

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

---

(подпись)

Проверил: Воронкин Р. А.

---

(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.

Ход работы:

1. Изучил теоретический материал работы.
2. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python и файл .gitignore.

**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 \*** **Repository name \***

odik8 / BSE7

✓ BSE7 is available.

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

**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**

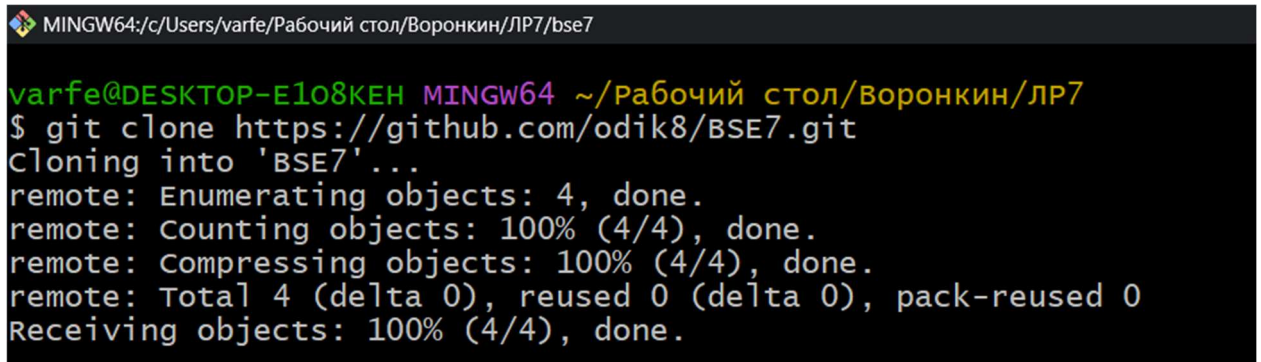
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

You are creating a public repository in your personal account.

**Create repository**

Рисунок 1 – Создание репозитория

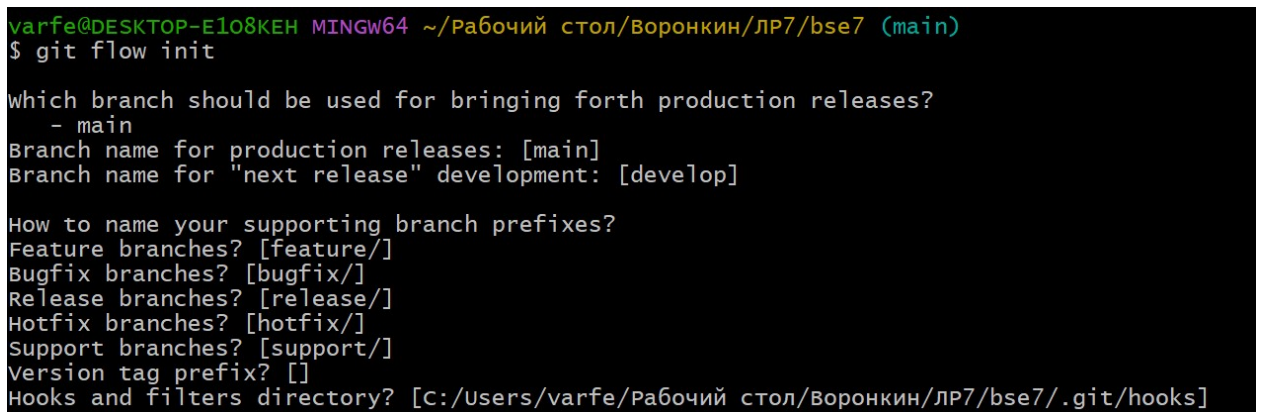
3. Выполнил клонирование созданного репозитория.



```
MINGW64:/c:/Users/varfe/Рабочий стол/Воронкин/ЛР7/bse7
varfe@DESKTOP-E108KEH MINGW64 ~/Рабочий стол/Воронкин/ЛР7
$ git clone https://github.com/odik8/BSE7.git
Cloning into 'BSE7'...
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 2. – Клонирование репозитория

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



```
varfe@DESKTOP-E108KEH MINGW64 ~/Рабочий стол/Воронкин/ЛР7/bse7 (main)
$ 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/varfe/Рабочий стол/Воронкин/ЛР7/bse7/.git/hooks]
```

Рисунок 3 – Инициализация git-flow

5. Создал проект PyCharm в папке репозитория.

6. Выполнил индивидуальные задания

Вариант 11

Задание 11\_1. Составить программу с использованием одномерных массивов для решения задачи. Ввести список А из 10 элементов, найти сумму отрицательных элементов кратных 7, их количество и вывести результаты на экран.

Код:

```
1.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  usage new*
5  def main():
6      A = [int(input(f"Введите {i + 1}-й элемент списка: ")) for i in range(10)]
7
8      sum_negative_multiples_of_7 = 0
9      count_negative_multiples_of_7 = 0
10
11     for num in A:
12         if num < 0 and num % 7 == 0:
13             sum_negative_multiples_of_7 += num
14             count_negative_multiples_of_7 += 1
15
16     print("С использованием циклов:")
17     print(f"Сумма отрицательных элементов, кратных 7: {sum_negative_multiples_of_7}")
18     print(f"Количество отрицательных элементов, кратных 7: {count_negative_multiples_of_7}")
19
20     sum_negative_multiples_of_7 = sum(num for num in A if num < 0 and num % 7 == 0)
21     count_negative_multiples_of_7 = sum(1 for num in A if num < 0 and num % 7 == 0)
22
23     print("\nC использованием List Comprehensions:")
24     print(f"Сумма отрицательных элементов, кратных 7: {sum_negative_multiples_of_7}")
25     print(f"Количество отрицательных элементов, кратных 7: {count_negative_multiples_of_7}")
26
27 if __name__ == "__main__":
28     main()
```

Рисунок 4 – Код решения первой задачи

```
C:\Users\varfe\AppData\Local\Programs\Python\Python312\python.exe
Введите 1-й элемент списка: 1
Введите 2-й элемент списка: -21
Введите 3-й элемент списка: -7
Введите 4-й элемент списка: -31
Введите 5-й элемент списка: 231
Введите 6-й элемент списка: 5
Введите 7-й элемент списка: 3
Введите 8-й элемент списка: 2
Введите 9-й элемент списка: 7
Введите 10-й элемент списка: 4
С использованием циклов:
Сумма отрицательных элементов, кратных 7: -28
Количество отрицательных элементов, кратных 7: 2

С использованием List Comprehensions:
Сумма отрицательных элементов, кратных 7: -28
Количество отрицательных элементов, кратных 7: 2

Process finished with exit code 0
```

Рисунок 5 - Результат выполнения первой программы

Задание 11\_2 Составить программу с использованием одномерных массивов для решения задачи на переупорядочивание элементов массива.

В списке, состоящем из вещественных элементов, вычислить:

1. номер максимального по модулю элемента списка;
2. сумму элементов списка, расположенных после первого положительного элемента.

Преобразовать список таким образом, чтобы сначала располагались все элементы, целая часть которых лежит в интервале  $[a, b]$ , а потом - все остальные.

Код решения:

```
2.py x
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  1 usage new *
5  def main():
6      elements = [1.5, -3.2, 4.7, -2.1, 5.8, -6.4, 0.9]
7
8      # 1. Найти номер максимального по модулю элемента списка
9      max_abs_index = elements.index(max(elements, key=abs))
10     print(f"Номер максимального по модулю элемента: {max_abs_index}")
11
12     # 2. Найти сумму элементов после первого положительного элемента
13     first_positive_index = next((i for i, x in enumerate(elements) if x > 0), None)
14     if first_positive_index is not None:
15         sum_after_positive = sum(elements[first_positive_index + 1:])
16         print(f"Сумма элементов после первого положительного: {sum_after_positive}")
17     else:
18         print("В списке нет положительных элементов.")
19
20     # Значения a и b для интервала [a, b]
21     a = 0
22     b = 5
23
24     # Преобразовать список
25     transformed_list = sorted(elements, key=lambda x: x // 1 if a <= x <= b else float(
26         'inf')) # float('inf') делает число бесконечно большим, что перемещает его в конец списка
27     print("Преобразованный список:", transformed_list)
28
29 if __name__ == "__main__":
30     main()
31
```

Рисунок 6 – Код решения второй задачи

```
Run 2 x
C:\Users\varfe\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\varf
Номер максимального по модулю элемента: 5
Сумма элементов после первого положительного: -0.30000000000000006
Преобразованный список: [0.9, 1.5, 4.7, -3.2, -2.1, 5.8, -6.4]
Process finished with exit code 0
```

Рисунок 7 – Результат выполнения второй программы

6. Зафиксировал сделанные изменения в репозитории.

7. Добавил отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксировал изменения.

8. Выполните слияние ветки для разработки с веткой main.

Вопросы для защиты работы

1. **Что такое списки в языке Python?** В Python список — это упорядоченная изменяемая коллекция объектов. Он может содержать элементы различных типов данных, таких как числа, строки, другие списки и т.д.

2. **Как осуществляется создание списка в Python?** Список создается с использованием квадратных скобок `[]` и содержит элементы, разделенные запятыми. Например:

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

3. **Как организовано хранение списков в оперативной памяти?** Списки в Python хранятся в виде динамических массивов, что позволяет эффективно изменять их размер, но также может привести к периодическому выделению новой памяти при изменении размера списка.

4. **Каким образом можно перебрать все элементы списка?** Используйте цикл **for** для перебора элементов списка:

```
for element in my_list:  
  
    print(element)
```

5. **Какие существуют арифметические операции со списками?** Списки поддерживают операции конкатенации (+) и умножения (\*):

```
new_list = my_list + [6, 7]  
  
repeated_list = my_list * 3
```

6. **Как проверить есть ли элемент в списке?** Используйте оператор **in**:

if 3 in my\_list:

```
    print("Элемент 3 находится в списке.")
```

**7. Как определить число вхождений заданного элемента в списке?**

Используйте метод **count()**:

```
count_of_3 = my_list.count(3)
```

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

Используйте методы **append()** для добавления в конец списка и **insert()** для вставки по индексу:

```
my_list.append(8)
```

```
my_list.insert(1, 10)
```

**9. Как выполнить сортировку списка? Используйте метод **sort()** для сортировки в порядке возрастания:**

```
my_list.sort()
```

**10. Как удалить один или несколько элементов из списка? Используйте метод **remove()** для удаления по значению и **del** или метод **pop()** для удаления по индексу:**

```
my_list.remove(3)
```

```
del my_list[1]
```

```
popped_element = my_list.pop(2)
```

**11. Что такое списковое включение и как с его помощью осуществлять обработку списков? Списковое включение — это синтаксическая конструкция для создания списка в одну строку. Пример:**

```
squared_numbers = [x**2 for x in range(10)]
```

**12. Как осуществляется доступ к элементам списков с помощью срезов? Используйте синтаксис срезов **start:stop:step**:**



```
sub_list = my_list[1:4] # элементы с индексами 1, 2, 3
```

**13. Какие существуют функции агрегации для работы со списками?**

Некоторые функции агрегации включают **sum()**, **len()**, **max()**, и **min()**:

```
total = sum(my_list)
```

```
length = len(my_list)
```

```
maximum = max(my_list)
```

```
minimum = min(my_list)
```

**14. Как создать копию списка? Используйте метод **copy()** или срез:**

```
copied_list = my_list.copy()
```

```
sliced_copy = my_list[:]
```

**15. Самостоятельно изучите функцию **sorted** языка Python. В чем ее отличие от метода **sort** списков? **sorted()** - это встроенная функция, которая возвращает новый отсортированный список из элементов итерируемого объекта. Метод **sort()** выполняет сортировку на месте, изменяя оригинальный список. **sorted()** не изменяет оригинальный список, а возвращает новый.**