

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №7
дисциплины «Основы программной инженерии»

Выполнил:
Звездин Алексей Сергеевич
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Ход работы

1. Я изучил теоретический материал работы

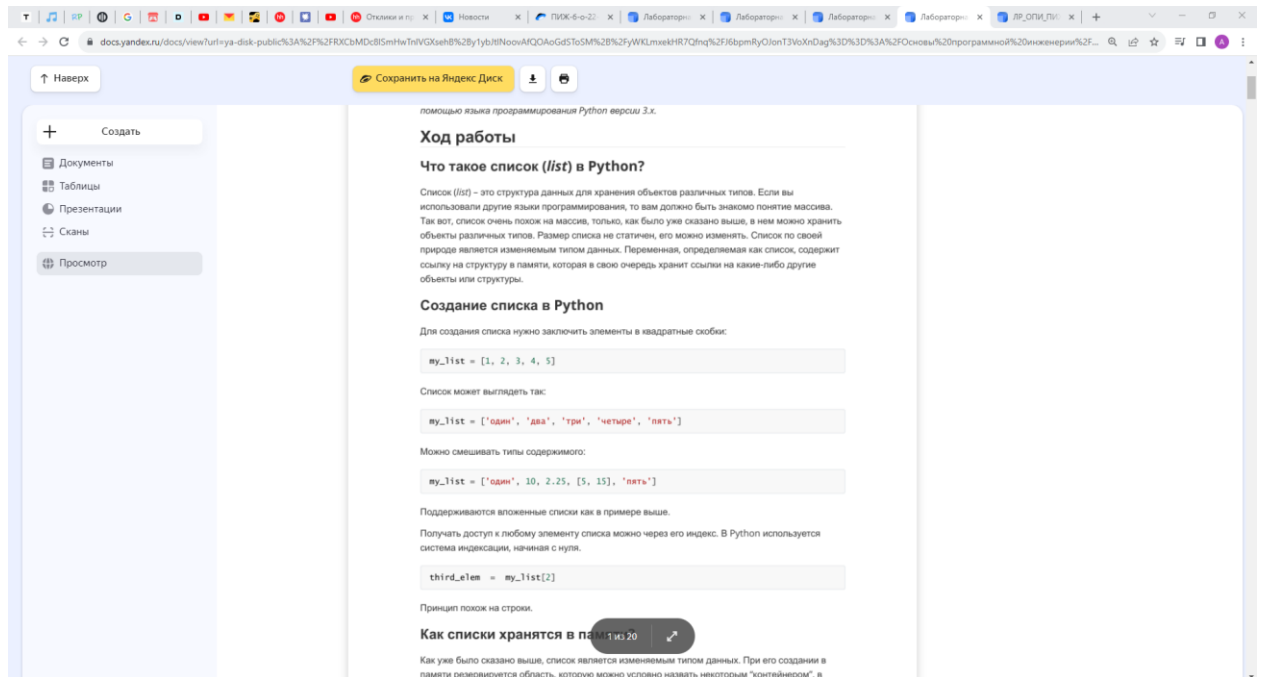


Рисунок 1.1 – Изучение материала для лабораторной работы

2. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python

Create a new repository

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

Required fields are marked with an asterisk (*).

Owner *

InternetHacker1123

/

Repository name *

laba_2(4)_7

✔

Your new repository will be created as lab_2-4-7.
The repository name can only contain ASCII letters, digits, and the characters `.`, `-`, and `_`.

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

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:

☒ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python

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

Choose a license

License: MIT License

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

This will set `main` as the default branch. Change the default name in your [settings](#).

ⓘ

 You are creating a public repository in your personal account.

Рисунок 2.1 – Настройка репозитория

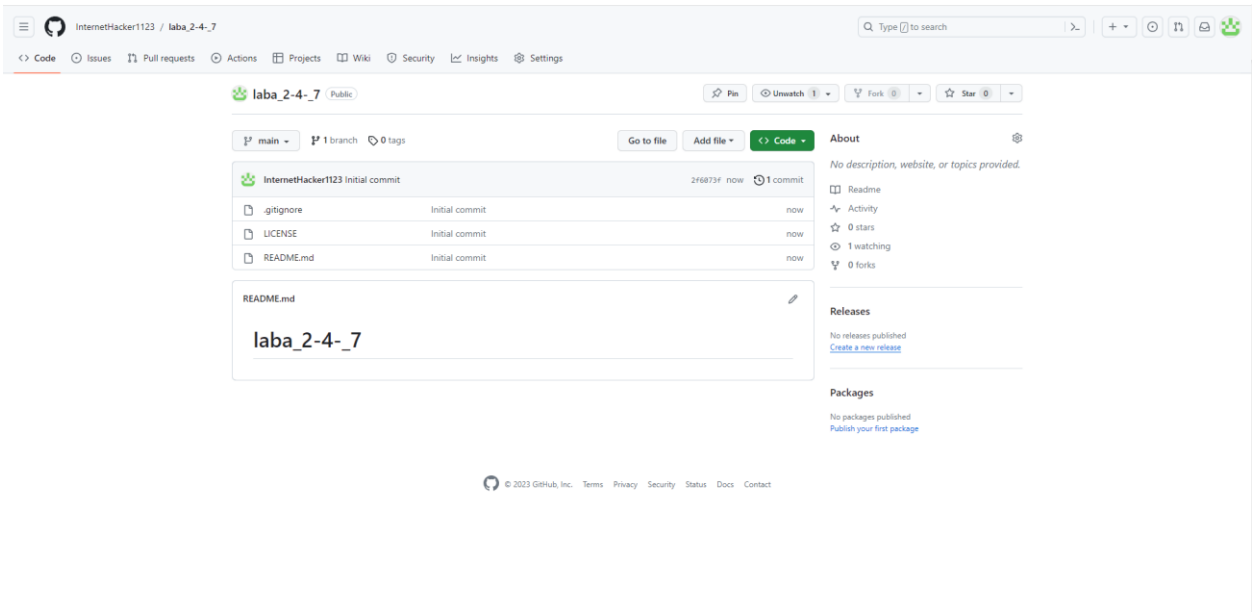


Рисунок 2.2 – Готовый репозиторий

3. Выполняю клонирование созданного репозитория



```
C:\Windows\system32\cmd.exe

C:\Users\tyt\Desktop\SE\laba7>git clone https://github.com/InternetHacker1123/laba_2-4-_7.git
Cloning into 'laba_2-4-_7'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

C:\Users\tyt\Desktop\SE\laba7>
```

Рисунок 3.1 – Клонирование репозитория на локальный диск

4. Дополнил файл .gitignore необходимыми правилами для работы с VS Code



```
# Cython debug symbols
cython_debug/

.vscode/*
!.vscode/settings.json
!.vscode/tasks.json
!.vscode/launch.json
!.vscode/extensions.json
!.vscode/*.code-snippets

# Local History for Visual Studio Code
.history/

# Built Visual Studio Code Extensions
*.vsix

# PyCharm
# JetBrains specific template is maintained in a separate JetBrains.gitignore that can
# be found at https://github.com/github/gitignore/blob/main/Global/JetBrains.gitignore
# and can be added to the global gitignore or merged into this file. For a more nuclear
# option (not recommended) you can uncomment the following to ignore the entire idea folder.
#.idea/
```

Рисунок 4.1 – .gitignore для VS Code

5. Организовал свой репозиторий в соответствии с моделью ветвления git-flow

```
C:\Windows\system32\cmd.exe

C:\Users\tyt\Desktop\SE\laba7\laba_2-4-_7>git branch develop
C:\Users\tyt\Desktop\SE\laba7\laba_2-4-_7>git checkout develop
Switched to branch 'develop'
M      .gitignore

C:\Users\tyt\Desktop\SE\laba7\laba_2-4-_7>git push -u origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/InternetHacker1123/laba_2-4-_7/pull/new/develop
remote:
To https://github.com/InternetHacker1123/laba_2-4-_7.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

C:\Users\tyt\Desktop\SE\laba7\laba_2-4-_7>
```

Рисунок 5.1 – Создание ветки develop от ветки main

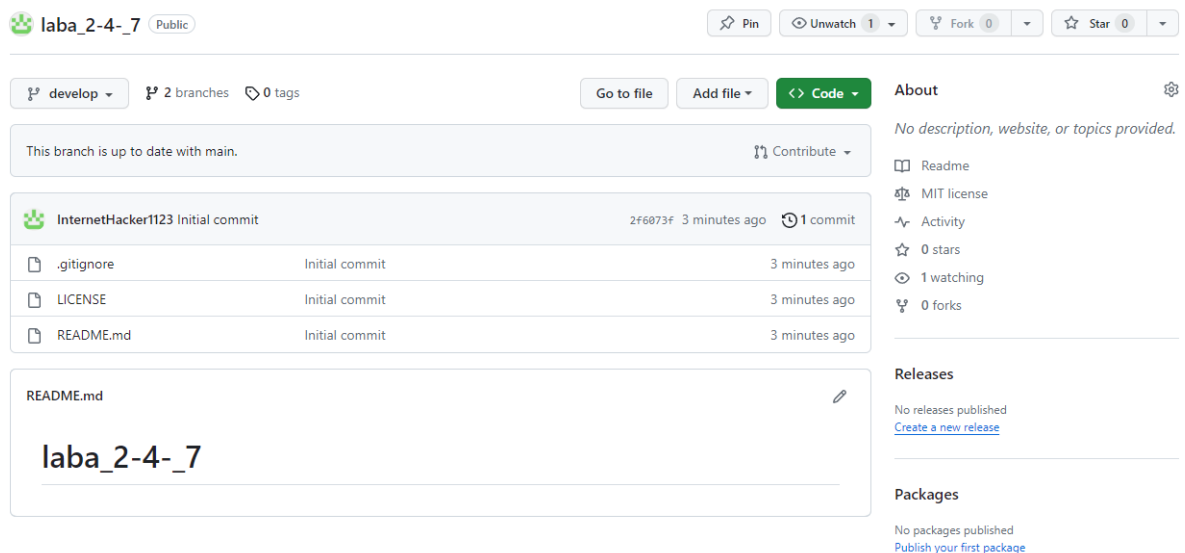


Рисунок 5.2 – Ветка develop на GitHub

6. Создал проект PyCharm в папке репозитория

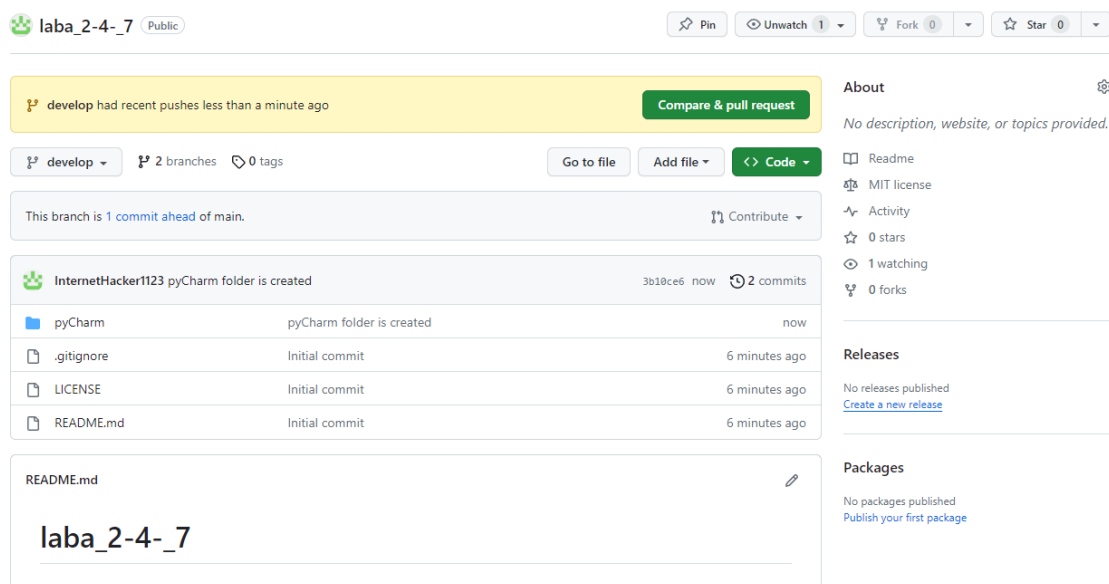


Рисунок 6.1 – Репозиторий с проектом PyCharm

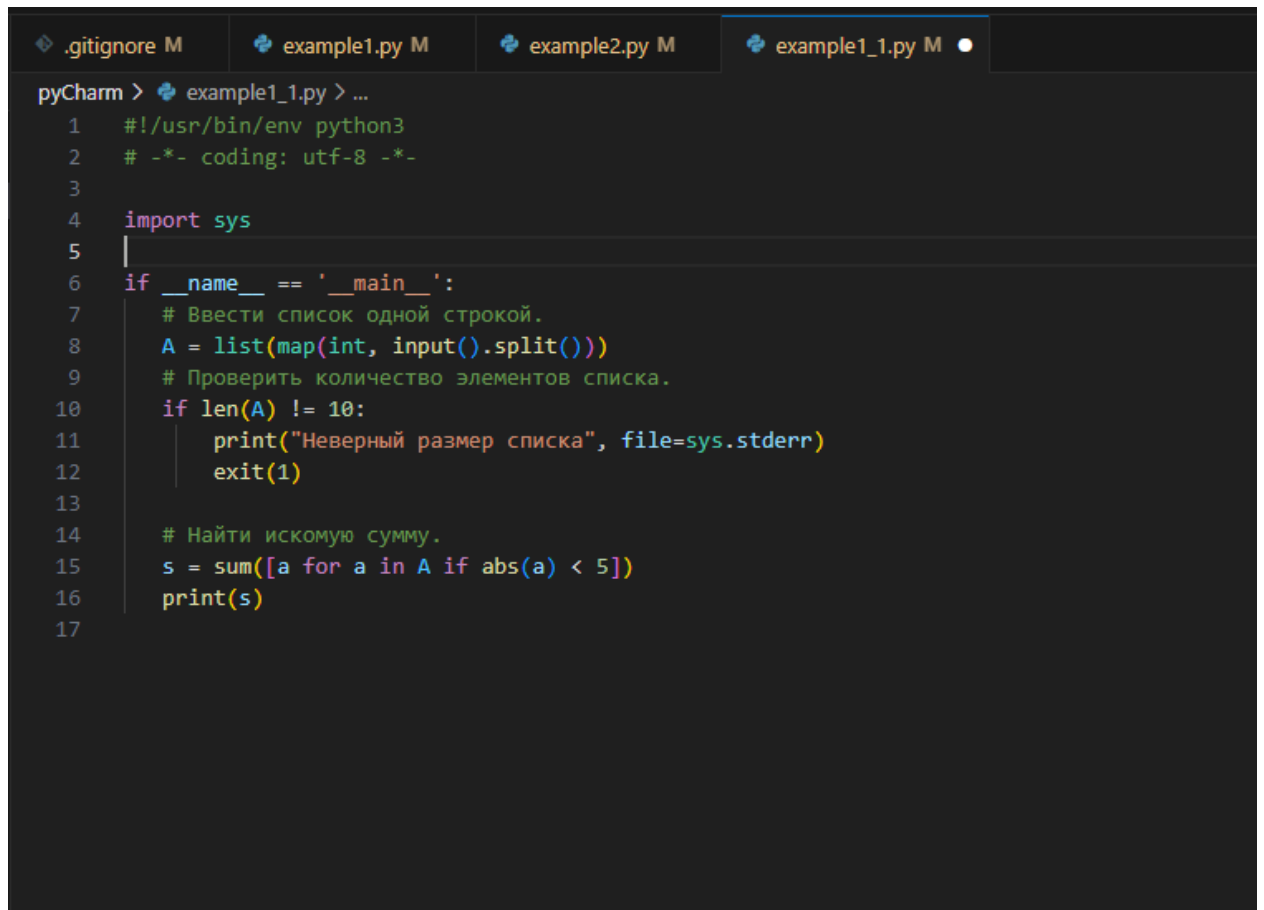
7. Проработал примеры лабораторной работы. Создала для каждого примера отдельный модуль языка Python. Зафиксировал изменения в репозитории.

```

.gitignore M  example1.py M  example2.py M  example1_1.py M
pyCharm > example1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      # Ввести список одной строкой.
8      A = list(map(int, input().split()))
9      # Проверить количество элементов списка.
10     if len(A) != 10:
11         print("Неверный размер списка", file=sys.stderr)
12         exit(1)
13
14     # Найти искомую сумму.
15     s = 0
16     for item in A:
17         if abs(item) < 5:
18             s += item
19
20     print(s)
21

```

Рисунок 7.1 – Проработка примера 1 с применением map()



```
pyCharm > example1_1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5  |
6  if __name__ == '__main__':
7      # Ввести список одной строкой.
8      A = list(map(int, input().split()))
9      # Проверить количество элементов списка.
10     if len(A) != 10:
11         print("Неверный размер списка", file=sys.stderr)
12         exit(1)
13
14     # Найти искомую сумму.
15     s = sum([a for a in A if abs(a) < 5])
16     print(s)
17
```

Рисунок 7.2 – Проработка примера 1 с применением списковых включений

```
.gitignore M example1.py M example2.py M x example1_1.py M
pyCharm > example2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import sys
4  if __name__ == '__main__':
5      # Ввести список одной строкой.
6      a = list(map(int, input().split()))
7      # Если список пуст, завершить программу.
8      if not a:
9          print("Заданный список пуст", file=sys.stderr)
10         exit(1)
11
12     # Определить индексы минимального и максимального элементов.
13     a_min = a_max = a[0]
14     i_min = i_max = 0
15     for i, item in enumerate(a):
16         if item < a_min:
17             i_min, a_min = i, item
18     if item >= a_max:
19         i_max, a_max = i, item
20
21     # Проверить индексы и обменять их местами.
22     if i_min > i_max:
23         i_min, i_max = i_max, i_min
24
25     # Посчитать количество положительных элементов.
26     count = 0
27     for item in a[i_min+1:i_max]:
28         if item > 0:
29             count += 1
30
31     print(count)
32
```

Рисунок 7.3 – Проработка примера 2

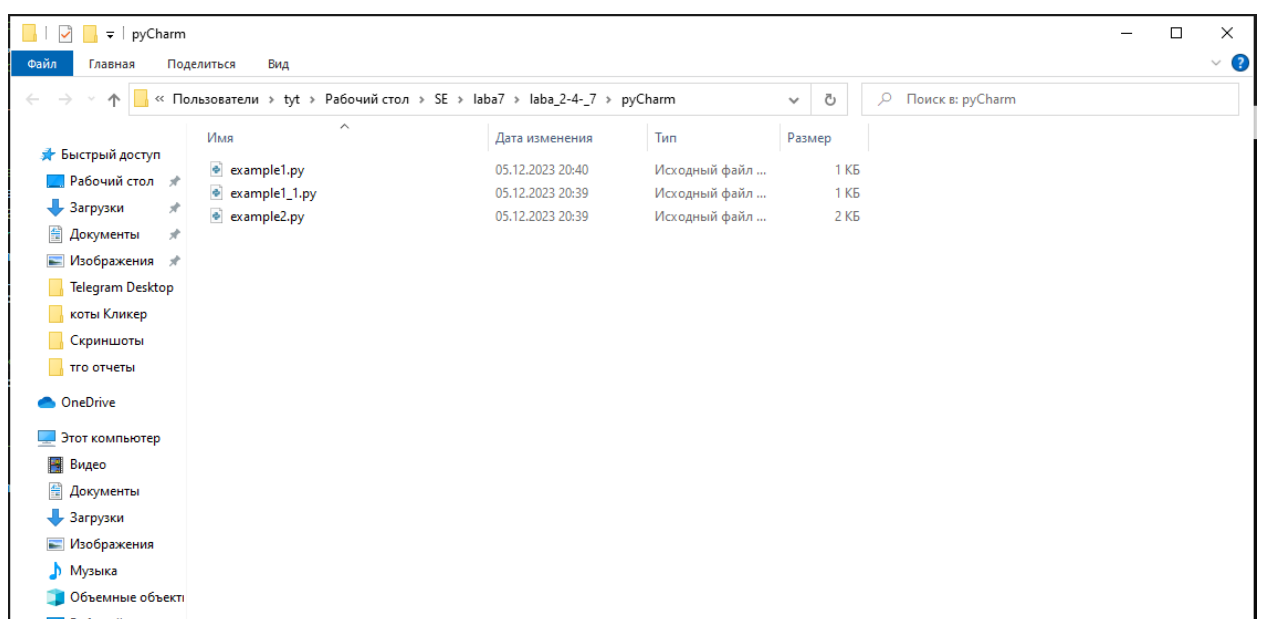
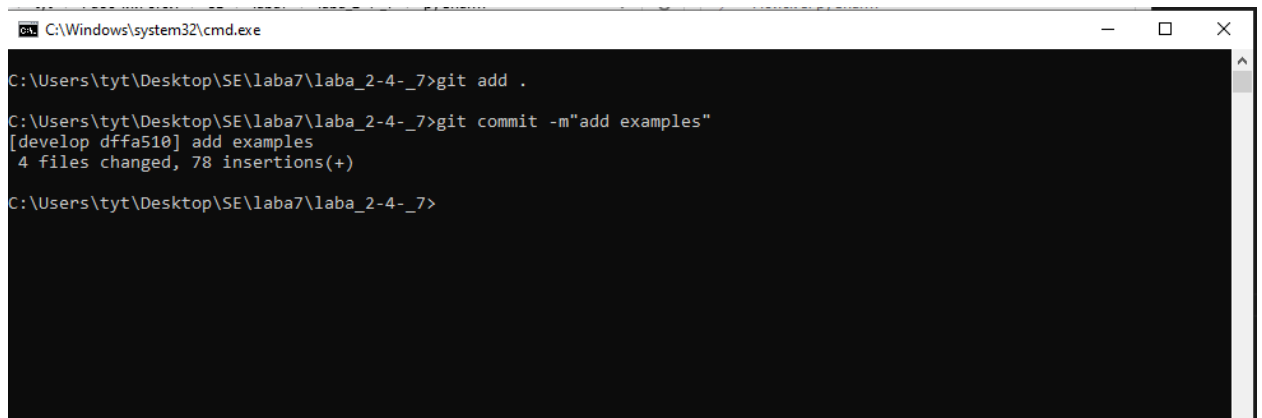


Рисунок 7.4 – Создание отдельных модулей для каждого из примеров

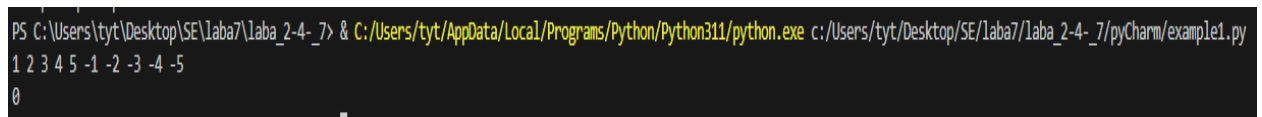


```
C:\Windows\system32\cmd.exe

C:\Users\tyt\Desktop\SE\laba7\laba_2-4-_7>git add .
C:\Users\tyt\Desktop\SE\laba7\laba_2-4-_7>git commit -m"add examples"
[develop dffa510] add examples
 4 files changed, 78 insertions(+)
C:\Users\tyt\Desktop\SE\laba7\laba_2-4-_7>
```

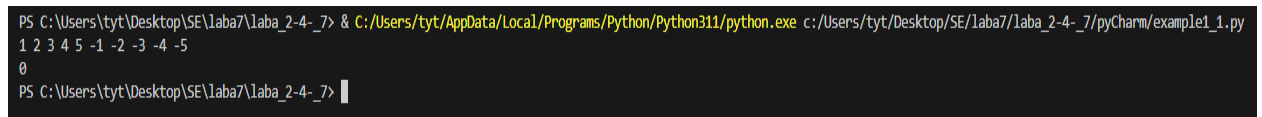
Рисунок 7.5 – Фиксирование изменений в репозитории

8. Привел в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных, вводимых с клавиатуры.



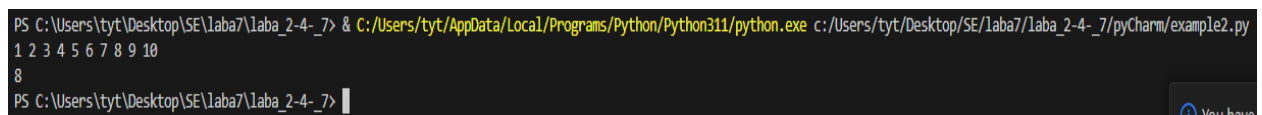
```
PS C:\Users\tyt\Desktop\SE\laba7\laba_2-4-_7> & C:/Users/tyt/AppData/Local/Programs/Python/Python311/python.exe c:/Users/tyt/Desktop/SE/laba7/laba_2-4-_7/pyCharm/example1.py
1 2 3 4 5 -1 -2 -3 -4 -5
0
```

Рисунок 8.1 – Результат примера 1 с применением map()



```
PS C:\Users\tyt\Desktop\SE\laba7\laba_2-4-_7> & C:/Users/tyt/AppData/Local/Programs/Python/Python311/python.exe c:/Users/tyt/Desktop/SE/laba7/laba_2-4-_7/pyCharm/example1_1.py
1 2 3 4 5 -1 -2 -3 -4 -5
0
PS C:\Users\tyt\Desktop\SE\laba7\laba_2-4-_7>
```

Рисунок 8.2 – Результат примера 2 с применением списковых включений



```
PS C:\Users\tyt\Desktop\SE\laba7\laba_2-4-_7> & C:/Users/tyt/AppData/Local/Programs/Python/Python311/python.exe c:/Users/tyt/Desktop/SE/laba7/laba_2-4-_7/pyCharm/example2.py
1 2 3 4 5 6 7 8 9 10
8
PS C:\Users\tyt\Desktop\SE\laba7\laba_2-4-_7>
```

Рисунок 8.3 – Результат примера 2

9. Привел в отчете скриншоты работы программ решения индивидуальных заданий

```

pyCharm > individual1_0.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      sp = list(map(int, input().split()))
8      zeroCounter = 0
9      if not sp:
10         print("Заданный список пуст", file=sys.stderr)
11         exit(1)
12     for i, item in enumerate(sp):
13         if item == 0:
14             zeroCounter += 1
15     print(i)

```

Рисунок 9.1 – Код программы Individual1.py

```

pyCharm > individual1_1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      sp = list(map(int, input().split()))
8      if not sp:
9         print("Заданный список пуст", file=sys.stderr)
10        exit(1)
11
12     zeros = [i for i, item in enumerate(sp) if item == 0]
13     zeroCounter = len([i for i in sp if i == 0])
14     print(zeros)

```

Рисунок 9.2 – Код программы Individual1_1.py

```

pyCharm > individual2_0.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      sp = list(map(float, input().split()))
8      sp_1 = []
9      sp_2 = []
10     sp_3 = []
11     finalSp = []
12     maks = -1
13     for i, item in enumerate(sp):
14         if abs(item) > maks:
15             maks = abs(item)
16         if item != 0:
17             sp_1.append(item)
18         if item == 0:
19             sp_2.append(item)
20         if item > 0:
21             sp_3.append(i)
22     finalSp = sp_1 + sp_2
23     sm = sum(sp[sp_3[0]+1:sp_3[1]])
24     print(f"{maks}\n{sm}")
25

```

Рисунок 9.3 – Код программы Individual2_0.py

10. Зафиксировал сделанные изменения в репозитории

```

C:\Windows\system32\cmd.exe

C:\Users\tyt\Desktop\SE\laba7\laba_2-4-_7>git add .

C:\Users\tyt\Desktop\SE\laba7\laba_2-4-_7>git commit -m"add examples"
[develop dffa510] add examples
 4 files changed, 78 insertions(+)

C:\Users\tyt\Desktop\SE\laba7\laba_2-4-_7>git add .

C:\Users\tyt\Desktop\SE\laba7\laba_2-4-_7>git commit -m"add all"
[develop 4fc050f] add all
 5 files changed, 58 insertions(+)
 create mode 100644 pyCharm/individual1_0.py
 create mode 100644 pyCharm/individual1_1.py
 create mode 100644 pyCharm/individual2_0.py

C:\Users\tyt\Desktop\SE\laba7\laba_2-4-_7>git status
On branch develop
Your branch is ahead of 'origin/develop' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

C:\Users\tyt\Desktop\SE\laba7\laba_2-4-_7>

```

Рисунок 10.1 – Коммит файлов в репозитории git

Контрольные вопросы

1. Что такое списки в языке Python?

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

2. Как осуществляется создание списка в Python?

Для создания списка нужно заключить элементы в квадратные скобки:

```
my_list = [1, 2, 3, 4, 5]
```

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?

Читать элементы списка можно с помощью следующего цикла:

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']  
for elem in my_list:  
    print(elem)
```

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

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

9. Как выполнить сортировку списка?

Для сортировки списка нужно использовать метод `sort`. Для сортировки списка в порядке убывания необходимо вызвать метод `sort` с аргументом `reverse=True`.

10. Как удалить один или несколько элементов из списка?

Удалить элемент можно, написав его индекс в методе `pop`. Элемент можно удалить с помощью метода `remove`. Оператор `del` можно использовать для тех же целей. Можно удалить несколько элементов с помощью оператора среза.

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. Проще всего работу list comprehensions показать на примере. Допустим вам необходимо создать список целых чисел от 0 до `n`, где `n` предварительно задается. Классический способ решения данной задачи выглядел бы так.

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?

Слайсы (срезы) являются очень мощной составляющей Python, которая позволяет быстро и лаконично решать задачи выборки элементов из списка.

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. 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?

Функция sort() предназначена для сортировки списка на месте, то есть изменяет сам список, а функция sorted() возвращает новый отсортированный список, не изменяя исходный.