

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего
образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ
УНИВЕРСИТЕТ»**

**Кафедра
инфокоммуникаций
Институт цифрового
развития**

**ОТЧЁТ
по лабораторной работе №2.10**

Дисциплина: «Основы программной инженерии»

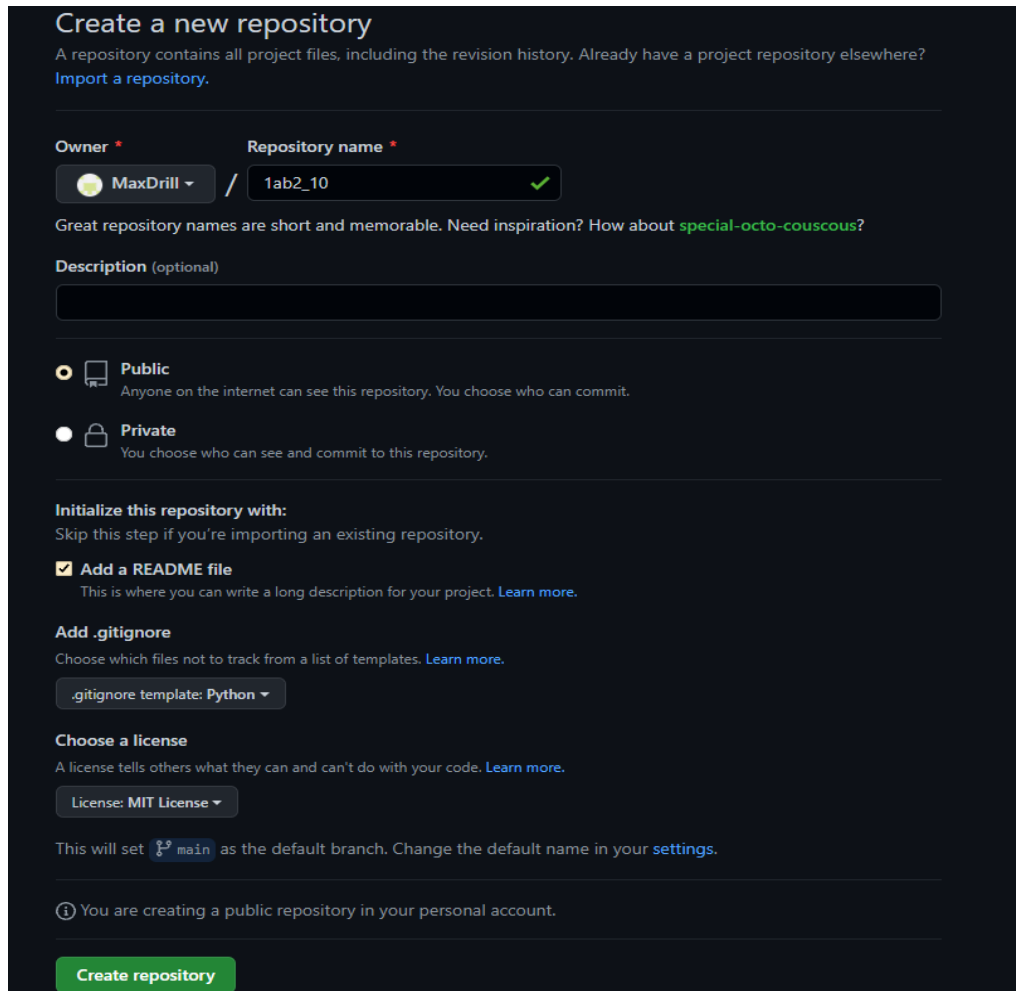
Тема: «Функции с переменным числом параметров в
Python»

Выполнил:
студент 2 курса
группы Пиж-б-о-21-1
Коныжев Максим
Викторович

Ставрополь 2022

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

1. Был создан репозиторий в Github в который были добавлены правила gitignore для работы IDE PyCharm, была выбрана лицензия MIT, сам репозиторий был клонирован на локальный сервер и был организован в соответствии с моделью ветвления git-flow.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * MaxDrill / **Repository name *** 1ab2_10 ✓

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

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

Create repository

Рисунок 1 – Создание репозитория

```
C:\Users\UESR\gitproj\lab2_10>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/UESR/gitproj/lab2_10/.git/hooks]

C:\Users\UESR\gitproj\lab2_10>
```

Рисунок 2 – Организация репозитория с моделью ветвления git-flow

```

C:\Users\UESR\Desktop\2.1\proj\venv\Scripts\python
None
6.0
4.5

Process finished with exit code 0

```

Рисунок 3 – Результат работы примера

2. 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.

Код программы:

```

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

def sr_geom(*args):
    if args:
        values = [float(arg) for arg in args]
        g = 1
        for arg in values:
            g = g * arg
        n = len(args)
        return pow(g, 1/n)
    else:
        return None

if __name__ == "__main__":
    print(f'Среднее геометрическое 1: {sr_geom(1, 2, 3, 4, 5)}')
    print(f'Среднее геометрическое 1: {sr_geom()}')
    print(f'Среднее геометрическое 1: {sr_geom(1.3, 7.9, 8.3)}')

```

```

C:\Users\UESR\Desktop\2.1\proj\venv\Scripts\python.e
Среднее геометрическое 1: 2.605171084697352
Среднее геометрическое 1: None
Среднее геометрическое 1: 4.400981186140995

Process finished with exit code 0

```

Рисунок 4 – Результат работы программы

3. Решить поставленную задачу: написать функцию, вычисляющую

среднее гармоническое своих аргументов a_1, a_2, \dots, a_n

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

Код программы:

```

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

def sr_harm(*args):
    if args:
        values = [float(arg) for arg in args]
        h = 0
        for arg in values:
            h += 1 / arg
        n = len(args)
        return n / h
    else:
        return None

if __name__ == "__main__":
    print(f'Среднее гармоническое: {sr_harm(1, 2, 3, 4, 5, 6)}')
    print(f'Среднее гармоническое: {sr_harm()}')
    print(f'Среднее гармоническое: {sr_harm(3.3, 6.5, 9.7, 8.0)}')

```

```

C:\Users\UESR\Desktop\2.1\proj\venv\Scripts\python.
Среднее гармоническое: 2.4489795918367347
Среднее гармоническое: None
Среднее гармоническое: 5.83967828653373

Process finished with exit code 0

```

Рисунок 5 – Результат работы программы

4. Было выполнено индивидуальное задание

Произведение аргументов, расположенных после максимального по модулю аргумента.

Код программы:

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

import math

def mult_after_max(*args):
    """
    Произведение аргументов, расположенных после
    максимального по модулю аргумента.
    """
    if args:
        mult = 1
        values = [float(arg) for arg in args]
        ind = max([math.fabs(item), i] for i, item in enumerate(values))
        values = values[ind[1]:]
        for arg in values:
            mult = mult * arg
        return mult
    else:
        return None

if __name__ == "__main__":
    print(f'Произведение: {mult_after_max()}')
    print(f'Произведение: {mult_after_max(1, 3, 6, 4, 5)}')
    print(f'Произведение: {mult_after_max(7, 8, 12, -76, 34, 7, 34)}')
```

```
C:\Users\UESR\Desktop\2.1\proj\venv\Scripts\python.exe
Произведение: None
Произведение: 120.0
Произведение: -614992.0

Process finished with exit code 0
```

Рисунок 6 – Результат работы программы

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

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

def print_cars(**cars):
    if cars:
        for name, mark in cars.items():
            print(f"{name}: {mark}")
    else:
        return None

if __name__ == "__main__":
    print_cars(
        Toyota="Mark2", Nissan="Silvia", Mazda="RX-7",
        Honda="Civic", Tesla="CyberTruck", Porsche="Macan",
    )
```

```
C:\Users\UESR\Desktop\2.1\proj\venv\Scripts\p
Toyota: Mark2
Nissan: Silvia
Mazda: RX-7
Honda: Civic
Tesla: CyberTruck
Porsche: Macan

Process finished with exit code 0
```

Рисунок 7 – Результат работы программы

```
C:\Users\UESR\gitproj\lab2_10>git merge develop
Updating 41775eb..0d2959a
Fast-forward
 proj/lab13_ex1.py | 23 ++++++
 proj/lab13_ex2.py | 20 ++++++
 proj/lab13_ex3.py | 20 ++++++
 proj/lab13_idz1.py | 27 ++++++
 4 files changed, 90 insertions(+)
 create mode 100644 proj/lab13_ex1.py
 create mode 100644 proj/lab13_ex2.py
 create mode 100644 proj/lab13_ex3.py
 create mode 100644 proj/lab13_idz1.py
```

Рисунок 7 – Коммит и пуш изменений переход на ветку main и слияние ее с веткой develop

Вывод: в ходе лабораторной работы были приобретены навыки по работе с функциями с переменным числом параметров при написании программ с помощью языка программирования Python версии 3.x.

Контрольные вопросы

1. Какие аргументы называются позиционными в Python?

Это аргументы, передаваемые в вызов в определённой последовательности (на определённых позициях), без указания их имён. Элементы объектов, поддерживающих итерирование, могут использоваться в качестве позиционных аргументов, если их распаковать при помощи `*`.

2. Какие аргументы называются именованными в Python?

Это аргументы, передаваемые в вызов при помощи имени (идентификатора), либо словаря с его распаковкой при помощи `**`.

3. Для чего используется оператор `*`?

Этот оператор позволяет «распаковывать» объекты, внутри которых хранятся некие элементы.

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

`*args` используется для передачи произвольного числа именованных аргументов функции.

`**kwargs` позволяет передавать произвольное число именованных аргументов в функцию.