

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 №2.4**  
**дисциплины «Основы программной инженерии»**

Выполнил:  
Соколов Михаил Романович  
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:  
Богданов С.С., ассистент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

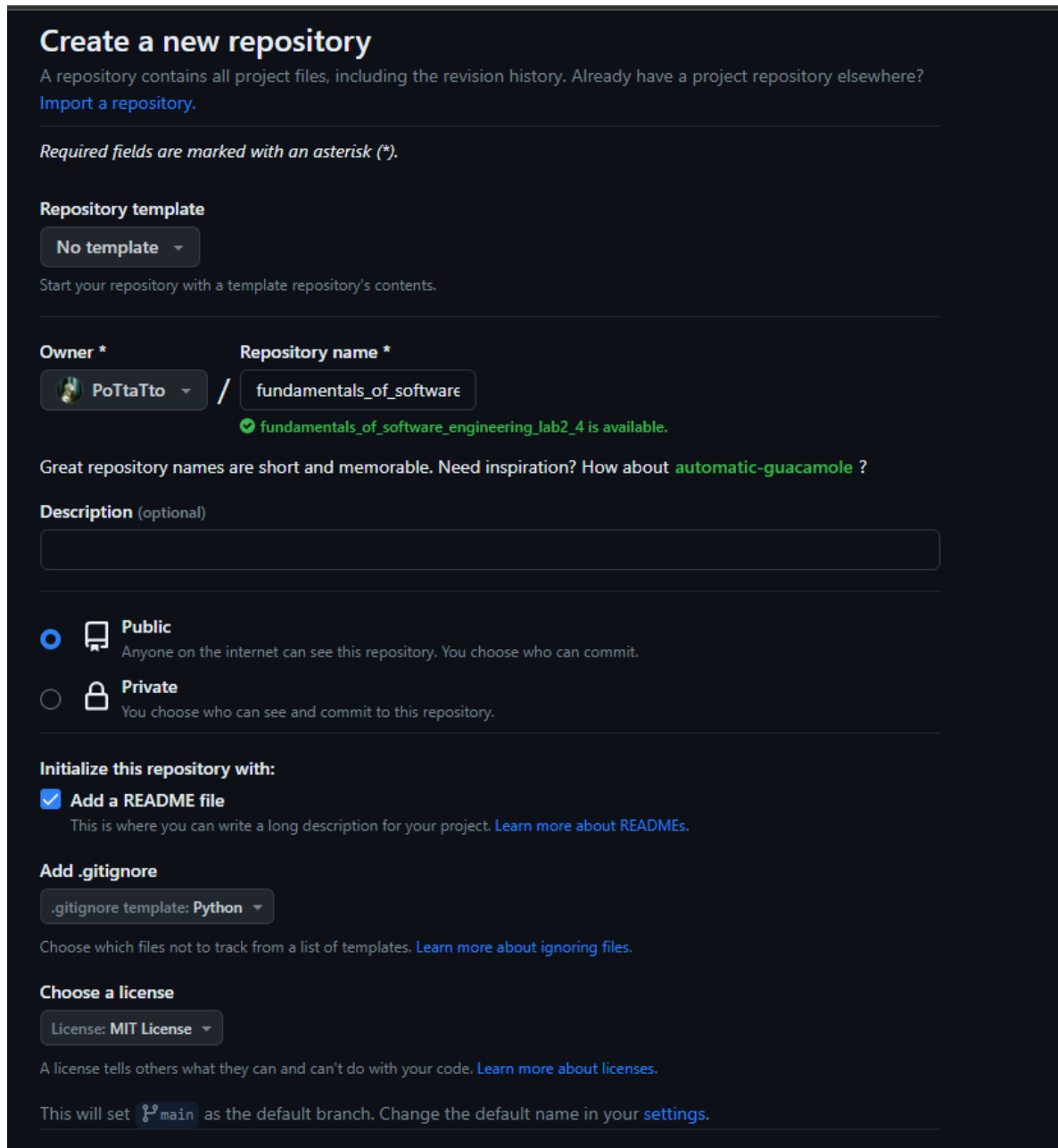
Ставрополь, 2023 г.

Тема: Работа со списками в языке Python.

Цель работы: приобретение навыков по работе со списками при написании программ с помощью языка программирования Python версии 3.x.

Ход выполнения работы:

1. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и .gitignore файл для языка программирования Python:



The screenshot shows the GitHub 'Create a new repository' interface. The 'Repository template' is set to 'No template'. The 'Owner' is 'PoTtaTto' and the 'Repository name' is 'fundamentals\_of\_software', with a suggestion for 'fundamentals\_of\_software\_engineering\_lab2\_4'. The 'Description' field is empty. The 'Public' option is selected for repository visibility. Under 'Initialize this repository with:', the 'Add a README file' checkbox is checked. The '.gitignore' template is set to 'Python'. The 'Choose a license' dropdown is set to 'MIT License'. At the bottom, it states the default branch will be 'main'.

**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 (\*).*

**Repository template**

No template ▾

Start your repository with a template repository's contents.

**Owner \*** PoTtaTto ▾ / **Repository name \*** fundamentals\_of\_software

✔ fundamentals\_of\_software\_engineering\_lab2\_4 is available.

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

**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](#).

Рисунок 1 – Создание репозитория с заданными настройками

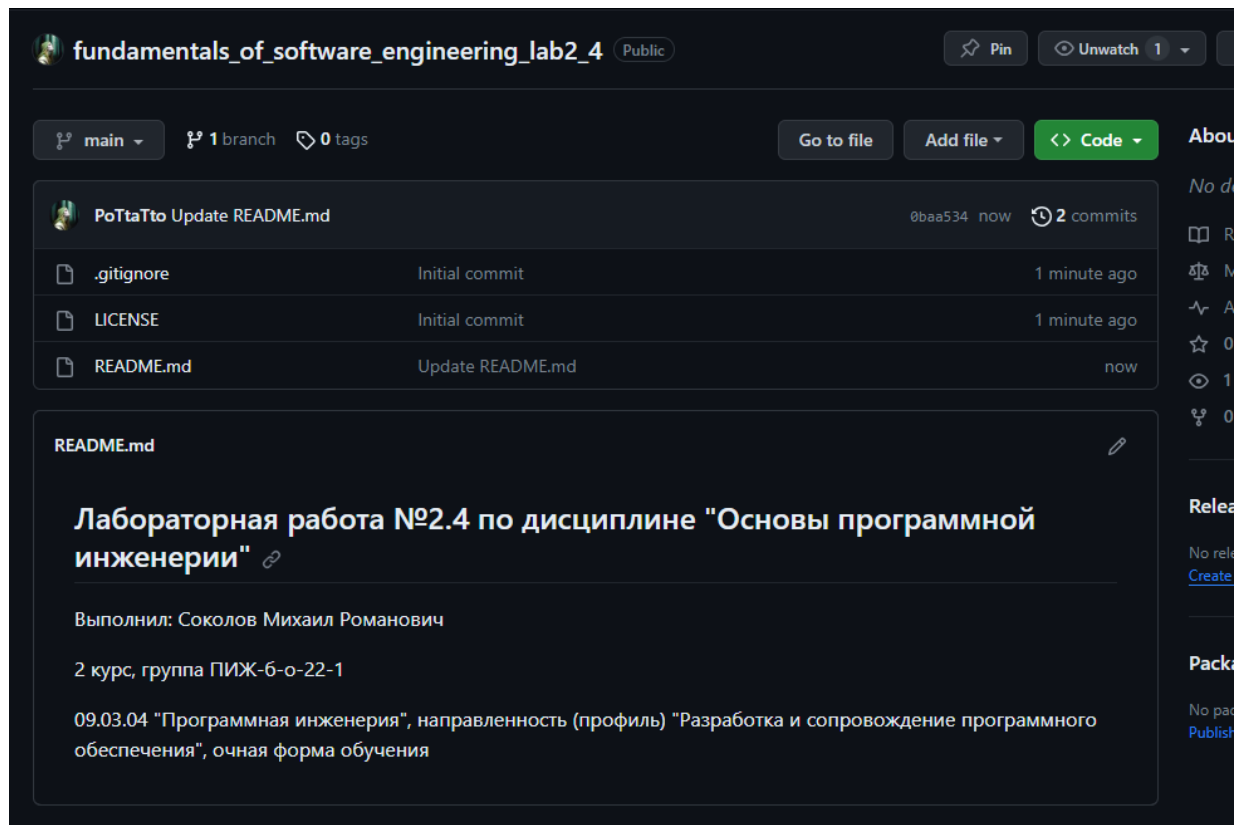


Рисунок 2 – Созданный репозиторий

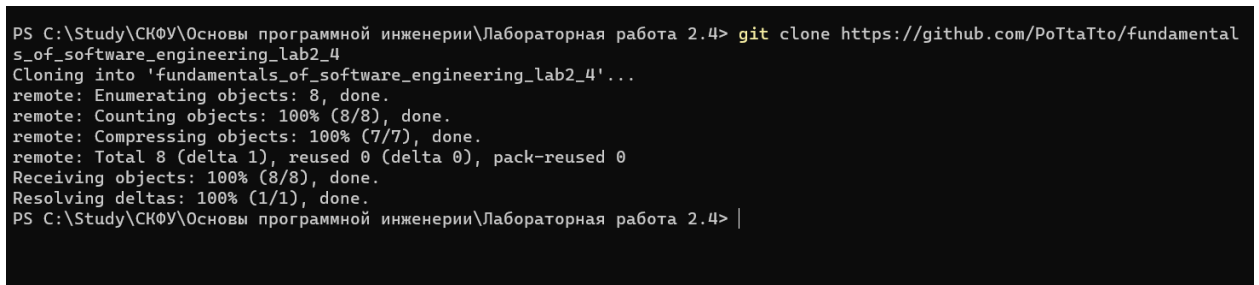


Рисунок 3 – Клонирование репозитория

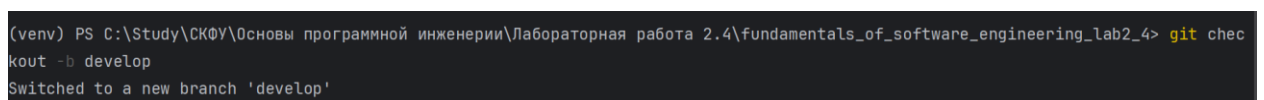


Рисунок 4 – Создание ветки develop, где будут происходить изменения проекта до его полного релиза

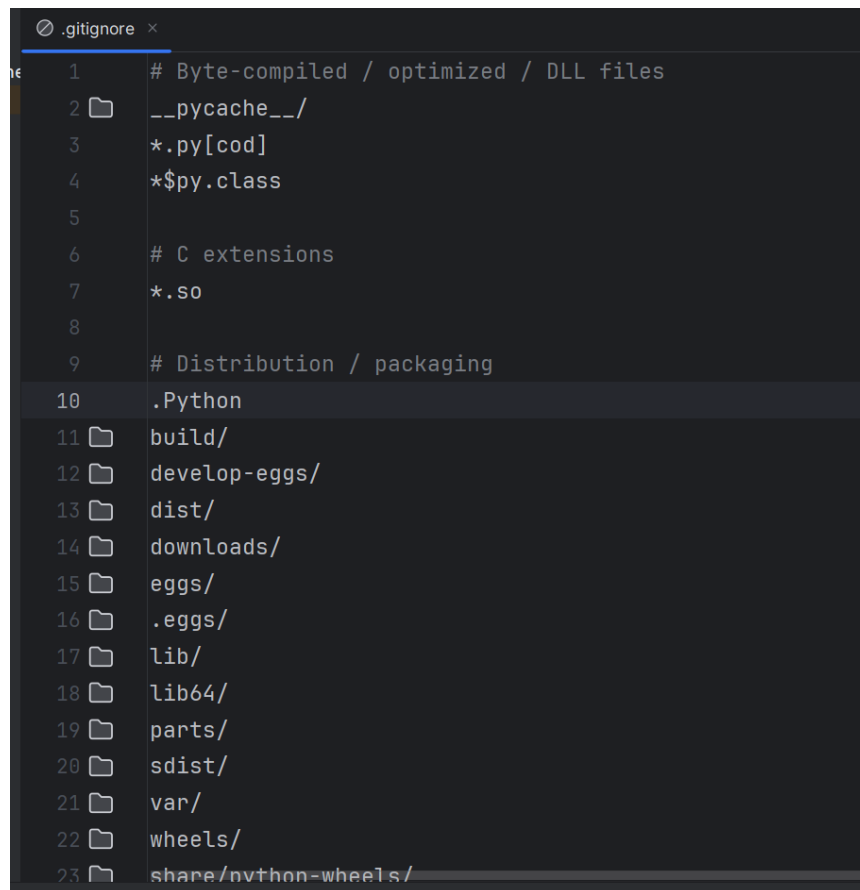


Рисунок 5 – Часть .gitignore, созданного GitHub

2. Проработаем примеры лабораторной работы, фиксируя изменения.  
Создадим для каждого примера отдельный модуль языка Python:

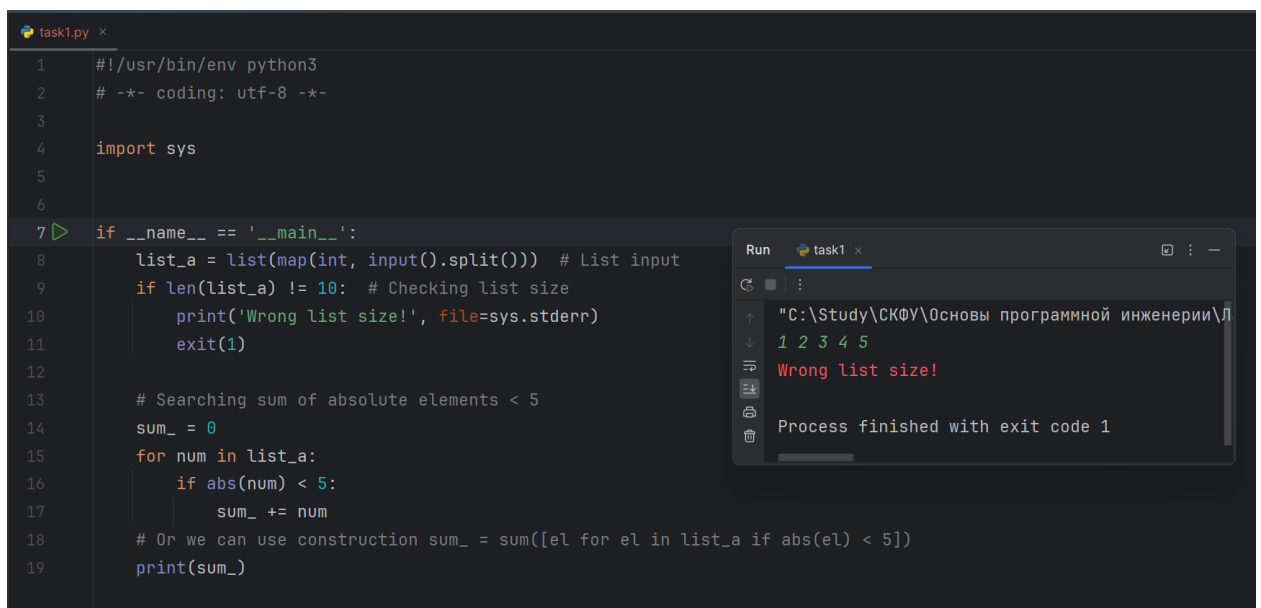


Рисунок 6 – Нахождение суммы списка из 10 элементов, меньших по модулю 5, и вывод ее на экран (пример №1) (1)

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6
7 if __name__ == '__main__':
8     list_a = list(map(int, input().split())) # List input
9     if len(list_a) != 10: # Checking list size
10         print('Wrong list size!', file=sys.stderr)
11         exit(1)
12
13     # Searching sum of absolute elements < 5
14     sum_ = 0
15     for num in list_a:
16         if abs(num) < 5:
17             sum_ += num
18     # Or we can use construction sum_ = sum([el for el in list_a if abs(el) < 5])
19     print(sum_)
```

Run task1 x

"C:\Study\СКФУ\Основы программной инженерии\Л

15 -100 12 1 3 6 -9 10 1 1

6

Process finished with exit code 0

Рисунок 7 – Нахождение суммы списка из 10 элементов, меньших по модулю 5, и вывод ее на экран (пример №1) (2)

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6
7 if __name__ == '__main__':
8     list_a = list(map(int, input().split())) # List input
9     if len(list_a) != 10: # Checking list size
10         print('Wrong list size!', file=sys.stderr)
11         exit(1)
12
13     # Searching sum of absolute elements < 5
14     sum_ = 0
15     for num in list_a:
16         if abs(num) < 5:
17             sum_ += num
18     # Or we can use construction sum_ = sum([el for el in list_a if abs(el) < 5])
19     print(sum_)
```

Run task1 x

"C:\Study\СКФУ\Основы программной инженерии\Л

6 6 6 6 6 6 6 6 6 6

0

Process finished with exit code 0

Рисунок 8 – Нахождение суммы списка из 10 элементов, меньших по модулю 5, и вывод ее на экран (пример №1) (3)

```
task2.py x
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6
7 if __name__ == '__main__':
8     list_a = list(map(int, input().split())) # List input
9     if not list_a: # Checking list is not empty
10         print('List is empty!', file=sys.stderr)
11         exit(1)
12
13     # Searching indexes of min and max elements of list
14     a_min = a_max = list_a[0]
15     i_min = i_max = 0
16     for i, item in enumerate(list_a):
17         if item < a_min:
18             i_min, a_min = i, item
19         if item > a_max:
20             i_max, a_max = i, item
21
22     # Checking indexes order
23     if i_min > i_max:
24         i_min, i_max = i_max, i_min
25
26     # Counting positive elements
27     count = 0
28     for item in list_a[i_min+1:i_max]:
29         if item > 0:
30             count += 1
31     print(count)
```

Run task2 x

"C:\Study\СКФУ\Основы программной инженерии\Л  
0 1 2 3 4 5 100  
5  
Process finished with exit code 0

Рисунок 9 – Нахождение количества положительных элементов, которые располагаются между максимальным и минимальным элементами (пример №2) (1)

```
task2.py x
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6
7 if __name__ == '__main__':
8     list_a = list(map(int, input().split())) # List input
9     if not list_a: # Checking list is not empty
10         print('List is empty!', file=sys.stderr)
11         exit(1)
12
13     # Searching indexes of min and max elements of list
14     a_min = a_max = list_a[0]
15     i_min = i_max = 0
16     for i, item in enumerate(list_a):
17         if item < a_min:
18             i_min, a_min = i, item
19         if item > a_max:
20             i_max, a_max = i, item
21
22     # Checking indexes order
23     if i_min > i_max:
24         i_min, i_max = i_max, i_min
25
26     # Counting positive elements
27     count = 0
28     for item in list_a[i_min+1:i_max]:
29         if item > 0:
30             count += 1
31     print(count)
```

Run task2 x

"C:\Study\СКФУ\Основы программной инженерии\Л  
List is empty!  
Process finished with exit code 1

Рисунок 10 – Нахождение количества положительных элементов, которые располагаются между максимальным и минимальным элементами (пример №2) (2)

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  if __name__ == '__main__':
8      list_a = list(map(int, input().split())) # List input
9      if not list_a: # Checking list is not empty
10         print('List is empty!', file=sys.stderr)
11         exit(1)
12
13     # Searching indexes of min and max elements of list
14     a_min = a_max = list_a[0]
15     i_min = i_max = 0
16     for i, item in enumerate(list_a):
17         if item < a_min:
18             i_min, a_min = i, item
19         if item > a_max:
20             i_max, a_max = i, item
21
22     # Checking indexes order
23     if i_min > i_max:
24         i_min, i_max = i_max, i_min
25
26     # Counting positive elements
27     count = 0
28     for item in list_a[i_min+1:i_max]:
29         if item > 0:
30             count += 1
31     print(count)

```

Run task2

```

"C:\Study\СКФУ\Основы программной инженерии\Л
100 99 98 0
2
Process finished with exit code 0

```

Рисунок 11 – Нахождение количества положительных элементов, которые располагаются между максимальным и минимальным элементами (пример №2) (3)

### 3. Выполним индивидуальные задания (вариант №7):

Задание №1. Ввести список А из 10 элементов, найти произведение отрицательных элементов и вывести его на экран:

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  from functools import reduce
5  import sys
6
7
8  # Using List Comprehensions for solving problem
9  usage
10 def find_negative_composition_comprehension_version(list_a: list):
11     print(reduce(lambda a, b: a * b, [el for el in list_a if el < 0]))
12
13 # Using ordinary loops for solving problem
14 usage
15 def find_negative_composition_loop_version(list_a: list):
16     composition = 1
17     for num in list_a:
18         if num < 0:
19             composition *= num
20     print(composition)
21
22 if __name__ == '__main__':
23     input_list = [int(el) for el in input().split()]
24     if len(input_list) != 10:
25         print('Invalid list length!', file=sys.stderr)
26     else:
27         find_negative_composition_comprehension_version(input_list)
28         find_negative_composition_loop_version(input_list)

```

Run individual\_task1

```

"C:\Study\СКФУ\Основы программной инженерии\Л
0 3 12
Invalid list length!
Process finished with exit code 0

```

Рисунок 12 – Индивидуальное задание №1 (1)

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  from functools import reduce
5  import sys
6
7
8  # Using List Comprehensions for solving problem
9  usage
10 def find_negative_composition_comprehension_version(list_a: list):
11     print(reduce(lambda a, b: a * b, [el for el in list_a if el < 0]))
12
13 # Using ordinary loops for solving problem
14 usage
15 def find_negative_composition_loop_version(list_a: list):
16     composition = 1
17     for num in list_a:
18         if num < 0:
19             composition *= num
20     print(composition)
21
22 if __name__ == '__main__':
23     input_list = [int(el) for el in input().split()]
24     if len(input_list) != 10:
25         print('Invalid list length!', file=sys.stderr)
26     else:
27         find_negative_composition_comprehension_version(input_list)
28         find_negative_composition_loop_version(input_list)

```

Run individual\_task1 x

```

"C:\Study\СКФУ\основы программной инженерии\И
-10 -20 0 12 34 3 2 1 8 -1
-200
-200
Process finished with exit code 0

```

Рисунок 13 – Индивидуальное задание №1 (1)

Задание №2. В списке, состоящем из вещественных элементов, вычислить: номер минимального элемента списка; сумму элементов списка, расположенных между первым и вторым отрицательными элементами. Преобразовать список таким образом, чтобы сначала располагались все элементы, модуль которых не превышает 1, а потом - все остальные:

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  if __name__ == '__main__':
8      list_ = [float(num) for num in input().split()]
9      if not list_: # Checking list is not empty
10         print('List is empty!', file=sys.stderr)
11     else:
12         neg_i = [] # List with indexes of first 2 negative elements
13         min_val, min_i = list_[0], 0
14         for i, num in enumerate(list_):
15             if num < min_val: # Searching min value and it's index
16                 min_val, min_i = num, i
17             if len(neg_i) != 2 and num < 0: # Appending indexes of first 2 negative elements
18                 neg_i.append(i)
19         print('Min element index:', min_i)
20         print('Sum between first 2 negative elements:', sum(list_[neg_i[0]+1:neg_i[1]]) if len(neg_i) == 2 else 'None')
21         print('List with abs(elements) < 1 into the beginning:', sorted(list_, key=lambda a: abs(a) > 1))
22
23

```

Run individual\_task2 x

```

"C:\Study\СКФУ\основы программной инженерии\Лабораторная работа 2.4\fundamentals_of_software_eng
0.1 0.4 -0.5 12 14 -100 12 -0.7
Min element index: 5
Sum between first 2 negative elements: 26.0
List with abs(elements) < 1 into the beginning: [0.1, 0.4, -0.5, -0.7, 12.0, 14.0, -100.0, 12.0]

```

Рисунок 14 – Индивидуальное задание №2 (1)



The screenshot shows a code editor with a file named `individual_task2.py`. The script is a Python program that takes input from the user, checks if the list is empty, and then processes the list based on the number of negative elements. The output window shows the following text:

```
"C:\Study\СКФУ\Основы программной и
List is empty!
Process finished with exit code 0
```

Рисунок 15 – Индивидуальное задание №2 (2)

The screenshot shows the same code editor with the file `individual_task2.py`. The output window shows the following text:

```
"C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.4\fundam
1 -1 0
Min element index: 1
Sum between first 2 negative elements: None
List with abs(elements) < 1 into the beginning: [1.0, -1.0, 0.0]
```

Рисунок 16 – Индивидуальное задание №2 (3)

4. Сольем ветки `develop` и `main / master` и отправим на удаленный репозиторий:

```

(venv) PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.4\fundamentals_of_software_engineering_lab2_4> git log --oneline
b059a93 (HEAD -> develop) individual_task2.py is added
b1456ef individual_task1.py is added
af550ff task2.py is added
fb0a26d task1.py is added
0baa534 (origin/main, origin/HEAD, main) Update README.md
f6e835e Initial commit
(venv) PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.4\fundamentals_of_software_engineering_lab2_4>

```

Рисунок 17 – История коммитов

```

(venv) PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.4\fundamentals_of_software_engineering_lab2_4> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
(venv) PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.4\fundamentals_of_software_engineering_lab2_4> git merge develop
Updating 0baa534..b059a93
Fast-forward
 .idea/.gitignore | 3 +++
 ...fundamentals_of_software_engineering_lab2_4.iml | 10 ++++++
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/modules.xml
 create mode 100644 .idea/vcs.xml
 create mode 100644 individual_task1.py
 create mode 100644 individual_task2.py
 create mode 100644 task1.py
 create mode 100644 task2.py
(venv) PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.4\fundamentals_of_software_engineering_lab2_4>

```

Рисунок 18 – Слияние ветки main с веткой develop

```

(venv) PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.4\fundamentals_of_software_engineering_lab2_4> git push origin main
Enumerating objects: 22, done.
Counting objects: 100% (22/22), done.
Delta compression using up to 12 threads
Compressing objects: 100% (20/20), done.
Writing objects: 100% (21/21), 4.04 KiB | 4.04 MiB/s, done.
Total 21 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/PoItaTo/fundamentals_of_software_engineering_lab2_4
 0baa534..b059a93 main -> main
(venv) PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.4\fundamentals_of_software_engineering_lab2_4>

```

Рисунок 19 – Отправка изменений на удаленный репозиторий

Ответы на 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?

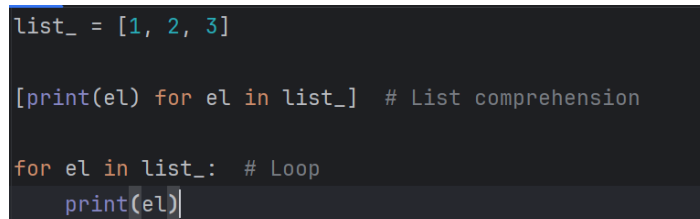
Для создания списка нужно заключить элементы в квадратные скобки.

Пример: `list_ = [1, 2, 3]`

3. Как организовано хранение списков в оперативной памяти?

Как уже было сказано выше, 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?



```
list_ = [1, 2, 3]

[print(el) for el in list_] # List comprehension

for el in list_: # Loop
    print(el)
```

Рисунок 20 – Через цикл или list comprehension

5. Какие существуют арифметические операции со списками?

Объединение (+) и умножение/дублирование (\*).

6. Как проверить есть ли элемент в списке?

Оператором in.

7. Как определить число вхождений заданного элемента в списке?

Методом `count(element)`.

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

Методами `append(element)` и `insert(index, element)`.

9. Как выполнить сортировку списка?

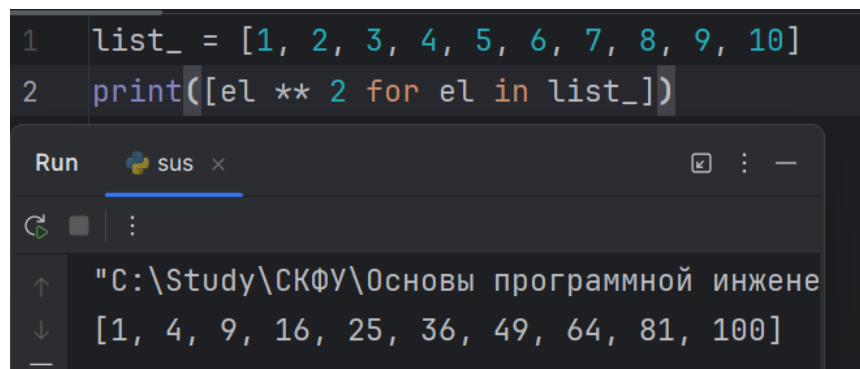
Методом `sort()` и функцией `sorted(list)`.

10. Как удалить один или несколько элементов из списка?

Методами `pop(index)`, `remove(element)` и оператором `del`. Оператор `del + slice` может удалить несколько элементов. Метод `clear()` удаляет все элементы из списка.

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.

A screenshot of a Python IDE window. The code editor shows two lines: `1 list_ = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]` and `2 print([el ** 2 for el in list_])`. Below the editor is a 'Run' button and a terminal window. The terminal shows the command prompt `"C:\Study\СКФУ\Основы программной инжене` followed by the output `[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]`.

```
1 list_ = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2 print([el ** 2 for el in list_])

Run sus x

"C:\Study\СКФУ\Основы программной инжене
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Рисунок 21 – Пример использования list comprehension

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

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?

Функции: `len()` [длина списка], `min()` [минимальный элемент списка], `max()` [максимальный элемент списка], `sum()` [сумма элементов списка].

14. Как создать копию списка?

Методом `copy()`.

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

`list.sort()` – это метод, который можно вызвать непосредственно на списке. Он сортирует список на месте, изменяя оригинальный список. `sorted()` – это встроенная функция Python, которая принимает итерируемый объект (например, список) в качестве входных данных и возвращает новый отсортированный список, не изменяя исходный.