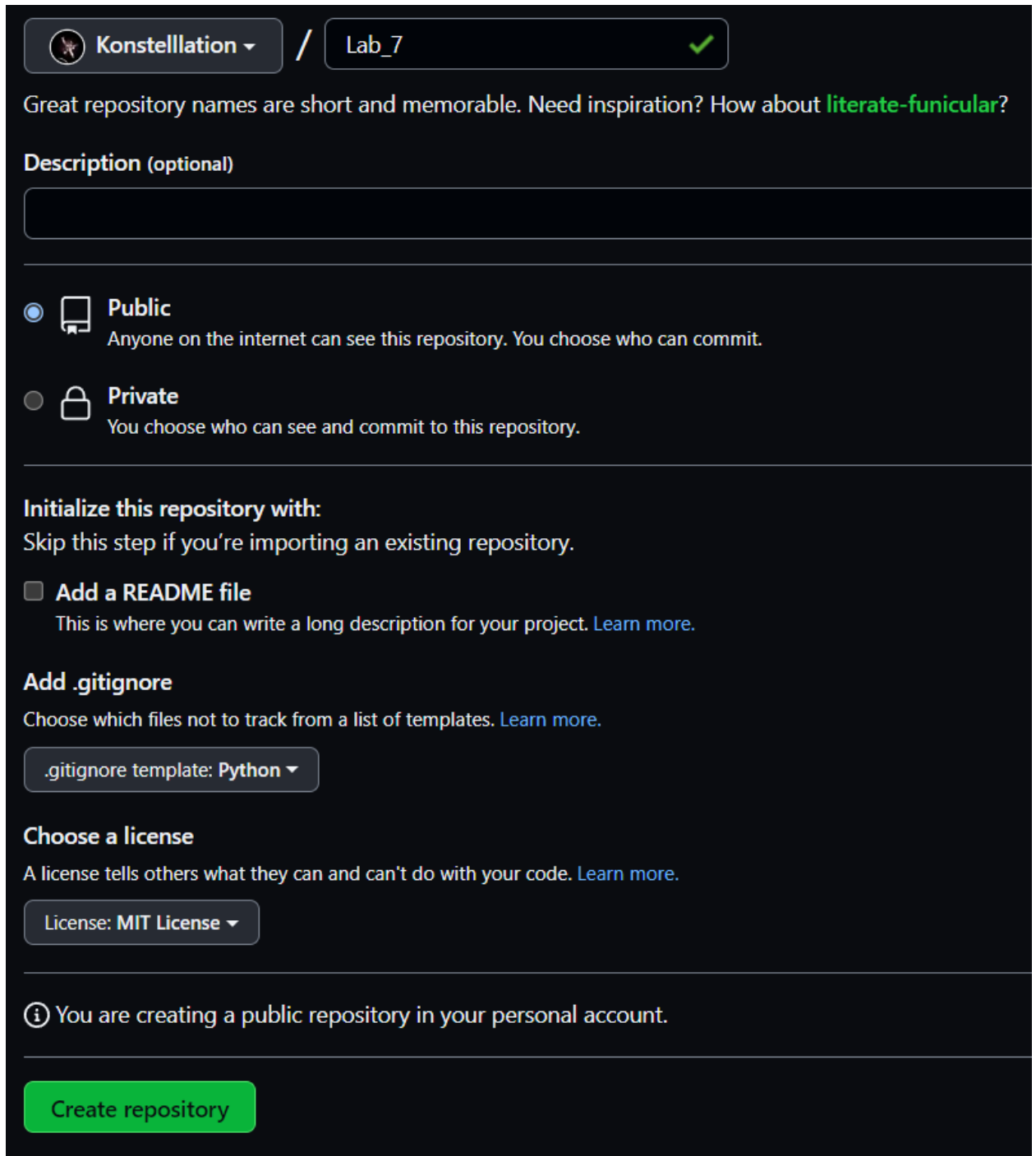
группы ИВТ-б-о-21-1

Харченко Богдан Романович

Ставрополь 2022

## Выполнение работы:

1. Создал репозиторий в GitHub «rep 2.4» 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.



The screenshot shows the GitHub 'Create repository' form. At the top, there's a header with the username 'Konstellation' and a dropdown arrow, followed by a slash and the repository name 'Lab\_7' with a green checkmark. Below this is a tip: 'Great repository names are short and memorable. Need inspiration? How about **literate-funicular**?'. The 'Description (optional)' field is empty. There are two radio buttons for visibility: 'Public' (selected) and 'Private'. Below these are instructions for each. The 'Initialize this repository with:' section has a checkbox for 'Add a README file'. The 'Add .gitignore' section has a dropdown menu showing '.gitignore template: Python'. The 'Choose a license' section has a dropdown menu showing 'License: MIT License'. At the bottom, there's an information icon and a message: 'You are creating a public repository in your personal account.' and a large green 'Create repository' button.

Konstellation / Lab\_7 ✓

Great repository names are short and memorable. Need inspiration? How about **literate-funicular**?

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 Создание репозитория

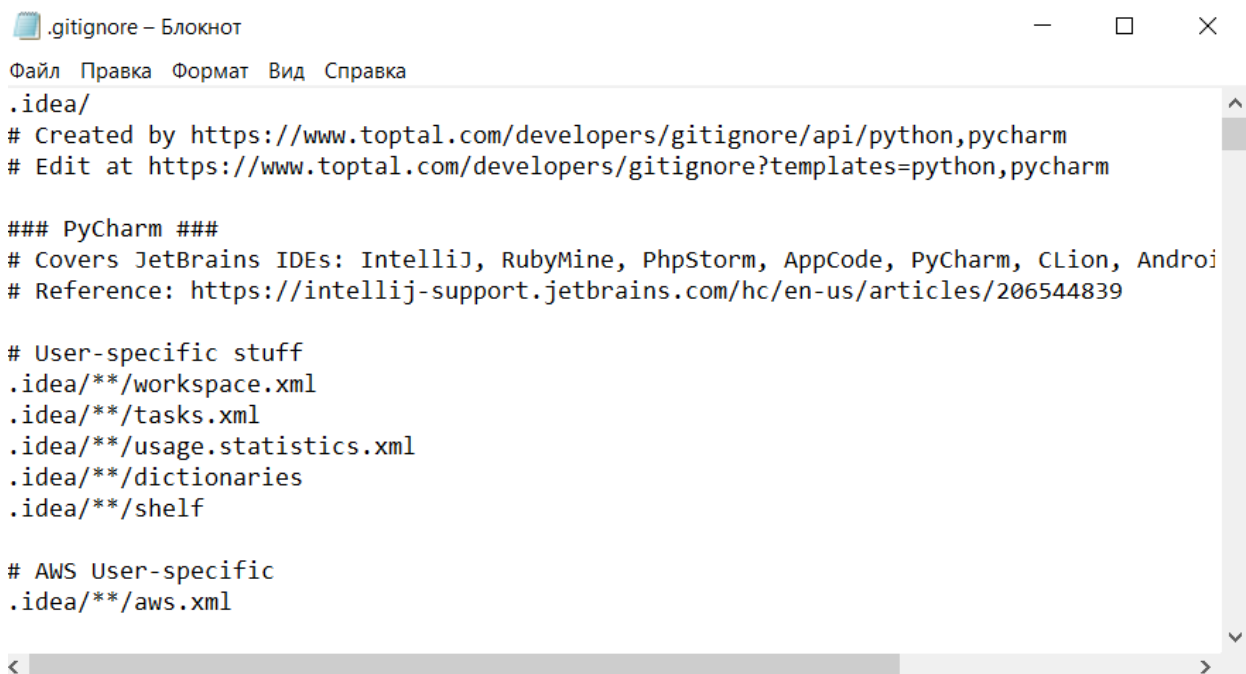
```
C:\>git clone https://github.com/Konstellation/Lab_7.git
Cloning into 'Lab_7'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Рисунок 1.2 Клонирование репозитория

```
C:\Lab_7>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:/Lab_7/.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
# 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
```

Рисунок 1.4 Изменение .gitignore

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

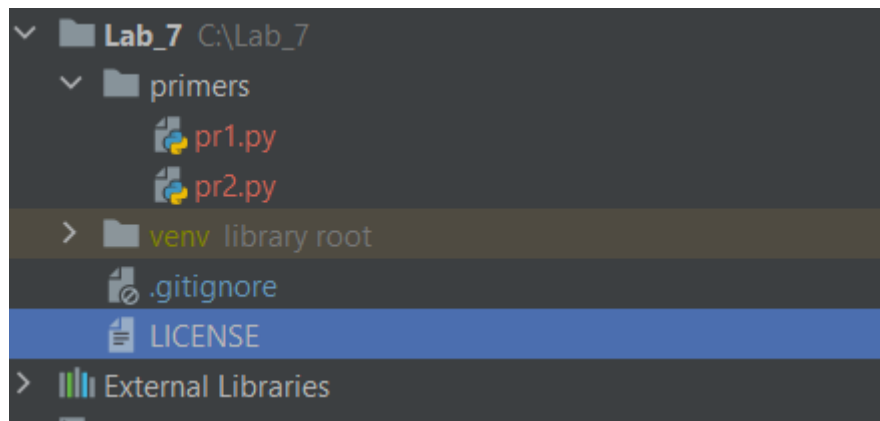


Рисунок 2.1 Создание проекта в PyCharm

```
1 5 4 6 8 7 9 5 1 3
9
```

Рисунок 2.2 Рез-т выполнения программы

```
2 3 6 5 4 8 9 4 8 7
13
```

Рисунок 2.3 Рез-т выполнения программы

3. (21 вариант). Выполнил 2 индивидуальных задания.

```
1 -й учащийся
Введите оценку за алгебру: 2
Введите оценку за геометрию: 5
Введите оценку за физику: 4
2 -й учащийся
Введите оценку за алгебру: 3
Введите оценку за геометрию: 6
Введите оценку за физику: 5
3 -й учащийся
Введите оценку за алгебру: 2
Введите оценку за геометрию: 4
Введите оценку за физику: 5
Лучше всего успеваемость по геометрии
```

Рисунок 3.1 Вывод программы индивидуального задания 1

```

Введите количество переменных: 7
1
9
2
3
4
10
12
Сумма положительных элементов списка = 33
Произведение элементов списка, расположенных между максимальным по модулю и минимальным по модулю элементами = 3600
Отсортированный список по убыванию: [12, 10, 5, 4, 2, -1, -9]

```

Рисунок 3.2 Вывод программы индивидуального задания 2

4. Сделал коммит, выполнил слияние с веткой main, и запустил изменения в уд. репозиторий.

```

C:\Lab_7>git add .

C:\Lab_7>git commit -m "added project"
[develop 9e29dd0] added project
5 files changed, 249 insertions(+), 3 deletions(-)
create mode 100644 ind/ind1.py
create mode 100644 ind/ind2.py
create mode 100644 primers/pr1.py
create mode 100644 primers/pr2.py

C:\Lab_7>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

```

Рисунок 4.1 коммит изменений и переход на ветку main

```

C:\Lab_7>git merge develop
Updating b97e126..9e29dd0
Fast-forward
 .gitignore      | 153 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ind/ind1.py     | 27 +++++
 ind/ind2.py     | 35 +++++
 primers/pr1.py  | 20 +++++
 primers/pr2.py  | 17 +++++
 5 files changed, 249 insertions(+), 3 deletions(-)
 create mode 100644 ind/ind1.py
 create mode 100644 ind/ind2.py
 create mode 100644 primers/pr1.py
 create mode 100644 primers/pr2.py

```

Рисунок 4.2 Слияние ветки main с develop

```
C:\Lab_7>git push
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 4.20 KiB | 2.10 MiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Konstellation/Lab_7.git
b97e126..9e29dd0  main -> main
```

Рисунок 4.3 Пуш изменений на удаленный сервер

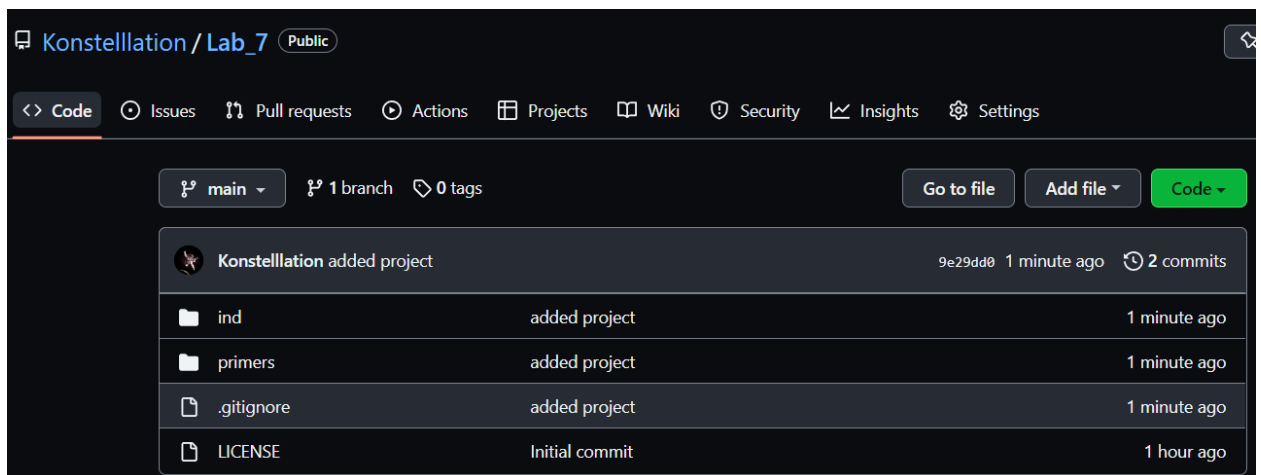


Рисунок 4.4 Изменения на удаленном сервере

### Контр. вопросы и ответы на них:

#### 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.