

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:
Магомедов Имран Борисович
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., кандидат технических
наук, доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____

Дата защиты _____

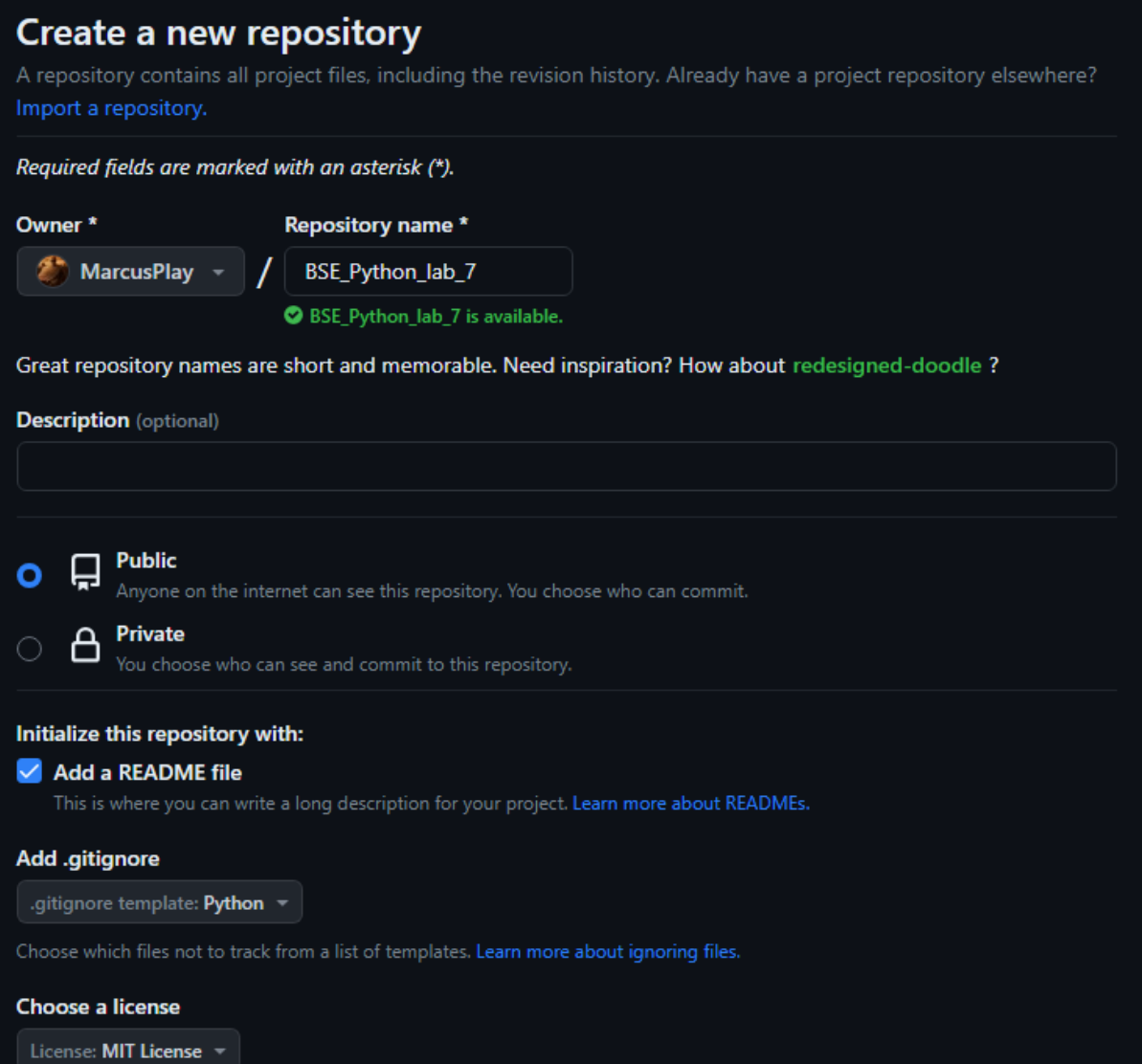
Ставрополь, 2023 г.

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.

Методика и порядок выполнения работы

1. Изучить теоретический материал работы.
2. Создать 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 * MarcusPlay / **Repository name *** BSE_Python_lab_7

✔ BSE_Python_lab_7 is available.

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

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

3. Выполните клонирование созданного репозитория.

4. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.

5. Проработайте примеры лабораторной работы. Создайте для каждого примера отдельный модуль языка Python. Зафиксируйте изменения в репозитории.

6. Приведите в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных вводимых с клавиатуры.

Пример 1.

```
example_1.1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  if __name__ == '__main__':
8      # Ввести список одной строкой.
9      A = list(map(int, input().split()))
10     # Проверить количество элементов списка.
11     if len(A) != 10:
12         print("Неверный размер списка", file=sys.stderr)
13         exit(1)
14
15     # Найти искомую сумму.
16     s = 0
17     for item in A:
18         if abs(item) < 5:
19             s += item
20
21     print(s)
```

1 2 3 4 5 6 7 8 10 1
● 11

```
example_1.2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  if __name__ == '__main__':
8      # Ввести список одной строкой.
9      A = list(map(int, input().split()))
10     # Проверить количество элементов списка.
11     if len(A) != 10:
12         print("Неверный размер списка", file=sys.stderr)
13         exit(1)
14
15     # Найти искомую сумму.
16     s = sum([a for a in A if abs(a) < 5])
17     print(s)
```

1 2 3 4 5 6 7 8 9 10
10

Пример 2.

```
example_2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  if __name__ == '__main__':
8      # Ввести список одной строкой.
9      a = list(map(int, input().split()))
10     # Если список пуст, завершить программу.
11     if not a:
12         print("Заданный список пуст", file=sys.stderr)
13         exit(1)
14
15     # Определить индексы минимального и максимального элементов.
16     a_min = a_max = a[0]
17     i_min = i_max = 0
18     for i, item in enumerate(a):
19         if item < a_min:
20             i_min, a_min = i, item
21         if item >= a_max:
22             i_max, a_max = i, item
23
24     # Проверить индексы и обменять их местами.
25     if i_min > i_max:
26         i_min, i_max = i_max, i_min
27
28     # Посчитать количество положительных элементов.
29     count = 0
30     for item in a[i_min+1:i_max]:
31         if item > 0:
32             count += 1
33
34     print(count)
```

● 12 23 34 45 56 67 78 89 90 13
7

7. Приведите в отчете скриншоты работы программ решения индивидуальных заданий.

Задание 1 (Вариант - 12).

Составить программу с использованием одномерных массивов для решения задачи. Решить индивидуальное задание как с

использованием циклов, так и с использованием List Comprehensions.

Ввести список A из 10 элементов, найти сумму элементов, больших 2 и меньших 20 и кратных 8, их количество и вывести результаты на экран.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Ввести список A из 10 элементов, найти сумму элементов, больших 2 и меньших 20
5  # и кратных 8, их количество и вывести результаты на экран.
6
7  import sys
8
9
10 if __name__=="__main__":
11     while True:
12         a = list(map(int, input("Введите 10 значений через пробел: ").split()))
13         if not a:
14             print("Заданный список пуст", file=sys.stderr)
15             exit(1)
16
17         if len(a) == 10:
18             break
19         else:
20             print("Количество элементов одномерного массива не равно 10.\n")
21
22     sum_num = 0
23     count = 0
24     for num in a:
25         if (num > 2) and (num < 20) and (num % 8 == 0):
26             sum_num += num
27             count += 1
28
29     print(sum_num, count)
```

```
Введите 10 значений через пробел: 1 2 3 4 5 6 7 8 9 10
8 1
```

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Ввести список A из 10 элементов, найти сумму элементов, больших 2 и меньших 20
5  # и кратных 8, их количество и вывести результаты на экран.
6
7  import sys
8
9
10 if __name__=="__main__":
11     while True:
12         a = [int(x) for x in input("Введите 10 значений через пробел: ").split()]
13
14         if not a:
15             print("Заданный список пуст", file=sys.stderr)
16             exit(1)
17
18         if len(a) == 10:
19             break
20         else:
21             print("Количество элементов одномерного массива не равно 10.\n")
22
23         s = [num for num in a if (num > 2) and (num < 20) and (num % 8 == 0)]
24
25         print(sum(s), len(s))

```

```

Введите 10 значений через пробел: 1 2 3 4 5 6 7 8 9 10
8 1

```

Задание 2 (Вариант - 12).

Составить программу с использованием одномерных массивов для решения задачи на переупорядочивание элементов массива. Для сортировки допускается использовать метод `sort` с заданным параметром `key` и объединение нескольких списков.

В списке, состоящем из вещественных элементов, вычислить:

- Количество элементов списка, лежащих в диапазоне от A до B;
- Сумму элементов списка, расположенных после максимального элемента.
- Упорядочить элементы списка по убыванию модулей элементов.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # В списке, состоящем из вещественных элементов, вычислить:
5  # 1. Количество элементов списка, лежащих в диапазоне от A до B;
6  # 2. Сумму элементов списка, расположенных после максимального элемента.
7  # 3. Упорядочить элементы списка по убыванию модулей элементов.
8
9  import sys
10
11
12  if __name__ == "__main__":
13      m = [float(x) for x in input("Вводите значения через пробел: ").split()]
14      a = float(input("Введите начальное значение диапазона: "))
15      b = float(input("Введите конечное значение диапазона: "))
16
17      s = [i for i in m if a < i < b]
18      print(f'Кличесиво элементов массива в диапазоне от {a} до {b}: {len(s)}')
19
20      s = [i for i in m[m.index(max(m)) + 1:]]
21      print(f"Сумму элементов списка, расположенных после максимального элемента равна: {sum(s)}")
22
23      s_sort = [abs(i) for i in m]
24      print("Элементы списка упорядочены по убыванию модулей элементов:", s_sort.sort(reverse=True))

```

Вводите значения через пробел: 1 2 3 4 5 1 2 3 4
Введите начальное значение диапазона: 1
Введите конечное значение диапазона: 5
Кличесиво элементов массива в диапазоне от 1.0 до 5.0: 6
Сумму элементов списка, расположенных после максимального элемента равна: 10.0
Элементы списка упорядочены по убыванию модулей элементов: [5.0, 4.0, 4.0, 3.0, 3.0, 2.0, 2.0, 1.0, 1.0]

8. Зафиксируйте сделанные изменения в репозитории.

```

$ git commit -m "added individual tasks"
[develop 15e7e24] added individual tasks
5 files changed, 96 insertions(+), 3 deletions(-)
create mode 100644 individual_task_1.1.py
create mode 100644 individual_task_1.2.py
create mode 100644 individual_task_2.1.py

```

9. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.

10. Выполните слияние ветки для разработки с веткой main / master.


```
$ git merge develop
Updating 61498ae..15e7e24
Fast-forward
 example_tasks/example_1.1.py | 21 ++++++
 example_tasks/example_1.2.py | 17 ++++++
 example_tasks/example_2.py   | 34 ++++++
 individual_task_1.1.py       | 37 ++++++
 individual_task_1.2.py       | 27 ++++++
 individual_task_2.1.py       | 29 ++++++
6 files changed, 165 insertions(+)
create mode 100644 example_tasks/example_1.1.py
create mode 100644 example_tasks/example_1.2.py
create mode 100644 example_tasks/example_2.py
create mode 100644 individual_task_1.1.py
create mode 100644 individual_task_1.2.py
create mode 100644 individual_task_2.1.py
```

11. Отправьте сделанные изменения на сервер GitHub.

```
$ git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MarcusPlay/BSE_Python_lab_7.git
 61498ae..15e7e24  main -> main
```

12. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

Контрольные вопросы:

1. Что такое списки в языке Python?

Списки в Python представляют собой упорядоченные коллекции элементов, которые могут содержать объекты разных типов. Элементы в списках могут быть изменяемыми, и списки позволяют хранить множество значений в одной переменной.

2. Как осуществляется создание списка в Python?

Для создания списка в Python используется квадратные скобки [], и элементы списка разделяются запятыми. Пример: `my_list = [1, 2, 3]`.

3. Как организовано хранение списков в оперативной памяти?

Списки в Python хранятся в памяти как динамически выделенные массивы, что позволяет эффективно изменять и обращаться к элементам списка.

4. Каким образом можно перебрать все элементы списка?

Для перебора всех элементов списка можно использовать цикл `for` или генераторы списков. Пример: `for item in my_list:` или `[item for item in my_list]`.

5. Какие существуют арифметические операции со списками?

Списки поддерживают операции сложения (+, объединение списков), умножения (*, повторение списка), и некоторые другие операции, такие как сравнение и индексация.

6. Как проверить есть ли элемент в списке?

Для проверки наличия элемента в списке можно использовать оператор `in`. Пример: `element in my_list` вернет `True`, если `element` присутствует в `my_list`.

7. Как определить число вхождений заданного элемента в списке?

Метод `count()` может быть использован для определения числа вхождений заданного элемента в список. Пример: `count = my_list.count(element)`.

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

Элемент можно добавить в список с помощью методов `append()` (в конец списка) и `insert()` (в указанную позицию). Пример: `my_list.append(new_element)` или `my_list.insert(index, new_element)`.

9. Как выполнить сортировку списка?

Список можно отсортировать с помощью методов `sort()` (сортировка на месте) или `sorted()` (возвращает новый

отсортированный список). Пример: `my_list.sort()` или `sorted_list = sorted(my_list)`.

10. Как удалить один или несколько элементов из списка?

Элементы можно удалить с помощью методов `remove()` (по значению) и `pop()` (по индексу). Пример: `my_list.remove(element)` или `my_list.pop(index)`.

11. Что такое списковое включение и как с его помощью осуществлять обработку списков?

Списковое включение (list comprehension) - это синтаксическая конструкция, позволяющая создавать новые списки на основе существующего списка с использованием выражений и фильтров. Пример: `[x*2 for x in my_list if x > 2]` создаст новый список, содержащий удвоенные значения элементов, которые больше 2.

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

Для доступа к под спискам списка используются срезы, которые определяются с использованием двоеточия. Пример: `my_list[start:end]` вернет подсписок элементов с индексами от `start` до `end - 1`.

13. Какие существуют функции агрегации для работы со списками?

Для выполнения агрегационных операций, таких как нахождение суммы, минимума, максимума и других, можно использовать встроенные функции, такие как `sum()`, `min()`, `max()`, `len()` и другие.

14. Как создать копию списка?

Копию списка можно создать с помощью среза или метода `copy()`. Пример: `new_list = my_list[:]` или `new_list = my_list.copy()`.

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

Функция `sorted()` возвращает новый отсортированный список на основе исходного, не изменяя исходный список. Метод `sort()` сортирует список на месте, изменяя исходный список. Таким образом, `sorted()` сохраняет исходный список неизменным, в то время как `sort()` изменяет его.