Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.10
дисциплины «Основы программной инженерии»

Выполнил:
Юрьев Илья Евгеньевич
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

_____
(подпись)

Руководитель практики:
Богданов С.С., ассистент кафедры
инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

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, в котором будет использована лицензия MIT и язык программирования Python:

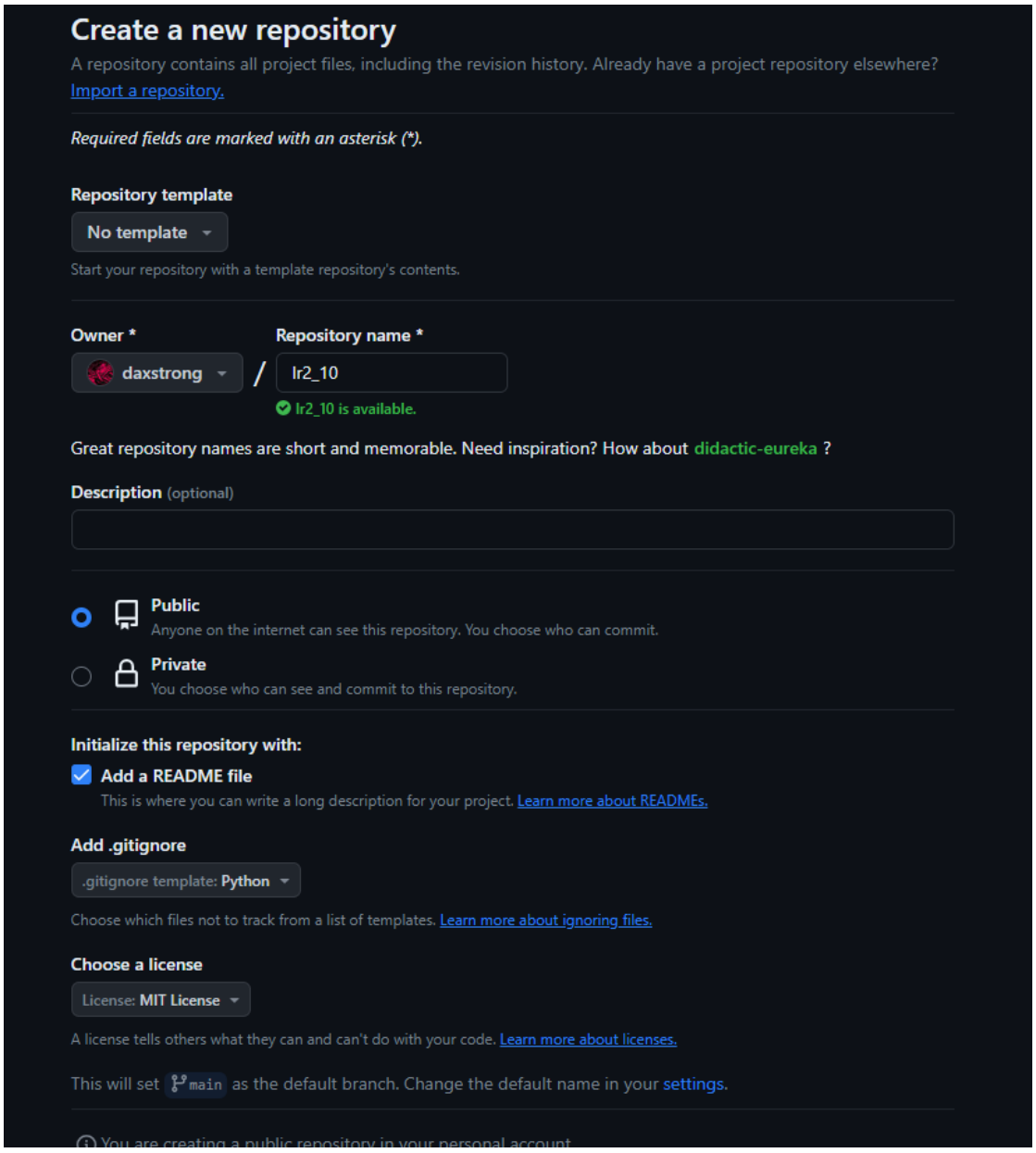

Рисунок 1 – Создание репозитория с заданными настройками

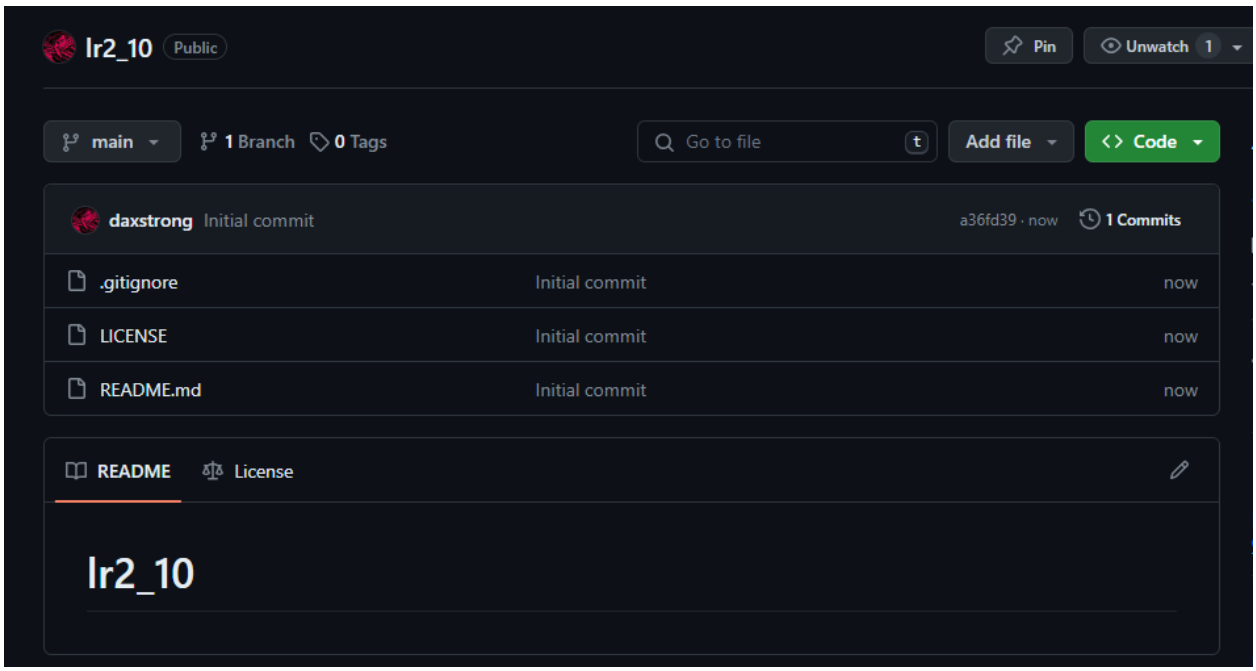

Рисунок 2 – Созданный репозиторий

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инжене
ии
$ git clone https://github.com/daxstrong/lr2_10.git
Cloning into 'lr2_10'...
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 3 – Клонирование репозитория

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженер
ии/lr2_10 (main)
$ git checkout -b develop
Switched to a new branch 'develop'
```

Рисунок 4 – Создание ветки develop

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

```python
#!/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))
```

Рисунок 5 – Пример №1

```
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
None
6.0
4.5
```

Рисунок 6 – Вывод программы (Пример №1)

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

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

def geometric_mean(*args):
    if not args:
        return None

    product = 1
    for arg in args:
        product *= arg

    return product ** (1 / len(args))


if __name__ == "__main__":
    values = list(int(i) for i in input("Введите числа: ").split())

    result = geometric_mean(*values)
    if result is None:
        print("Список аргументов пуст, среднее геометрическое не может быть вычислено.")
    else:
        print(f"Среднее геометрическое: {result}")
```

Рисунок 7 –Задание №1

```
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
Введите числа: 2 3 5 2 1 8
Среднее геометрическое: 2.7981664143395273
```

Рисунок 8 – Вывод программы (Задание №1)

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

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

def harmonic_mean(*args):
    if not args:
        return None

    return len(args) / sum(1 / arg for arg in args)


if __name__ == "__main__":
    arguments = [float(i) for i in input("Введите числа: ").split()]

    result = harmonic_mean(*arguments)
    if result is None:
        print("Список аргументов пуст, среднее гармоническое не может быть вычислено.")
    else:
        print(f"Среднее гармоническое: {result}")
```

Рисунок 9 – Задание №2

```
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
Введите числа: 1 2 3 8 9
Среднее гармоническое: 2.416107382550336
```

Рисунок 10 – Вывод программы (Задание №2)

5. Выполним индивидуальное задание:

Функция принимает произвольное количество аргументов и возвращает сумму аргументов, расположенных после максимального аргумента:

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

def sum_after_max(*args):
    if not args:
        return None

    max_arg = max(args)
    max_index = args.index(max_arg)

    if max_index == len(args) - 1:
        return None  # Если максимальный аргумент находится в конце списка

    return sum(args[max_index + 1:])


# Пример использования:
arguments = [3, 7, 2, 8, 5, 2]
result = sum_after_max(*arguments)

if result is None:
    print("Список аргументов пуст или максимальный аргумент находится в конце списка.")
else:
    print(f"Сумма аргументов после максимального: {result}")
```

Рисунок 11 – Решение индивидуального задания

```
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
Сумма аргументов после максимального: 7
```

Рисунок 12 – Вывод программы (Индивидуальное задание)

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

Задача: Написать функцию, которая принимает произвольное количество именованных аргументов и возвращает их количество.

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

def count_arguments(**kwargs):
    return len(kwargs)


# Пример использования:
arguments_count = count_arguments(name="Alice", age=30, city="London", occupation="Engineer")

print(f"Количество аргументов: {arguments_count}")

```

Рисунок 13 – Решение задания

```
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
Количество аргументов: 4
```

Рисунок 14 – Вывод программы

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

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженер
ии/lr2_10 (develop)
$ git log --oneline
b2a8311 (HEAD -> develop) финальные изменения
a36fd39 (origin/main, origin/HEAD, main) Initial commit
```

Рисунок 15 – Коммиты ветки develop во время выполнения лабораторной Работы

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженер
ии/lr2_10 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженер
ии/lr2_10 (main)
$ git merge develop
Updating a36fd39..b2a8311
Fast-forward
 .idea/.gitignore                                  |  8 ++++++++
 .idea/inspectionProfiles/profiles_settings.xml    |  6 ++++++
 .idea/lr2_10.iml                                  |  8 ++++++++
 .idea/misc.xml                                    |  4 ++++
 .idea/modules.xml                                 |  8 ++++++++
 .idea/vcs.xml                                     |  6 ++++++
 ex1.py                                            | 21 +++++++++++++++++++++
 individual1.py                                    | 24 ++++++++++++++++++++++++
 individual2.py                                    | 11 +++++++++++
 task1.py                                          | 22 ++++++++++++++++++++++
 task2.py                                          | 18 ++++++++++++++++++
 11 files changed, 136 insertions(+)
 create mode 100644 .idea/.gitignore
 create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
 create mode 100644 .idea/lr2_10.iml
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/modules.xml
 create mode 100644 .idea/vcs.xml
 create mode 100644 ex1.py
 create mode 100644 individual1.py
 create mode 100644 individual2.py
 create mode 100644 task1.py
 create mode 100644 task2.py
```

Рисунок 16 – Слияние веток main и develop

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженер
ии/lr2_10 (main)
$ git push origin main
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 12 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (15/15), 3.06 KiB | 3.06 MiB/s, done.
Total 15 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/daxstrong/lr2_10.git
   a36fd39..b2a8311  main -> main
```

Рисунок 17 – Отправка изменений на удаленный репозиторий

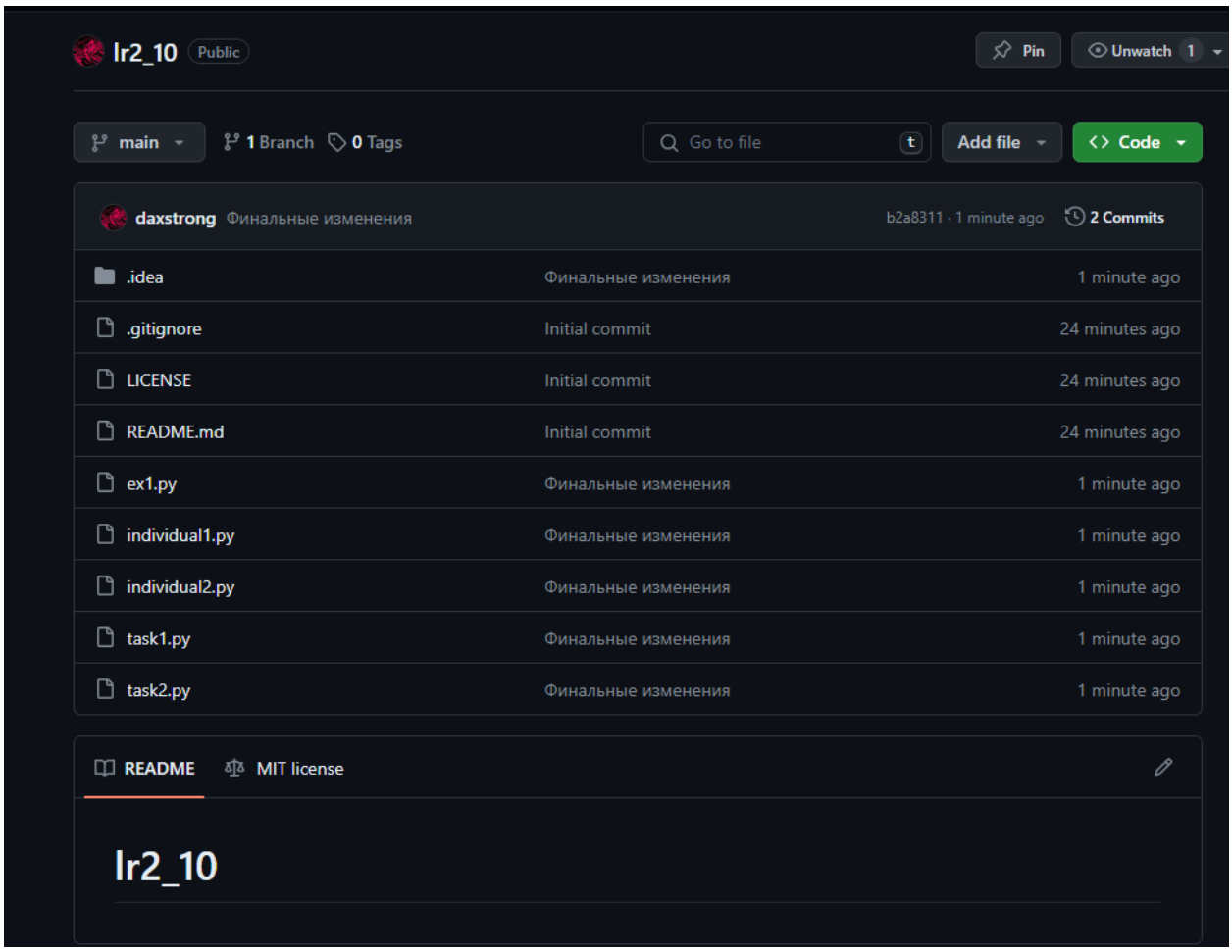

Рисунок 18 – Изменения удаленного репозитория

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.