

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

**Кафедра
инфокоммуникаций
Институт цифрового
развития**

**ОТЧЁТ
по лабораторной работе №2.10**

Дисциплина: «Основы программной инженерии»

Тема: «Функции с переменным числом параметров в Python»

Выполнила:
студентка 2 курса
группы Пиж-б-о-21-1
Джолдошова Мээрим
Бекболотовна

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.

1. Был создан репозиторий в Github в который были добавлены правила gitignore для работы IDE PyCharm, была выбрана лицензия MIT, сам репозиторий был клонирован на локальный сервер и был организован в соответствии с моделью ветвления git-flow.

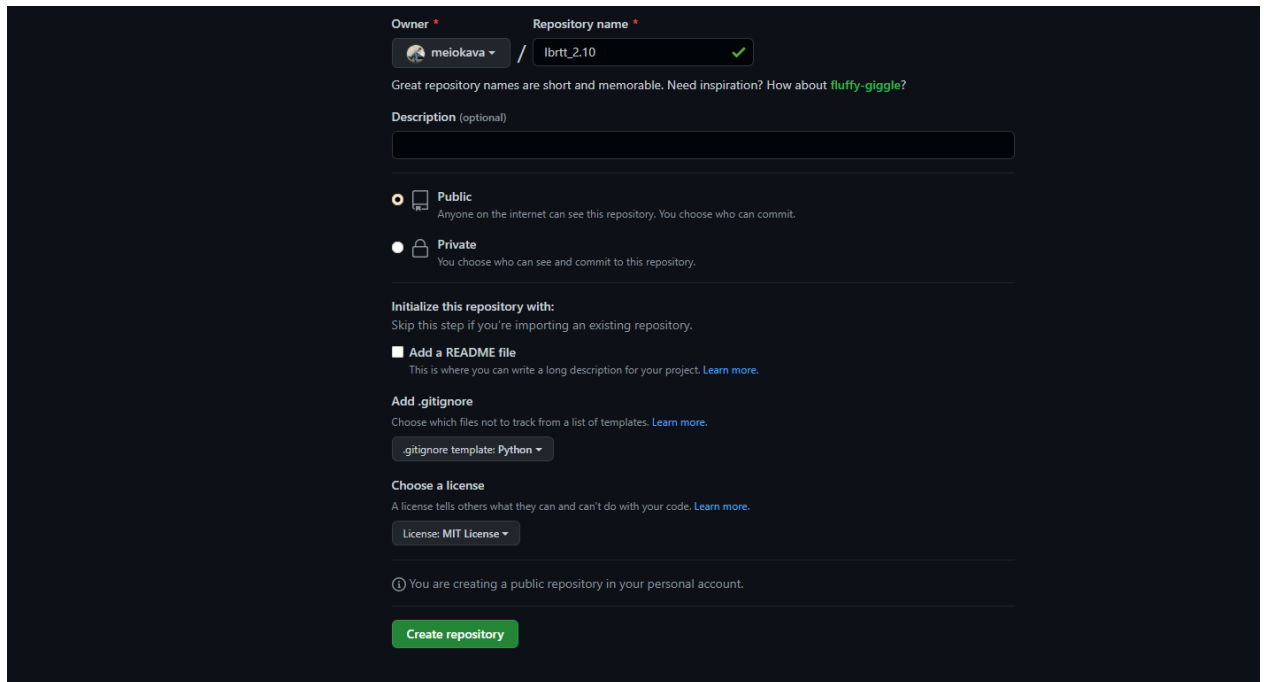


Рисунок 1 – Создание репозитория

```
c:\Users\мвидео>cd/d c:\lbrtt_2.10
c:\lbrtt_2.10>git clone https://github.com/meiokava/lbrtt_2.10.git
Cloning into 'lbrtt_2.10'...
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.
c:\lbrtt_2.10>
```

Рисунок 2 – Клонирование репозитория

```
c:\lbrtt_2.10>cd/d c:\lbrtt_2.10\lbrtt_2.10
c:\lbrtt_2.10\lbrtt_2.10>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:/lbrtt_2.10/lbrtt_2.10/.git/hooks]
c:\lbrtt_2.10\lbrtt_2.10>
```

Рисунок 3 – Организация репозитория с моделью ветвления git-flow

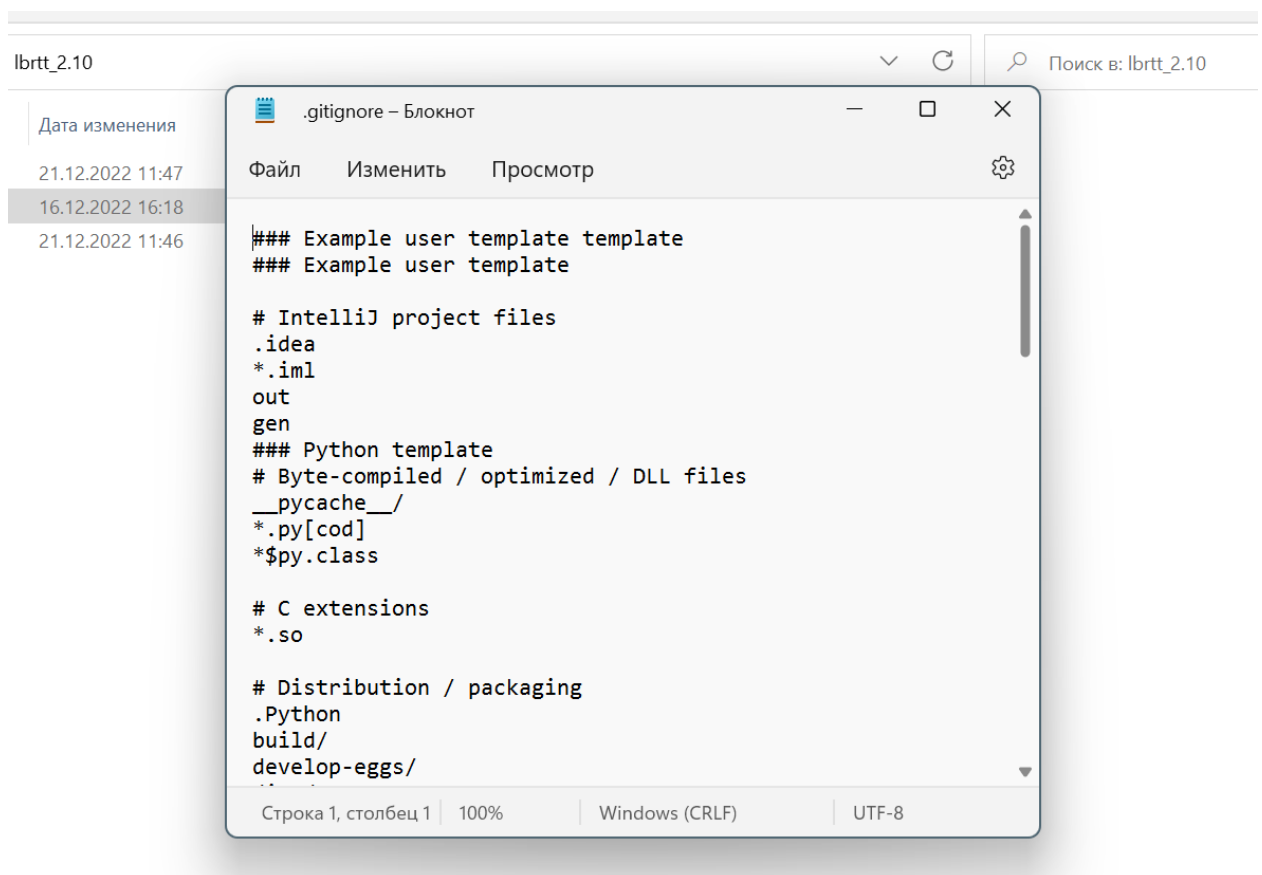


Рисунок 4 – Дополнение файла gitignore

2. Была создана папка PyCharm в которой хранятся примеры из лабораторной работы.

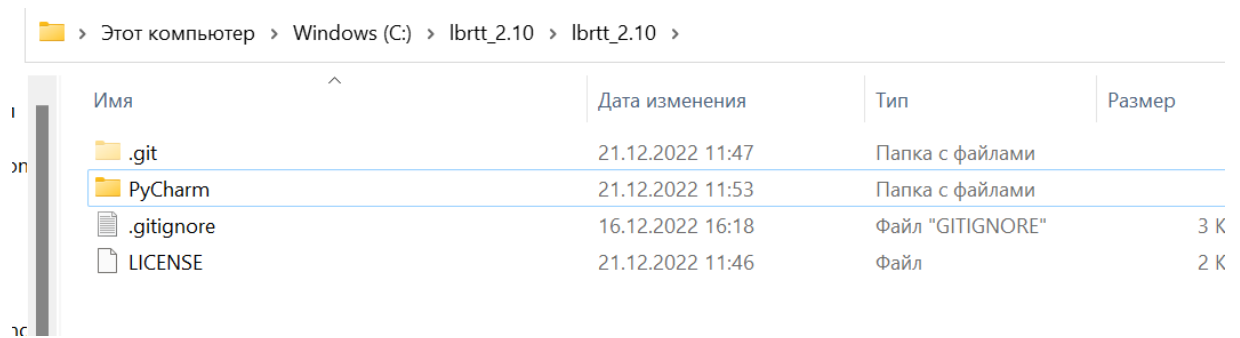


Рисунок 5 – Папка для хранения примеров

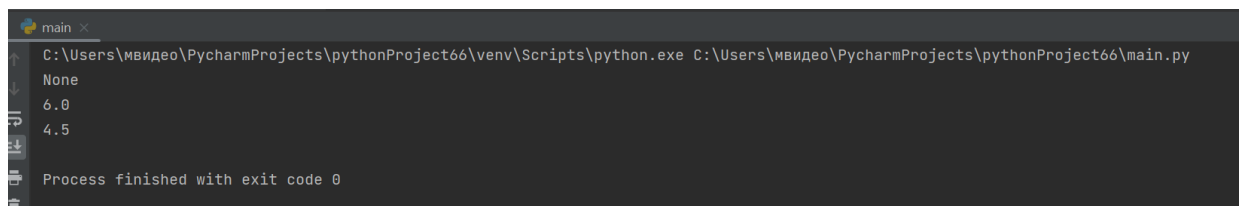


Рисунок 6 – Результат работы примера

3. 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 программы:

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

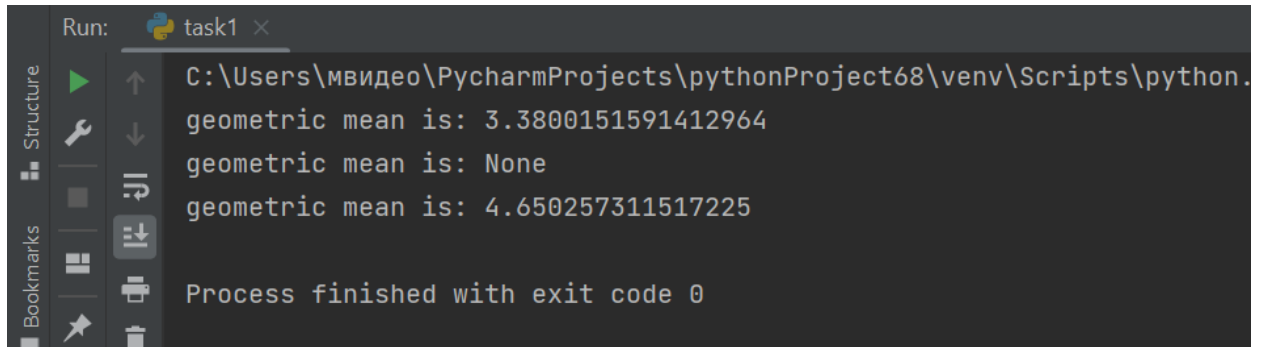
import math

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

```

if __name__ == "__main__":
    print(f'geometric mean is: {aver_geom(1, 2, 3, 4, 5, 6, 7)}')
    print(f'geometric mean is: {aver_geom()}')
    print(f'geometric mean is: {aver_geom(2.3, 6.5, 9.2, 3.4)}')

```



```

Run: task1 x
C:\Users\мвидео\PycharmProjects\pythonProject68\venv\Scripts\python.
geometric mean is: 3.3800151591412964
geometric mean is: None
geometric mean is: 4.650257311517225
Process finished with exit code 0

```

Рисунок 7 – Результат работы программы

4. 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 программы:

```

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

import math

def aver_harm(*args):
    if args:
        values = [float(arg) for arg in args]
        sz = len(args)
        s = 0
        for arg in values:
            s += 1 / arg
        g = sz / s
        return g
    else:
        return None

if __name__ == "__main__":
    print(f'harmonic mean is: {aver_harm(1, 2, 3, 4, 5, 6, 7, 8)}')
    print(f'harmonic mean is: {aver_harm()}')
    print(f'harmonic mean is: {aver_harm(1.5, 4.6, 9.3, 8.0, 10.5)}')

```

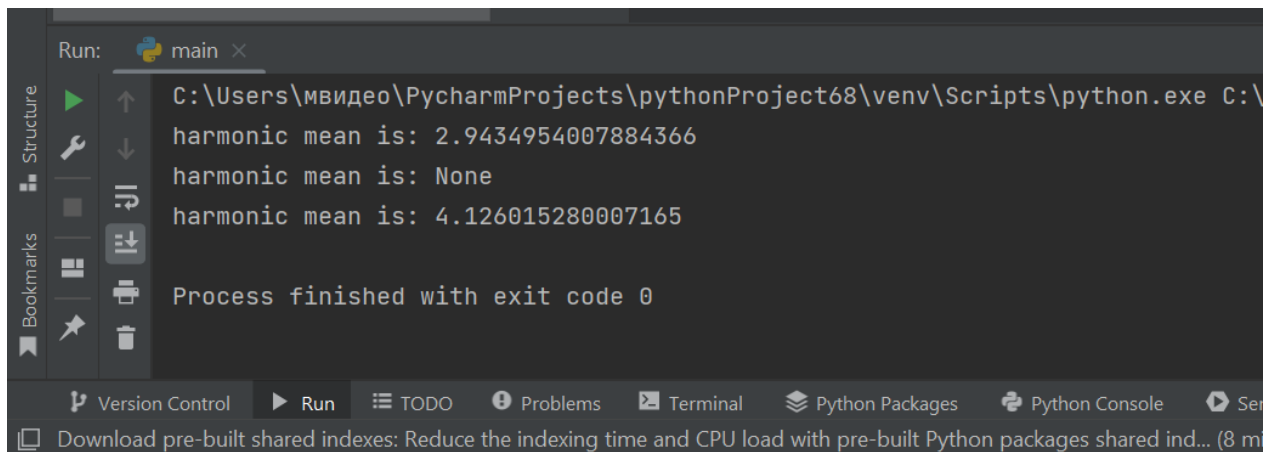


Рисунок 8 – Результат работы программы

5. Было выполнено индивидуальное задание

Вариант 5

Сумму аргументов, расположенных до последнего положительного аргумента.

Код программы:

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

import math

def sum_pos(*args):
    """
    The sum of the arguments located up to the last positive argument
    """
    if args:
        values = [float(arg) for arg in args]
        sum_ = 0
        for num in values:
            if num >= 0:
                sum_ += num
            else:
                break
        return sum_

if __name__ == "__main__":
    print(f'sum positive arguments: {sum_pos(1, 2, 3, 4, 5, -6, 7, 8)}')
    print(f'sum positive arguments: {sum_pos(1.5, 4.6, -9.3, 8.0, 10.5)}')
```

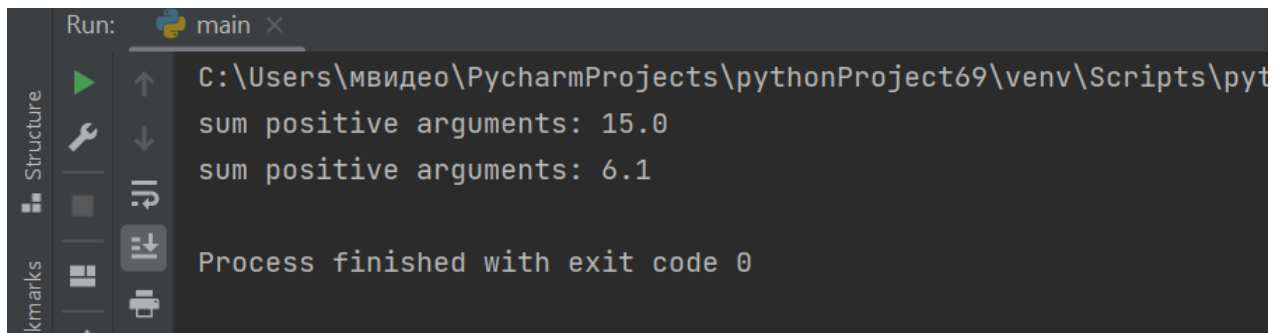


Рисунок 9 – Результат работы программы

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

```

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

def print_idols_group(**group):
    """
    which idol belongs to which group
    """
    if group:
        for name, group in group.items():
            print(f"{name}: {group}")
    else:
        return None

if __name__ == "__main__":
    print_idols_group(
        Nayeon="Twice", Momo="Twice", Karina="Aespa",
        Tzuyu="Twice", Winter="Aespa", Bibi="None",
        Ryujin="Itzy", DPR="None", Lia="Itzy", Lisa="Blackpink"
    )

```

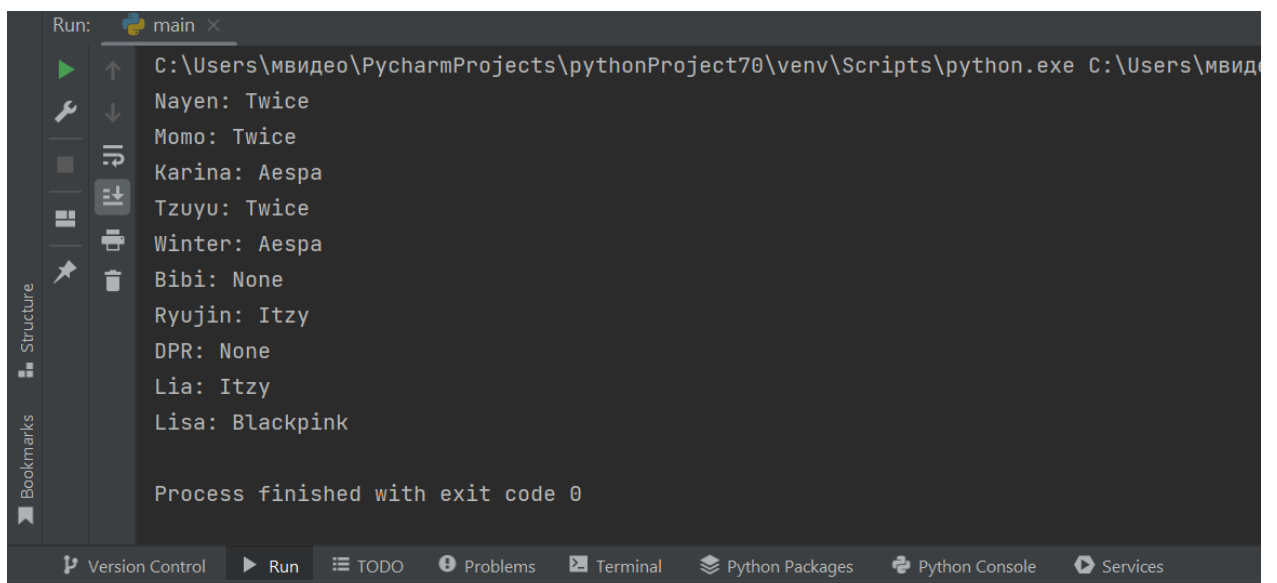


Рисунок 10 – Результат работы программы

```
c:\lbrtt_2.10\lbrtt_2.10>git add .

c:\lbrtt_2.10\lbrtt_2.10>git commit -m "finished programs"
[develop f242271] finished programs
6 files changed, 138 insertions(+), 2 deletions(-)
create mode 100644 pycharm/examp.py
create mode 100644 indiv/indiv1.py
create mode 100644 indiv/indiv2.py
create mode 100644 tasks/task1.py
create mode 100644 tasks/task2.py
```

```
c:\lbrtt_2.10\lbrtt_2.10>git push --set-upstream origin develop
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 8 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (11/11), 2.21 KiB | 754.00 KiB/s, done.
Total 11 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/meiokava/lbrtt_2.10/pull/new/develop
remote:
To https://github.com/meiokava/lbrtt_2.10.git
* [new branch] develop -> develop
branch 'develop' set up to track 'origin/develop'.

c:\lbrtt_2.10\lbrtt_2.10>git chekout main
git: 'chekout' is not a git command. See 'git --help'.

The most similar command is
    checkout

c:\lbrtt_2.10\lbrtt_2.10>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

c:\lbrtt_2.10\lbrtt_2.10>git merge develop
Updating 780db2a..f242271
Fast-forward
```

```
c:\lbrtt_2.10\lbrtt_2.10>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

c:\lbrtt_2.10\lbrtt_2.10>git merge develop
Updating 780db2a..f242271
Fast-forward
 .gitignore      | 24 ++++++++
 PyCharm/examp.py | 23 ++++++++
 indiv/indiv1.py  | 25 ++++++++
 indiv/indiv2.py  | 22 ++++++++
 tasks/task1.py   | 23 ++++++++
 tasks/task2.py   | 23 ++++++++
 6 files changed, 138 insertions(+), 2 deletions(-)
 create mode 100644 PyCharm/examp.py
 create mode 100644 indiv/indiv1.py
 create mode 100644 indiv/indiv2.py
 create mode 100644 tasks/task1.py
 create mode 100644 tasks/task2.py

c:\lbrtt_2.10\lbrtt_2.10>git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/meiokava/lbrtt_2.10.git
 780db2a..f242271 main -> main
```

Рисунок 11 – Коммит и пуш изменений переход на ветку main и слияние ее с веткой develop

вывод: в ходе лабораторной работы были приобретены навыки по работе с функциями с переменным числом параметров при написании программ с

помощью языка программирования Python версии 3.x.

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 `*`?

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.