

**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

Халимендик Я. Д. « » 2022г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

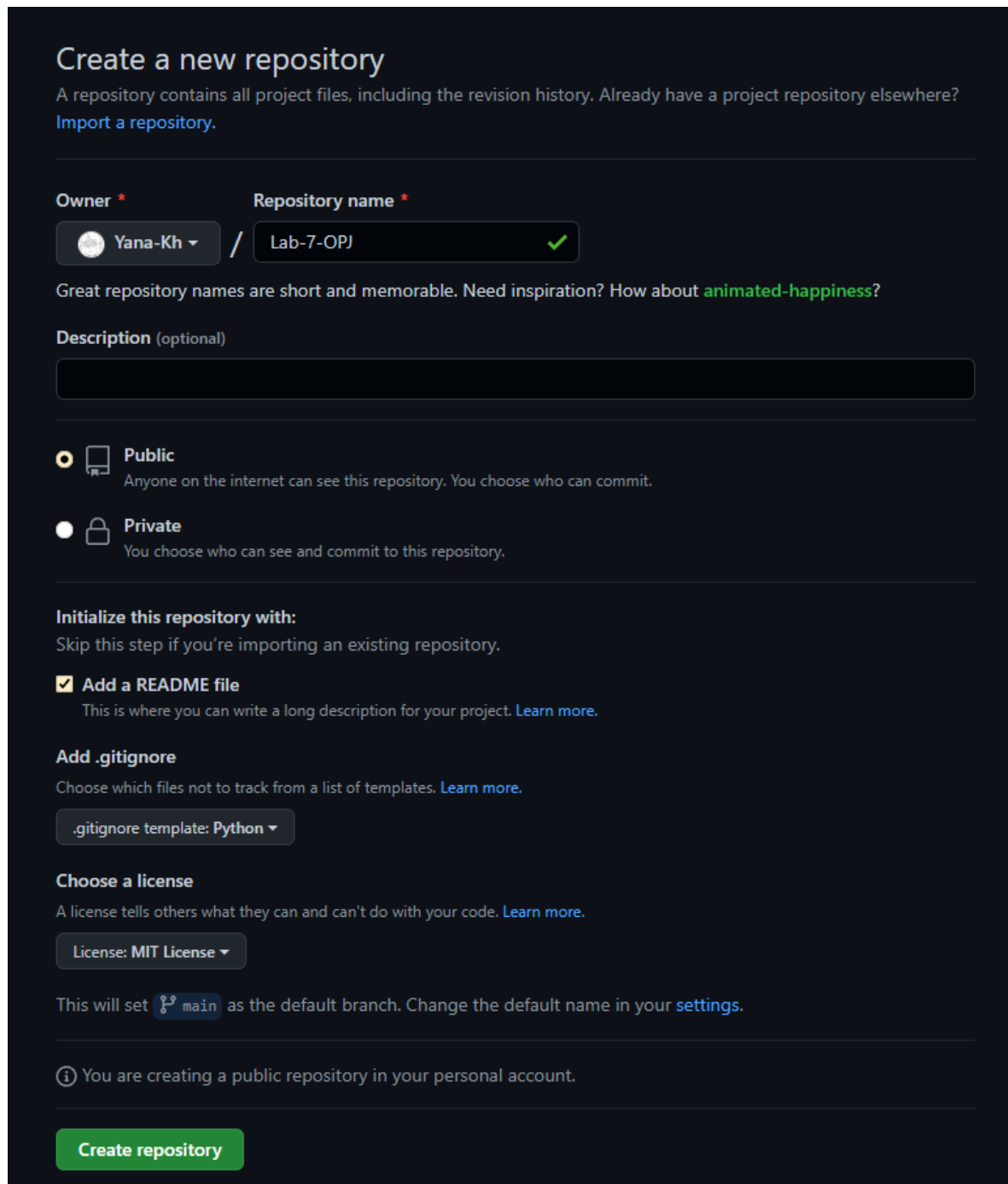
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2022

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

Ход 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.](#)

Owner * Repository name *

Yana-Kh / Lab-7-OPJ ✓

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

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:
Skip this step if you're importing an existing repository.

☒ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python ▾

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

License: MIT License ▾

This will set `main` as the default branch. Change the default name in your [settings](#).

i You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание репозитория

3. Выполните клонирование созданного репозитория.

```
C:\Users\ynakh\OneDrive\Рабочий стол\Git>git clone https://github.com/Yana-Kh/Lab-7-OPJ.git
Cloning into 'Lab-7-OPJ'...
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.

C:\Users\ynakh\OneDrive\Рабочий стол\Git>cd C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-7-OPJ
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-7-OPJ>
```

Рисунок 2 – Клонирование репозитория

4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.

```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-7-OPJ>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   .gitignore

C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-7-OPJ>
```

Рисунок 3 – Дополнение файла .gitignore

5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-7-OPJ>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/ynakh/OneDrive/Рабочий стол/Git/Lab-7-OPJ/.git/hooks]

C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-7-OPJ>
```

Рисунок 4 – Организация репозитория в соответствии с моделью git-flow

6. Создайте проект PyCharm в папке репозитория.

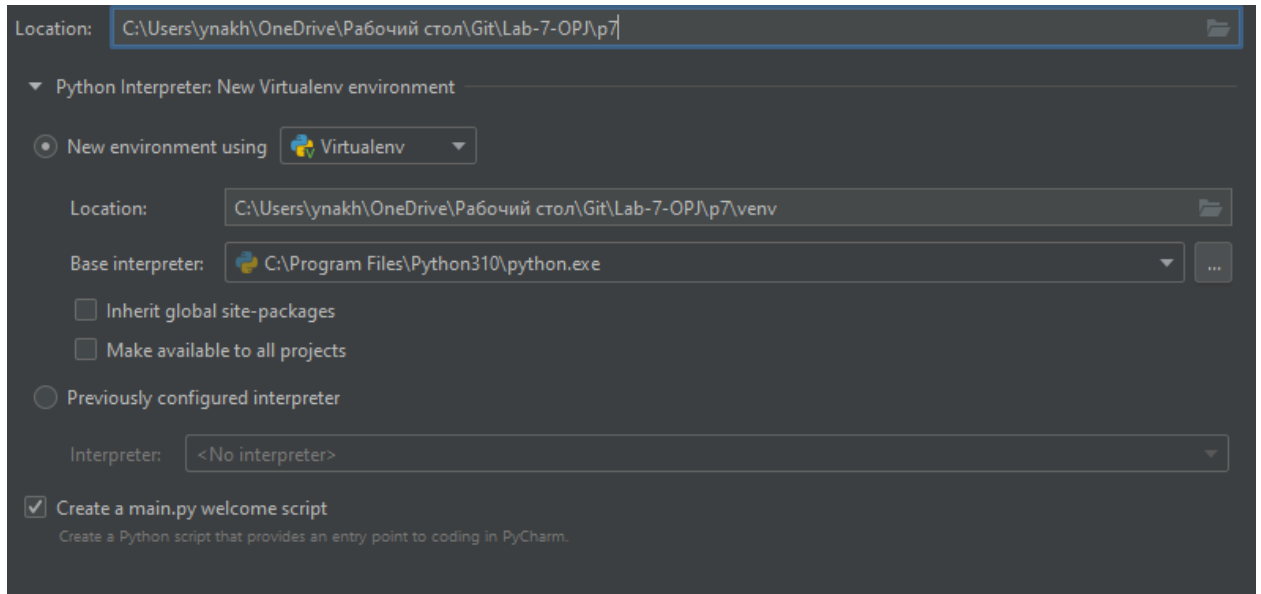


Рисунок 5 – Создание проекта PyCharm в папке репозитория

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.

Код:

```
#!/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)
```

```
1 -4 6 27 -95 34 -6 3 1 0
1
Process finished with exit code 0
```

Рисунок 6 – Результат работы программы

Решение задачи с помощью списковых включений:

Код:

```
#!/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)
```

```
3 5 81 4 7 36 5 1 0 -4
4
Process finished with exit code 0
```

Рисунок 7 – Результат работы программы

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

Код:

```
#!/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)

```

```

8 9 76 9 1 8 68 0
4
Process finished with exit code 0

```

Рисунок 8 – Результат работы программы

```

9 -55 9 6 4 -7 0 78
3
Process finished with exit code 0

```

Рисунок 9 – 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 4(32). Ввести список А из 10 элементов, найти сумму отрицательных элементов и вывести ее на экран.

Решение задачи с помощью цикла:

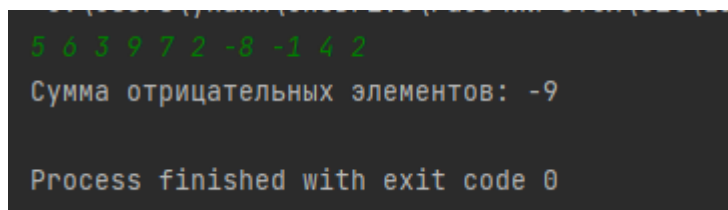
Код:

```
#!/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 отрицательных элементов
    summ = 0
    for i in a:
        if i < 0:
            summ += i
    print(f"Сумма отрицательных элементов: {summ}")
```



```
5 6 3 9 7 2 -8 -1 4 2
Сумма отрицательных элементов: -9

Process finished with exit code 0
```

Рисунок 10 – Результат работы программы

Решение задачи с помощью списковых включений:

Код:

```
#!/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 отрицательных элементов
summ = sum(i for i in a if i < 0)
print(f"Сумма отрицательных элементов: {summ}")
```

```
9 5 -9 -8 -3 5 6 2 1 8
Сумма отрицательных элементов: -20
Process finished with exit code 0
```

Рисунок 11 – 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 13(32). В списке, состоящем из вещественных элементов, вычислить:

1. количество элементов списка, равных 0;
2. сумму элементов списка, расположенных после минимального элемента.

Упорядочить элементы списка по возрастанию модулей элементов

Код:

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

import functools

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

    # Нахождение нулевых элементов и минимального элемента
    n = 0
    a_min = a[0]
    i_min = 0
    for ind, value in enumerate(a):
        if value == 0:
            n += 1
        if value < a_min:
            a_min = value
            i_min = ind
    print(f"Количество нулевых элементов: {n}")
```



```
# Нахождение суммы элементов после минимального
a1 = a[i_min + 1:]
s = sum(i for i in a1)
print(f"Сумма элементов после минимального : {s}")

# Сортировка
i = 0
while i < len(a):
    j = 0
    while j < len(a) - 1:
        if abs(a[j]) > abs(a[j + 1]):
            a[j], a[j+1] = a[j+1], a[j]
        j += 1
    i += 1
print(f"Отсортированный список: {a}")
```

```
7 -4 0 9 0 -3 6 19
Количество нулевых элементов: 2
Сумма элементов после минимального : 31.0
Отсортированный список: [0.0, 0.0, -3.0, -4.0, 6.0, 7.0, 9.0, 19.0]

Process finished with exit code 0
```

Рисунок 12 – 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, например:

```
s = [1, 2, 3, 5]
for i in s:
    print(i)
```

Результат:

```
1
2
3
5
```

5. Какие существуют арифметические операции со списками?

1) Объединение списков (+)

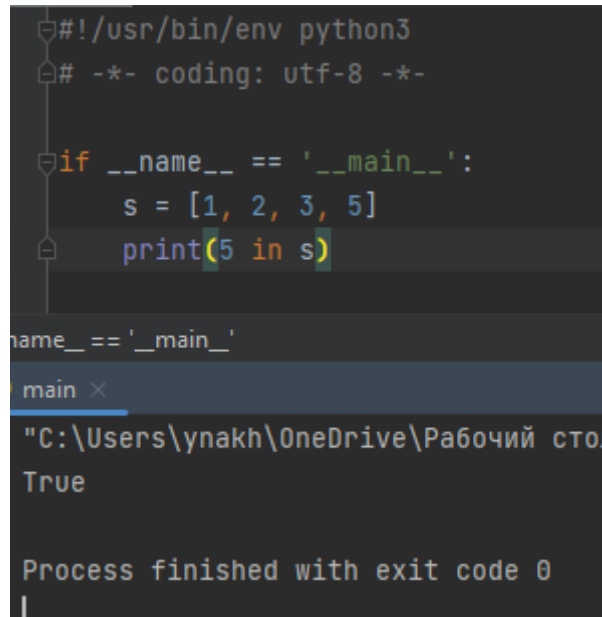
2) Умножение на число (*)

6. Как проверить есть ли элемент в списке?

Для этого можно использовать оператор in/not in. Например:

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

if __name__ == '__main__':
    s = [1, 2, 3, 5]
    print(5 in s)
```



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, удалить 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.