

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙ-
СКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования**

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Основы кроссплатформенного программирования

Отчет по лабораторной работе №2.10

Тема: «Функции с переменным числом параметров в Python»

Выполнил студент группы

ИВТ-б-о-21-1

Назаров Н.Ю. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил доцент

Кафедры инфокоммуникаций, старший
преподаватель

Воронкин Р.А.

(подпись)

Выполнение работы:

```
H:\cross\git\nazarov>git clone https://github.com/NikitaNazarov179/2.10.git
Cloning into '2.10'...
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.

H:\cross\git\nazarov>cd /d H:\cross\git\nazarov\2.10
```

Рисунок 1 – клонирование репозитория

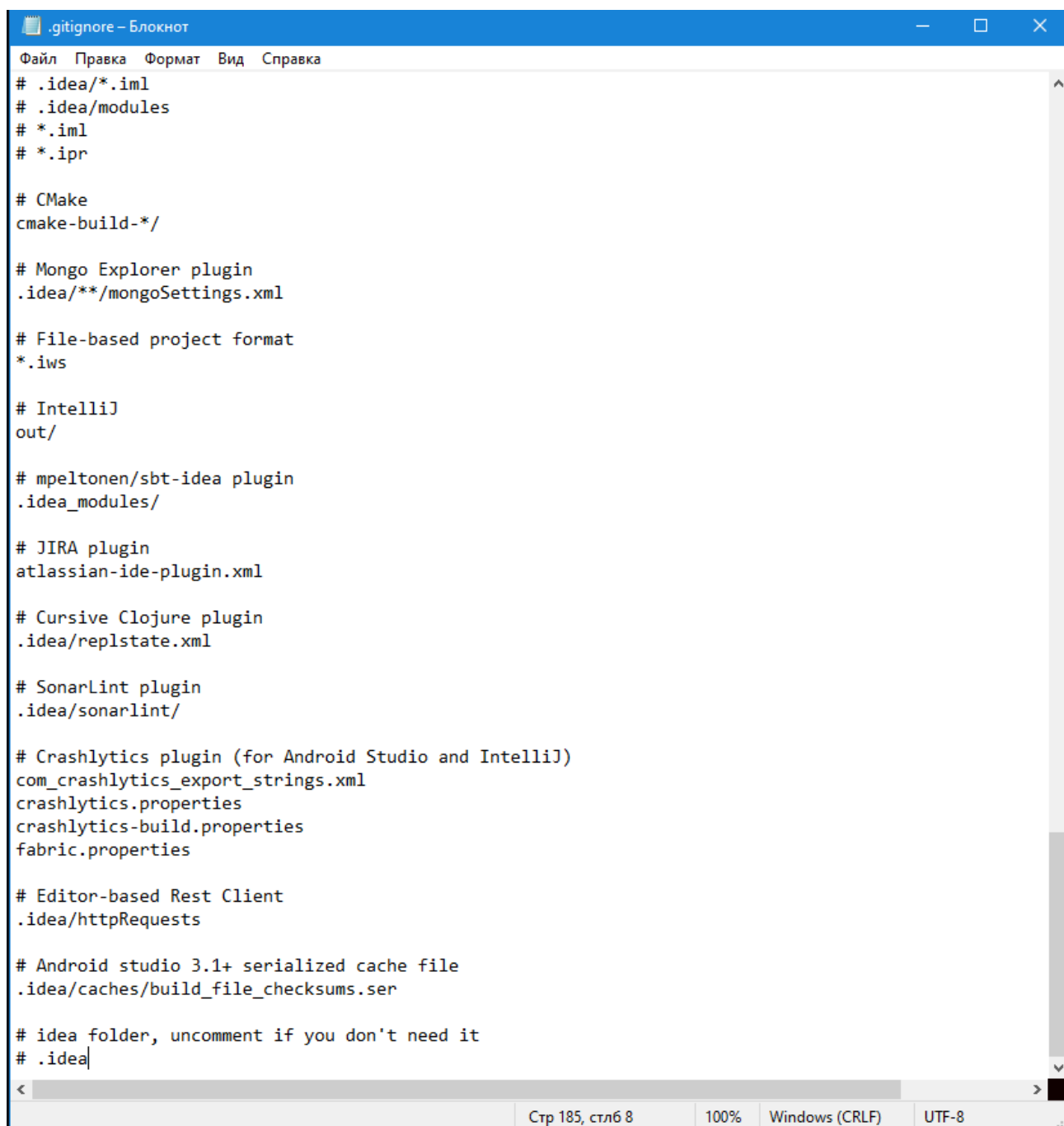


Рисунок 2 – редактирование файла readme

```

H:\cross\git\nazarov\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? [H:/cross/git/nazarov/2.10/.git/hooks]

```

Рисунок 3 – организация репозитория в соответствии с моделью ветвления git-flow

The screenshot shows an IDE window titled '5python(prim) - H:\cross\git\nazarov\2.10\1.py'. The editor displays a Python script for calculating the median of a list of numbers. The script includes a function 'median(*args)' that sorts the input values and returns the median. The main block calls this function with different sets of numbers and prints the results.

```

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
9          n = len(values)
10         idx = n // 2
11         if n % 2:
12             return values[idx]
13         else:
14             return (values[idx - 1] + values[idx]) / 2
15     else:
16         return None
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))

```

Below the editor, the 'Run' window shows the execution output:

```

Run: 1 x
H:\Python\python.exe H:/cross/git/nazarov/2.10/1.py
None
6.0
4.5
Process finished with exit code 0

```

The output matches the expected results for the median function: no arguments returns None, [3, 7, 1, 6, 9] returns 6.0, and [1, 5, 8, 4, 3, 9] returns 4.5.

Рисунок 4 – проработал пример

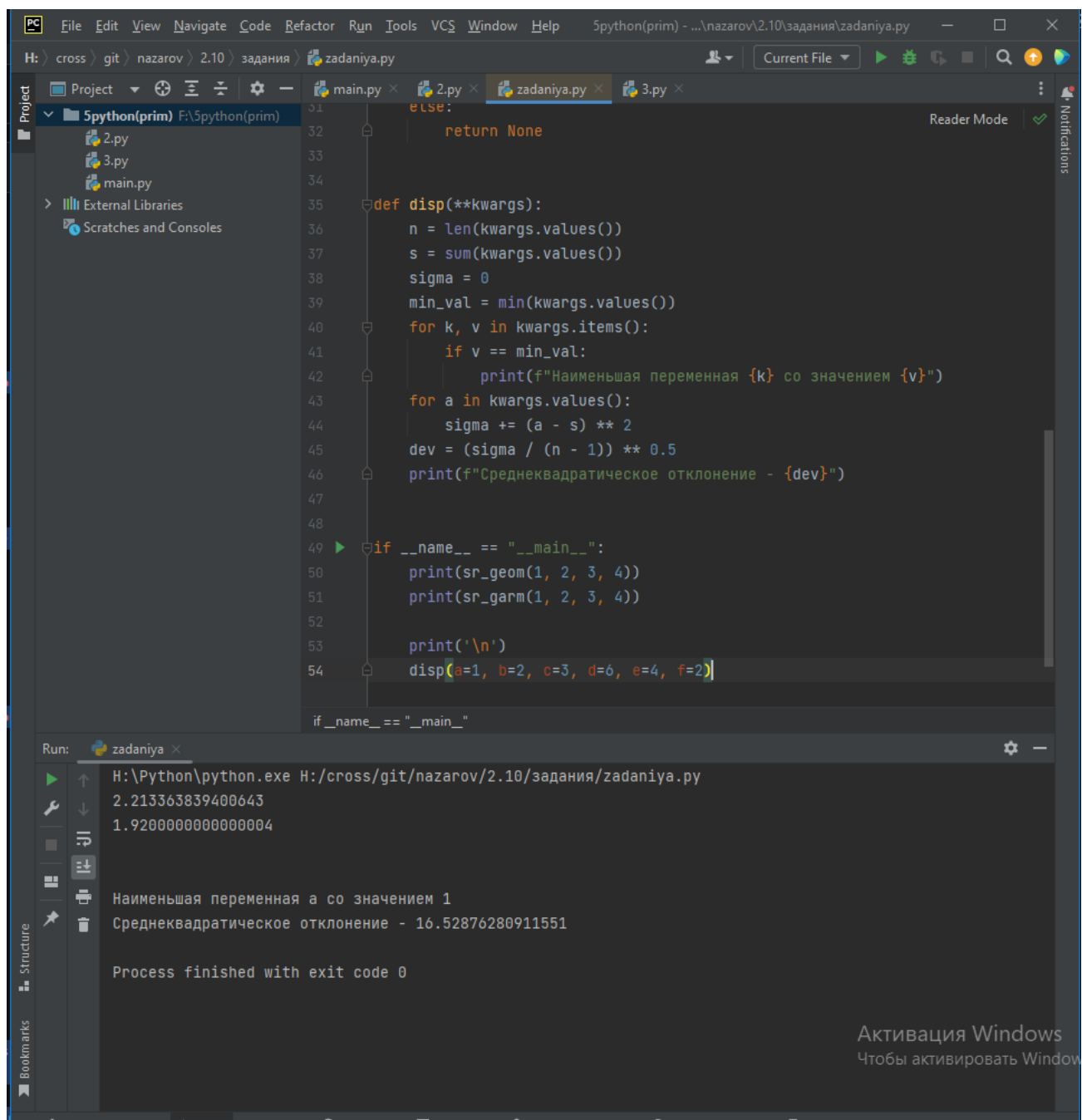


Рисунок 5 – выполнил 1, 2 и 3 задания

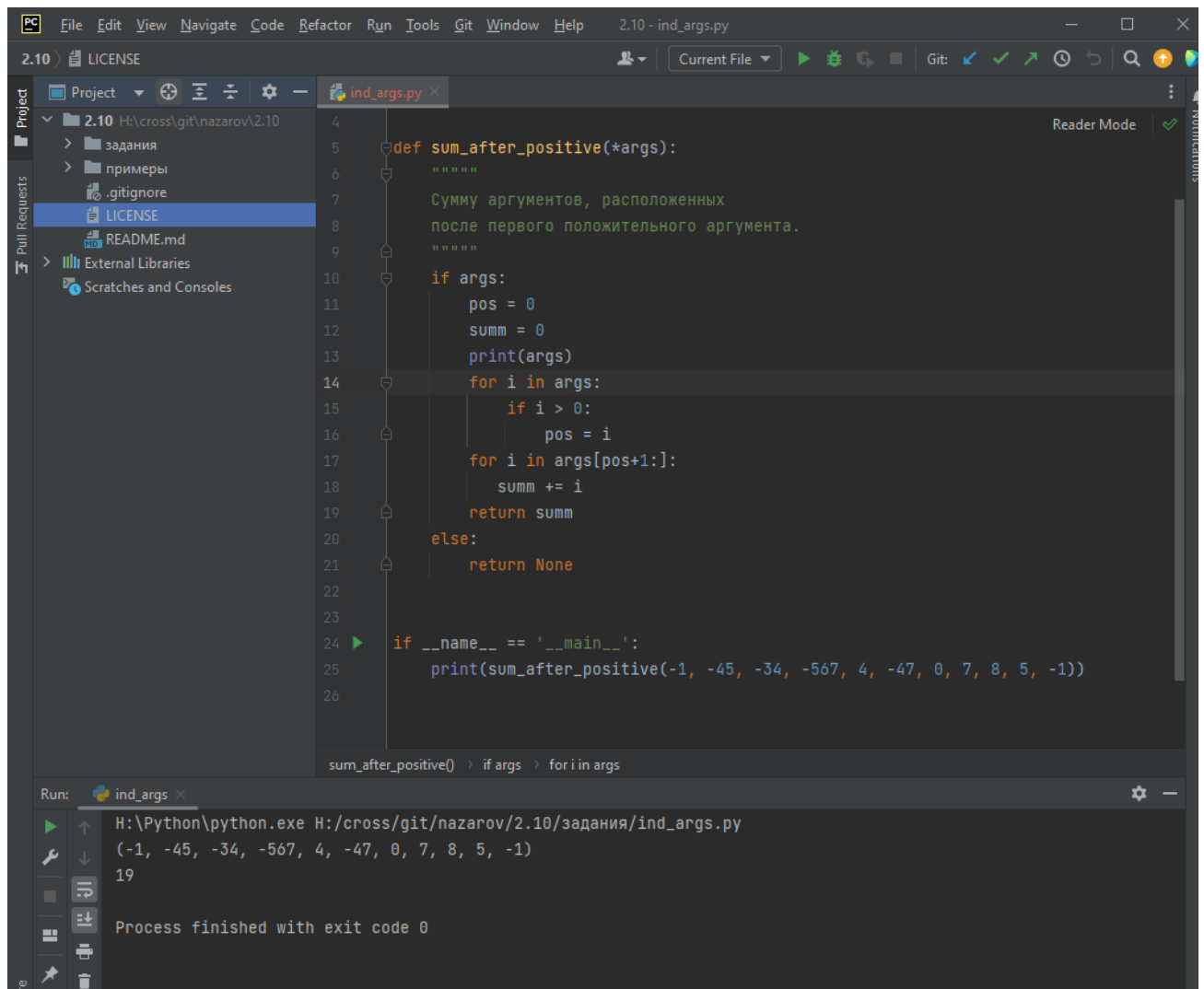


Рисунок 6 – выполнил индивидуальное задание с данным условием

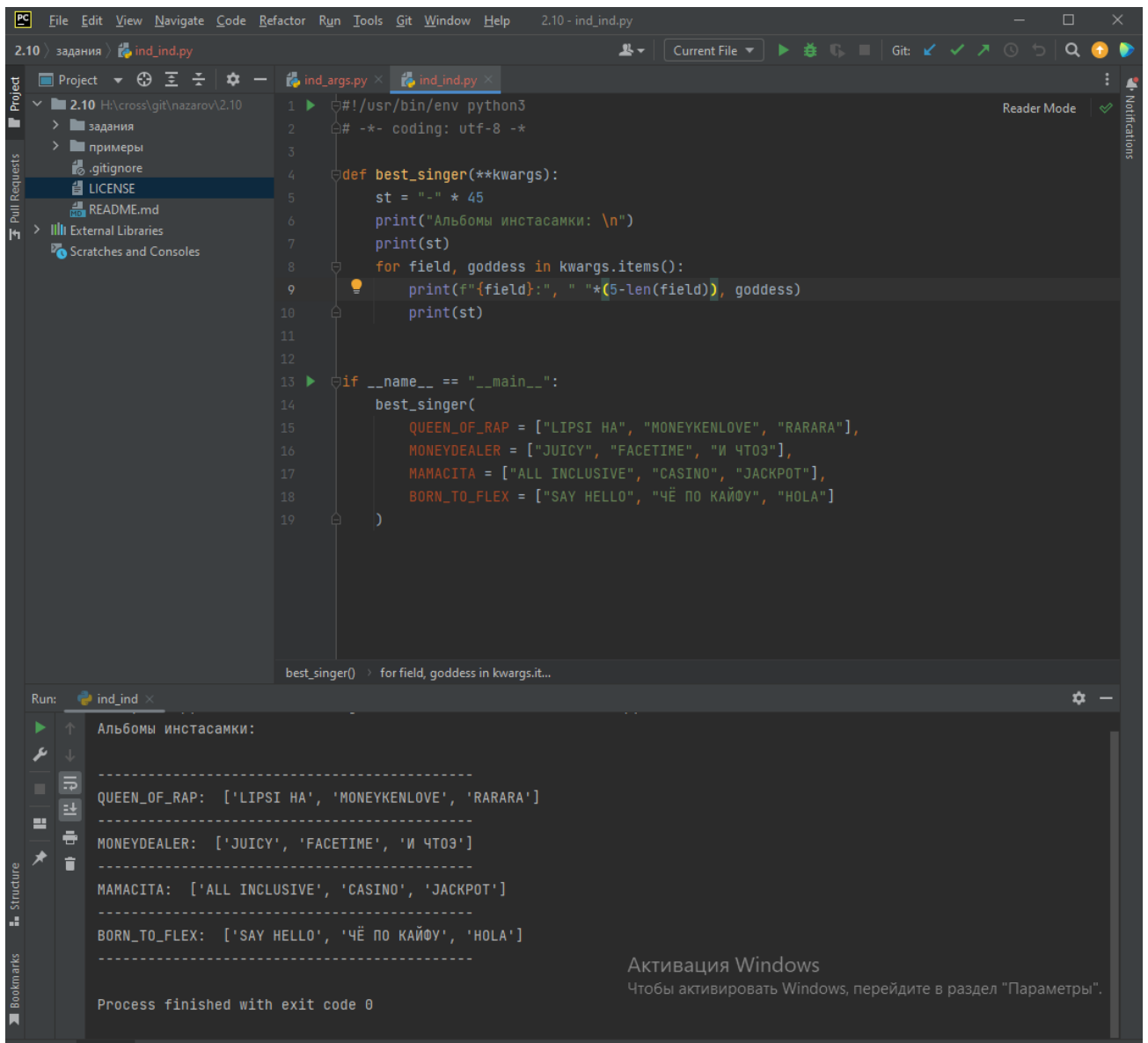


Рисунок 7 – выполнил самостоятельное задание.

Ответы на контрольные вопросы:

1. Какие аргументы называются позиционными в Python?

Аргументы, которые передаются без указания имен называются позиционными, потому что именно по позиции, расположению аргумента, функция понимает, какому параметру он соответствует.

2. Какие аргументы называются именованными в Python?

Аргументы, передаваемые с именами, называются именованными. При вызове функции можно использовать имена параметров из ее определения.

3. Для чего используется оператор *?

Оператор `*` чаще всего ассоциируется у людей с операцией умножения, но в Python он имеет и другой смысл.

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

Вот пример:

```
a = [1, 2, 3]
```

```
b = [*a, 4, 5, 6]
```

```
print(b) # [1, 2, 3, 4, 5, 6]
```

Тут берётся содержимое списка `a`, распаковывается, и помещается в список `b`.

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

Итак, мы знаем о том, что оператор «звёздочка» в Python способен «вытаскивать» из объектов составляющие их элементы. Знаем мы и о том, что существует два вида параметров функций. А именно, `*args` — это сокращение от «arguments» (аргументы), а `**kwargs` — сокращение от «keyword arguments» (именованные аргументы).

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

Важно помнить, что «args» — это всего лишь набор символов, которым принято обозначать аргументы. Самое главное тут — это оператор `*`. А то, что именно идёт после него, особой роли не играет. Благодаря использованию `*` мы создали список позиционных аргументов на основе того, что было передано функции при вызове.

После того, как мы разобрались с `*args`, с пониманием `**kwargs` проблем быть уже не должно.

Имя, опять же, значения не имеет. Главное — это два символа `**`. Благодаря им создаётся словарь, в котором содержатся именованные аргументы, переданные функции при её вызове.

