

**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: «Функции с переменным числом параметров в Python»

Тема: «Рекурсия в языке Python»

Выполнил: студент 1 курса

группы ИВТ-б-о-21-1

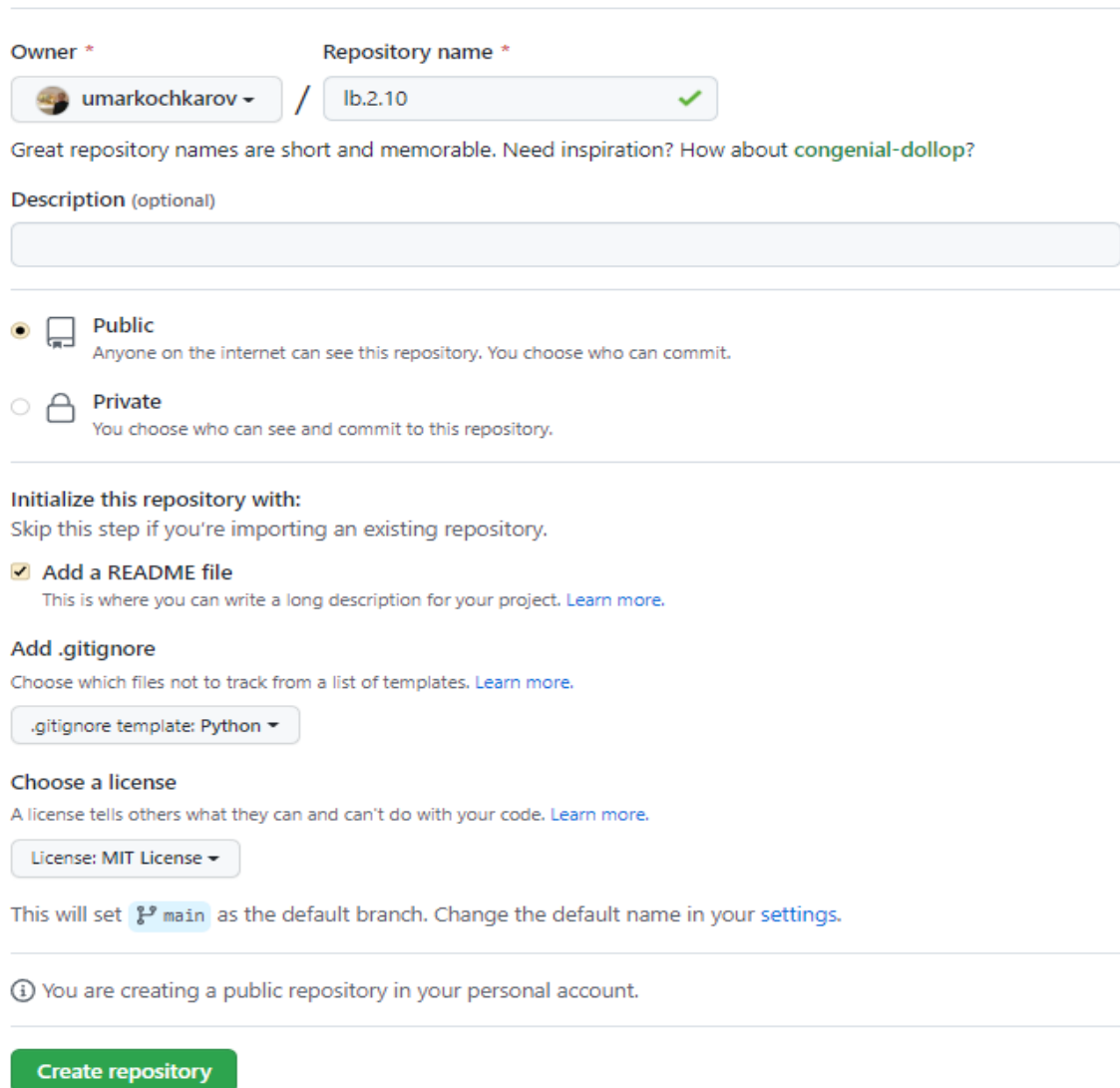
Кочкаров Умар Ахматович

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

1. Создать общедоступный репозиторий с лицензией MIT и языком Python.



Owner * Repository name *

umarkochkarov / lb2.10 ✓

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

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

2. Клонировать репозиторий на ПК:

```
erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/python
$ git clone https://github.com/umarkochkarov/lb2.10.git
Cloning into 'lb2.10'...
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.
```

Рисунок 2. Клонирование

3. Организовать репозиторий в соответствии с моделью ветвления git-flow.

```
erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/python/Ла62.10/1b2.10 (main)
$ 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/erken/Desktop/python/Ла62.10/.git/
hooks]
```

Рисунок 3. Организация репозитория в соответствии с git-flow

4. Решить поставленную задачу: написать функцию, вычисляющую среднее геометрическое своих аргументов

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

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5
6  def Geom(*args):
7      if args:
8          values = [float(arg) for arg in args]
9          values.sort()
10
11         a = 1
12         n = len(values)
13         for i in values:
14             a = a * pow(i, (1/n))
15         return a
16     else:
17         return None
18
19 if __name__ == "__main__":
20     print(Geom())
21     print(Geom(3,4,5))
```

Run - 2.10

Run: 1 x

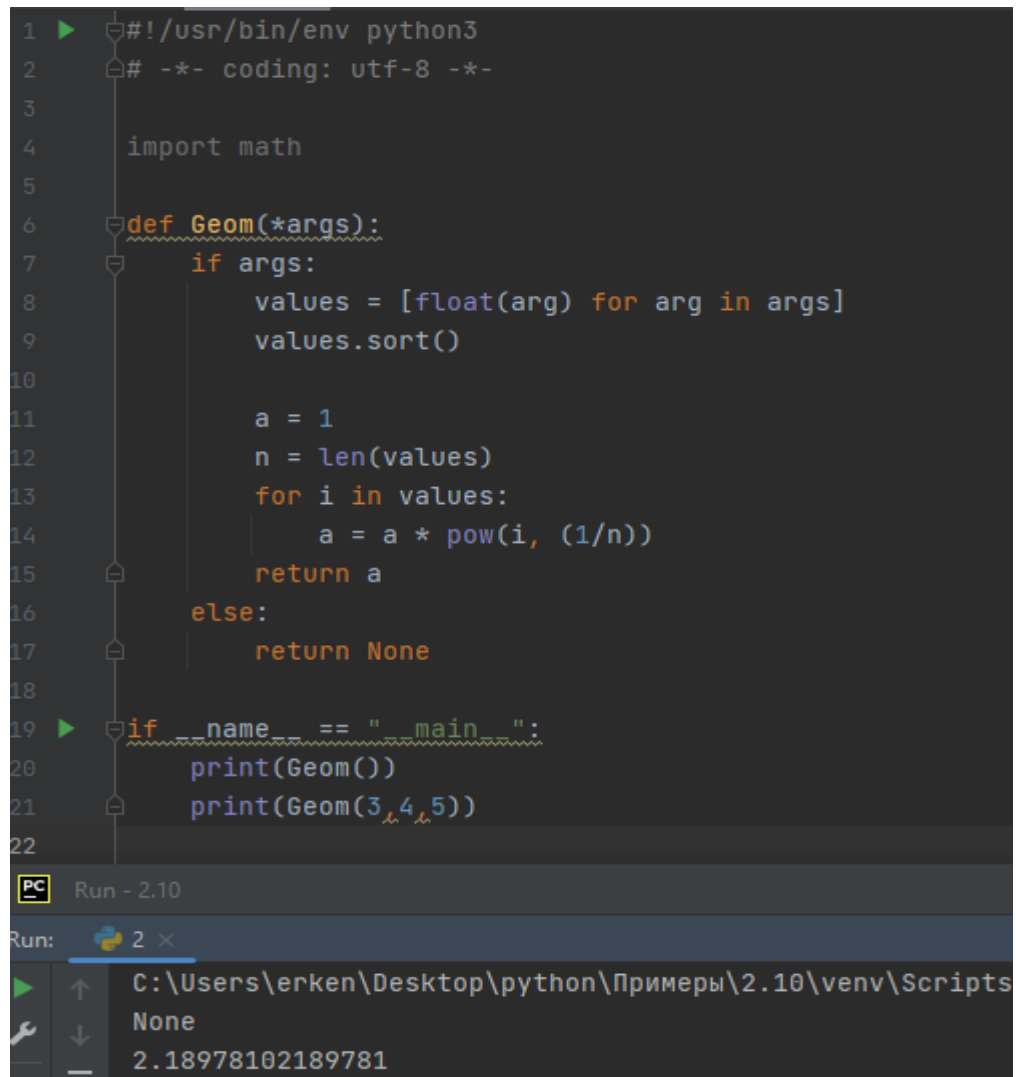
C:\Users\erken\Desktop\python\Примеры\2.10\venv\Scripts\

None

3.9148676411688625

Рисунок 4. Результат выполнения программы

5. Решить поставленную задачу: написать функцию, вычисляющую среднее гармоническое своих аргументов



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5
6  def Geom(*args):
7      if args:
8          values = [float(arg) for arg in args]
9          values.sort()
10
11          a = 1
12          n = len(values)
13          for i in values:
14              a = a * pow(i, (1/n))
15          return a
16      else:
17          return None
18
19  if __name__ == "__main__":
20      print(Geom())
21      print(Geom(3,4,5))
22
```

Run - 2.10

Run: 2 x

C:\Users\erken\Desktop\python\Примеры\2.10\venv\Scripts

None

2.18978102189781

Рисунок 5. Результат выполнения программы

6. Индивидуальное задание(вариант 8)

Сумму аргументов, расположенных между первым и вторым положительными аргументами.

```
1  ▶  #!/usr/bin/env python3
2      #- coding: utf-8 -*-
3
4      import math
5
6      def Summa(*args):
7          first = 0
8          second = 0
9          summa = 0
10         if args:
11             for i in args:
12                 if i > 0:
13                     first = args.index(i)
14                     break
15
16             for i in args:
17                 if i > 0 and args.index(i) > first:
18                     second = args.index(i)
19                     break
20
21             for i in args:
22                 if (args.index(i) > first) and (args.index(i) < second):
23                     summa += i
24             return summa
25
26         else:
27             return None
28
29  ▶  if name == "main ":
30      print(Summa())
31      print(Summa(-2, -4, 2, -6, -1, -9, 1, 4, 9, -5))
```

Run - 2.10

Run: main ×

C:\Users\erken\Desktop\python\Примеры\2.10\venv\Scripts\python.exe C:/Users/e

None

-16

Рисунок 6. Индивидуальное задание

Ответы на вопросы:

1. Какие аргументы называются позиционными в Python?

Это аргументы, передаваемые в вызов в определенной последовательности (на определенных позициях), без указания их имен.

Элементы объектов, поддерживающих итерирование, могут использоваться в качестве позиционных аргументов, если их распаковывать при помощи *.

2. Какие аргументы называются именованными в Python?

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

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.

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.

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