

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙ-
СКОЙ ФЕДЕРАЦИИ**

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

Кафедра инфокоммуникаций

Основы кроссплатформенного программирования

Отчет по лабораторной работе №2.10

Тема: «Функции с переменным числом параметров в Python»

Выполнил студент группы

ИВТ-б-о-21-1

Криворот В.Г. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил доцент

Кафедры инфокоммуникаций, старший
преподаватель

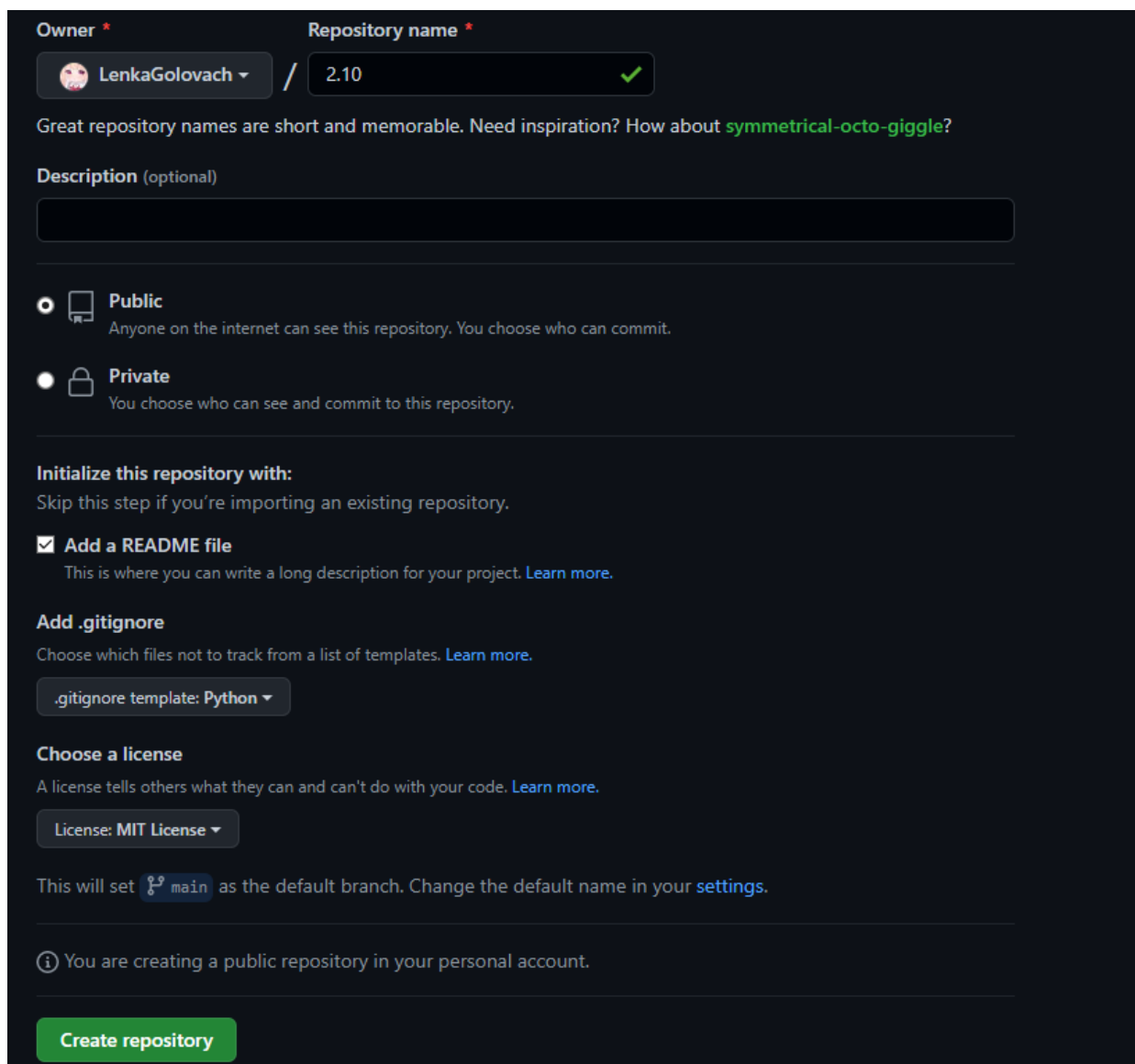
Воронкин Р.А.

(подпись)

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.

Ход работы:



The screenshot shows the GitHub 'Create repository' form. At the top, the 'Owner' is set to 'LenkaGolovach' and the 'Repository name' is '2.10', which is marked as valid with a green checkmark. Below this, a hint suggests repository names should be short and memorable. The 'Description' field is empty. Under the 'Visibility' section, 'Public' is selected, indicating that anyone on the internet can see the repository. The 'Initialize this repository with' section has 'Add a README file' checked. The '.gitignore' section shows a template for 'Python'. The 'Choose a license' section has 'MIT License' selected. At the bottom, it states the default branch will be 'main'. A green 'Create repository' button is at the very bottom.

1.

Создал репозиторий в GitHub, дополнил правила в .gitignore для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на компьютер и организовал в соответствии с моделью ветвления git-flow.

Рисунок 1.1 – Созданный репозиторий

```
.gitignore – Блокнот
Файл Правка Формат Вид Справка
.idea/
# Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
# Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm

### PyCharm ###
# Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio, WebStorm and Rider
# Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839

# User-specific stuff
.idea/**/workspace.xml
.idea/**/tasks.xml
.idea/**/usage.statistics.xml
.idea/**/dictionaries
.idea/**/shelf

# AWS User-specific
.idea/**/aws.xml

# Generated files
.idea/**/contentModel.xml

# Sensitive or high-churn files
.idea/**/dataSources/
.idea/**/dataSources.ids
.idea/**/dataSources.local.xml
.idea/**/sqlDataSources.xml
```

< Стр 14, слб 15 100% Windows (CRLF)

Рисунок 1.2 – Дополнил правила в .gitignore

```

c:\Users\Admin\Desktop\git\Python5>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/Admin/Desktop/git/Python5/.git/hooks]

c:\Users\Admin\Desktop\git\Python5>

```

Рисунок 1.3 – Организация репозитория в соответствии с моделью ветвления git-flow

2. Создал проект Pycharm в папке репозитория, проработал примеры ЛР.

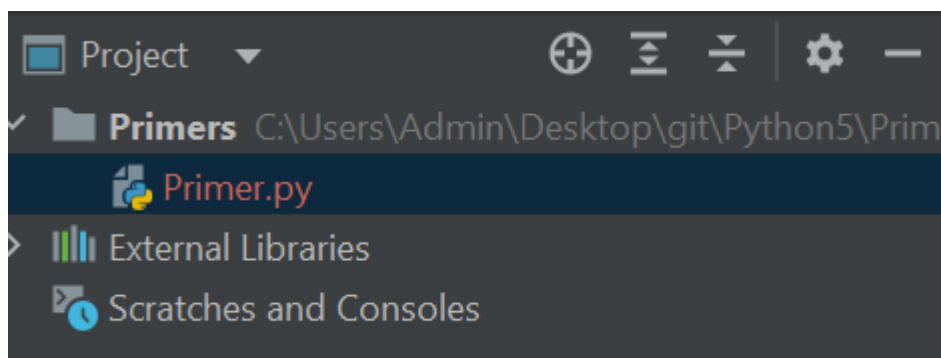


Рисунок 2.1 – Созданные проекты

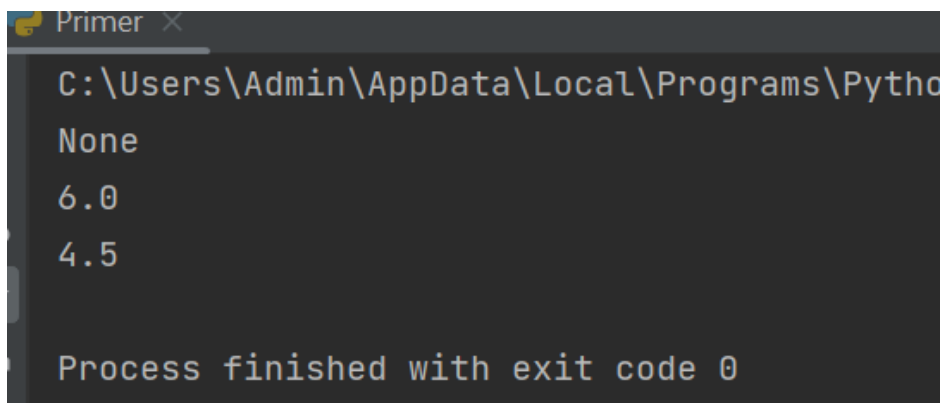


Рисунок 2.2 – Результат работы примера №1

Задание №1. Решить поставленную задачу: написать функцию, вычисляющую среднее геометрическое своих аргументов a_1, a_2, \dots, a_n . Если функции передается пустой список аргументов, то она должна возвращать значение `None`.

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

Рисунок 3.1 – Формула для задания №1

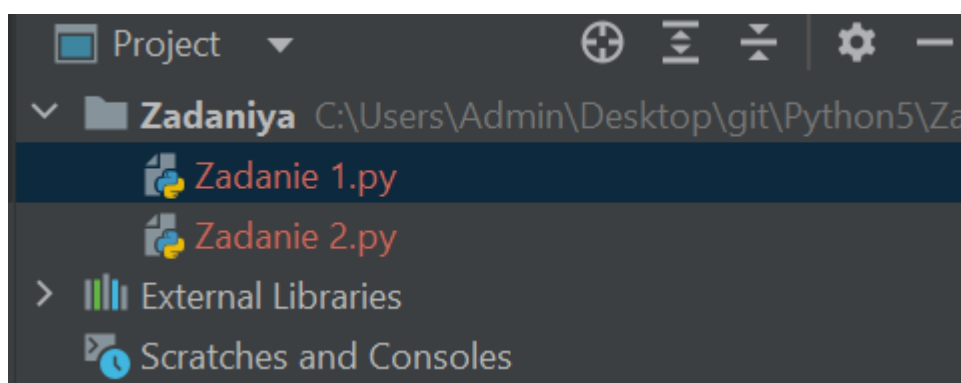


Рисунок 3.2 – Созданные проекты

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Написать функция, вычисляющую среднее геометрическое своих аргументов a1,a2,...an.
Если функции передается пустой список аргументов,то она должна возвращать значение None
"""

def geo(*args):
    if args:
        values = [float(arg) for arg in args]
        values.sort()
        a = 1
        n = len(values)

if __name__ == "__main__":
    # Zadanie 1
    C:\Users\Admin\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/Admin/Desktop/git/
4.282254736676649

Process finished with exit code 0
```

Рисунок 3.3 – Результат работы программы

Задание №2. Решить поставленную задачу: написать функцию, вычисляющую среднее гармоническое своих аргументов $a_1, a_2 \dots a_n$. Если функции передается пустой список аргументов, то она должна возвращать значение None.

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

Рисунок 3.4 – Формула для задания №2

```
n = len(values)
a = 0
for i in values:
    a = a + (1 / i)
return n / a

else:
    return None

if __name__ == "__main__":
    print(garmonic(1, 6, 2, 2, 21))

if __name__ == "__main__":
    print(garmonic(1, 6, 2, 2, 21))

Zadanie 2 x
C:\Users\Admin\AppData\Local\Programs\Python\Python38-64\python.exe
2.258064516129032

Process finished with exit code 0
```

Рисунок 3.5 – Результат работы программы

№3. Самостоятельно подберите или придумайте задачу с переменным числом именованных аргументов. Приведите решение этой задачи.

Дано время забега на дистанцию 100 метров студентами в виде ключевых значений. Определить лучшее время и среднее время среди всех студентов.

```
        min = keywords[kw]
    print("Лучшее время:", min)
    print("Среднее время среди всех студентов - ", summa / n)

if __name__ == "__main__":
    beg(Вася=10.2,
        Виталик=11.3,
        Адам=10.1,
        Вова=9.98,
        Дина=12.3,
        )

beg()
```

ZadanieIndividual x

C:\Users\Admin\AppData\Local\Programs\Python\Python39\python.exe

Лучшее время: 9.98

Среднее время среди всех студентов - 10.776

Process finished with exit code 0

Рисунок 3.6 – Результат выполнения программы

Индивидуальное задание. В – 1. Напишите функцию, принимающую произвольное количество аргументов, и возвращающую требуемое значение. Если функции передается пустой список аргументов, то она должна возвращать значение None. В процессе решения не использовать преобразования конструкции `*args` в список или иную структуру данных.

Задание: 1. Произведение аргументов, расположенных между максимальным и минимальным аргументами.

```
def proiz(*args):  
    if args:  
        last_index = len(args) - 1 - args[::-1].index(0)  
        return sum(args[last_index + 1:])  
    else:  
        return None
```

Рисунок 4.1 – Результат выполнения программы

Вывод: в результате выполнения лабораторной работы были приобретены практические навыки и теоретические сведения по работе с функциями с переменным числом параметров при написании программ с помощью языка программирования Python версии 3.x.

Ответы на контрольные вопросы:

1. Какие аргументы называются позиционными в Python?

Аргументы, которые передаются без указания имен называются позиционными, потому что именно по позиции, расположению аргумента, функция понимает, какому параметру он соответствует.

2. Какие аргументы называются именованными в Python?

Аргументы, передаваемые с именами, называются именованными. При вызове функции можно использовать имена параметров из ее определения.

3. Для чего используется оператор *?

Оператор `*` чаще всего ассоциируется у людей с операцией умножения, но в Python он имеет и другой смысл.

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

Вот пример:

```
a = [1, 2, 3]
```

```
b = [*a, 4, 5, 6]
```

```
print(b) # [1, 2, 3, 4, 5, 6]
```

Тут берётся содержимое списка `a`, распаковывается, и помещается в список `b`.

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

Итак, мы знаем о том, что оператор «звёздочка» в Python способен «вытаскивать» из объектов составляющие их элементы. Знаем мы и о том, что существует два вида параметров функций. А именно, `*args` — это сокращение от «arguments» (аргументы), а `**kwargs` — сокращение от «keyword arguments» (именованные аргументы).

Каждая из этих конструкций используется для распаковки аргументов соответствующего типа, позволяя вызывать функции со списком аргументов переменной длины.

Важно помнить, что «args» — это всего лишь набор символов, которым принято обозначать аргументы. Самое главное тут — это оператор `*`. А то, что именно идёт после него, особой роли не играет. Благодаря использованию `*` мы создали список позиционных аргументов на основе того, что было передано функции при вызове.

После того, как мы разобрались с `*args`, с пониманием `**kwargs` проблем быть уже не должно.

Имя, опять же, значения не имеет. Главное — это два символа **. Благодаря им создаётся словарь, в котором содержатся именованные аргументы, переданные функции при её вызове.