

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

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

**Кафедра инфокоммуникаций
«Функции с переменным числом параметров в Python»**

**Отчет по лабораторной работе № 2.10
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-21-1

Халимендик Я. Д. « » 2022г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил 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:


1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия IT и язык программирования Python.

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 *

 Yana-Kh ▾


/ Lab-13-OPJ ✓

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

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

3. Выполните клонирование созданного репозитория.

```
C:\Users\ynakh\OneDrive\Рабочий стол\Git>git clone https://github.com/Yana-Kh/Lab-13-OPJ.git
Cloning into 'Lab-13-OPJ'...
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 репозитория

4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.

```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-13-OPJ>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   .gitignore

C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-13-OPJ>_
```

Рисунок 3 – Дополнение файла .gitignore

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

```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-13-OPJ>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/ynakh/OneDrive/Рабочий стол/Git/Lab-13-OPJ/.git/hooks]

C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-13-OPJ>_
```

Рисунок 4 – Организация репозитория в соответствии с моделью git-flow

6. Создайте проект PyCharm в папке репозитория.

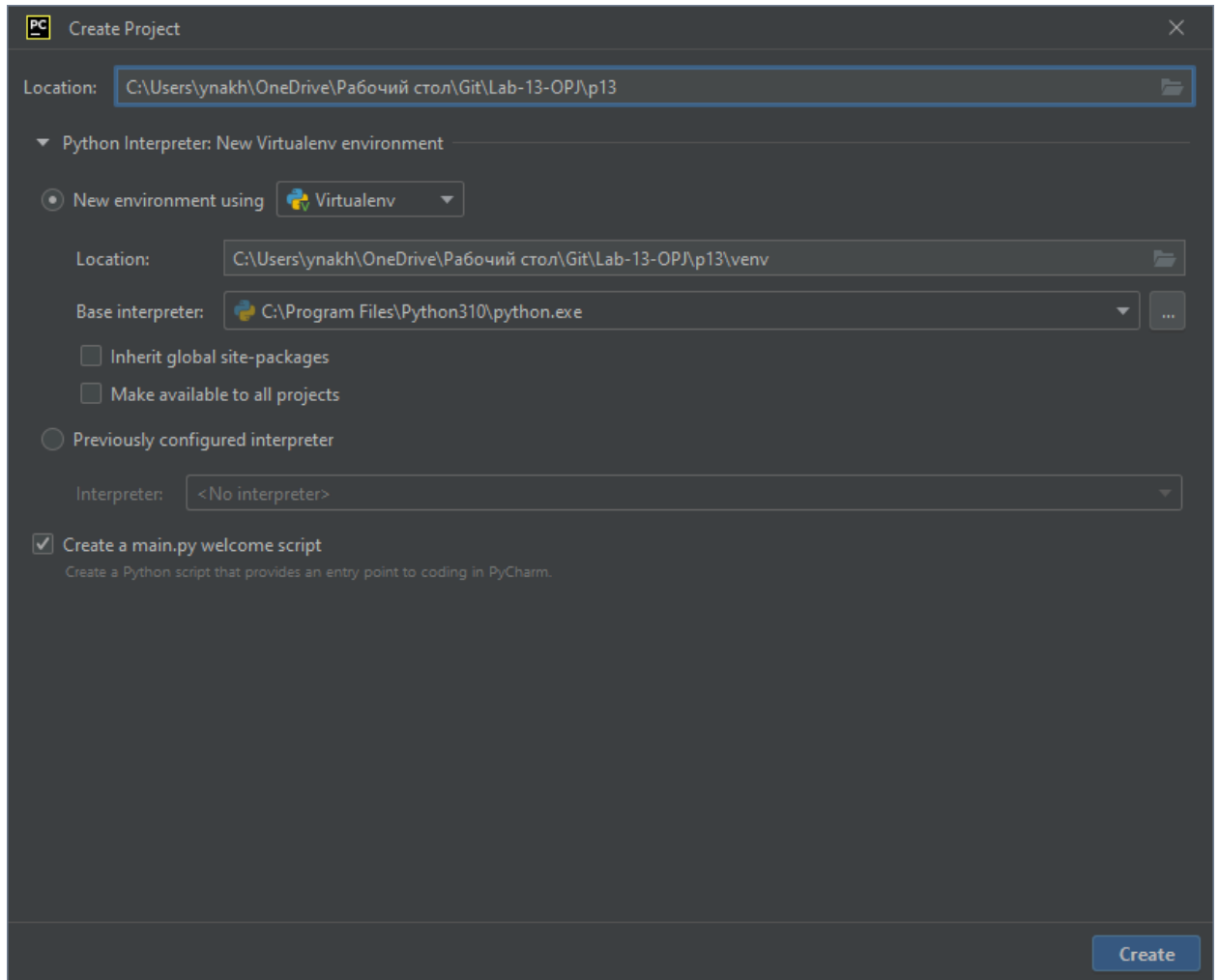


Рисунок 5 – Создание проекта PyCharm в папке репозитория

7. Проработайте пример лабораторной работы.

Пример 1. Разработать функцию для определения медианы значений аргументов функции. Если функции передается пустой список аргументов, то она должна возвращать значение None.

Код:

```
#!/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))

```

```

"C:\Users\ynakh\OneDrive\Рабочий стол\
None
6.0
4.5

Process finished with exit code 0

```

Рисунок 6 – Результат работы программы

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

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

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

Код:

```

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

def sr_geom(*args):
    if args:
        values = [float(arg) for arg in args]
        g = 1
        for arg in values:
            g = g * arg
        n = len(args)
        return pow(g, 1/n)
    else:
        return None

if __name__ == "__main__":
    print(f'Среднее геометрическое 1: {sr_geom(1, 2, 3, 4, 5)}')
    print(f'Среднее геометрическое 1: {sr_geom()}')
    print(f'Среднее геометрическое 1: {sr_geom(1.3, 7.9, 8.3)}')

```

```
Среднее геометрическое 1: 2.605171084697352
Среднее геометрическое 1: None
Среднее геометрическое 1: 4.400981186140995
```

Рисунок 6 – Результат работы программы

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

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

Код:

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

def sr_harm(*args):
    if args:
        values = [float(arg) for arg in args]
        h = 0
        for arg in values:
            h += 1 / arg
        n = len(args)
        return n / h
    else:
        return None

if __name__ == "__main__":
    print(f'Среднее гармоническое: {sr_harm(1, 2, 3, 4, 5, 6)}')
    print(f'Среднее гармоническое: {sr_harm()}')
    print(f'Среднее гармоническое: {sr_harm(3.3, 6.5, 9.7, 8.0)}')
```

```
Среднее гармоническое: 2.4489795918367347
Среднее гармоническое: None
Среднее гармоническое: 5.83967828653373
```

Рисунок 7 – Результат работы программы

10. Зафиксируйте сделанные изменения в репозитории.




 ex1.py	ex
 ex2.py	ex
 ex3.py	ex

Рисунок 8 – Фиксирование изменений в репозитории

11. Решите индивидуальное задание согласно своему варианту.

Вариант 32(14). Произведение аргументов, расположенных после максимального по модулю аргумента.

Код:

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

import math

def mult_after_max(*args):
    """
    Произведение аргументов, расположенных после
    максимального по модулю аргумента.
    """
    if args:
        mult = 1
        values = [float(arg) for arg in args]
        max_item = 0
        max_ind = 0
        for i, item in enumerate(values):
            if math.fabs(item) > max_item:
                max_ind = i
                max_item = math.fabs(item)
        values = values[max_ind:]
        for arg in values:
            mult = mult * arg
        return mult
    else:
        return None

if __name__ == "__main__":
    print(f'Произведение: {mult_after_max()}')
    print(f'Произведение: {mult_after_max(1, 3, 6, 4, 5)}')
    print(f'Произведение: {mult_after_max(7, 8, 12, 76, 34, 7, 34)}')
```

```
Произведение: None
Произведение: 120.0
Произведение: 614992.0
```

Рисунок 9 – Результат работы программы

12. Зафиксируйте сделанные изменения в репозитории.





 ex1.py	ex
 ex2.py	ex
 ex3.py	ex
 id.py	id

Рисунок 10 – Фиксирование изменений в репозитории

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

Код:

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

def print_anime(**kwargs):
    """
    Названия аниме по именам героям
    """
    print("На ошибках учатся, после ошибок лечатся")
    if kwargs:
        for name, anime in kwargs.items():
            print(f"{anime}: {name}")
    else:
        return None

if __name__ == "__main__":
    print_anime(
        Kakashi="Naruto", Sakura="Naruto",
        Narumi="Otaku ni Koi wa Muzukashii",
        Violet="Violet Evergarden",
        Ban="Nanatsu no Taizai: The Seven Deadly Sins",
        Akutagava="Bungo sutorei doggusu"
    )
```

```
На ошибках учатся, после ошибок лечатся
Naruto: Kakashi
Naruto: Sakura
Otaku ni Koi wa Muzukashii: Narumi
Violet Evergarden: Violet
Nanatsu no Taizai: The Seven Deadly Sins: Ban
Bungo sutorei doggusu: Akutagava

Process finished with exit code 0
```

Рисунок 11 – Результат работы программы

12. Зафиксируйте сделанные изменения в репозитории.






 ex1.py	ex
 ex2.py	ex
 ex3.py	ex
 id.py	id
 id_kwargs.py	id_kwargs

Рисунок 12 – Фиксирование изменений в репозитории

Вопросы для защиты работы

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 `*`?

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

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

`*args` используется для передачи произвольного числа именованных аргументов функции.

`**kwargs` позволяет передавать произвольное число именованных аргументов в функцию.