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 №2.8

Тема: «Работа с функциями в языке 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:\Python\python.exe H:/cross/git/nazarov/2.8/Примеры/1.py
>>> help
Список команд:

add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - вывести список команд;
exit - завершить работу с программой.
>>> add
Фамилия и инициалы: Харченко Б.Р.
Должность: Жадина
Год поступления: 2021
>>> list
+------+----------------------------------+----------------------+----------+
|  №   |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Харченко Б.Р.                    | Жадина               |     2021 |
+------+----------------------------------+----------------------+----------+
>>> select 1
+------+----------------------------------+----------------------+----------+
|  №   |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Харченко Б.Р.                    | Жадина               |     2021 |
+------+----------------------------------+----------------------+----------+
>>>
```

Рисунок 1 – проработал 1 пример

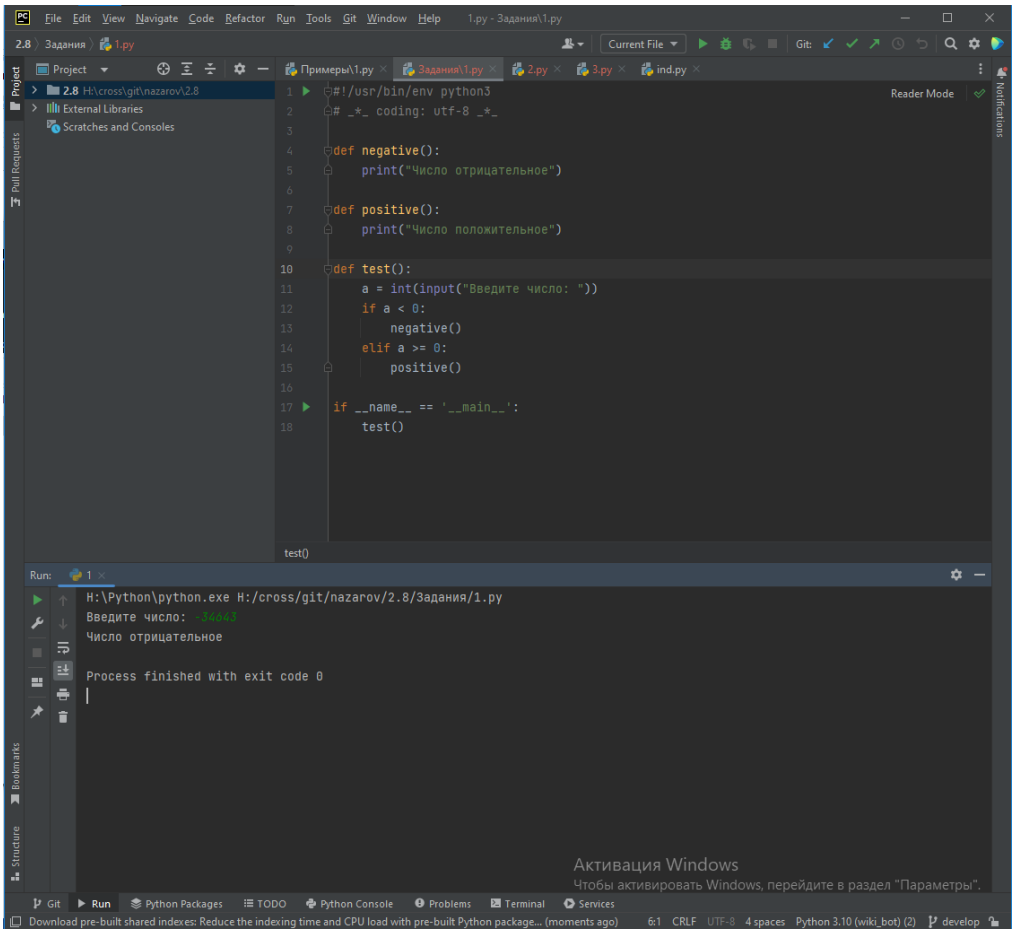

Рисунок 2 – 1 заданием с 1 вариантом расположения функций

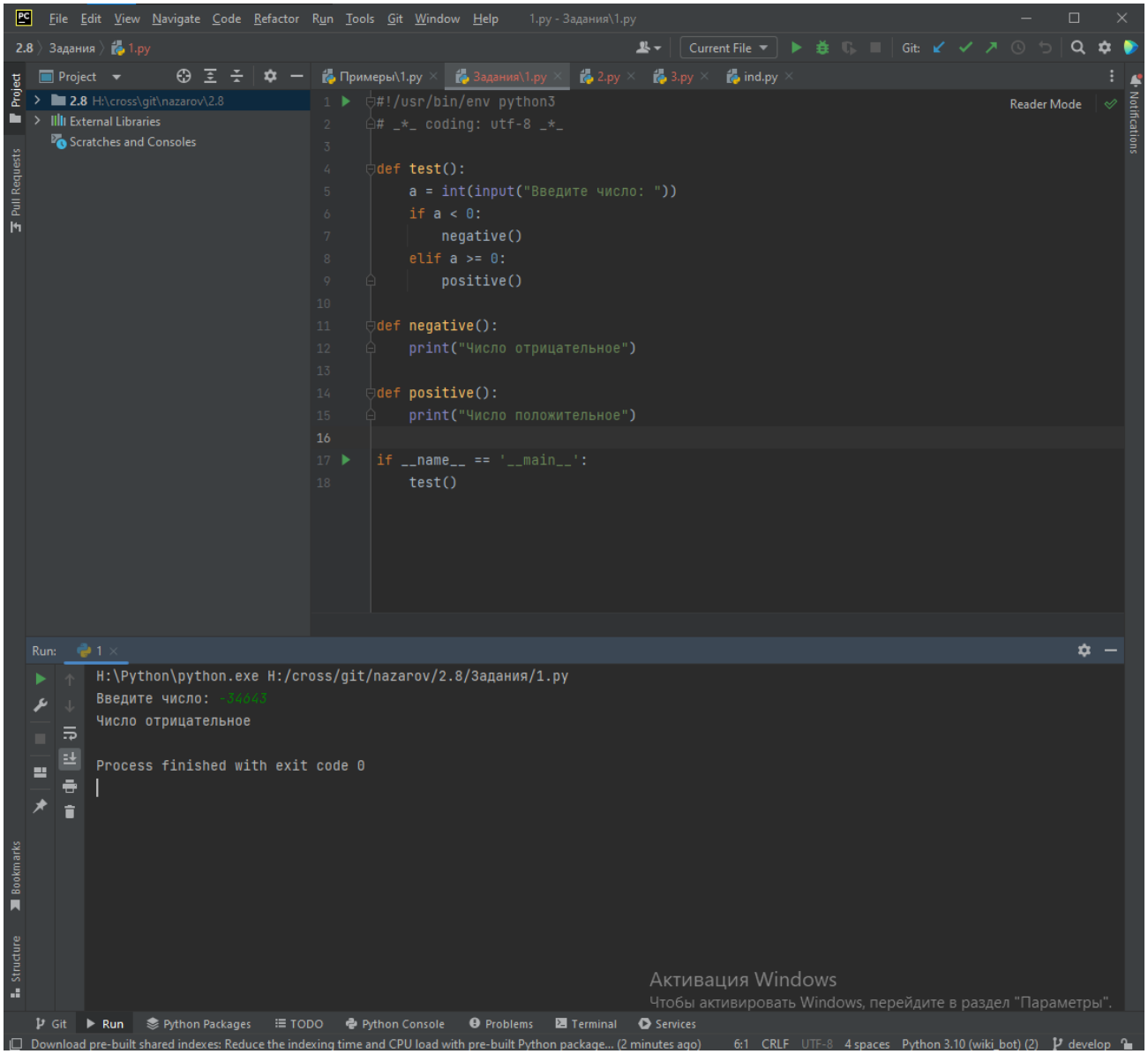

Рисунок 3 – 1 задание со вторым вариантом расположения функций

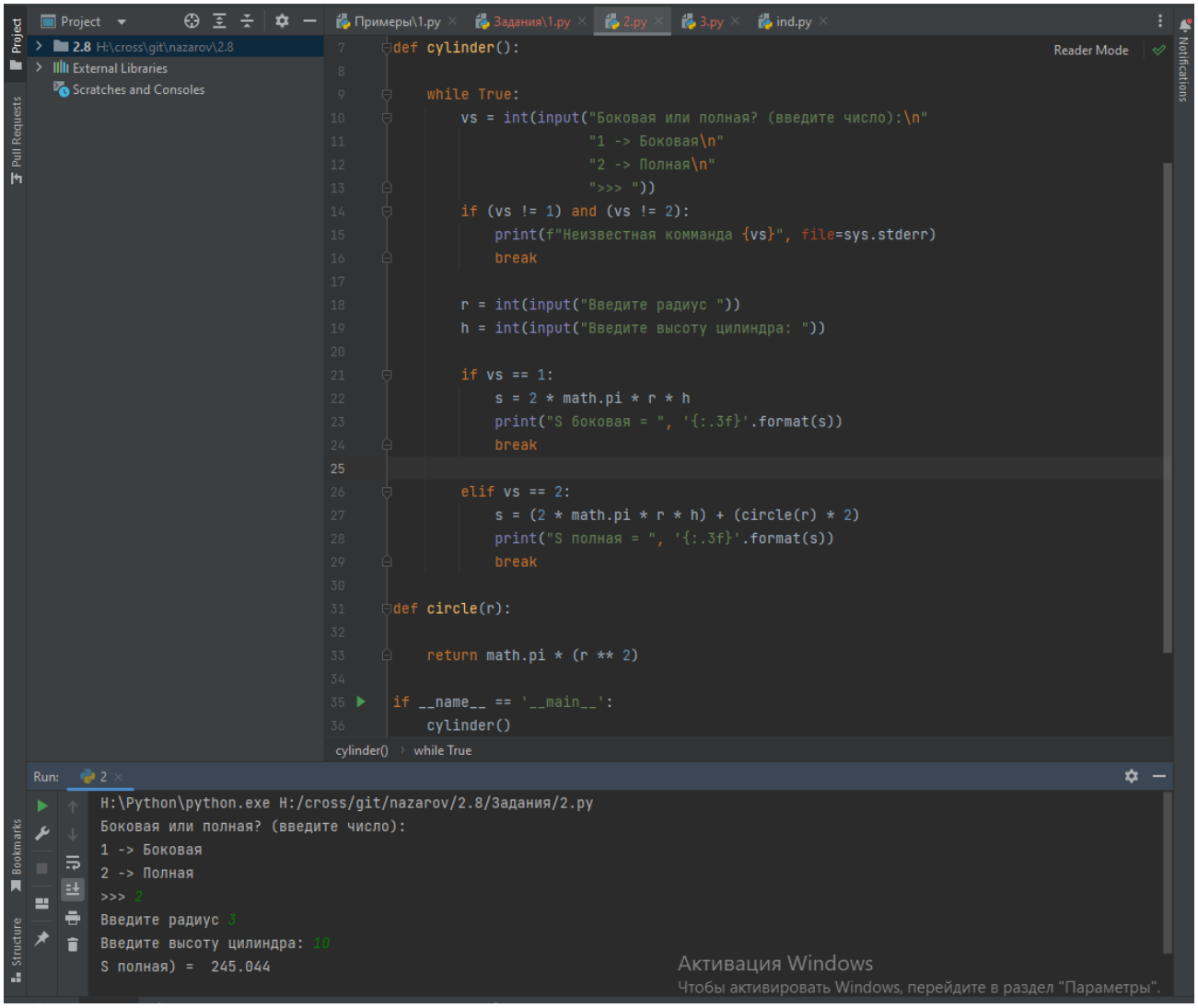

Рисунок 4 – 2 задание

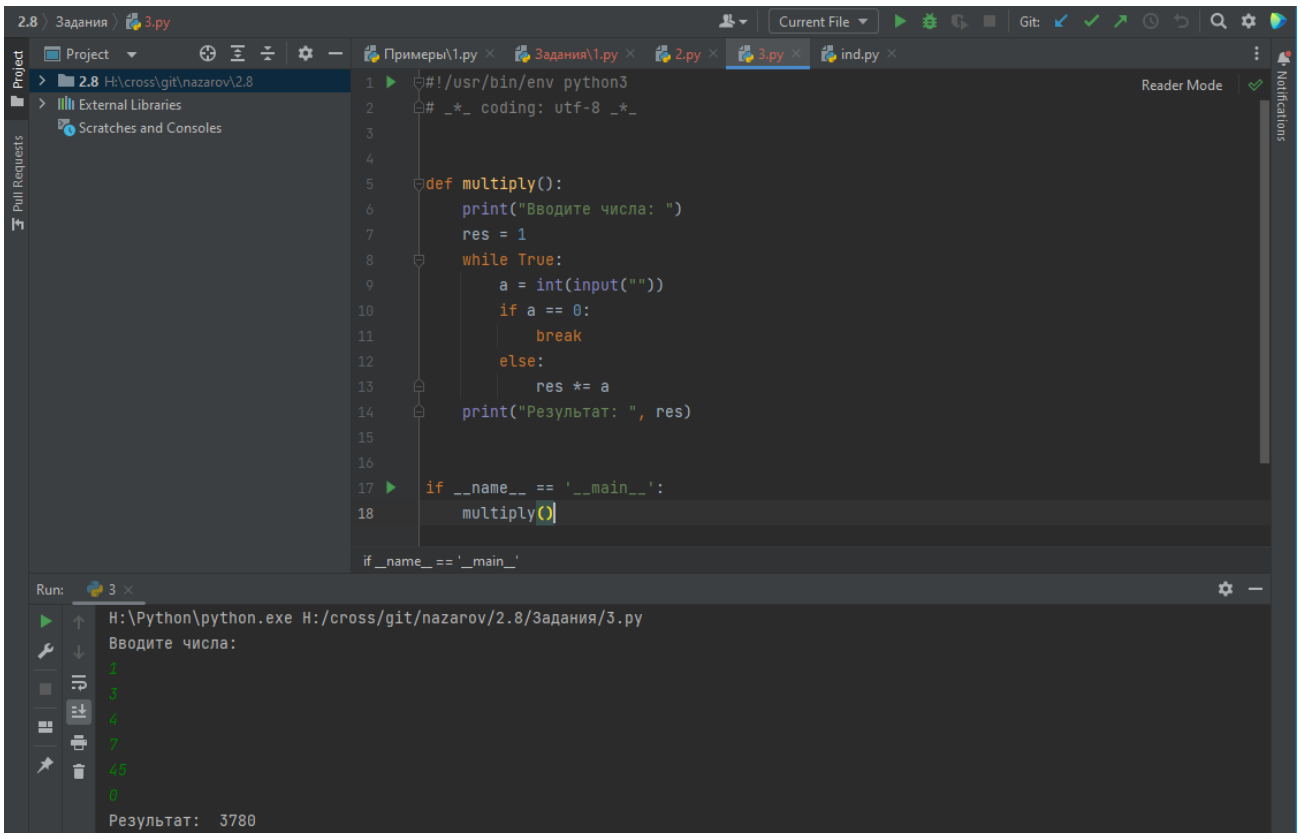

Рисунок 5 – 3 задание

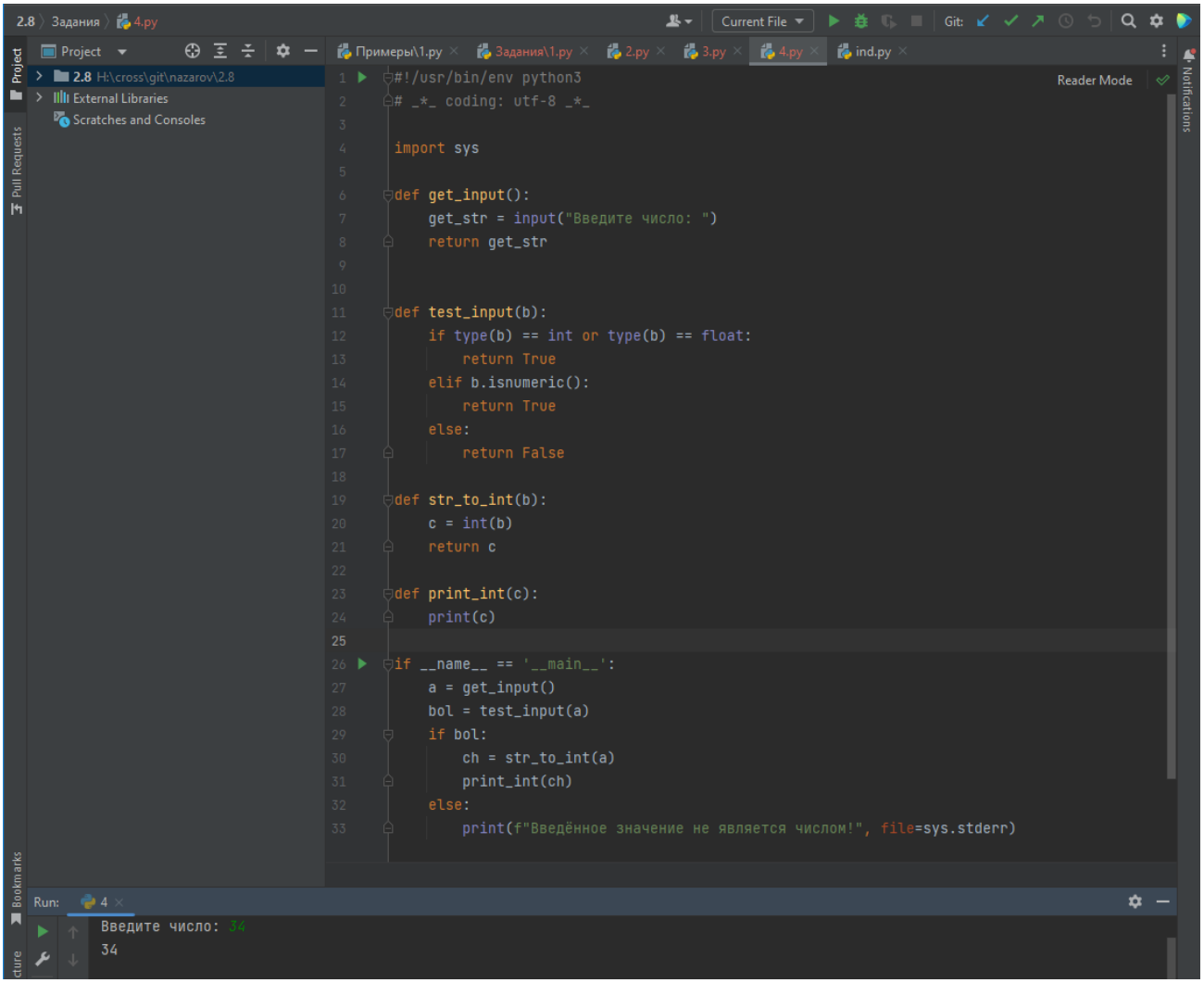

Рисунок 6 – 4 задание

```
H:\Python\python.exe H:/cross/git/nazarov/2.8/ind.py
help - список всех команд
>>> add
Название товара: сосиски
Магазин: пятерочка
Стоимость товара: 220
>>> add
Название товара: колбаса
Магазин: магнит
Стоимость товара: 300
>>> list
+------+--------------------------------+----------------------+-----------------+
|  №   |             Товар              |       Магазин        | Стоимость товара |
+------+--------------------------------+----------------------+-----------------+
|  1   | колбаса                        | магнит               | 300             |
|  2   | сосиски                        | пятерочка            | 220             |
+------+--------------------------------+----------------------+-----------------+
>>> select
Введите товар, информацию о котором хотите получить:
сосиски
+------+--------------------------------+----------------------+-----------------+
|  №   |             Товар              |       Магазин        | Стоимость товара |
+------+--------------------------------+----------------------+-----------------+
|  1   | сосиски                        | пятерочка            | 220             |
+------+--------------------------------+----------------------+-----------------+
```

Рисунок 7 – индивидуальное задание

```
H:\cross\git\nazarov\2.8>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

H:\cross\git\nazarov\2.8>git merge develop
Updating ce54c7b..392340d
Fast-forward
 .gitignore                                    | 150 +++++++++++++++-------
 ind.py                                        | 115 ++++++++++++++++++
 .../1.py"                                     |  18 +++
 .../2.py"                                     |  36 +++++
 .../3.py"                                     |  18 +++
 .../4.py"                                     |  33 +++++
 .../1.py"                                     | 123 ++++++++++++++++++++
 7 files changed, 446 insertions(+), 47 deletions(-)
 create mode 100644 ind.py
 create mode 100644 "\320\227\320\260\320\264\320\260\320\275\320\270\321\217/1.py"
 create mode 100644 "\320\227\320\260\320\264\320\260\320\275\320\270\321\217/2.py"
 create mode 100644 "\320\227\320\260\320\264\320\260\320\275\320\270\321\217/3.py"
 create mode 100644 "\320\227\320\260\320\264\320\260\320\275\320\270\321\217/4.py"
 create mode 100644 "\320\237\321\200\320\270\320\274\320\265\321\200\321\213/1.py"
```

Рисунок 8 – слияние веток

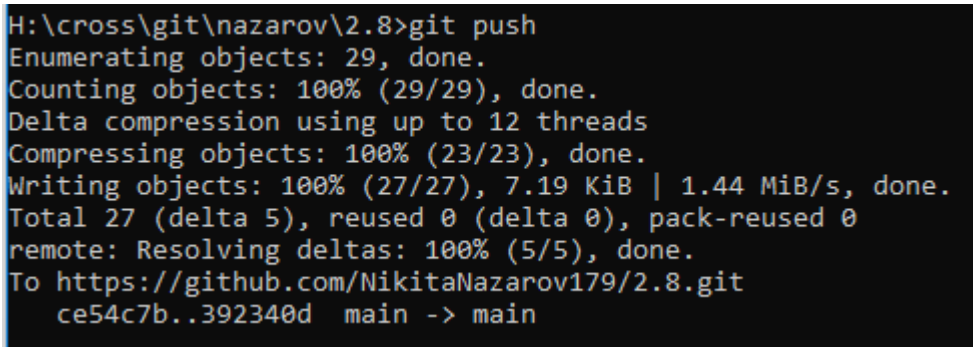

Рисунок 9 – пуш на удаленный репозиторий

Ответы на контрольные вопросы:

1. Каково назначение функций в языке программирования Python?

Оптимизировать код, сократить его.

Функция в программировании представляет собой обособленный участок кода, который можно вызывать, обратившись к нему по имени, которым он был назван. При вызове происходит выполнение команд тела функции.

Функции можно сравнить с небольшими программками, которые сами по себе, т. е. автономно, не исполняются, а встраиваются в обычную программу. Нередко их так и называют – подпрограммы.

Других ключевых отличий функций от программ нет. Функции также при необходимости могут получать и возвращать данные. Только обычно они их получают не с ввода (клавиатуры, файла и др.), а из вызывающей программы. Сюда же они возвращают результат своей работы.

2. Каково назначение операторов def и return?

В языке программирования Python функции определяются с помощью оператора def.

Рассмотрим код:

```
def countFood():
a = int(input())
b = int(input())
print("Всего", a+b, "шт.")
```

Это пример определения функции. Как и другие сложные инструкции вроде условного оператора и циклов функция состоит из заголовка и тела. Заголовок оканчивается двоеточием и переходом на новую строку. Тело имеет отступ.

Ключевое слово def сообщает интерпретатору, что перед ним определение функции. За def следует имя функции. Оно может быть любым, также как и всякий идентификатор, например, переменная. В программировании весьма желательно давать всему осмысленные имена. Так в данном случае функция названа "посчитатьЕду" в переводе на русский.

После имени функции ставятся скобки. В приведенном примере они пустые. Это значит, что функция не принимает никакие данные из вызывающей ее программы. Однако она могла бы их принимать, и тогда в скобках были бы указаны так называемые параметры.

После двоеточия следует тело, содержащее инструкции, которые выполняются при вызове функции. Следует различать определение функции и ее вызов. В программном коде они не рядом и не вместе. Можно определить функцию, но ни разу ее не вызвать. Нельзя вызвать функцию, которая не была определена. Определив функцию, но ни разу не вызвав ее, вы никогда не выполните ее тела.

Возврат значений из функции. Оператор return

Функции могут передавать какие-либо данные из своих тел в основную ветку программы.

Говорят, что функция возвращает значение. В большинстве языков программирования, в том числе Python, выход из функции и передача данных в то место, откуда она была вызвана, выполняется оператором return.

Если интерпретатор Питона, выполняя тело функции, встречает return, то он "забирает" значение, указанное после этой команды, и "уходит" из функции.

```
import    math
```

```
def cylinder():

r = float(input())
h = float(input())

# площадь боковой поверхности цилиндра:

side = 2 * math.pi * r * h

# площадь одного основания цилиндра:

circle = math.pi * r**2

# полная площадь цилиндра:

full = side + 2 * circle

return full

square = cylinder()

print(square)
```

Результат: 3 7
188.4

В данной программе в основную ветку из функции возвращается значение локальной переменной full. Не сама переменная, а ее значение, в данном случае – какое-либо число, полученное в результате вычисления площади цилиндра.

В основной ветке программы это значение присваивается глобальной переменной square. То есть выражение square = cylinder() выполняется так:

1. Вызывается функция cylinder() .

2. Из нее возвращается значение.

3. Это значение присваивается переменной square .

3. Каково назначение локальных и глобальных переменных при написании функций в Python?

В программировании особое внимание уделяется концепции о локальных и глобальных переменных, а также связанное с ними представление об

областях видимости. Соответственно, локальные переменные видны только в локальной области видимости, которой может выступать отдельно взятая функция. Глобальные переменные видны во всей программе. "Видны" – значит, известны, доступны. К ним можно обратиться по имени и получить связанное с ними значение.

К глобальной переменной можно обратиться из локальной области видимости. К локальной переменной нельзя обратиться из глобальной области видимости, потому что локальная переменная существует только в момент выполнения тела функции. При выходе из нее, локальные переменные исчезают. Компьютерная память, которая под них отводилась, освобождается. Когда функция будет снова вызвана, локальные переменные будут созданы заново.

4. Как вернуть несколько значений из функции Python?

Возврат нескольких значений:

В Питоне позволительно возвращать из функции несколько объектов, перечислив их через запятую после команды return пример: return side, full

5. Какие существуют способы передачи значений в функцию?

По ссылке и по значению

6. Как задать значение аргументов функции по умолчанию?

Аргументы по умолчанию в функциях Python – это те аргументы, которые принимают значения по умолчанию, если никакие явные значения не передаются этим аргументам из вызова функции. Давайте определим функцию с одним аргументом по умолчанию. **(В скобках после объявления функции)**

```
def find_square(integer1=2):
    result = integer1 * integer1
    return result
```

7. Каково назначение lambda-выражений в языке Python?

lambda – это выражение, а не инструкция. По этой причине ключевое слово lambda может появляться там, где синтаксис языка Python не позволяет

использовать инструкцию def , – внутри литералов или в вызовах функций, например.

8. Как осуществляется документирование кода согласно PEP257?

Если пояснение функции содержит одну строку, то достаточно двух кавычек с каждой стороны строки. Пример: ““Пояснение””. Если это многострочное пояснение, то необходимо три кавычки с каждой стороны. Пояснение находится в теле функции, сразу после еѐ появления.

9. В чем особенность однострочных и многострочных форм строк документации?

Однострочные строки используются для очевидных случаев и они должны действительно находится на одной строке.

Пример:

```
def kos_root():
    """Вернѐт путь к папке root KOS"""
    global _kos_root
    if _kos_root: return _kos_root
    ...
```

Многострочные документации состоят из сводной строки (summary line) имеющей такую же структуру, как и однострочный docstring, после которой следует пустая линия, а затем более сложное описание. «Summary line» может быть использована средствами автоматического документирования; поэтому так важно располагать еѐ на одной строке и после делать пропуск в одну линию. Сводная строка пишется сразу после открывающихся кавычек, но допускается сделать перенос и начать со следующей строки. [прим. после этого

предложения я был счастлив, ведь находились люди, которые упорно пыта- лись мне доказать, что делать перенос ни в коем случае нельзя :-)] При этом, весь docstring должен иметь такой же отступ, как и открывающие кавычки пер-вой строки (см. пример ниже).