МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ 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)

Ставрополь 2022

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

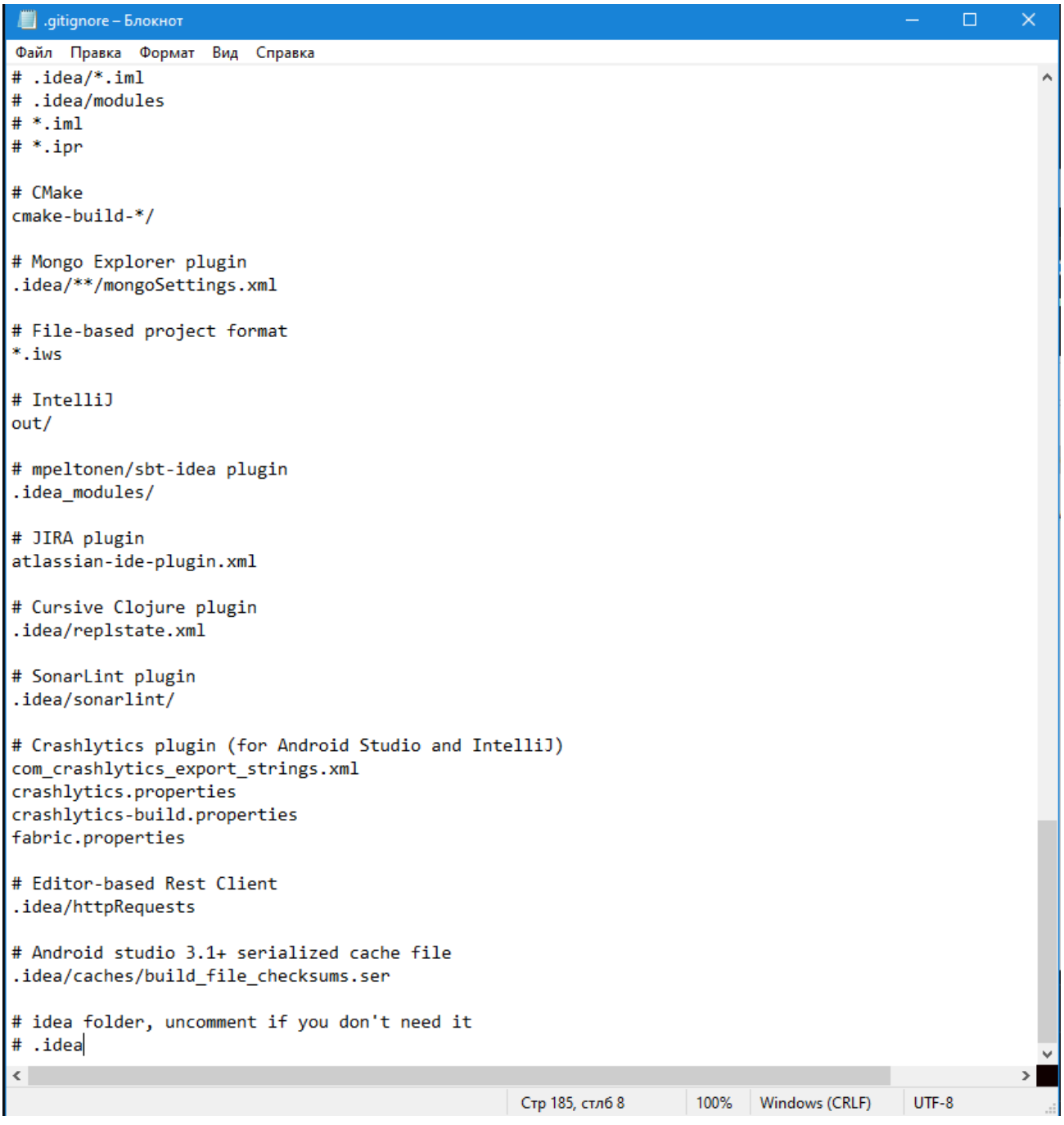

Рисунок 2 – редактирование файла readme

```
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

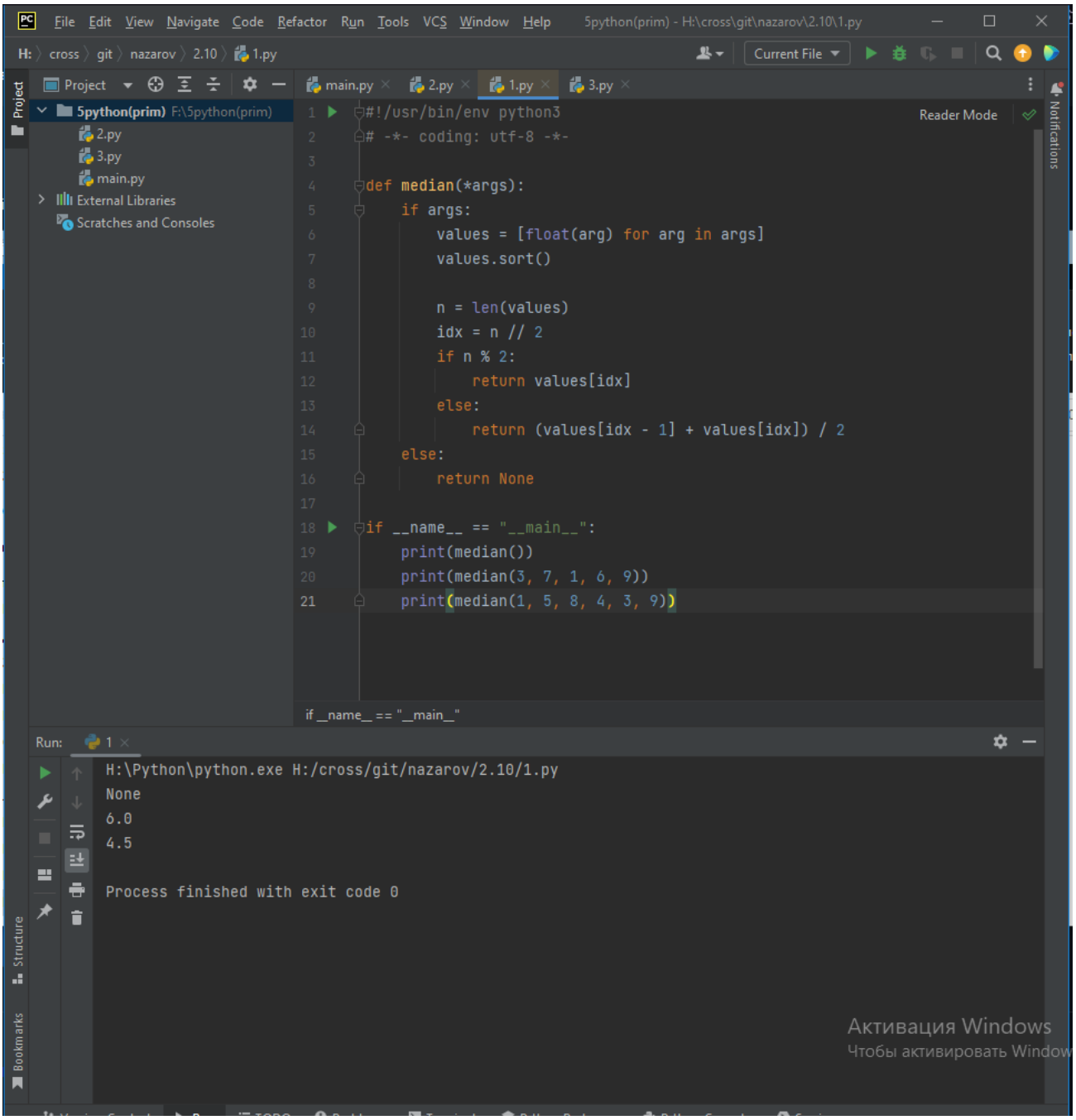

Рисунок 4 – проработал пример

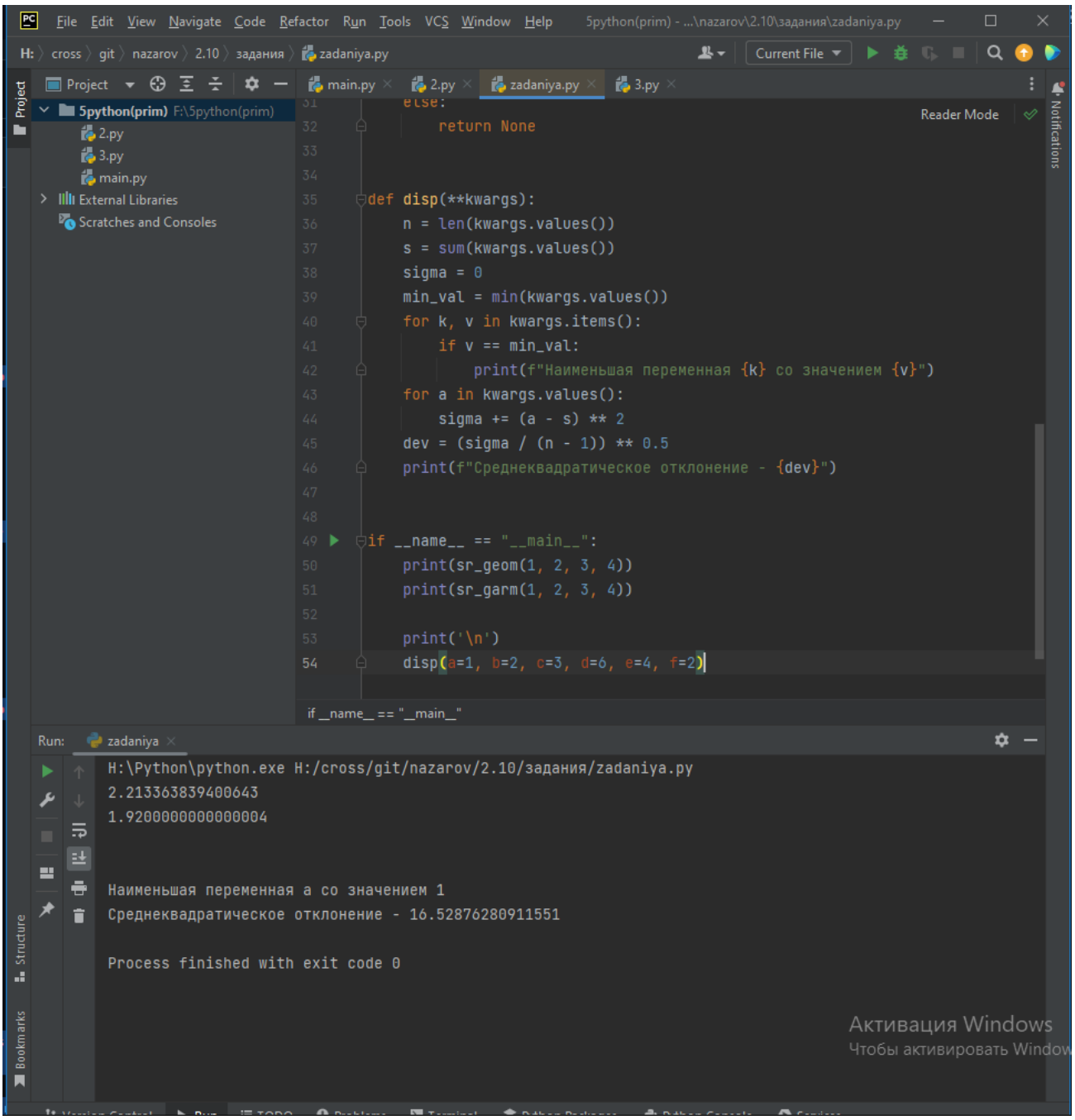

Рисунок 5 – выполнил 1, 2 и 3 задания

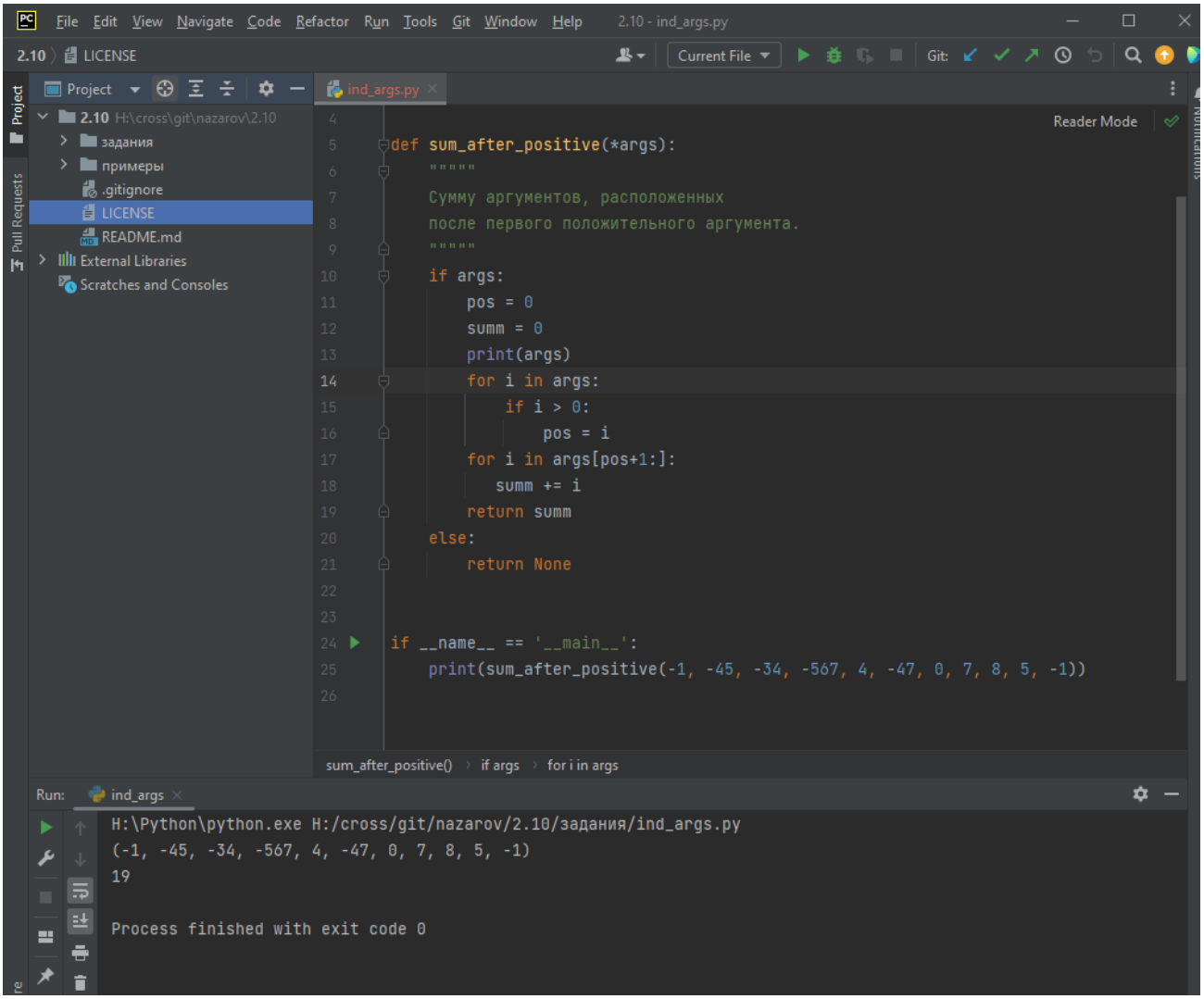

Рисунок 6 – выполнил индивидуальное задание с данным условием

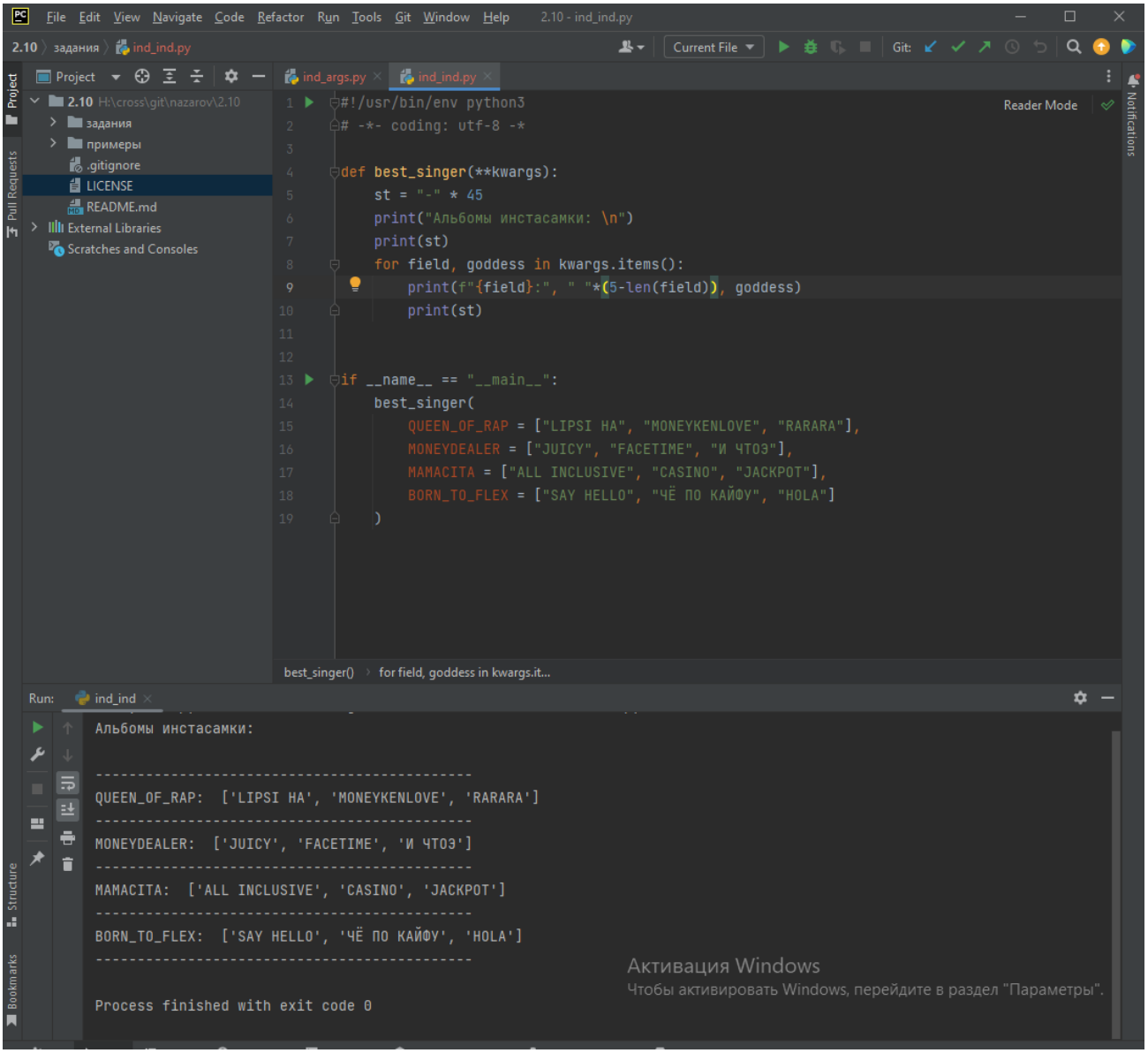

Рисунок 7 – выполнил самостоятельное задание.

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.