

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

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

Выполнил студент группы ПИЖ-б-о-22-1

Душин Александр Владимирович.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_

(подпись)

Ставрополь 2023

Тема: 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.

Ход выполнения работы:


1. Создать общедоступный репозиторий на GitHub с использованием лицензии MIT 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.](#)

Required fields are marked with an asterisk (\*).

Owner \*

 MrPlatinum ▾

Repository name \*

ProgrammEngineering13

✔ ProgrammEngineering13 is available.

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

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

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License ▾

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



You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание общедоступного репозитория на GitHub с заданными настройками

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents
$ git clone https://github.com/MrPlatynum/ProgrammEngineering13.git
Cloning into 'ProgrammEngineering13'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Рисунок 2 – Клонирование созданного репозитория на локальный компьютер

```
env.bak/
venv.bak/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

# mkdocs documentation
/site

# mypy
.mypy_cache/
.dmypy.json
dmypy.json

# Pyre type checker
.pyre/

# pytype static type analyzer
.pytype/

# Cython debug symbols
cython_debug/

# PyCharm
# JetBrains specific template is maintained in a separate JetBrains.gitignore that can
# be found at https://github.com/github/gitignore/blob/main/Global/JetBrains.gitignore
# and can be added to the global gitignore or merged into this file. For a more nuclear
# option (not recommended) you can uncomment the following to ignore the entire idea folder.
.idea/
```

Рисунок 3 – файл .gitignore

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering13 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering13 (develop)
$ |
```

Рисунок 4 – организация репозитория в соответствии с моделью ветвления git flow

2. Проработать примеры лабораторной работы, оформляя код согласно PEP-8:

```

example1.py x
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(3, 7, 1, 6, 9))
21      print(median(1, 5, 8, 4, 3, 9))
22

```

Рисунок 5 – Пример 1

```

"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ/3 семестр/Основы программной инженерии/ЛР13_ДушинаВ/example1.py"
None
6.0
4.5

```

Рисунок 6 – Вывод программы

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

```
example1.py x
1  ▶  #!/usr/bin/env python3
2     # -*- coding: utf-8 -*-
3
4     def geometric_mean(*args):
5         if not args:
6             return None
7
8         product = 1
9         for arg in args:
10             product *= arg
11
12         return product ** (1 / len(args))
13
14
15  ▶  if __name__ == "__main__":
16      values = list(int(i) for i in input("Введите числа: ").split())
17
18      result = geometric_mean(*values)
19      if result is None:
20          print("Список аргументов пуст, среднее геометрическое не может быть вычислено.")
21      else:
22          print(f"Среднее геометрическое: {result}")
23
```

Рисунок 7 – Задание №1

```
"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/D
Введите числа: 1 2 3 4 5 6 7 8 9
Среднее геометрическое: 4.147166274396913
```

Рисунок 8 – Вывод программы

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

```

task2.py x
1 def harmonic_mean(*args):
2     if not args:
3         return None
4
5     reciprocal_sum = 0
6     for arg in args:
7         reciprocal_sum += 1 / arg
8
9     return len(args) / reciprocal_sum
10
11
12 if __name__ == "__main__":
13     values = list(int(i) for i in input("Введите числа: ").split())
14
15     result = harmonic_mean(*values)
16     if result is None:
17         print("Список аргументов пуст, среднее гармоническое не может быть вычислено.")
18     else:
19         print(f"Среднее гармоническое: {result}")
20

```

Рисунок 9 – Задание №2

```

"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/
Введите числа: 1 2 3 4 5 6 7 8 9
Среднее гармоническое: 3.181371861411138

```

Рисунок 10 – Вывод программы

5. Выполним индивидуальные задания:

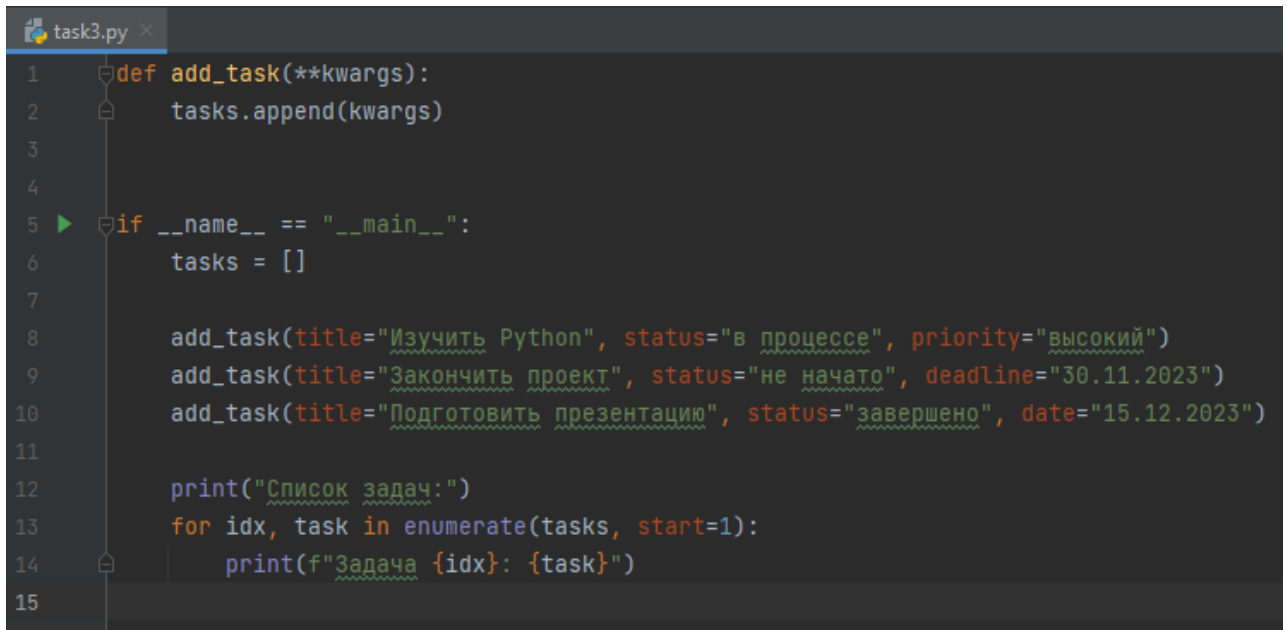
```
individual1.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5  def product_between_zeros(*args):
6      zero_count = 0
7      result = 1
8
9      for arg in args:
10         if arg == 0:
11             zero_count += 1
12             if zero_count == 2:
13                 return result
14             elif zero_count == 1:
15                 result *= arg
16
17     return None
18
19
20  ▶  if __name__ == "__main__":
21      values = list(int(i) for i in input("Введите числа: ").split())
22
23      result = product_between_zeros(*values)
24      if result is None:
25          print("Нельзя вычислить произведение аргументов между первым и вторым нулями.")
26      else:
27          print(f"Произведение аргументов между первым и вторым нулями: {result}")
28
```

Рисунок 11 – Решение индивидуального задания

```
"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/
Введите числа: 1 2 0 3 4 5 6 7 8 9 0
Произведение аргументов между первым и вторым нулями: 181440
```

Рисунок 12 – Вывод программы

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

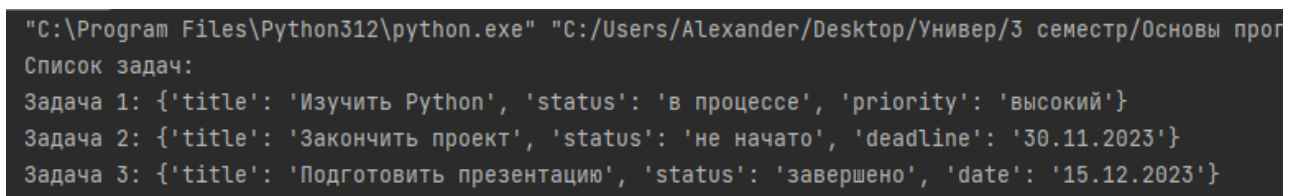


```

1  def add_task(**kwargs):
2      tasks.append(kwargs)
3
4
5  if __name__ == "__main__":
6      tasks = []
7
8      add_task(title="Изучить Python", status="в процессе", priority="высокий")
9      add_task(title="Закончить проект", status="не начато", deadline="30.11.2023")
10     add_task(title="Подготовить презентацию", status="завершено", date="15.12.2023")
11
12     print("Список задач:")
13     for idx, task in enumerate(tasks, start=1):
14         print(f"Задача {idx}: {task}")
15

```

Рисунок 13 – Задача с переменным числом именованных аргументов



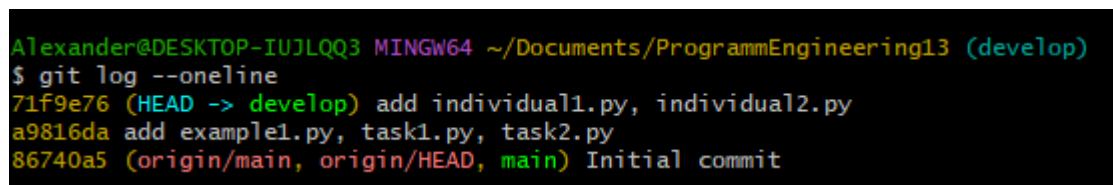
```

"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ/3 семестр/Основы прог
Список задач:
Задача 1: {'title': 'Изучить Python', 'status': 'в процессе', 'priority': 'высокий'}
Задача 2: {'title': 'Закончить проект', 'status': 'не начато', 'deadline': '30.11.2023'}
Задача 3: {'title': 'Подготовить презентацию', 'status': 'завершено', 'date': '15.12.2023'}

```

Рисунок 14 – Вывод программы

7. Зафиксируем проделанные изменения, сольем ветки и отправим на удаленный репозиторий:



```

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering13 (develop)
$ git log --oneline
71f9e76 (HEAD -> develop) add individual1.py, individual2.py
a9816da add example1.py, task1.py, task2.py
86740a5 (origin/main, origin/HEAD, main) Initial commit

```

Рисунок 15 – Коммиты ветки develop во время выполнения лабораторной работы



```

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering13 (develop)
$ git checkout main
Switched to branch 'main'
M       .gitignore
Your branch is up to date with 'origin/main'.

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering13 (main)
$ git merge develop
Updating 86740a5..71f9e76
Fast-forward
 example1.py      | 21 ++++++
 individual1.py   | 27 ++++++
 individual2.py   | 17 ++++++
 task1.py         | 22 ++++++
 task2.py         | 22 ++++++
 5 files changed, 109 insertions(+)
 create mode 100644 example1.py
 create mode 100644 individual1.py
 create mode 100644 individual2.py
 create mode 100644 task1.py
 create mode 100644 task2.py

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering13 (main)

```

Рисунок 16 – Слияние ветки develop в ветку main

```

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering13 (main)
$ git push origin main
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 12 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 2.11 KiB | 2.11 MiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/MrPlatynum/ProgrammEngineering13.git
 86740a5..71f9e76  main -> main

```

Рисунок 17 – Отправка на удаленный репозиторий

Ответы на контрольные вопросы:

1. Какие аргументы называются позиционными в Python?

Позиционные аргументы в Python – это аргументы, передаваемые функции по порядку их расположения в вызове функции. Порядок передачи значений важен для правильной интерпретации аргументов.

2. Какие аргументы называются именованными в Python?

Именованные аргументы в Python – это аргументы, которые передаются функции с указанием их имени и значения. Они позволяют явно указать, какому параметру функции присваивается передаваемое значение.

3. Для чего используется оператор \*?

Оператор \* в Python используется для распаковки элементов из структуры данных, такой как список или кортеж. Например, он может быть использован для передачи аргументов в функцию или объединения нескольких структур данных.

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

\*args и \*\*kwargs – это соглашения для обработки переменного числа аргументов в функциях в Python. \*args используется для передачи переменного числа позиционных аргументов, а \*\*kwargs – для передачи переменного числа именованных аргументов (они представлены в виде словаря). Эти конструкции позволяют функциям работать с различным числом аргументов без необходимости определения заранее фиксированного числа параметров.