

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций
«Работа со списками в языке Python»**

**Отчет по лабораторной работе № 2.4
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-21-1

Кучеренко С.Ю. « » 2022г.

Подпись студента _____

Работа защищена « » _____ 2022г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2022

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

Выполнение работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

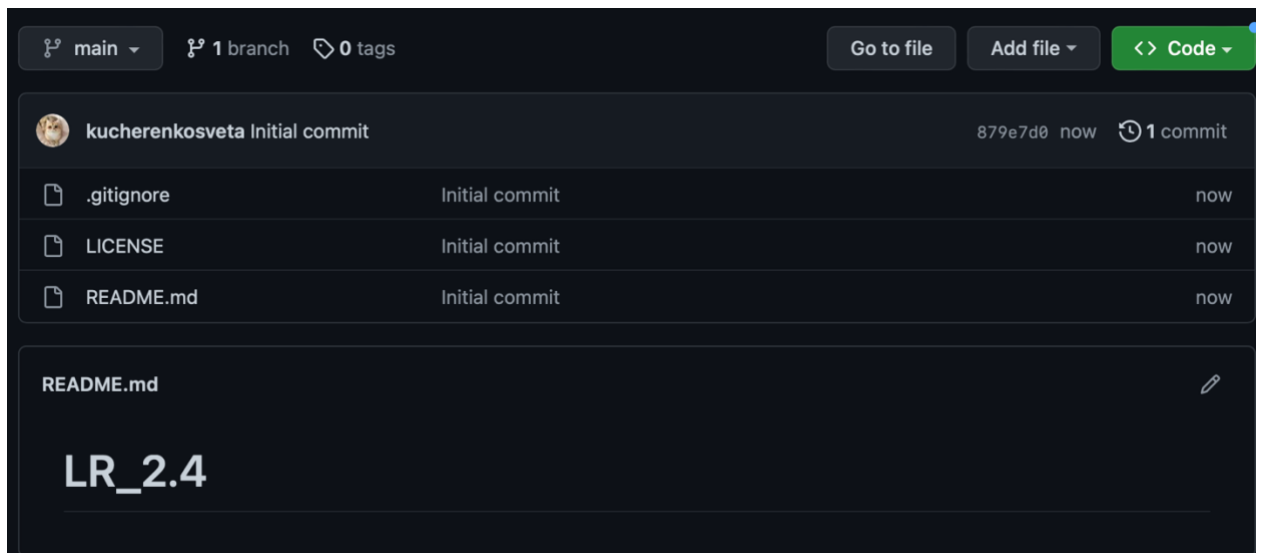


Рисунок 1 – Создание репозитория

3. Выполните клонирование созданного репозитория.

```
[(base) svetik@MacBook-Air-Svetik Laba7 % git clone https://github.com/kucherenko]
sveta/LR_2.4.git
Cloning into 'LR_2.4'...
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.
(base) svetik@MacBook-Air-Svetik Laba7 %
```

Рисунок 2 – Клонирование репозитория

4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```

((base) svetik@MacBook-Air-Svetik Laba7 % git flow init
No branches exist yet. Base branches must be created now.
Branch name for production releases: [master]
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? [/Users/svetik/.git/hooks]

```

Рисунок 3 – Организация репозитория в соответствии с моделью git-flow

6. Создайте проект PyCharm в папке репозитория.

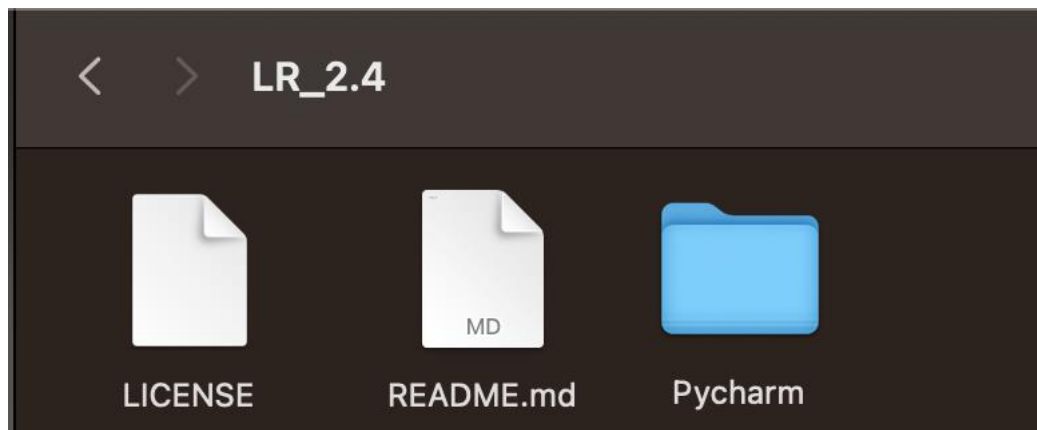


Рисунок 4 – Создание проекта PyCharm в папке репозитория

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

Пример 1. Ввести список A из 10 элементов, найти сумму элементов, меньших по модулю 5, и вывести ее на экран.

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == '__main__':
    # Ввести список одной строкой.
    A = list(map(int, input().split()))
    # Проверить количество элементов списка.
    if len(A) != 10:
        print("Неверный размер списка", file=sys.stderr)

```

```

        exit(1)
# Найти искомую сумму.
s = 0
for item in A:
    if abs(item) < 5:
        s += item
print(s)

```

```

/Users/svetik/.conda/envs/Pycharm/bin/
1 2 3 4 5 6 7 -4 5 23
6
Process finished with exit code 0

```

Рисунок 5 – Результат работы программы

Решение задачи с помощью списковых включений:

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == '__main__':
    # Ввести список одной строкой.
    A = list(map(int, input().split()))
    # Проверить количество элементов списка.
    if len(A) != 10:
        print("Неверный размер списка", file=sys.stderr)
        exit(1)

    # Найти искомую сумму.
    s = sum([a for a in A if abs(a) < 5])
    print(s)

```

```

/Users/svetik/.conda/envs/Pycharm/bin/
1 2 3 4 -56 81 5 1 -2 3
12
Process finished with exit code 0

```

Рисунок 6 – Результат работы программы

Пример 2. Написать программу, которая для целочисленного списка определяет, сколько положительных элементов располагается между его максимальным и минимальным элементами.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == '__main__':
    # Ввести список одной строкой.
    a = list(map(int, input().split()))
    # Если список пуст, завершить программу.
    if not a:
        print("Заданный список пуст", file=sys.stderr)
        exit(1)

    # Определить индексы минимального и максимального элементов.
    a_min = a_max = a[0]
    i_min = i_max = 0
    for i, item in enumerate(a):
        if item < a_min:
            i_min, a_min = i, item
        if item >= a_max:
            i_max, a_max = i, item

    # Проверить индексы и обменять их местами.
    if i_min > i_max:
        i_min, i_max = i_max, i_min

    # Посчитать количество положительных элементов.
    count = 0
    for item in a[i_min+1:i_max]:
        if item > 0:
            count += 1

    print(count)
```

```
/Users/svetik/.conda/envs/Pycharm/bi
-5 3 2 7 -4 5 1 33
5
Process finished with exit code 0
```

Рисунок 7 – Результат работы программы

8. Выполните индивидуальные задания, согласно своему варианту. Для заданий повышенной сложности номер варианта должен быть получен у преподавателя.

Задание 1. Составить программу с использованием одномерных массивов для решения задачи. Номер варианта необходимо получить у преподавателя. Решить индивидуальное задание как с использованием циклов, так и с использованием List Comprehensions.

Вариант 16. Ввести список А из 10 элементов, найти сумму элементов кратных 2, их количество и вывести результаты на экран.

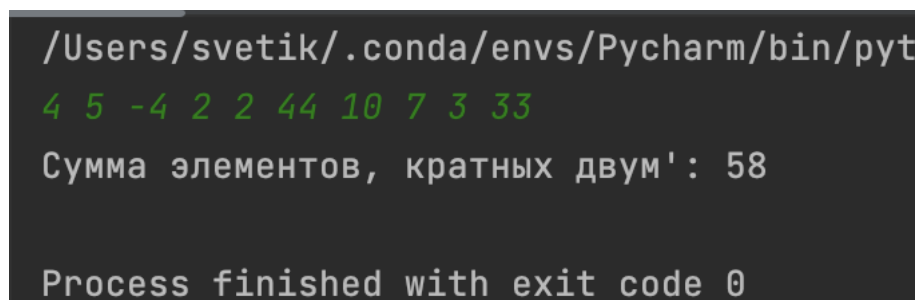
Решение задачи с помощью цикла:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == '__main__':
    # Ввести список одной строкой.
    a = list(map(int, input().split()))
    # Проверить количество элементов списка.
    if len(a) != 10:
        print("Неверный размер списка", file=sys.stderr)
        exit(1)

    # Нахождение суммы элементов, кратных двум
    summ = 0
    for i in a:
        if i % 2 == 0:
            summ += i
    print(f"Сумма элементов, кратных двум': {summ}")
```



```
/Users/svetik/.conda/envs/Pycharm/bin/python
4 5 -4 2 2 44 10 7 3 33
Сумма элементов, кратных двум': 58
Process finished with exit code 0
```

Рисунок 8 – Результат работы программы

Решение задачи с помощью списковых включений:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == '__main__':
    # Ввести список одной строкой.
    A = list(map(int, input().split()))
    # Проверить количество элементов списка.
    if len(A) != 10:
```

```

print("Неверный размер списка", file=sys.stderr)
exit(1)

# Найти сумму элементов, кратных двум.
s = sum([a for a in A if a % 2 == 0])
print(s)

```

```

/Users/svetik/.conda/envs/Pycharm/bin/
2 44 -2 3 6 7 11 6 10 12
78
Process finished with exit code 0

```

Рисунок 9 – Результат работы программы

Задание 2. Составить программу с использованием одномерных массивов для решения задачи на переупорядочивание элементов массива. Для сортировки допускается использовать метод `sort` с заданным параметром `key` (<https://docs.python.org/3/howto/sorting.html>) и объединение нескольких списков. Номер варианта необходимо получить у преподавателя.

Вариант 16. В списке, состоящем из вещественных элементов, вычислить:

1. количество положительных элементов списка;
2. сумму элементов списка, расположенных после последнего элемента, равного нулю.

Преобразовать список таким образом, чтобы сначала располагались все элементы, целая часть которых не превышает 1, а потом - все остальные.

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == '__main__':
    # Ввести список одной строкой.
    a = list(map(float, input().split()))

    # Нахождение количества положительных элементов
    n = 0
    for i in range(len(a)):
        if a[i] > 0:
            n += 1
    print(f"Количество положительных элементов: {n}")

```

```
# Нахождение суммы элементов, расположенных после последнего элемента,
# равного нулю
if 0 in a:
    ind = len(a) - a[::-1].index(0)
    sum = sum(a[ind:])
    print(f"Сумма элементов, расположенных после последнего элемента,
    равного нулю: {sum}")

# Сортировка
a = [i for i in a if i // 10 <= 1] + [i for i in a if i // 10 > 1]
print(f"Отсортированный список: {a}")
```

```
/Users/svetik/.conda/envs/Pycharm/bin/python /Users/svetik/Desktop/Laba7/LR_2.4,
1 0 22 33 -4 88 0 2 11 4
Количество положительных элементов: 9
Сумма элементов, расположенных после последнего элемента, равного нулю: 17.0
Отсортированный список: [1.0, 0.0, -4.0, 0.0, 2.0, 11.0, 4.0, 22.0, 33.0, 88.0]
|
Process finished with exit code 0
```

Рисунок 10 – Результат работы программы

Вопросы для защиты работы:

1. Что такое списки в языке Python?

Список — это изменяемый упорядоченный тип данных предоставляющий возможность хранения объектов разных типов.

2. Как осуществляется создание списка в Python?

Для этого необходимо воспользоваться следующей конструкцией:

имя_переменной = [перечисление элементов через запятую]

или

имя_переменной = []

3. Как организовано хранение списков в оперативной памяти?

Объект списка хранит указатели на объекты, а не на сами объекты, при этом элементы могут быть «разбросаны» по памяти.

4. Каким образом можно перебрать все элементы списка?

С помощью цикла

5. Какие существуют арифметические операции со списками?

1) Объединение списков (+)

2) Умножение на число (*)

6. Как проверить есть ли элемент в списке?

Для этого можно использовать оператор in/not in.

7. Как определить число вхождений заданного элемента в списке?

Для этого используется метод count (имя_списка.count(элемент))

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

Существует несколько методов:

`имя_списка.append(элемент)` – добавляет в конец

`имя_списка.insert(индекс, элемент)` – добавляет по индексу со смещением всех последующих элементов.

9. Как выполнить сортировку списка?

Для сортировки списка нужно использовать метод `sort` (`имя_списка.sort()`) и `sort(reverse=True)` для сортировки в порядке убывания.

10. Как удалить один или несколько элементов из списка?

Для этого существуют методы `.pop(индекс)` – удаляет по индекс и возвращает удаленное значение; `.remove(элемент)` – удаляет первое вхождение. Также можно использовать оператор `del имя_списка[индекс]`, если поместить срез, удалиться несколько элементов. Удалить все элементы можно с помощью метода `.clear()`.

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

Списковое включение – это некий синтаксический сахар, позволяющий упростить генерацию последовательностей (списков, множеств, словарей, генераторов).

`новый_список = [«операция» for «элемент списка» in «список»]`

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

Срез имеет вид: `имя_списка[start:stop:step]`, где `start` – индекс первого элемента, `stop` – индекс крайнего элемента (сам он не включается), `step` – шаг. При этом `start`, `stop`, `step` необязательно должны принимать значения, так отсутствие `start` означает срез с начала, `stop` – до конца, `step` – каждый элемент.

Также их они могут принимать отрицательные значения, тогда -1 = последний элемент, -2 = предпоследний, отрицательный шаг = шаг назад. Важно, что элементы должны идти «в направлении» шага.

13. Какие существуют функции агрегации для работы со списками?

`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.

Важно, что для `min` и `max` элементы должны быть сравнимы

14. Как создать копию списка?

Это можно сделать с помощью срезов типа `a[:]`

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

Если `sort()` изменяет список, ничего не возвращая, то `sorted` возвращает измененный список, при этом не меняя исходный.