

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»  
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

**Отчет о лабораторной работе №13 по дисциплине основы программной  
инженерии**

Выполнил:  
Выходцев Егор Дмитриевич,  
2 курс, группа ПИЖ-б-о-20-1,

Проверил:  
Доцент кафедры инфокоммуникаций,  
Воронкин Р.А.

Ставрополь, 2021 г.

## 1. Функции с переменным числом параметров в Python

Примеры из методических указаний

Позиционные и именованные аргументы

```
e1.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      def print_these(a, b, c):
5          print(a, "is stored in a")
6          print(b, "is stored in b")
7          print(c, "is stored in c")
8
9
10 ▶  if __name__ == "__main__":
11      print_these(1, 2, 3)
12
```

```
e1 x
C:\Users\Evil\PycharmProjects\LB13\
1 is stored in a
2 is stored in b
3 is stored in c

Process finished with exit code 0
```

```
e2.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      def print_these(a, b, c=None):
5          print(a, "is stored in a")
6          print(b, "is stored in b")
7          print(c, "is stored in c")
8
9
10 ▶  if __name__ == "__main__":
11      print_these(1, 2)
12
```

```
e2 x
C:\Users\Evil\PycharmProjects\LB13\venv
1 is stored in a
2 is stored in b
None is stored in c
Process finished with exit code 0
```

```
e3.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 def print_these(a=None, b=None, c=None):
5     print(a, "is stored in a")
6     print(b, "is stored in b")
7     print(c, "is stored in c")
8
9
10 ▶ if __name__ == "__main__":
11     print_these(c=3, a=1)
12
```

```
e3 x
C:\Users\Evil\PycharmProjects\LB13\venv
1 is stored in a
None is stored in b
3 is stored in c
Process finished with exit code 0
```

Оператор «звёздочка»

```
e4.py x
1 ▶ #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == "__main__":
5     a = [1, 2, 3]
6     b = [*a, 4, 5, 6]
7     print(b)
8
```

```
e4 x
C:\Users\Evil\PycharmProjects\LB13\ve
[1, 2, 3, 4, 5, 6]

Process finished with exit code 0
```

Как пользоваться \*args и \*\*kwargs ?

```
e5.py x
1 ▶ #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3
4   def print_scores(student, *scores):
5       print(f"Student Name: {student}")
6       for score in scores:
7           print(score)
8
9
10 ▶ if __name__ == "__main__":
11     print_scores("Jonathan", 100, 95, 88, 92, 99)
12
```

```
e5 x
C:\Users\Evil\PycharmProjects\LB13\venv
Student Name: Jonathan
100
95
88
92
99

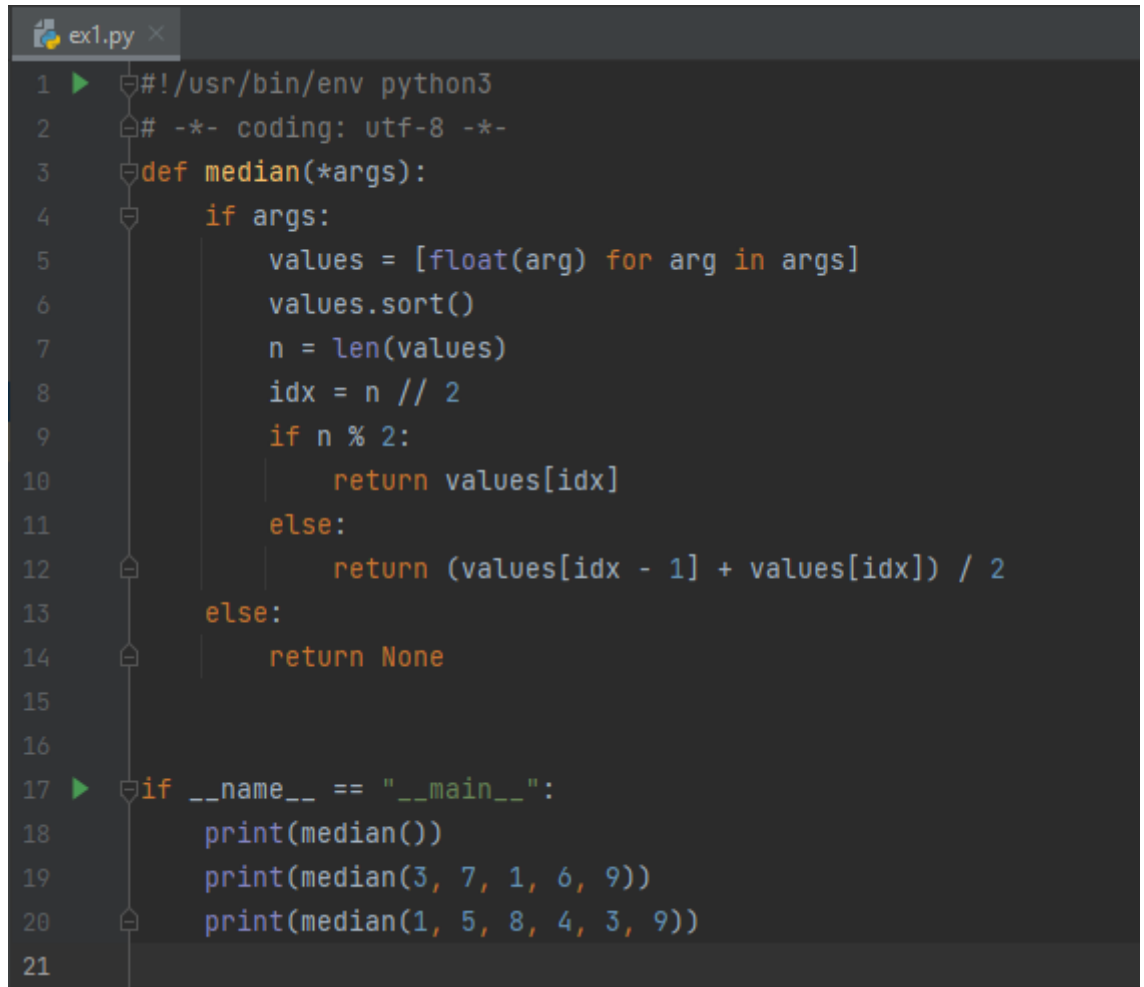
Process finished with exit code 0
```

```
e6.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 def print_pet_names(owner, **pets):
5     print(f"Owner Name: {owner}")
6     for pet, name in pets.items():
7         print(f"{pet}: {name}")
8
9
10 ▶ if __name__ == "__main__":
11     print_pet_names(
12         "Jonathan",
13         dog="Brock", fish=["Larry", "Curly", "Moe"],
14         turtle="Shelldon"
15     )
16
```

```
e6 x
C:\Users\Evil\PycharmProjects\LB13\venv
Owner Name: Jonathan
dog: Brock
fish: ['Larry', 'Curly', 'Moe']
turtle: Shelldon

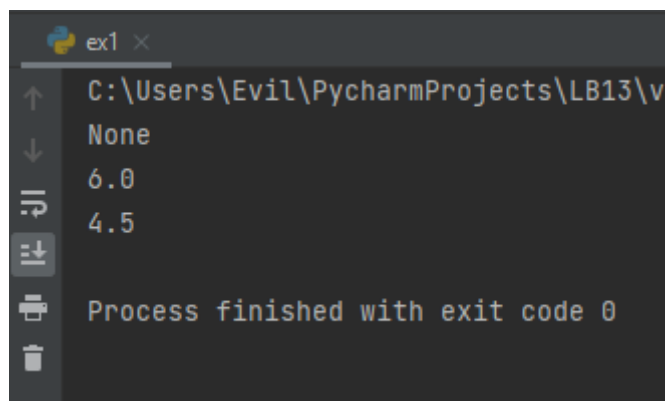
Process finished with exit code 0
```

### 1.1 Пример 1 (рис. 1,2)



```
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3      def median(*args):
4          if args:
5              values = [float(arg) for arg in args]
6              values.sort()
7              n = len(values)
8              idx = n // 2
9              if n % 2:
10                 return values[idx]
11             else:
12                 return (values[idx - 1] + values[idx]) / 2
13         else:
14             return None
15
16
17  ▶  if __name__ == "__main__":
18      print(median())
19      print(median(3, 7, 1, 6, 9))
20      print(median(1, 5, 8, 4, 3, 9))
21
```

Рисунок 1 – Код программы



```
ex1 ×
C:\Users\Evil\PycharmProjects\LB13\venv
None
6.0
4.5
Process finished with exit code 0
```

Рисунок 2 – Результат выполнения программы

### 1.2 Задача 1 (рис. 3-5).

```

pr1.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 def mid_geom(*args):
6     if args:
7         multiplication = 1
8         values = [float(arg) for arg in args]
9         n = len(values)
10        for elem in values:
11            multiplication *= elem
12        return multiplication ** (1 / n)
13    else:
14        return None
15
16
17 ▶ if __name__ == "__main__":
18     arguments = [float(i) for i in input("Enter the arguments: ").split()]
19     print(f"The geometric mean of these args is: {mid_geom(*arguments)}")
20

```

Рисунок 3 – Код программы

```

pr1 (1) x
C:\Users\Evil\PycharmProjects\LB13\venv\Scripts\python.exe
Enter the arguments: 1 4 7 10 5
The geometric mean of these args is: 4.258195602745703
Process finished with exit code 0

```

Рисунок 4 – Результат выполнения программы

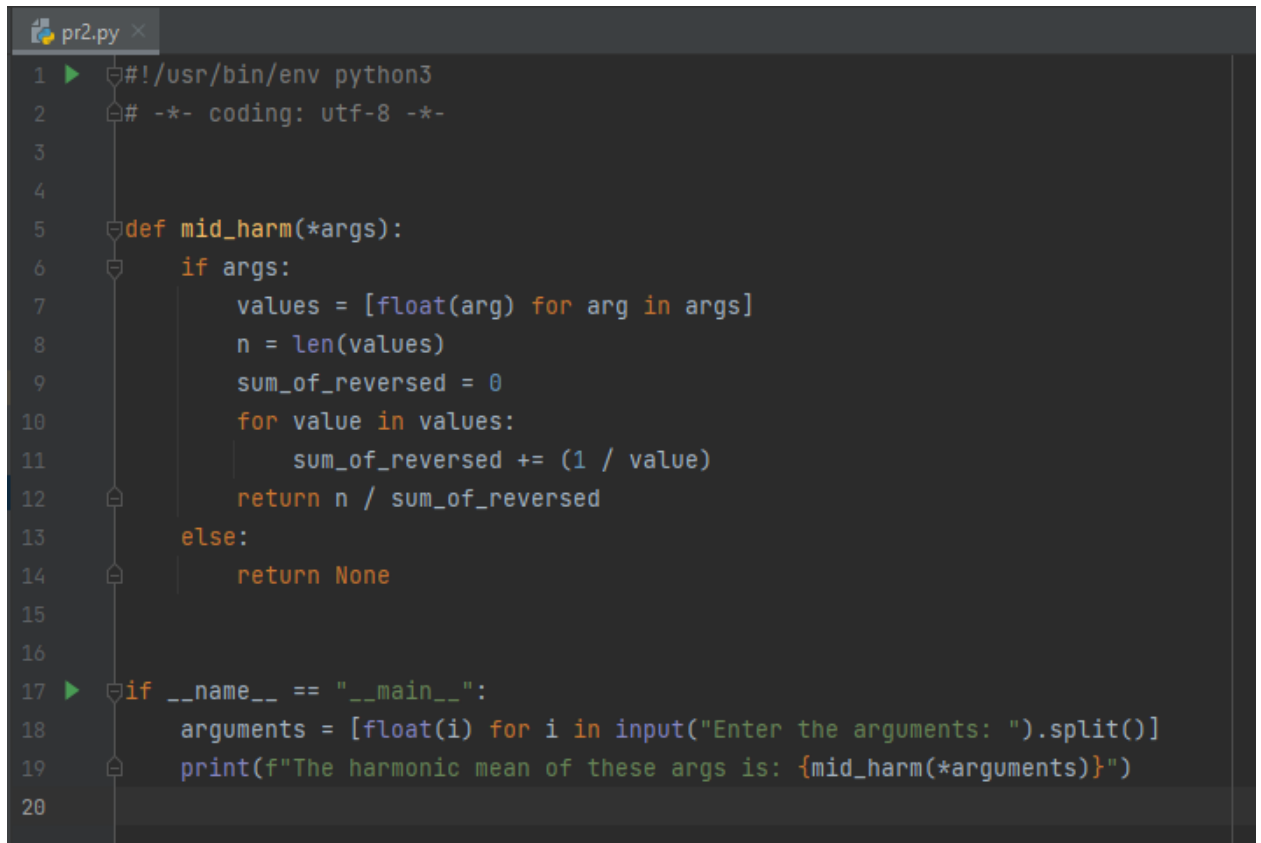
```

pr1 (1) x
C:\Users\Evil\PycharmProjects\LB13\venv\Scripts\python.exe
Enter the arguments:
The geometric mean of these args is: None
Process finished with exit code 0

```

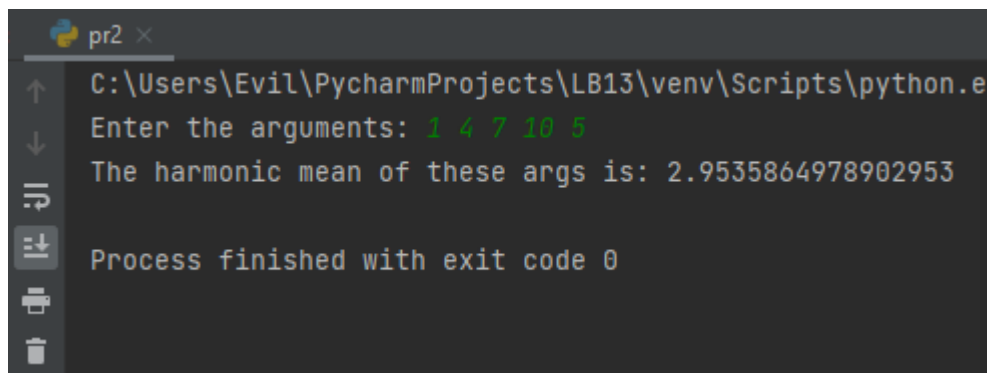
Рисунок 5 – Вывод программы при передаче пустого списка аргументов

### 1.3 Задача 2 (рис. 6-8).



```
pr2.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  def mid_harm(*args):
6      if args:
7          values = [float(arg) for arg in args]
8          n = len(values)
9          sum_of_reversed = 0
10         for value in values:
11             sum_of_reversed += (1 / value)
12         return n / sum_of_reversed
13     else:
14         return None
15
16
17  if __name__ == "__main__":
18      arguments = [float(i) for i in input("Enter the arguments: ").split()]
19      print(f"The harmonic mean of these args is: {mid_harm(*arguments)}")
20
```

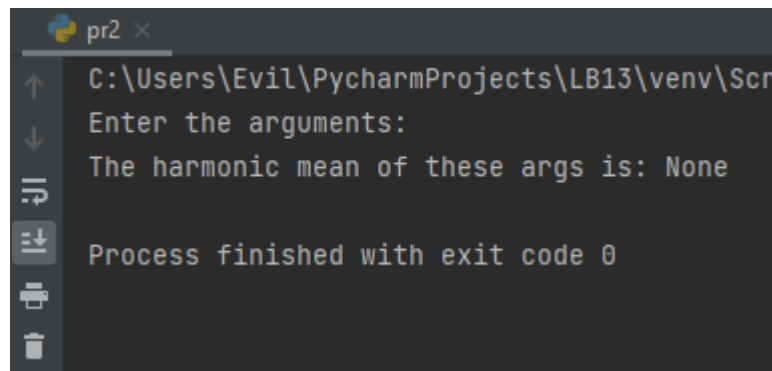
Рисунок 6 – Код программы



```
pr2 x
C:\Users\Evil\PycharmProjects\LB13\venv\Scripts\python.exe
Enter the arguments: 1 4 7 10 5
The harmonic mean of these args is: 2.9535864978902953
Process finished with exit code 0
```

Рисунок 7 – Результат выполнения программы



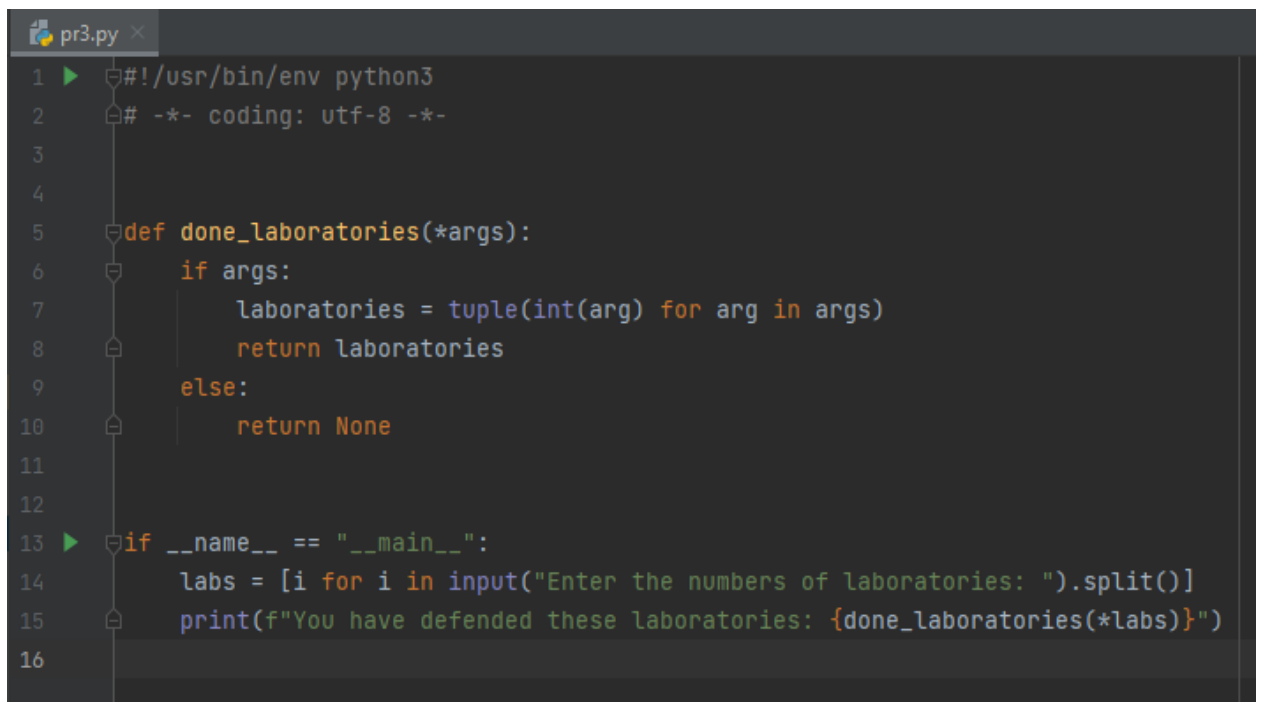


```
pr2 x
C:\Users\Evil\PycharmProjects\LB13\venv\Scripts\python.exe C:/Users/Evil/PycharmProjects/LB13/venv/Scripts/python.exe
Enter the arguments:
The harmonic mean of these args is: None
Process finished with exit code 0
```

Рисунок 8 – Вывод программы при передаче пустого списка аргументов

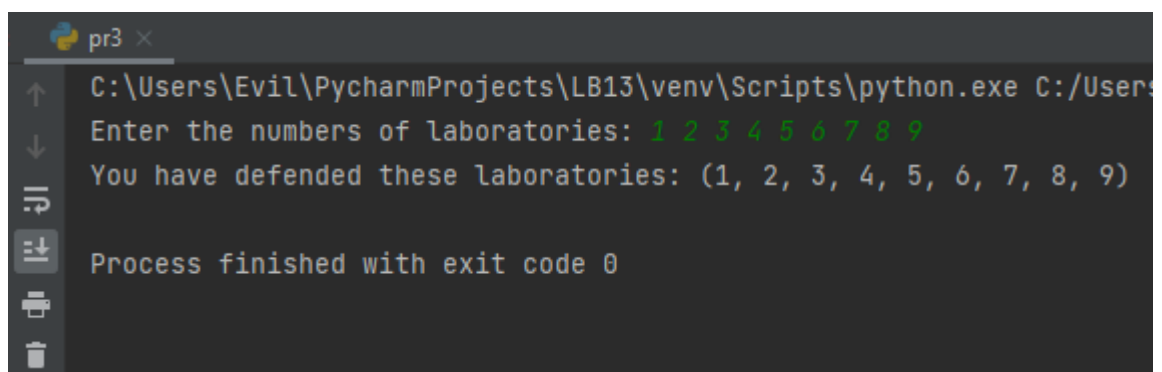
#### 1.4 Задача 3 (рис. 9-11).

Условие: в функцию подаётся количество защищённых работ, если таковых нет (список аргументов пуст), то функция возвращает None.



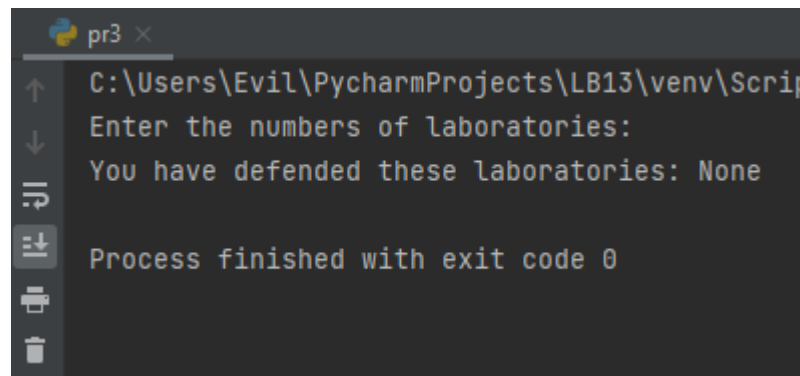
```
pr3.py x
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 def done_laboratories(*args):
6     if args:
7         laboratories = tuple(int(arg) for arg in args)
8         return laboratories
9     else:
10        return None
11
12
13 if __name__ == "__main__":
14     labs = [i for i in input("Enter the numbers of laboratories: ").split()]
15     print(f"You have defended these laboratories: {done_laboratories(*labs)}")
16
```

Рисунок 9 – Код программы



```
pr3 x
C:\Users\Evil\PycharmProjects\LB13\venv\Scripts\python.exe C:/Users/Evil/PycharmProjects/LB13/venv/Scripts/python.exe C:/Users/Evil/PycharmProjects/LB13/venv/Scripts/python.exe
Enter the numbers of laboratories: 1 2 3 4 5 6 7 8 9
You have defended these laboratories: (1, 2, 3, 4, 5, 6, 7, 8, 9)
Process finished with exit code 0
```

Рисунок 10 – Результат выполнения программы



```
pr3 x
C:\Users\Evil\PycharmProjects\LB13\venv\Scripts>
Enter the numbers of laboratories:
You have defended these laboratories: None
Process finished with exit code 0
```

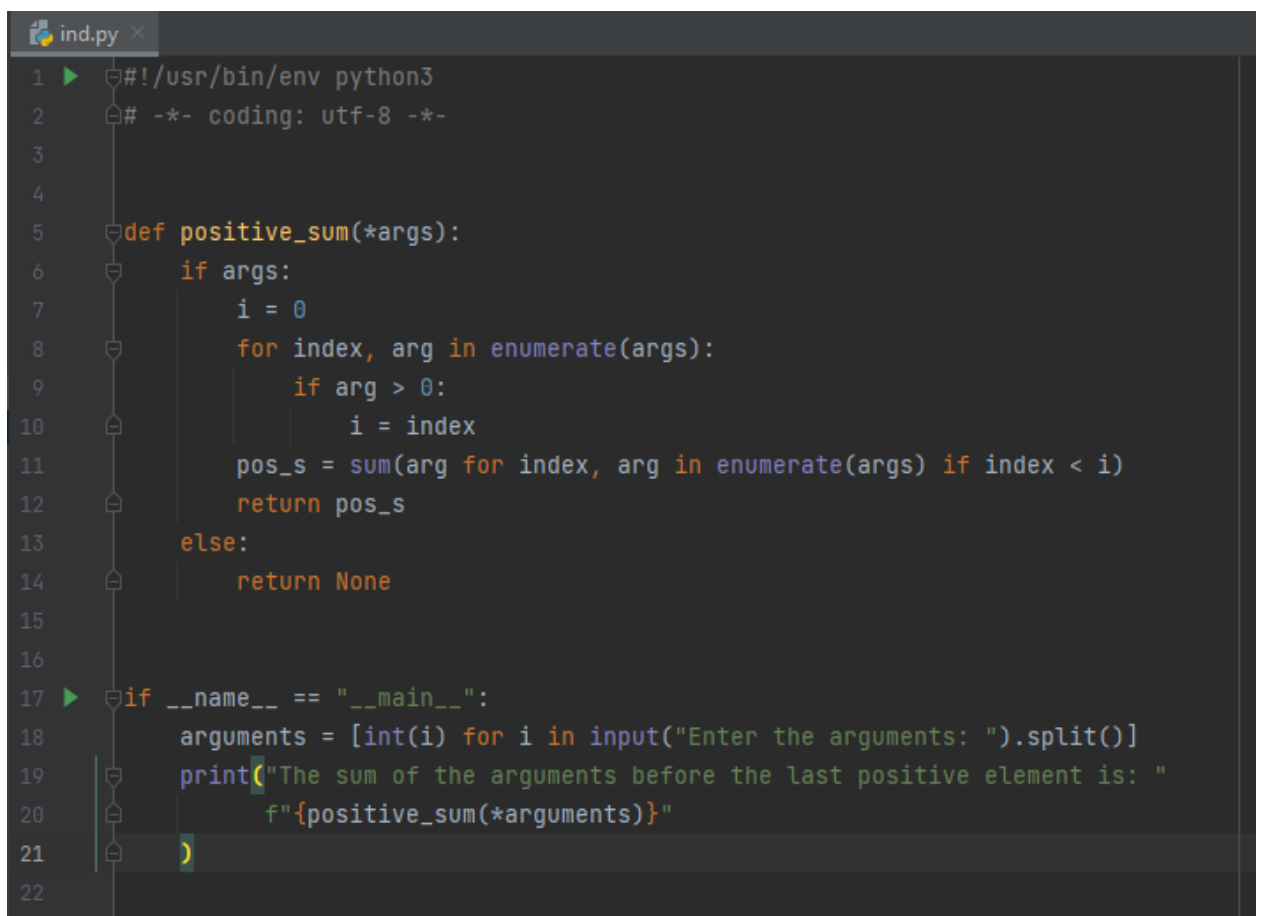
Рисунок 11 – Вывод программы при передаче пустого списка аргументов

## 1.5 Индивидуальное задание (рис. 12)

### Вариант 5

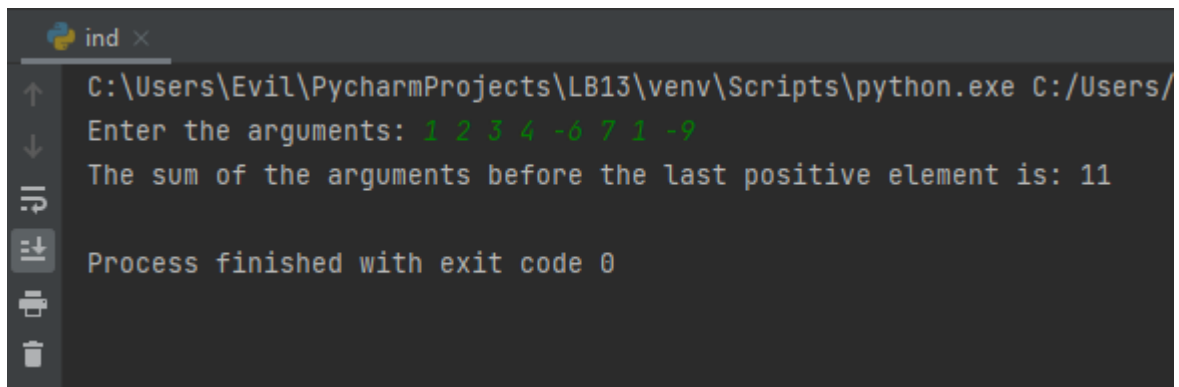
Условие: Напишите функцию, принимающую произвольное количество аргументов, и возвращающую требуемое значение. Если функции передается пустой список аргументов, то она должна возвращать значение `None`. В процессе решения не использовать преобразования конструкции `*args` в список или иную структуру данных:

5. Сумму аргументов, расположенных до последнего положительного аргумента.



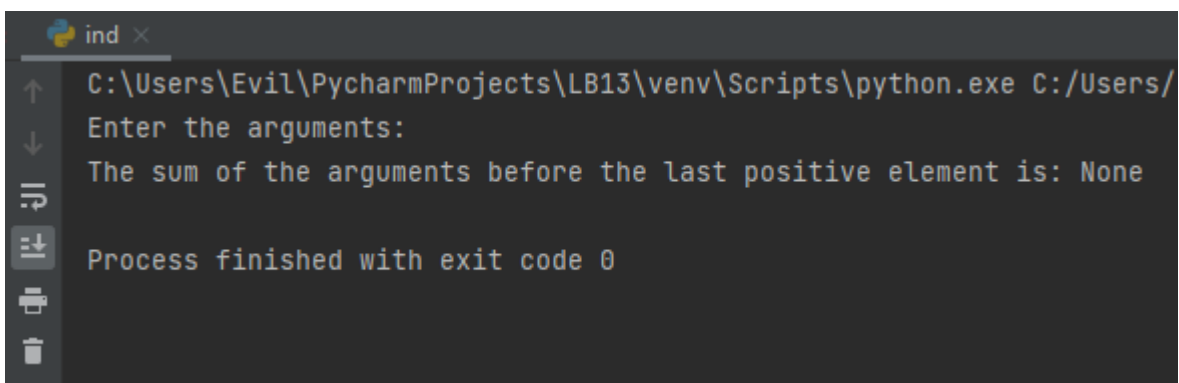
```
ind.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  def positive_sum(*args):
6      if args:
7          i = 0
8          for index, arg in enumerate(args):
9              if arg > 0:
10                 i = index
11             pos_s = sum(arg for index, arg in enumerate(args) if index < i)
12             return pos_s
13         else:
14             return None
15
16
17  if __name__ == "__main__":
18     arguments = [int(i) for i in input("Enter the arguments: ").split()]
19     print("The sum of the arguments before the last positive element is: "
20           f"{positive_sum(*arguments)}")
21
22
```

Рисунок 12 – Код программы



```
ind x
C:\Users\Evil\PycharmProjects\LB13\venv\Scripts\python.exe C:/Users/
Enter the arguments: 1 2 3 4 -6 7 1 -9
The sum of the arguments before the last positive element is: 11
Process finished with exit code 0
```

Рисунок 13 – Результат выполнения программы



```
ind x
C:\Users\Evil\PycharmProjects\LB13\venv\Scripts\python.exe C:/Users/
Enter the arguments:
The sum of the arguments before the last positive element is: None
Process finished with exit code 0
```

Рисунок 14 – Вывод программы при передаче пустого списка аргументов

## 2. Ответы на контрольные вопросы

### 1. Какие аргументы называются позиционными в Python?

При вызове функций значения в такие аргументы подставляются согласно позиции имён аргументов в определении функции.

### 2. Какие аргументы называются именованными в Python?

Это аргументы, передаваемые в функцию вместе с именем.

### 3. Для чего используется оператор \* ?

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

Пример:

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

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

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

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

`*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. Благодаря использованию `*` мы создаём список позиционных аргументов на основе того, что было передано функции при вызове. Благодаря символам `**` создаётся словарь, в котором содержатся именованные аргументы, переданные функции при её вызове.