

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 _____

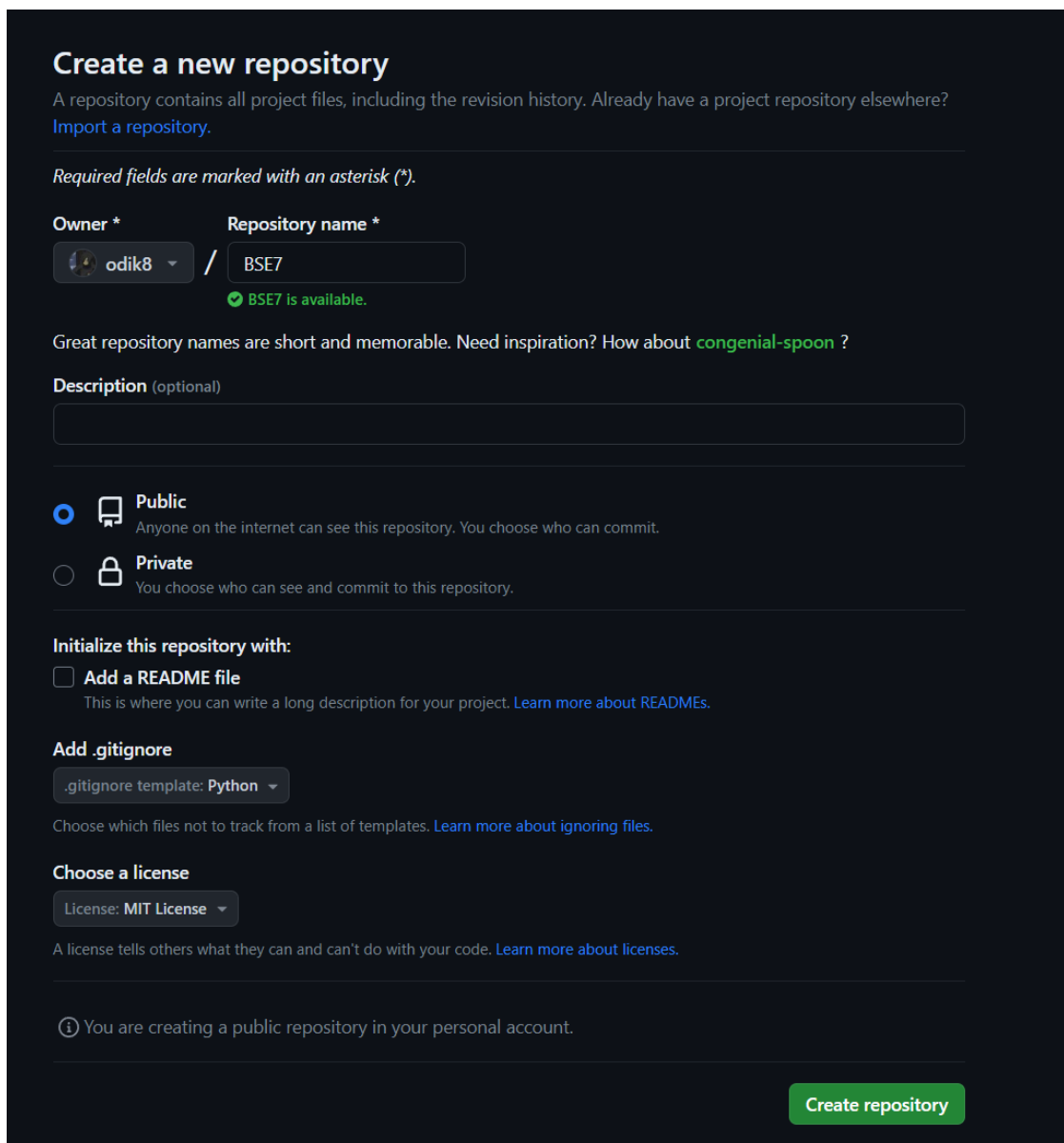
Ставрополь, 2023 г

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:

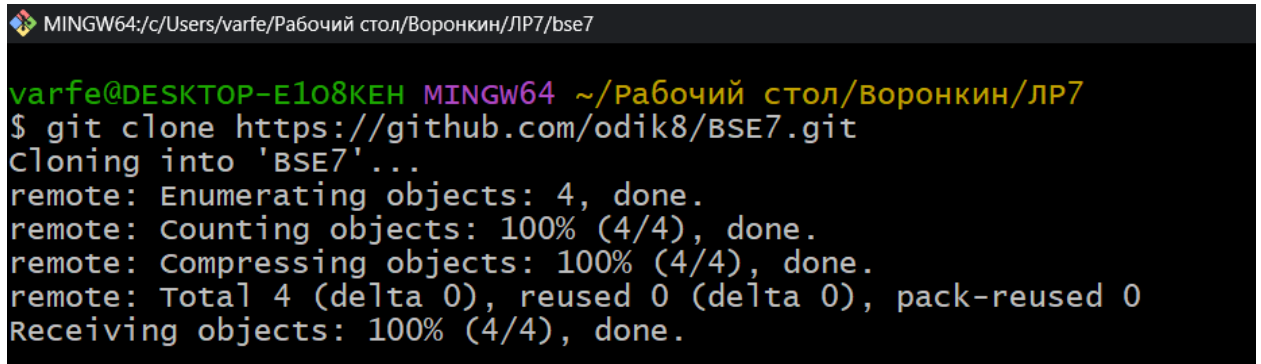
1. Изучил теоретический материал работы.
2. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python и файл .gitignore.



The screenshot shows the GitHub 'Create a new repository' page. At the top, it says 'Create a new repository' and provides a brief explanation of what a repository is. Below this, there are two required fields: 'Owner *' and 'Repository name *'. The 'Owner' field is set to 'odik8' and the 'Repository name' field is set to 'BSE7'. A green checkmark indicates that 'BSE7' is available. Below these fields, there is a text input for 'Description (optional)'. Further down, there are two radio buttons for repository visibility: 'Public' (selected) and 'Private'. Below these, there is a section 'Initialize this repository with:' with a checkbox for 'Add a README file'. Underneath, there is a section 'Add .gitignore' with a dropdown menu set to 'Python'. Below that, there is a section 'Choose a license' with a dropdown menu set to 'MIT License'. At the bottom, there is a green button labeled 'Create repository'.

Рисунок 1 – Создание репозитория

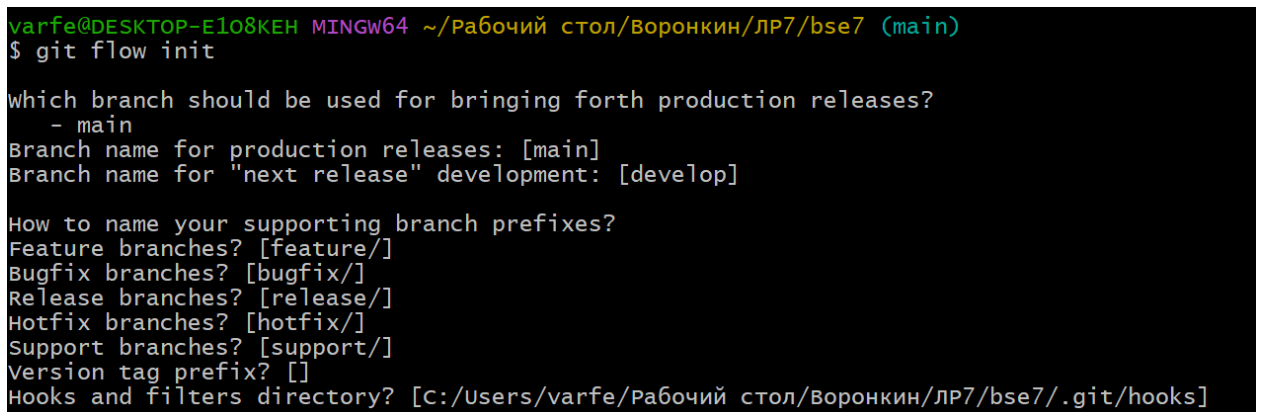
3. Выполнил клонирование созданного репозитория.



```
MINGW64:/c:/Users/varfe/Рабочий стол/Воронкин/ЛР7/bse7
varfe@DESKTOP-E108KEN 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-E108KEN 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  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  usage  odysey *
5  def main():
6      user_input = list(map(float, input("Введите числа через пробел: ").split()))
7
8      sum_negative_multiples_of_7 = 0
9      count_negative_multiples_of_7 = 0
10
11     for num in user_input:
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 user_input if num < 0 and num % 7 == 0)
21     count_negative_multiples_of_7 = sum(1 for num in user_input if num < 0 and num % 7 == 0)
22
23     print("\nС 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()
29

```

Рисунок 4 – Код решения первой задачи

```

C:\Users\varfe\AppData\Local\Programs\Python\Python312\python.exe
Введите числа через пробел: 1 7 14 -7 0 -777
С использованием циклов:
Сумма отрицательных элементов, кратных 7: -784.0
Количество отрицательных элементов, кратных 7: 2

С использованием List Comprehensions:
Сумма отрицательных элементов, кратных 7: -784.0
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:

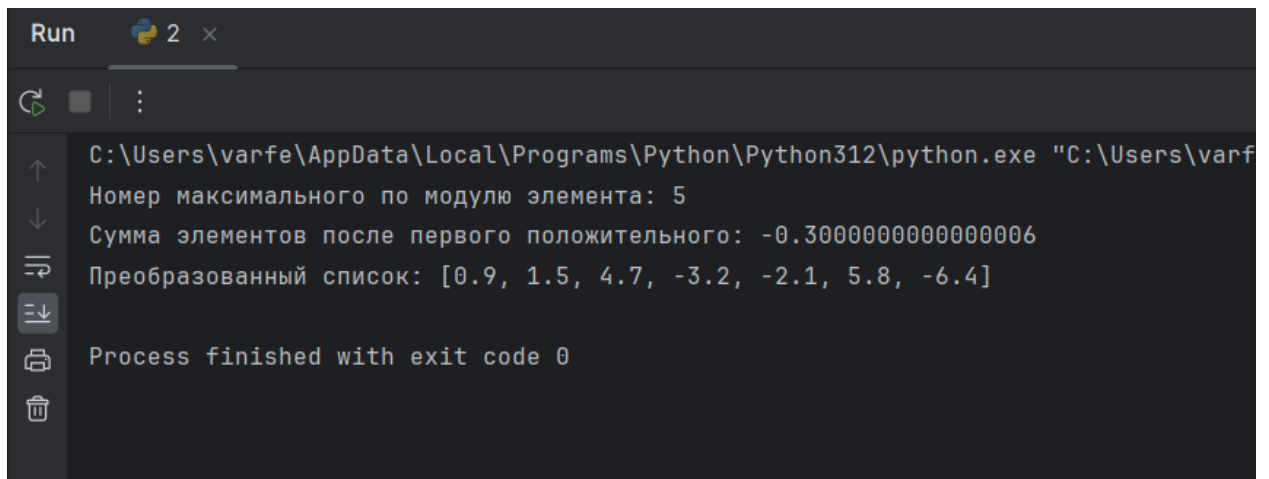
1. номер максимального по модулю элемента списка;
2. сумму элементов списка, расположенных после первого положительного элемента.

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

Код решения:

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  1 usage  _odyssey *
5  def main():
6      elements = list(map(float, input("Введите числа через пробел: ").split()))
7
8      max_abs_index = elements.index(max(elements, key=abs))
9      print(f"Номер максимального по модулю элемента: {max_abs_index}")
10
11     first_positive_index = next((i for i, x in enumerate(elements) if x > 0), None)
12     if first_positive_index is not None:
13         sum_after_positive = sum(elements[first_positive_index + 1:])
14         print(f"Сумма элементов после первого положительного: {sum_after_positive}")
15     else:
16         print("В списке нет положительных элементов.")
17
18     a = 0
19     b = 5
20
21     transformed_list = sorted(elements, key=lambda x: x // 1 if a <= x <= b else float(
22         'inf'))
23     print("Преобразованный список:", transformed_list)
24
25  if __name__ == "__main__":
26     main()
```

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.

