

# Лабораторная работа №5

Выполнил Эсеналиев Арсен

ИВТ-б-о-21-1

**Цель:** приобретение навыков по работе с рекурсивными функциями при написании программ с помощью языка программирования Python версии 3.x.

## 1. Создал общедоступный репозиторий на GitHub с MIT

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

---

**Owner \*** **Repository name \***

Arsen445 / LB2.9

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

**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:** None ▾

**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**

## 2. Выполнил клонирование созданного репозитория.

```

C:\Users\GG_Force>d:
D:\>cd REP4
D:\REP4>git clone https://github.com/Arsen445/LB2.9.git
Cloning into 'LB2.9'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
D:\REP4>

```

3. Организовал свой репозиторий в соответствии с моделью ветвления git-flow. (Перешел с главной main на develop)

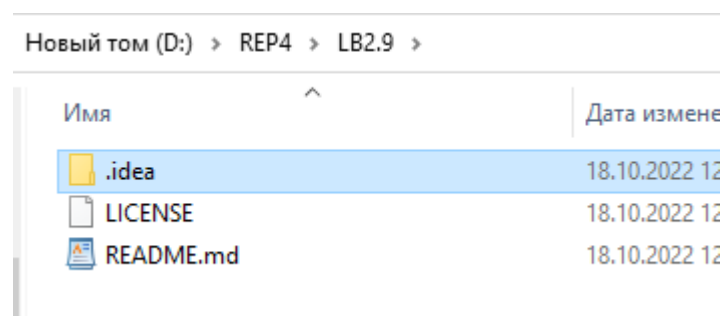
```

D:\REP4\LB2.9>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? [D:/REP4/LB2.9/.git/hooks]
D:\REP4\LB2.9>_

```

4. Создал проект PyCharm в папке репозитория.



5. Дополнил файл .gitignore необходимыми правилами для работы с IDE PyCharm.

```

# 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
# 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/

```

## 6. Проработал примеры лабораторной работы.

The screenshot shows a Python IDE window titled 'prim1.py - D:\REP5\2.10\prim1.py (3.9.13)'. The code defines a function `print_scores` that takes a student name and a list of scores, and prints the student name followed by each score on a new line. The `__main__` block calls this function with 'DK' and the list `[100, 123, 35, 46, 46]`. The IDLE Shell window on the right shows the output: 'student Name: DK' followed by the scores 100, 123, 35, 46, and 46 on separate lines.

```

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

def print_scores(student, *scores):
    print(f"student Name: {student}")
    for score in scores:
        print(score)

if __name__ == '__main__':
    i = [100, 123, 35, 46, 46]
    print_scores("DK", *i)

```

```

Python 3.9.13 (tags/v3.9.13:6de5448, Jan 10 2022) on win32
Type "help", "copyright", "credits()" or "quit()" for more
>>>
student Name: DK
100
123
35
46
46
>>>

```

The screenshot shows a Python IDE window titled 'prim2.py - D:\REP5\2.10\prim2.py (3.9.13)'. The code defines a function `print_pet_names` that takes an owner name and a dictionary of pets, and prints the owner name followed by each pet's name on a new line. The `__main__` block calls this function with 'Jonathan' and a dictionary containing 'Brock' (dog), 'Larry', 'Curly', and 'Moe' (fish), and 'Shelddon' (turtle). The IDLE Shell window on the right shows the output: 'Owner Name: Jonathan' followed by 'dog: Brock', 'fish: ['Larry', 'Curly', 'Moe']', and 'turtle: Shelddon'.

```

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

def print_pet_names(owner, **pets):
    print(f"Owner Name: {owner}")
    for pet, name in pets.items():
        print(f"{pet}: {name}")

if __name__ == '__main__':
    print_pet_names(
        "Jonathan",
        dog="Brock", fish=["Larry", "Curly", "Moe"],
        turtle="Shelddon"
    )

```

```

Python 3.9.13 (tags/v3.9.13:6de5448, Jan 10 2022) on win32
Type "help", "copyright", "credits()" or "quit()" for more
>>>
Owner Name: Jonathan
dog: Brock
fish: ['Larry', 'Curly', 'Moe']
turtle: Shelddon
>>>

```

```

prim3.py - D:\REP5\2.10\prim3.py (3.9.13)
File Edit Format Run Options Window Help

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

def median(*args):
    if args:
        values = [float(arg) for arg in args]
        values.sort()

        n = len(values)
        idx = n // 2
        if n % 2:
            return values[idx]
        else:
            return (values[idx - 1] + values[idx]) / 2
    else:
        return None

if __name__ == "__main__":
    print(median())
    print(median(3, 7, 1, 6, 9))
    print(median(1, 5, 8, 4, 3, 9))

```

7.

8. Решить поставленную задачу: написать функцию, вычисляющую среднее геометрическое своих аргументов  $a_1, a_2, \dots, a_n$

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

Если функции передается пустой список аргументов, то она должна возвращать значение `None`.

```

5.637124348539527
None
>>>

*zadanie1.py - D:\REP5\2.10\zadanie1.py (3.9.13)*
File Edit Format Run Options Window Help

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

def sred_geom(*args):
    if args:
        a = 1.0
        k = 0
        for arg in args:
            a = a * arg
            k = k + 1
        n = a ** (1 / k)
        return n
    else:
        return None

if __name__ == "__main__":
    print(sred_geom(1, 5, 8.9, 4, 3, 9.5, 67, 3))
    print(sred_geom())

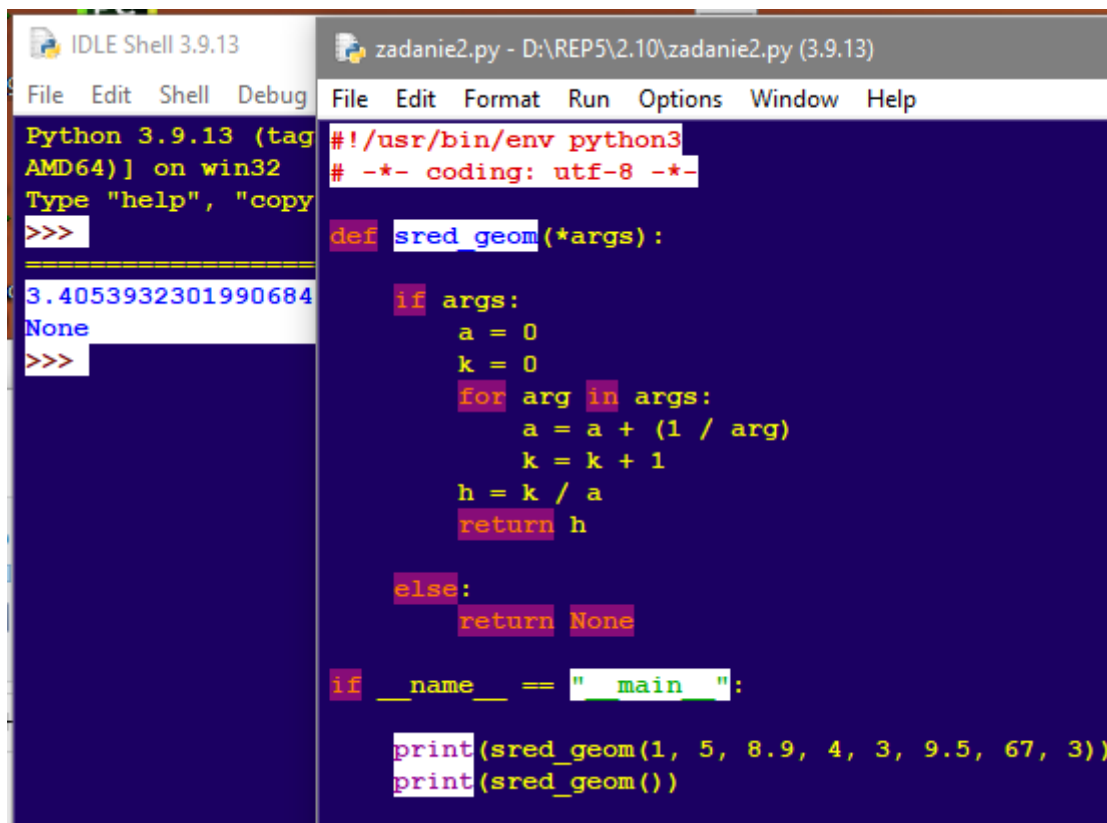
```

8.

9. Решить поставленную задачу: написать функцию, вычисляющую среднее гармоническое своих аргументов  $a_1, a_2, \dots, a_n$

$$\frac{n}{H} = \sum_{k=1}^n \frac{1}{a_k}. \quad (2)$$

Если функции передается пустой список аргументов, то она должна возвращать значение `None`.



The screenshot shows a Python IDE with two windows. The left window, titled 'IDLE Shell 3.9.13', shows the execution of a script. It displays the prompt 'Python 3.9.13 (tag AMD64) on win32', followed by 'Type "help", "copy" >>>', then the output '3.4053932301990684', then 'None', and finally another prompt '>>>'. The right window, titled 'zadanie2.py - D:\REP5\2.10\zadanie2.py (3.9.13)', shows the source code of the script. The code defines a function 'sred\_geom(\*args):' which calculates the harmonic mean. It includes a docstring, a loop to calculate the sum of reciprocals, and a return statement. It also has a main block that prints the result of 'sred\_geom(1, 5, 8.9, 4, 3, 9.5, 67, 3)' and 'sred\_geom()'.

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

def sred_geom(*args):
    """
    Функция вычисления среднего гармонического
    """
    if args:
        a = 0
        k = 0
        for arg in args:
            a = a + (1 / arg)
            k = k + 1
        h = k / a
        return h
    else:
        return None

if __name__ == "__main__":
    print(sred_geom(1, 5, 8.9, 4, 3, 9.5, 67, 3))
    print(sred_geom())
```

## 9. Индивидуальное задание

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.

Сумму модулей аргументов, расположенных после первого аргумента, равного нулю.

```

File Edit Shell Python 3.9 AMD64] on win32
Type "help" for more
>>>
100.4
None
>>>

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

def sred_geom(*args):
    if args:
        a = 0
        for arg in args:
            a = a + abs(arg)
        return a
    else:
        return None

if __name__ == "__main__":
    print(sred_geom(0, -5, 8.9, 4, 3, -9.5, 67, 3))
    print(sred_geom())

```

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

Вывести 2 списка. Список жертв и нападающих.

```

IDLE Shell 3.9.13
Python 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 16:36:56) on win32
Type "help", "copyright", "credits" or "license()" for more
>>>
Restart: D:\REP5\2.10\ind2.py
Traceback (most recent call last):
  File "D:\REP5\2.10\ind2.py", line 11, in <module>
    n_run(виды_ночных_поедателей_мышей=["сова", "филин", "owl"])
  File "D:\REP5\2.10\ind2.py", line 5, in n_run
    for bird,name in birds.items():
NameError: name 'birds' is not defined
>>>
Restart: D:\REP5\2.10\ind2.py
виды_ночных_поедателей_мышей: ['сова', 'филин', 'owl']
виды_жертв: ['заяц', 'крыса', 'мышь']
>>>

ind2.py - D:\REP5\2.10\ind2.py (3.9.13)
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

def n_run(**birds):
    for bird,name in birds.items():
        print(f"{bird}: {name}")

if __name__ == "__main__":
    n_run(виды_ночных_поедателей_мышей=["сова", "филин", "owl"],
          виды_жертв = ["заяц", "крыса", "мышь"])

```

11. Зафиксируйте сделанные изменения в репозитории.

```

"\320\233\320\2214 .docx"
nothing added to commit but untracked files present
D:\REP4\LB2.9>git add .
D:\REP4\LB2.9>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   Zadanie2.py
    new file:   ind.py
    new file:   prim1.py
    new file:   zadanie1.py
    new file:   "~$\320\233\320\2214 .docx"
    new file:   "\320\233\320\2214 .docx"
D:\REP4\LB2.9>git commit -m "last"2

D:\REP5\2.10>git commit -m "f"
[main a8e54b0] f
16 files changed, 301 insertions(+), 3 deletions(-)
create mode 100644 .idea/.name
create mode 100644 .idea/LB2.8.iml
create mode 100644 .idea/inspectionProfiles/profiles_settings
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 "doc/~$\320\233\320\2215 .docx"
create mode 100644 "doc/\320\233\320\2215 .docx"
create mode 100644 "doc/\320\233\320\260\320\261\320\276\321\
7 \321\200\320\260\320\261\320\276\321\202\320\260 2.10 (5).pd
create mode 100644 ind.py
create mode 100644 ind2.py
create mode 100644 prim1.py
create mode 100644 prim2.py
create mode 100644 prim3.py
create mode 100644 zadanie1.py
create mode 100644 zadanie2.py
D:\REP5\2.10>git push --all
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 8 threads
Compressing objects: 100% (19/19), done.
Writing objects: 100% (21/21), 519.32 KiB | 19.97 MiB/s, done.
Total 21 (delta 2), reused 0 (delta 0), pack-reused 0

```

12. Выполните слияние ветки для разработки с веткой main/master.

```

D:\REP5\2.10>git merge develop
merge: develop - not something
D:\REP5\2.10>git merge main
Already up to date.
D:\REP5\2.10>

```

### 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 \*?**

Функция также может принимать переменное количество позиционных аргументов, тогда перед именем ставится \*.

### **4) Каково назначение конструкций \*args и \*\*kwargs?**

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