

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»**

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 \_\_\_\_\_

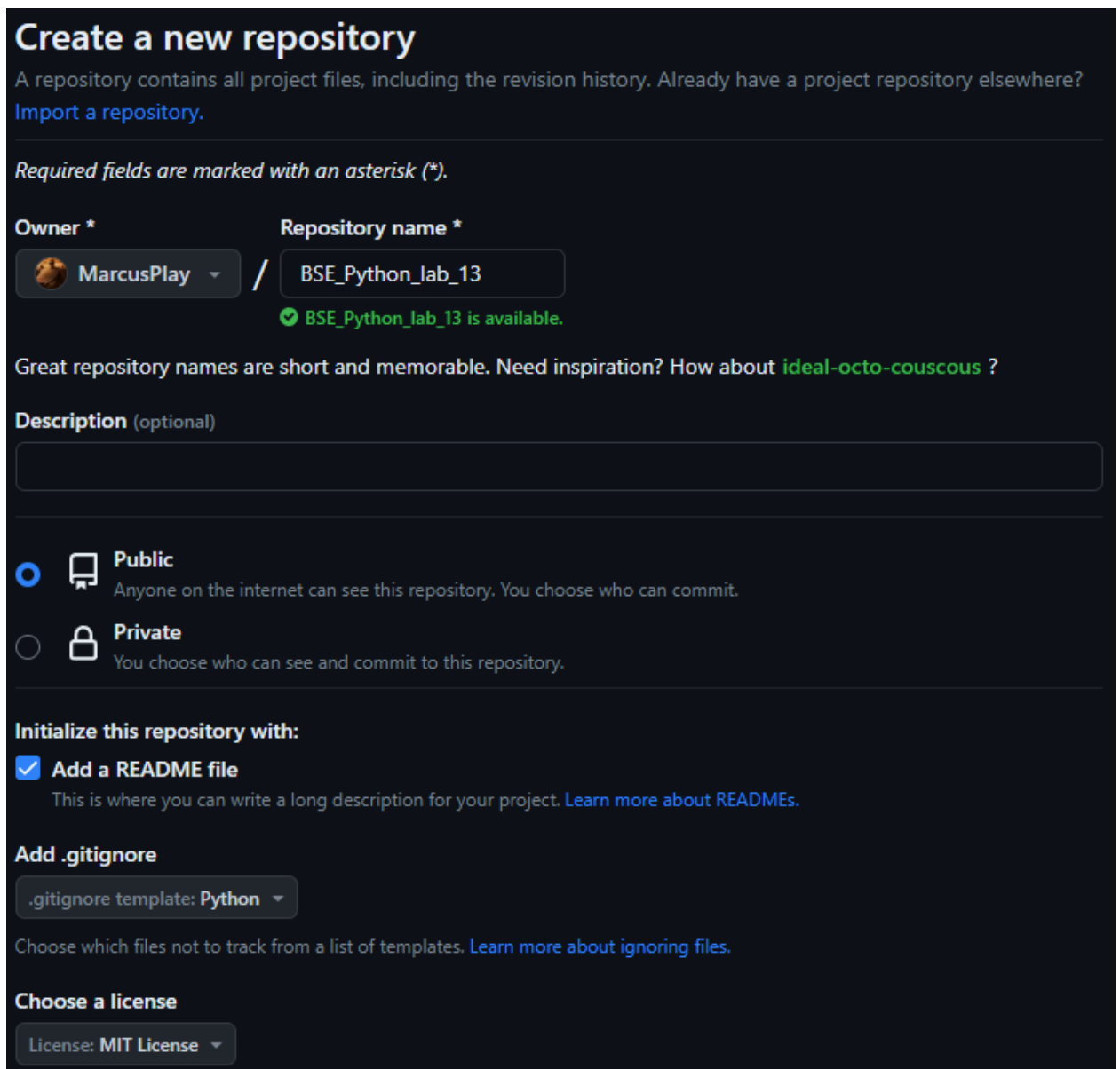
Ставрополь, 2023 г.

**ТЕМА:** Функции с переменным числом параметров в Python

**ЦЕЛЬ РАБОТЫ:** приобретение навыков по работе с функциями с переменным числом параметров 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\_13 is available.

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

**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**

3. Выполните клонирование созданного репозитория.

```
Imran@Kaskad MINGW64 ~/Desktop/Лабораторная работа №13
$ git clone https://github.com/MarcusPlay/BSE_Python_lab_13.git
Cloning into 'BSE_Python_lab_13'...
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.
```

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

```
Imran@Kaskad MINGW64 ~/Desktop/Лабораторная работа №13/BSE_Python_lab_13 (main)
$ git checkout -b develop
Switched to a new branch 'develop'
```

5. Проработать примеры лабораторной работы.

Пример 1.

```
#!/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))
```

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

$$G = \sqrt[n]{\prod_{k=1}^n a_k}$$

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

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

def geometric_mean(*args):
    if len(args) == 0:
        return None

    ans = 1
    for i in args:
        ans *= i
    return (ans)**(1/(len(args)))

if __name__=="__main__":
    a = list(map(int, input("Введите значения через пробел:\n").split()))
    print(geometric_mean(*a))
```

```
Imran@Kaskad MINGW64 ~/Desktop/Лабораторная работа №13/BSE_Python_lab_13 (develop)
$ python task_1.py
Введите значения через пробел:
1 2 3 4 5 6
2.993795165523909
```

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

$$\frac{n}{H} = \sum_{k=1}^n \frac{1}{a_k}$$

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

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

def garmonic_mean(*args):
    if len(args) == 0:
        return None

    ans = 0
    for i in args:
        ans += 1/i

    return (len(args)/ans)

if __name__=="__main__":
    a = list(map(int, input("Введите значения через пробел:\n").split()))
    print(garmonic_mean(*a))
```

```
Imran@Kaskad MINGW64 ~/Desktop/Лабораторная работа №13/BSE_Python_lab_13 (develop)
$ python task_2.py
Введите значения через пробел:
2 3 4
2.7692307692307696
```

8. Приведите в отчете скриншоты результатов выполнения примера при различных исходных данных, вводимых с клавиатуры.

9. Зафиксируйте изменения в репозитории.

```
Imran@Kaskad MINGW64 ~/Desktop/Лабораторная работа №13/BSE_Python_lab_13 (develop)
$ git commit -m "added example, task_1 and task_2"
[develop 438515c] added example, task_1 and task_2
3 files changed, 59 insertions(+)
create mode 100644 examples/example_1.py
create mode 100644 task_1.py
create mode 100644 task_2.py
```

10. Решите индивидуальное задание согласно своему варианту. Напишите функцию, принимающую произвольное количество аргументов, и возвращающую требуемое значение. Если функции передается пустой список аргументов, то она должна возвращать значение None. Номер варианта 12. В процессе решения не использовать преобразования конструкции `*args` в список или иную структуру данных. Найдите сумму аргументов, расположенных после максимального аргумента.

```
def f(*args):
    if len(args) == 0:
        return None

    arg_max = max(*args)

    if args.index(arg_max) == (len(args) - 1):
        return 0

    arg_sum = sum(args[args.index(arg_max) + 1:])
    return arg_sum

if __name__ == "__main__":
    ls = list(map(float, input("Введите значение через пробел:\n$ ").split()))
    print(f(*ls))
```

```
Imran@Kaskad MINGW64 ~/Desktop/Лабораторная работа №13/BSE_Python_lab_13 (develop)
$ python3 individual_task.py
Введите значение через пробел:
$ 1 2 45 32 2
34.0
```

11. Самостоятельно подберите или придумайте задачу с переменным числом именованных аргументов. Приведите решение этой задачи.

Реализуйте функцию `calculate`, которая принимает различные арифметические операции и переменное число операндов в виде именованных аргументов. Поддерживаемые операции могут включать сложение, вычитание, умножение и деление.

```
def calculate(**kwargs):
    result = None

    if 'operation' in kwargs and 'operands' in kwargs:
        operation = kwargs['operation']
        operands = kwargs['operands']

    match operation:
        case 'add':
            result = sum(operands)
        case 'subtract':
            result = operands[0] - sum(operands[1:])
        case 'multiply':
            result = 1
            for operand in operands:
                result *= operand
        case 'divide':
            if all(operands[1:]):
                result = operands[0] / operands[1]

    return result

if __name__ == "__main__":
    operation = input("Введите какую операцию хотите сделать:\nadd, subtract, multiply, divide\n$ ")
    operands = list(map(int, input("Введите значения через пробел:\n").split()))
    result = calculate(operation=operation, operands=operands)
    print("Результат:", result)
```

```
Imran@Kaskad MINGW64 ~/Desktop/Лабораторная работа №13/BSE_Python_lab_13 (develop)
$ python3 task_3.py
Введите какую операцию хотите сделать:
add, subtract, multiply, divide
$ add
Введите значения через пробел:
2 3 4
Результат: 9
```

```
Imran@Kaskad MINGW64 ~/Desktop/Лабораторная работа №13/BSE_Python_lab_13 (develop)
$ python3 task_3.py
Введите какую операцию хотите сделать:
add, subtract, multiply, divide
$ divide
Введите значения через пробел:
2 3
Результат: 0.6666666666666666
```

## 12. Зафиксируйте изменения в репозитории.

```
Imran@Kaskad MINGW64 ~/Desktop/Лабораторная работа №13/BSE_Python_lab_13 (develop)
$ git commit -m "added individual_task and task_3"
[develop dbecdc6] added individual_task and task_3
4 files changed, 68 insertions(+)
create mode 100644 individual_task.py
create mode 100644 task_3.py
```



13. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.
14. Выполните слияние ветки для разработки с веткой master/main.
15. Отправьте сделанные изменения на сервер GitHub.
16. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

## **ВОПРОСЫ ДЛЯ ЗАЩИТЫ РАБОТЫ**

1. Какие аргументы называются позиционными в Python?

Позиционные аргументы в Python — это аргументы, передаваемые в функцию в порядке, определенном при объявлении функции. Позиция аргумента в вызове функции соответствует порядку объявления в функции.

2. Какие аргументы называются именованными в Python?

Именованные аргументы в Python — это аргументы, передаваемые в функцию с явным указанием их имени в виде "имя=значение". Порядок их указания не важен, так как они идентифицируются по имени.

3. Для чего используется оператор \* ?

Оператор \* в Python используется для распаковки последовательности, такой как список или кортеж. Например, он может использоваться для передачи элементов списка в виде аргументов функции.

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

Конструкция \*args в определении функции позволяет передавать переменное количество позиционных аргументов. Она собирает оставшиеся аргументы в кортеж.

Конструкция \*\*kwargs аналогично позволяет передавать переменное количество именованных аргументов. Она собирает их в словарь, где ключами являются имена аргументов, а значениями — соответствующие значения.

Эти конструкции полезны, когда вы хотите создать функцию, которая может принимать разное количество аргументов без предварительного указания их числа.