

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра информационных систем и технологий

Отчет по лабораторной работе №13.

Дисциплина: «Основы программной инженерии»

Выполнил:

Студент группы ПИЖ-б-о-22-1,

направление подготовки: 09.03.04

«Программная инженерия»

ФИО: Рядская Мария Александровна

Проверил:

Воронкин Р. А.

Ставрополь 2022

Тема: Лабораторная работа 2.10 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.

Выполнение работы:

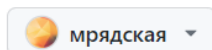
1. Изучила теоретический материал работы.
2. Создала репозиторий на git.hub.

Создайте новый репозиторий

Репозиторий содержит все файлы проекта, включая историю изменений. У вас уже есть репозиторий проекта в другом месте? [Импортируйте репозиторий.](#)

Обязательные поля отмечены звездочкой (*).

Владелец *




Название репозитория *

/ lab13


✓ lab13.

Отличные названия репозитория короткие и запоминающиеся. Нужно вдохновение? Как насчет [solid-meme?](#)

Описание (необязательно)

☒  **Общедоступный**

Этот репозиторий может видеть любой пользователь Интернета. Вы сами выбираете, кто может совершать коммиты.

☐  **Приватный**

Вы сами выбираете, кто может просматривать этот репозиторий и фиксировать его в нем.

Инициализируйте этот репозиторий с помощью:

☒ **Добавьте файл README**

Здесь вы можете написать подробное описание вашего проекта. [Узнайте больше о README.](#)

Добавить файл .gitignore

.шаблон gitignore: Отсутствует ▾

Рисунок 1 – создание репозитория

3. Клонировала репозиторий.

```
C:\git1>git clone https://github.com/mryadskaya/lab13.git
Cloning into 'lab13'...
remote: Enumerating objects: 5, done.
Receiving objects: 100% (5/5), 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

C:\git1>cd lab13

C:\git1\lab13>
```

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

```
C:\git1>cd lab13

C:\git1\lab13>git branch develop

C:\git1\lab13>git push -u origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/mryadskaya/lab13/pull/new/develop
remote:
To https://github.com/mryadskaya/lab13.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

C:\git1\lab13>git checkout develop
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.

C:\git1\lab13>
```

Проработала примеры из методички.

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

if __name__ == "__main__"

Run пример 1 ×

↻ [icon] :

↑ C:\Users\ADMIN\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:\git1\lab9\PyCharm\пример 1.py"

↓ None

⇅ 6.0

⇅ 4.5

Process finished with exit code 0

> |

> qit1 > lab9 > PyCharm > [icon] пример 1.py

Рисунок 5 – пример 1

решить поставленную задачу: написать функцию, вычисляющую среднее геометрическое своих аргументов. Если функции передается пустой список аргументов, то она должна возвращать значение None.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5
6  def sredgeom(*a):
7      if a:
8          n = len(a)
9          y = 1
10         for i in a:
11             y *= i
12
13         s = math.pow(y, 1/n)
14         return s
15
16     else:
17         return None
18
19  if __name__ == "__main__":
20      p = list(int(i) for i in input("Введите значения: ").split())
21      result = sredgeom(*p)
22      print(result)
```

if __name__ == "__main__"

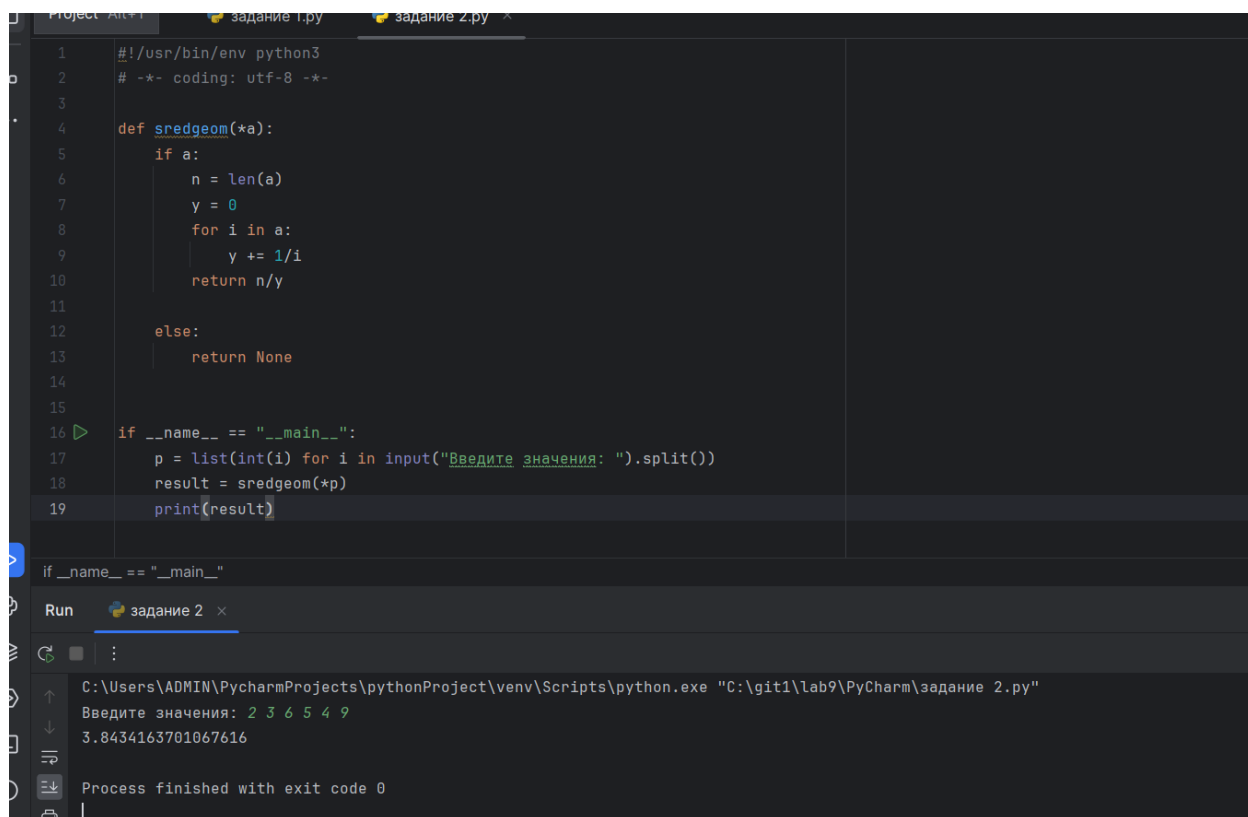
Run задание 1

Введите значения: 2 5 4 3 6 7 8 9 0

0.0

Рисунок 6 – пример выполнения 8 задания

Решить поставленную задачу: написать функцию, вычисляющую среднее гармоническое своих аргументов. Если функции передается пустой список аргументов, то она должна возвращать значение None



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def sredgeom(*a):
5      if a:
6          n = len(a)
7          y = 0
8          for i in a:
9              y += 1/i
10             return n/y
11
12     else:
13         return None
14
15
16  if __name__ == "__main__":
17      p = list(int(i) for i in input("Введите значения: ").split())
18      result = sredgeom(*p)
19      print(result)
```

Run задание 2

C:\Users\ADMIN\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:\git1\lab9\PyCharm\задание 2.py"

Введите значения: 2 3 6 5 4 9

3.8434163701067616

Process finished with exit code 0

Рисунок 7 – результат выполнения задания 9

Напишите функцию, принимающую произвольное количество аргументов, и возвращающую требуемое значение. Если функции передается пустой список аргументов, то она должна возвращать значение None . Номер варианта определяется по согласованию с преподавателем. В процессе решения не использовать преобразования конструкции *args в список или иную структуру данных

15. Сумму модулей аргументов, расположенных после минимального по модулю аргумента.

```
1  ✓ #!/usr/bin/env python3
2    # -*- coding: utf-8 -*-
3
4
5  ✓ def sum_of_abs_after_min(*args):
6      if len(args) == 0:
7          return None
8      min_arg = min(args, key=abs)
9      min_arg_index = args.index(min_arg)
10     sum_of_abs = 0
11     for i, arg in enumerate(args[min_arg_index + 1:]):
12         sum_of_abs += abs(arg)
13     return sum_of_abs
14
15  ▶ ✓ if __name__ == "__main__":
16      p = list(int(i) for i in input("Введите значения: ").split())
17      result = sum_of_abs_after_min(*p)
18      print(result)
19
20
21  sum_of_abs_after_min()
22
23  up  индивидуальное ×
24
25  C:\Users\ADMIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\git1\lab9\PyCharm\индивиду
26  Введите значения: 1 2 3 4 5 6 7 8 9
27  44
28
29  Process finished with exit code 0
```

Рисунок 8 – результат выполнения индивидуального задания

Зафиксировала все изменения в github в ветке develop и сливание ветки develop в ветку main

```

C:\git1\lab13>git add .
C:\git1\lab13>git status
On branch develop
Your branch is up to date with 'origin/develop'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   "PyCharm/\320\267\320\260\320\264\320\260\320\275\320\270\320\265 1.py"
        new file:   "PyCharm/\320\267\320\260\320\264\320\260\320\275\320\270\320\265 2.py"
        new file:   "PyCharm/\320\270\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\320\276\320\265.py"
        new file:   "PyCharm/\320\277\321\200\320\270\320\274\320\265\321\200 1.py"

C:\git1\lab13>git commit -m "схранение "
[develop 22c8ad2] схранение
 4 files changed, 83 insertions(+)
 create mode 100644 "PyCharm/\320\267\320\260\320\264\320\260\320\275\320\270\320\265 1.py"
 create mode 100644 "PyCharm/\320\267\320\260\320\264\320\260\320\275\320\270\320\265 2.py"
 create mode 100644 "PyCharm/\320\270\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\320\276\320\265.py"
 create mode 100644 "PyCharm/\320\277\321\200\320\270\320\274\320\265\321\200 1.py"

C:\git1\lab13>git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 2 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1.36 KiB | 198.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/mryadskaya/lab13.git
   fa19902..22c8ad2  develop -> develop

C:\git1\lab13>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\git1\lab13>

```

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.