

**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: Джараян Арег Александрович

Проверил:

Воронкин Р. А.

Ставрополь 2022

Тема: Лабораторная работа 2.10 Функции с переменным число 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. Изучил теоретический материал работы.
2. Создал репозиторий на git.hub.

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 * / Repository name *


 aregdz / lab13

✔ lab13 is available.

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

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

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

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

3. Клонировал репозиторий.

```
aregd@DESKTOP-5KV9QA9 MINGW64 ~
$ cd "D:\Рабочий стол\git 13"

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/git 13
$ git clone https://github.com/aregdz/lab13.git
Cloning into 'lab13'...
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.

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/git 13
$
```

4.

Рисунок 2 – клонирование репозитория 4.

Дополнить файл gitignore необходимыми правилами.

```
.gitignore – Блокнот
Файл Правка Формат Вид Справка
# Created by .ignore support plugin (hsz.mobi)
### Python template
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
env/
build/
develop-eggs/
dist/
downloads/
```

Рисунок 3 – .gitignore для IDE PyCharm

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

```
aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/git 13/lab13 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/git 13/lab13 (develop)
$ |
```

Рисунок 4 – создание ветки develop

6. Проработал примеры из методички.

```

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

```

Рисунок 5 – пример 1

```

C:\Users
None
6.0
4.5

```

Рисунок 6 – пример выполнения примера 1

7. 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  usage
7  def sredgeom(*a):
8      if a:
9          n = len(a)
10         y = 1
11         for i in a:
12             y *= i
13
14         s = math.pow(y, 1/n)
15         return s
16
17     else:
18         return None
19
20 if __name__ == "__main__":
21     p = list(int(i) for i in input("Введите значения: ").split())
22     result = sredgeom(*p)
23     print(result)

```

Рисунок 7 – задание 8

```

Введите значения: 1 2 3 4 5 6 7 8 9 10
4.528728688116765

```

Рисунок 8 – пример выполнения 8 задания

8. 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)
```

Рисунок 9 – выполнение задания 9

```
C:\Users\aregd\AppData\Local\Programs\
Введите значения: 1 2 3 4 5 6
2.4489795918367347
```

Рисунок 10 – результат выполнения задания 9

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.

5. Сумму аргументов, расположенных до последнего положительного аргумента.

```

1  ✓ #!/usr/bin/env python3
2    # -*- coding: utf-8 -*-
3
4  1 usage
5  ✓ def sum_before_last_positive(*args):
6      if args:
7          l = 0
8          for i, u in enumerate(args):
9              if u > 0:
10                 l = i
11
12             return sum(args[:l])
13
14         else:
15             return None
16
17 if __name__ == '__main__':
18     p = list(int(i) for i in input("Введите значения: ").split())
19     result = sum_before_last_positive(*p)
20     print(result)

```

Рисунок 11 – выполнение индивидуального задания

```

Введите значения: 1 2 3 4 5 -8 3 -6
7

```

Рисунок 12 – результат выполнения индивидуального задания

9. Зафиксировал все изменения в github в ветке develop.

```

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/git 13/lab13 (develop)
$ git add .

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/git 13/lab13 (develop)
$ git commit -m"Сохранение"
[develop b15061a] Сохранение
4 files changed, 81 insertions(+)
create mode 100644 PyCharm/8zadanie.py
create mode 100644 PyCharm/9zadanie.py
create mode 100644 PyCharm/individual.py
create mode 100644 PyCharm/primer 1.py

```

Рисунок 13 – фиксация изменений в ветку develop

10. Слил ветки.

```

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/git 13/lab13 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/git 13/lab13 (main)
$ git merge develop
Updating e7d578b..b15061a
Fast-forward
 PyCharm/8zadanie.py | 22 ++++++
 PyCharm/9zadanie.py | 19 ++++++
 PyCharm/individual.py | 19 ++++++
 PyCharm/primer 1.py | 21 ++++++
 4 files changed, 81 insertions(+)
 create mode 100644 PyCharm/8zadanie.py
 create mode 100644 PyCharm/9zadanie.py
 create mode 100644 PyCharm/individual.py
 create mode 100644 PyCharm/primer 1.py

```

Рисунок 14 – сливание ветки develop в ветку main

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.