

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 №13
дисциплины «Основы программной инженерии»

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

(подпись)

Доцент кафедры инфокоммуникаций
Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: Лабораторная работа 2.10. Функции с переменным числом параметров.

Цель работы: приобретение навыков по работе с функциями с переменным числом параметров при написании программ с помощью языка программирования Python версии 3.x.


Порядок выполнения работы

1. Создание репозитория GitHub.

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 *
 ksenia-lamskaya ▾	/ 12laba
	✔ 12laba is available.

Great repository names are short and memorable. Need inspiration? How about [ideal-octo-chainsaw](#) ?

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

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

Choose a license

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

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

 You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание репозитория

2. Проработайте пример из методички.

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

def median(*args):
    if args:
        values = [float(arg) for arg in args]
        values.sort()

        n = len(values)
        idx = n // 2
        if n % 2:
            return values[idx]
        else:
            return (values[idx - 1] + values[idx]) / 2

    else:
        return None

if __name__ == "__main__":
    print(median())
    print(median(3, 7, 1, 6, 9))
    print(median(1, 5, 8, 4, 3, 9))
```

Рисунок 2.1 – Пример кода

```
None
6.0
4.5
```

Рисунок 2.2 – Вывод программы

3.

8. Решить поставленную задачу: написать функцию, вычисляющую среднее геометрическое своих аргументов a_1, a_2, \dots, a_n

$$G = \sqrt[n]{\prod_{k=1}^n a_k}. \quad (1)$$

Если функции передается пустой список аргументов, то она должна возвращать значение `None`.

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

import math

def average(*a):
    if a:
        n = len(a)
        y = 1
        for i in a:
            y *= i

        g = math.pow(y, 1/n)
        return g

    else:
        return None

if __name__ == "__main__":
    arg = list(int(i) for i in input("Введите значения: ").split())
    result = average(*arg)
    print(result)
```

Рисунок 3.1 – Код программы

```
Введите значения: 1 2 3 4 4 6
2.8844991406148166
```

Рисунок 3.2 – Вывод программы

4.

9. Решить поставленную задачу: написать функцию, вычисляющую среднее гармоническое своих аргументов a_1, a_2, \dots, a_n

$$\frac{n}{H} = \sum_{k=1}^n \frac{1}{a_k}. \quad (2)$$

Если функции передается пустой список аргументов, то она должна возвращать значение `None`.

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

import math

def harmonycal(*a):
    if a:
        n = len(a)
        y = 0
        for i in a:
            y += 1/i

        return n/y

    else:
        return None

if __name__ == "__main__":
    arg = list(int(i) for i in input("Введите значения: ").split())
    result = harmonycal(*arg)
    print(result)
```

Рисунок 4.1 – Код программы

```
Введите значения: 1 2 3 4
1.9200000000000004
```

Рисунок 4.2 – Вывод программы

5. Сумму модулей аргументов, расположенных после первого аргумента, равного нулю.

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

def abss(*a):
    if a:
        summ = 0
        flag_zero = False

        for arg in a:
            if arg == 0:
                flag_zero = True
            elif flag_zero:
                summ += abs(arg)

        return summ

    else:
        return None

if __name__ == "__main__":
    p = list(int(i) for i in input("Введите значения: ").split())
    result = abss(*p)
    print(result)
```

Рисунок 5.1 - Код программы

```
Введите значения: 1 45 5 0 9 -4
13
```

Рисунок 5.2 – Вывод программы

Ответы на контрольные вопросы

1. Какие аргументы называются позиционными в Python?

Позиционные аргументы передаются в функцию в том порядке, в котором они объявлены в сигнатуре функции. Значения, переданные в качестве аргументов, присваиваются параметрам в том порядке, в котором они объявлены в определении функции.

2. Какие аргументы называются именованными в Python?

Именованные аргументы передаются с указанием имени параметра и значения, которое вы хотите присвоить этому параметру. Именованные аргументы могут быть переданы в любом порядке.

3. Для чего используется оператор * ?

Благодаря использованию * мы создаем список позиционных аргументов на основе того, что было передано функции при вызове. Также наоборот раскрываем список, раскладывая по элементам.

4. Каково назначение конструкций *args и **kwargs ?

Конструкции *args и **kwargs в Python используются для передачи переменного числа аргументов в функцию. Они облегчают работу с функциями, которые могут принимать разное количество аргументов. *args позволяет передавать переменное количество позиционных аргументов в функцию. Звездочка (*) перед именем args означает, что все аргументы, следующие после *args, будут собраны в кортеж. **kwargs позволяет передавать переменное количество именованных (ключевых) аргументов в функцию. Звездочки с двумя знаками перед именем kwargs означают, что все переданные именованные аргументы будут собраны в словарь.