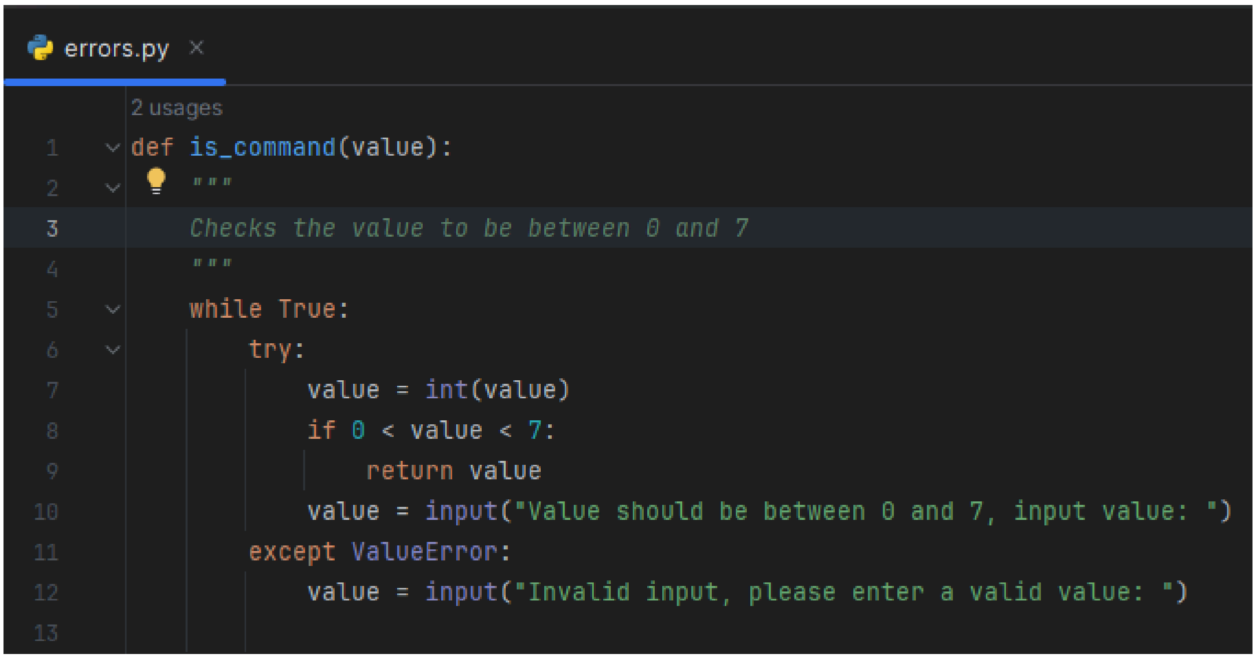
Лабораторная работа №3

Тема: Стандартные типы данных, коллекции, функции, модули.

Цель: освоить базовый синтаксис языка Python, приобрести навыки работы со стандартными типами данных, коллекциями, функциями, модулями и закрепить их на примере разработки интерактивных приложений.

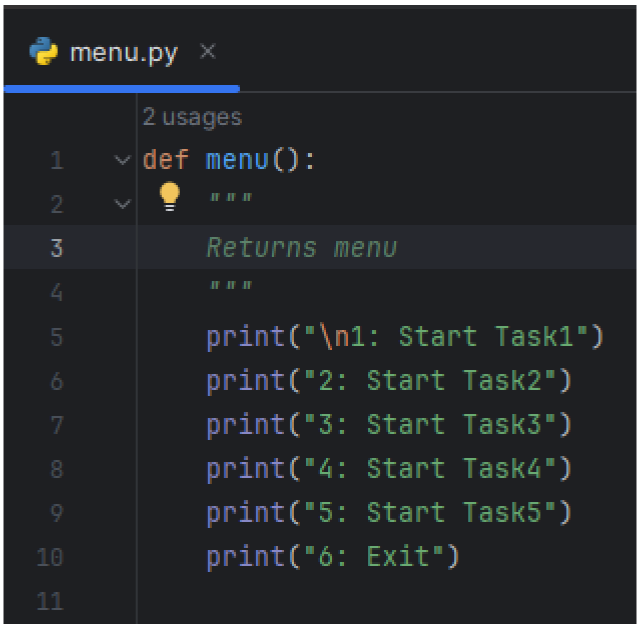
Выполнил: Трошко Александр Олегович

Группа: 253503



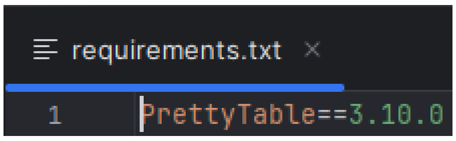
errors.py

```python
def is_command(value):
    """
    Checks the value to be between 0 and 7
    """
    while True:
        try:
            value = int(value)
            if 0 < value < 7:
                return value
            value = input("Value should be between 0 and 7, input value: ")
        except ValueError:
            value = input("Invalid input, please enter a valid value: ")
```



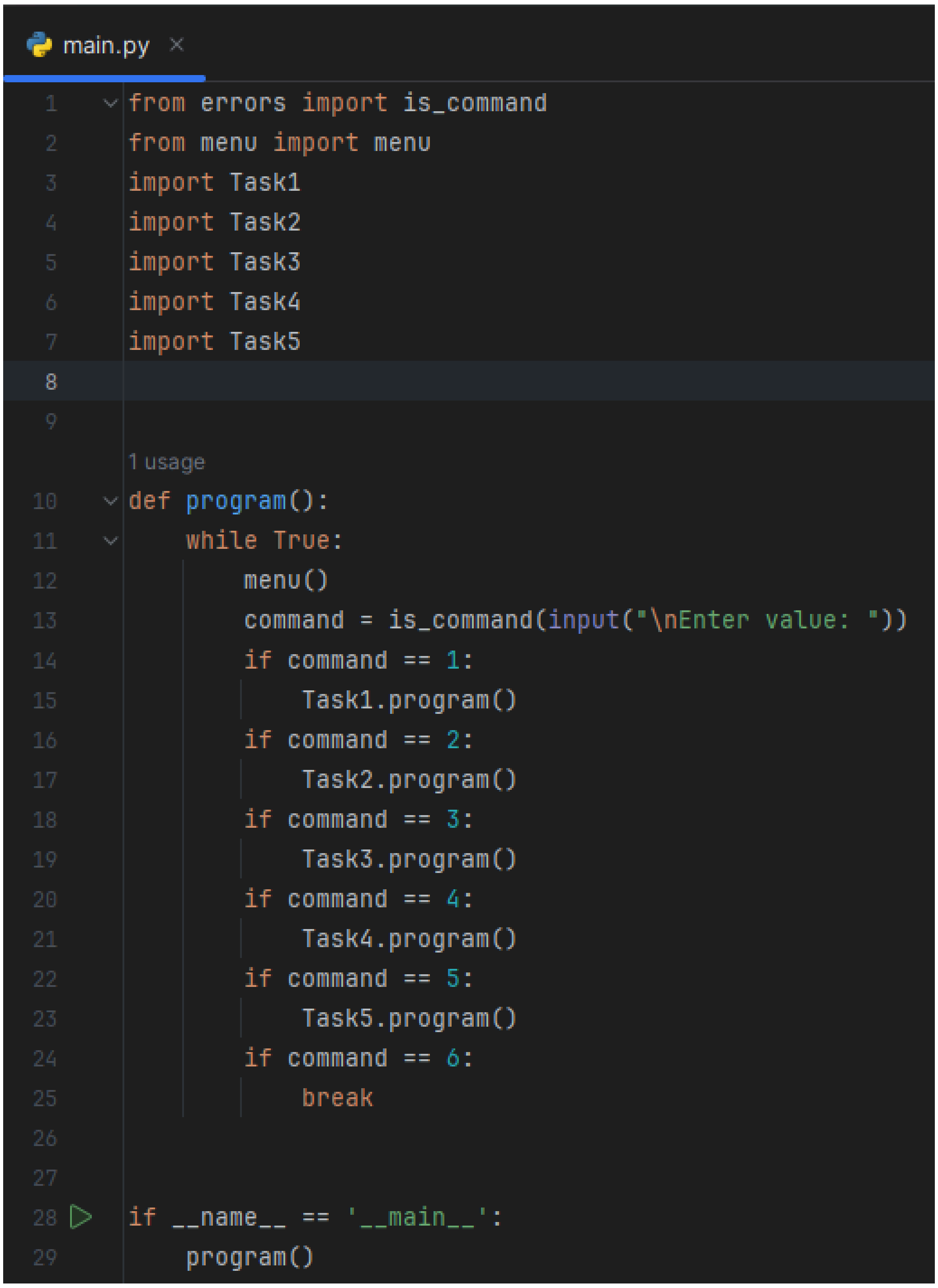
menu.py

```python
def menu():
    """
    Returns menu
    """
    print("\n1: Start Task1")
    print("2: Start Task2")
    print("3: Start Task3")
    print("4: Start Task4")
    print("5: Start Task5")
    print("6: Exit")
```



requirements.txt

```
PrettyTable==3.10.0
```


main.py

```python
from errors import is_command
from menu import menu
import Task1
import Task2
import Task3
import Task4
import Task5


1 usage
def program():
    while True:
        menu()
        command = is_command(input("\nEnter value: "))
        if command == 1:
            Task1.program()
        if command == 2:
            Task2.program()
        if command == 3:
            Task3.program()
        if command == 4:
            Task4.program()
        if command == 5:
            Task5.program()
        if command == 6:
            break


if __name__ == '__main__':
    program()
```

Задание 1. В соответствии с заданием своего варианта составить программу для вычисления значения функции с помощью разложения функции в степенной ряд. Задать точность вычислений eps.

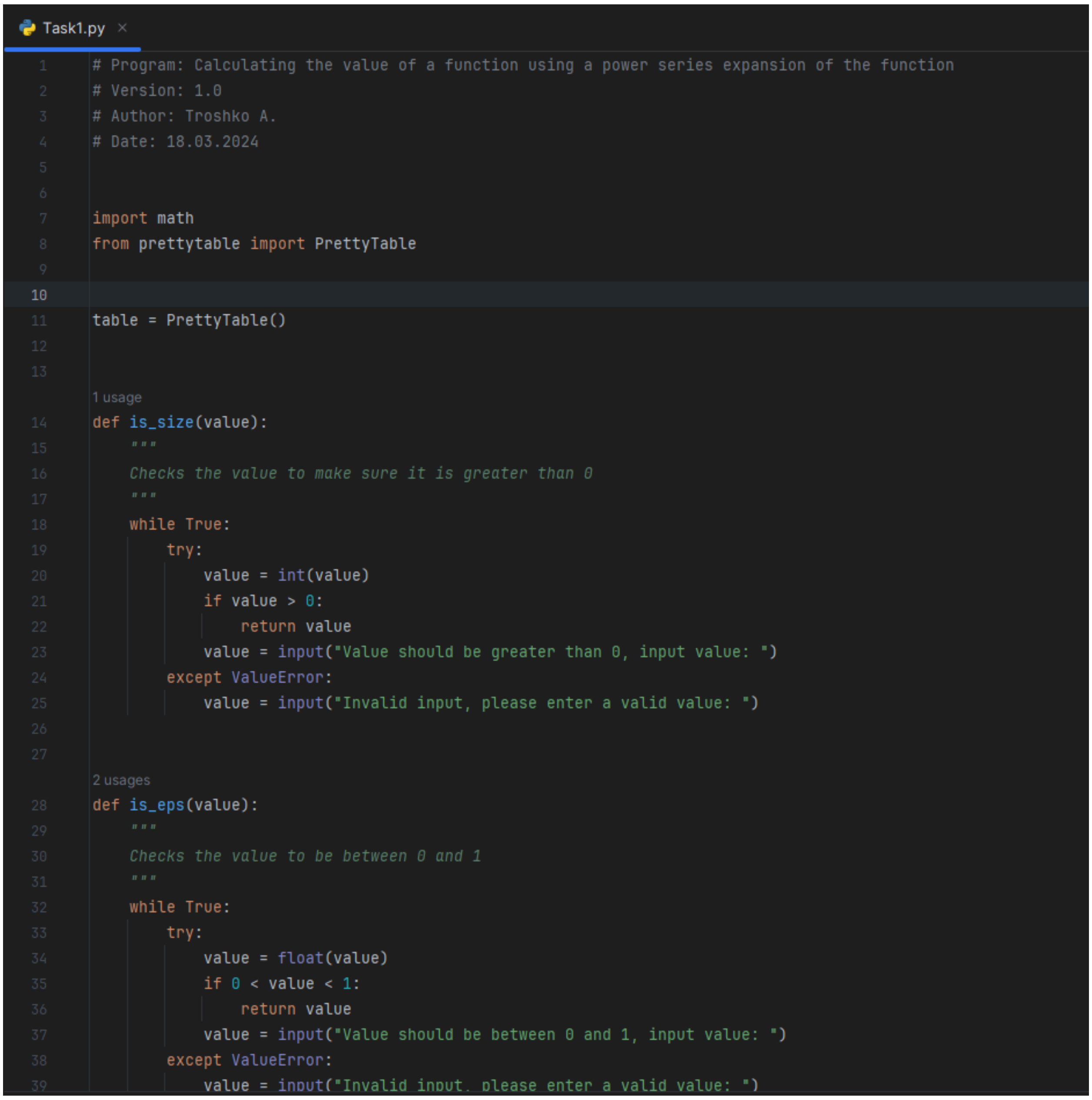
Предусмотреть максимальное количество итераций, равное 500.

Вывести количество членов ряда, необходимых для достижения указанной точности вычислений. Результат получить в виде:

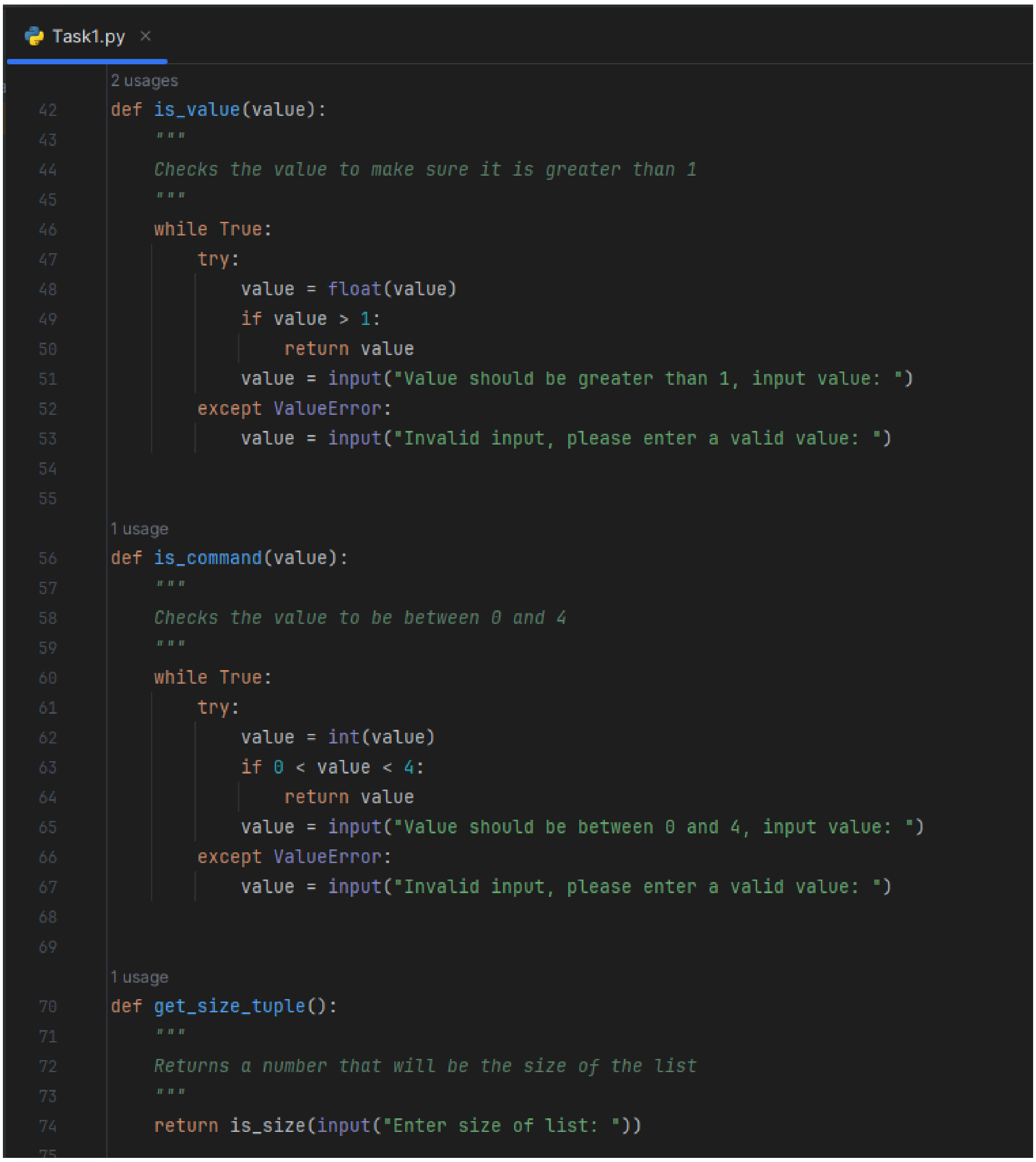
x	n	$F(x)$	$Math\ F(x)$	eps

Здесь x – значение аргумента, F(x) – значение функции, n – количество просуммированных членов ряда, Math F(x) – значение функции, вычисленное с помощью модуля math.

Условие
$\ln\frac{x+1}{x-1} = 2\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}} = 2(\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\ldots), \lvert x\rvert > 1$



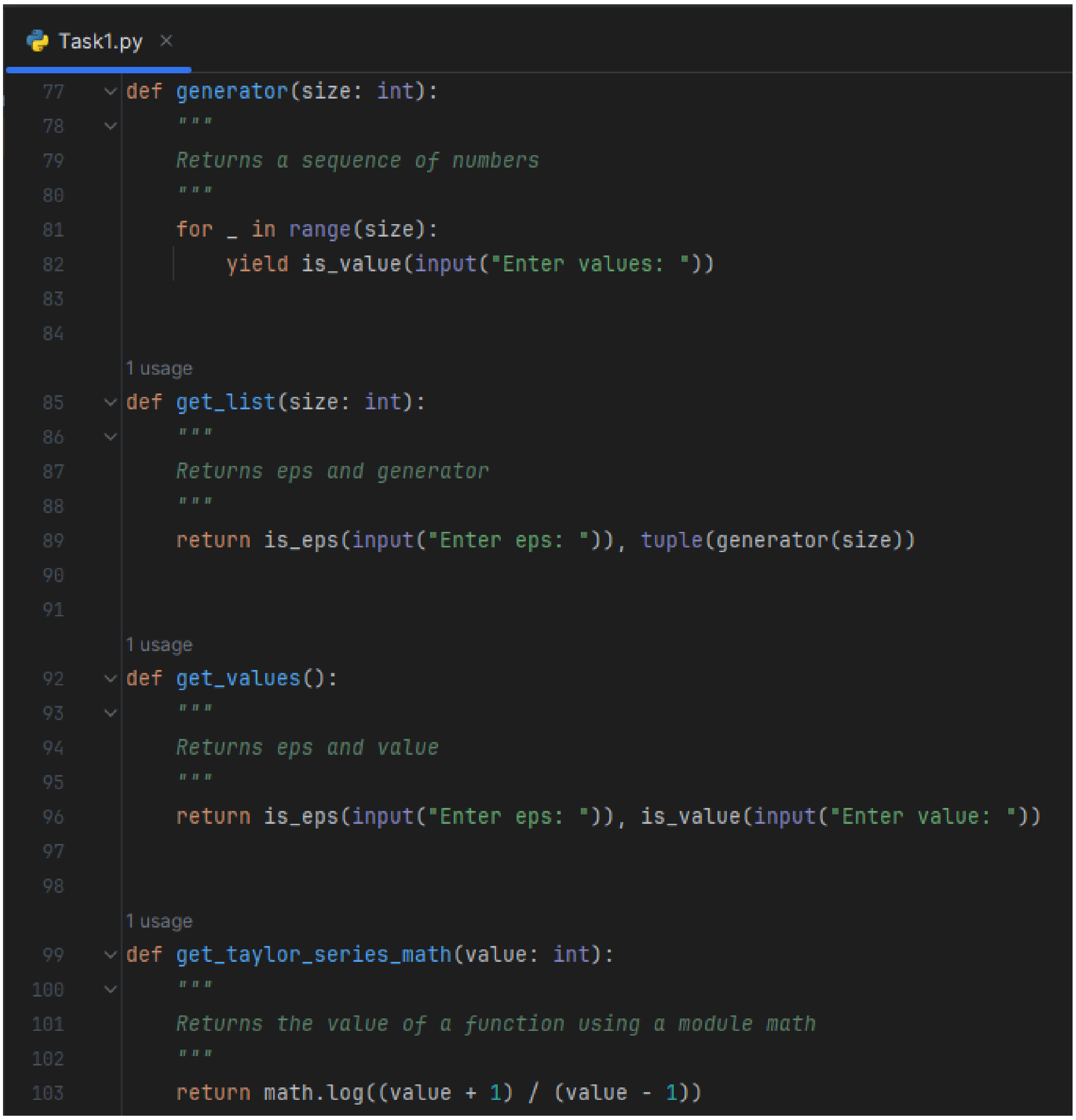
```python
# Program: Calculating the value of a function using a power series expansion of the function
# Version: 1.0
# Author: Troshko A.
# Date: 18.03.2024


import math
from prettytable import PrettyTable


table = PrettyTable()


1 usage
def is_size(value):
    """
    Checks the value to make sure it is greater than 0
    """
    while True:
        try:
            value = int(value)
            if value > 0:
                return value
            value = input("Value should be greater than 0, input value: ")
        except ValueError:
            value = input("Invalid input, please enter a valid value: ")


2 usages
def is_eps(value):
    """
    Checks the value to be between 0 and 1
    """
    while True:
        try:
            value = float(value)
            if 0 < value < 1:
                return value
            value = input("Value should be between 0 and 1, input value: ")
        except ValueError:
            value = input("Invalid input, please enter a valid value: ")
```



```
def is_value(value):
    """
    Checks the value to make sure it is greater than 1
    """
    while True:
        try:
            value = float(value)
            if value > 1:
                return value
            value = input("Value should be greater than 1, input value: ")
        except ValueError:
            value = input("Invalid input, please enter a valid value: ")


```

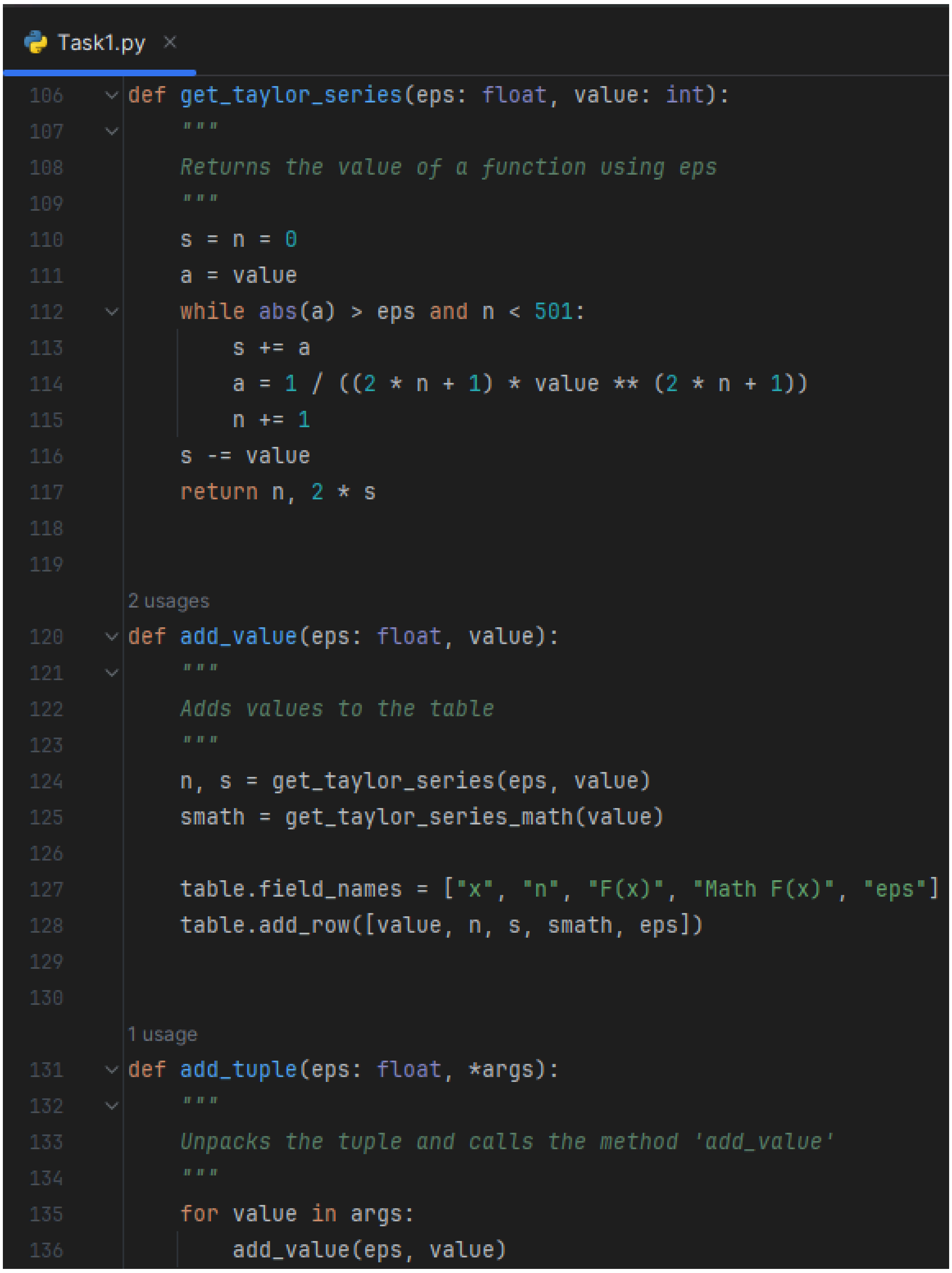
```
def is_command(value):
    """
    Checks the value to be between 0 and 4
    """
    while True:
        try:
            value = int(value)
            if 0 < value < 4:
                return value
            value = input("Value should be between 0 and 4, input value: ")
        except ValueError:
            value = input("Invalid input, please enter a valid value: ")


```

```
def get_size_tuple():
    """
    Returns a number that will be the size of the list
    """
    return is_size(input("Enter size of list: "))
```



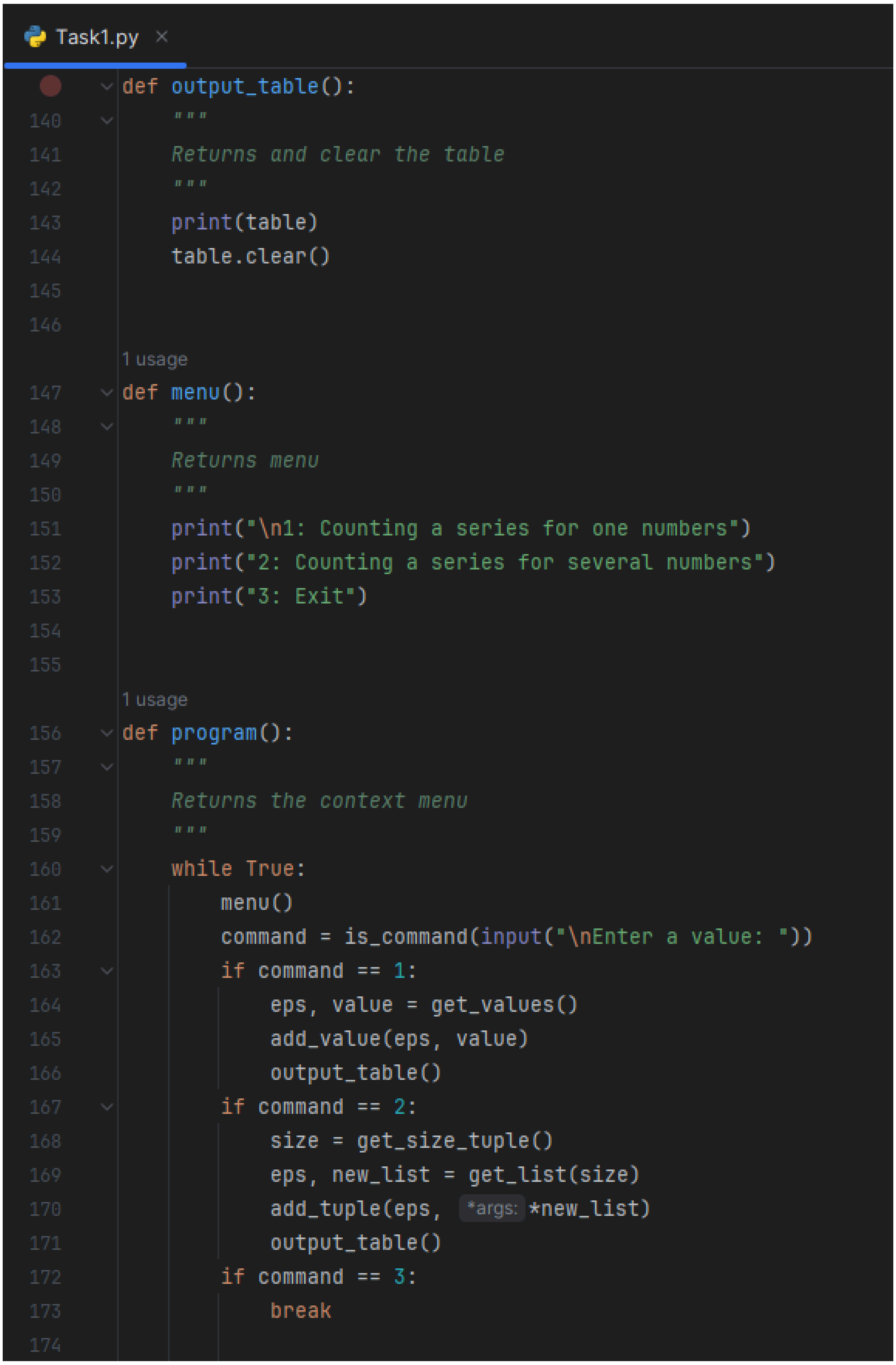
Task1.py

```python
def generator(size: int):
    """
    Returns a sequence of numbers
    """
    for _ in range(size):
        yield is_value(input("Enter values: "))


1 usage
def get_list(size: int):
    """
    Returns eps and generator
    """
    return is_eps(input("Enter eps: ")), tuple(generator(size))


1 usage
def get_values():
    """
    Returns eps and value
    """
    return is_eps(input("Enter eps: ")), is_value(input("Enter value: "))


1 usage
def get_taylor_series_math(value: int):
    """
    Returns the value of a function using a module math
    """
    return math.log((value + 1) / (value - 1))
```



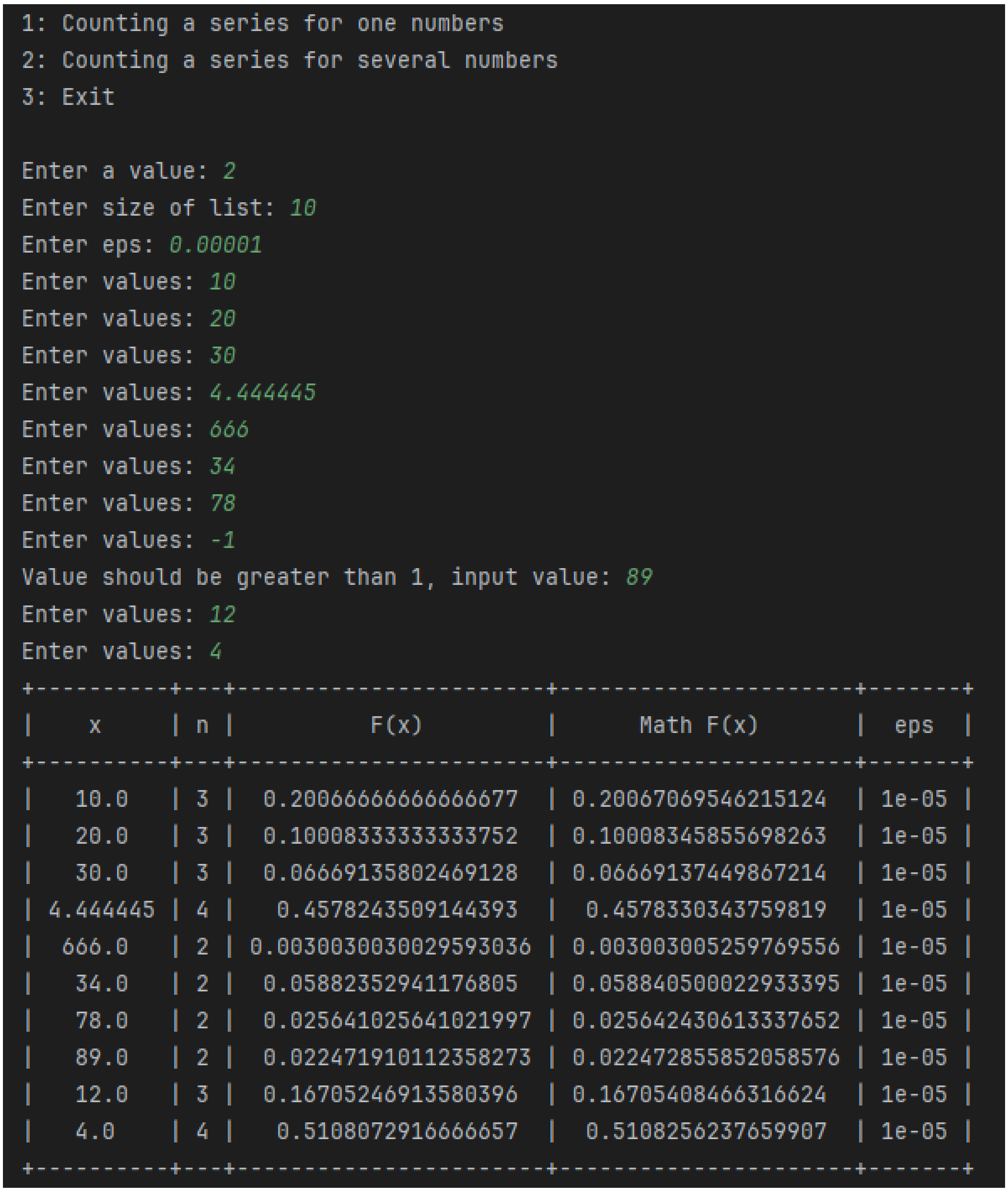
Task1.py

```python
def get_taylor_series(eps: float, value: int):
    """
    Returns the value of a function using eps
    """
    s = n = 0
    a = value
    while abs(a) > eps and n < 501:
        s += a
        a = 1 / ((2 * n + 1) * value ** (2 * n + 1))
        n += 1
    s -= value
    return n, 2 * s


2 usages
def add_value(eps: float, value):
    """
    Adds values to the table
    """
    n, s = get_taylor_series(eps, value)
    smath = get_taylor_series_math(value)

    table.field_names = ["x", "n", "F(x)", "Math F(x)", "eps"]
    table.add_row([value, n, s, smath, eps])


1 usage
def add_tuple(eps: float, *args):
    """
    Unpacks the tuple and calls the method 'add_value'
    """
    for value in args:
        add_value(eps, value)
```



Task1.py

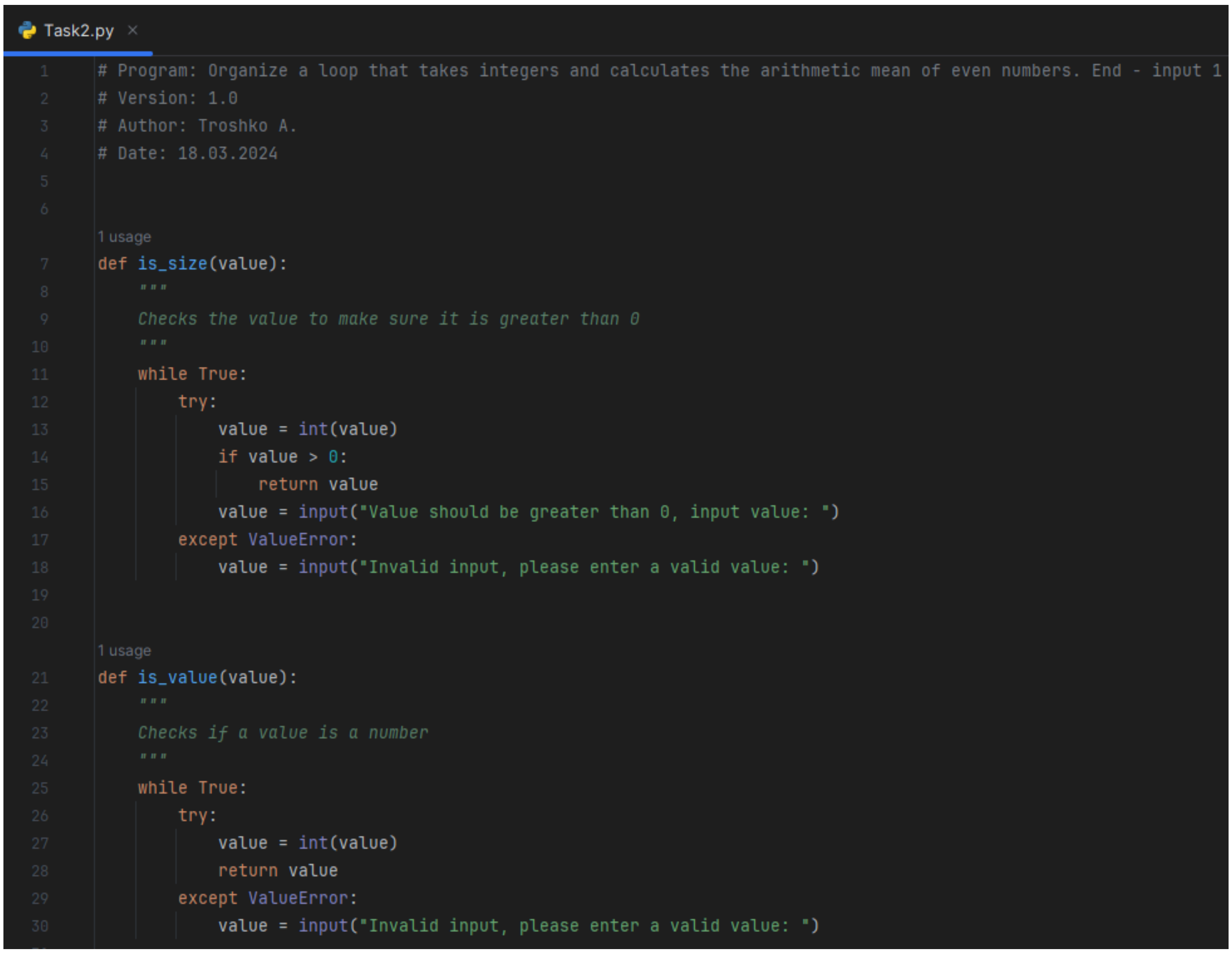
```python
def output_table():
    """
    Returns and clear the table
    """
    print(table)
    table.clear()


1 usage
def menu():
    """
    Returns menu
    """
    print("\n1: Counting a series for one numbers")
    print("2: Counting a series for several numbers")
    print("3: Exit")


1 usage
def program():
    """
    Returns the context menu
    """
    while True:
        menu()
        command = is_command(input("\nEnter a value: "))
        if command == 1:
            eps, value = get_values()
            add_value(eps, value)
            output_table()
        if command == 2:
            size = get_size_tuple()
            eps, new_list = get_list(size)
            add_tuple(eps, *args: *new_list)
            output_table()
        if command == 3:
            break

```



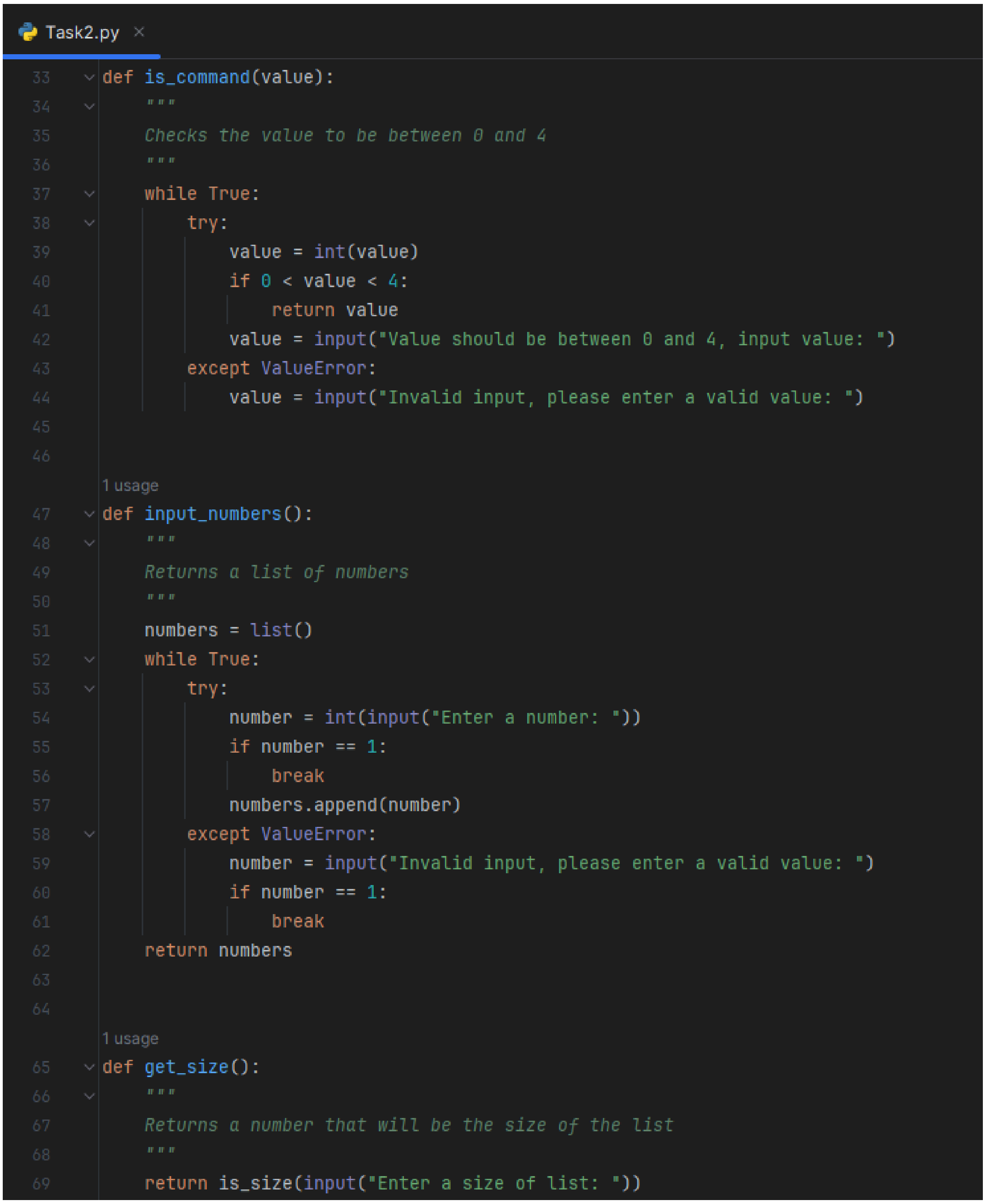
```
1: Counting a series for one numbers
2: Counting a series for several numbers
3: Exit

Enter a value: 2
Enter size of list: 10
Enter eps: 0.00001
Enter values: 10
Enter values: 20
Enter values: 30
Enter values: 4.444445
Enter values: 666
Enter values: 34
Enter values: 78
Enter values: -1
Value should be greater than 1, input value: 89
Enter values: 12
Enter values: 4
+----------+---+-----------------------+----------------------+-------+
|    x     | n |          F(x)         |       Math F(x)      |  eps  |
+----------+---+-----------------------+----------------------+-------+
|   10.0   | 3 |  0.20066666666666677  | 0.20067069546215124  | 1e-05 |
|   20.0   | 3 |  0.10008333333333752  | 0.10008345855698263  | 1e-05 |
|   30.0   | 3 |  0.06669135802469128  | 0.06669137449867214  | 1e-05 |
| 4.444445 | 4 |   0.4578243509144393  |  0.4578330343759819  | 1e-05 |
|  666.0   | 2 | 0.0030030030029593036 | 0.003003005259769556 | 1e-05 |
|   34.0   | 2 |  0.05882352941176805  | 0.05884050002293395  | 1e-05 |
|   78.0   | 2 |  0.025641025641021997 | 0.025642430613337652 | 1e-05 |
|   89.0   | 2 |  0.022471910112358273 | 0.022472855852058576 | 1e-05 |
|   12.0   | 3 |  0.16705246913580396  | 0.16705408466316624  | 1e-05 |
|   4.0    | 4 |   0.5108072916666657  |  0.5108256237659907  | 1e-05 |
+----------+---+-----------------------+----------------------+-------+
```

Задание 2. В соответствии с заданием своего варианта составить программу для нахождения суммы последовательности чисел.

Условие
Организовать цикл, который принимает целые числа и вычисляет среднее арифметическое четных чисел. Окончание – ввод 1

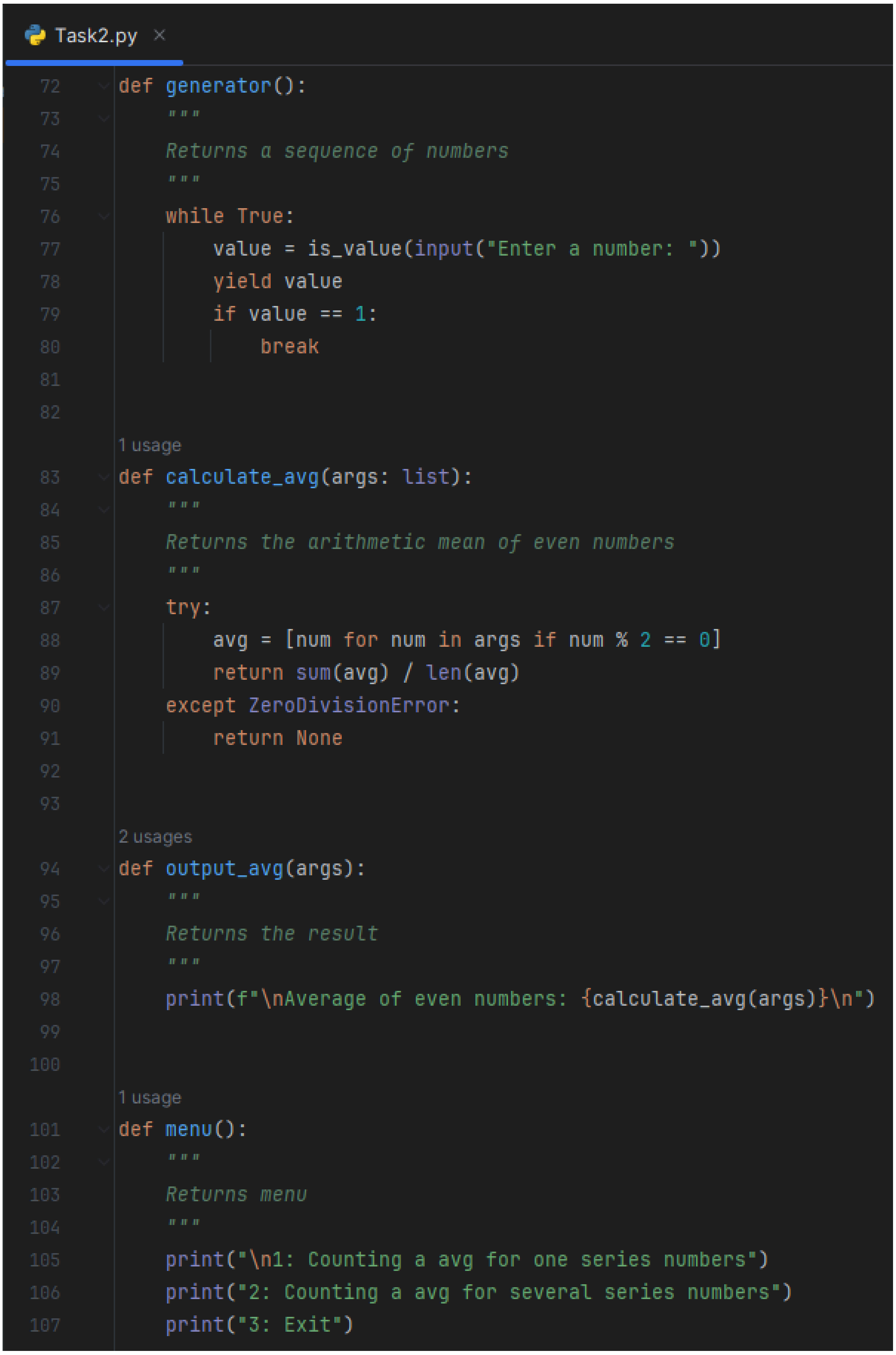


```python
# Program: Organize a loop that takes integers and calculates the arithmetic mean of even numbers. End - input 1
# Version: 1.0
# Author: Troshko A.
# Date: 18.03.2024


1 usage
def is_size(value):
    """
    Checks the value to make sure it is greater than 0
    """
    while True:
        try:
            value = int(value)
            if value > 0:
                return value
            value = input("Value should be greater than 0, input value: ")
        except ValueError:
            value = input("Invalid input, please enter a valid value: ")


1 usage
def is_value(value):
    """
    Checks if a value is a number
    """
    while True:
        try:
            value = int(value)
            return value
        except ValueError:
            value = input("Invalid input, please enter a valid value: ")
```

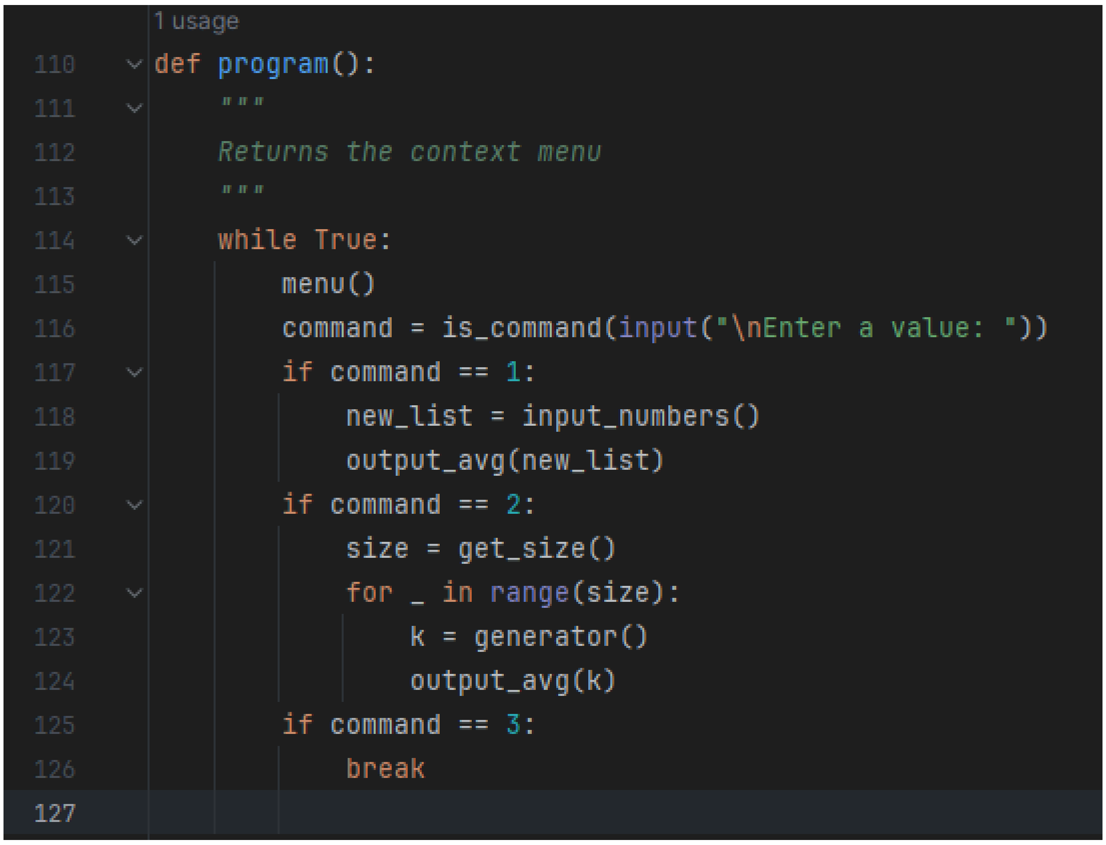


Task2.py

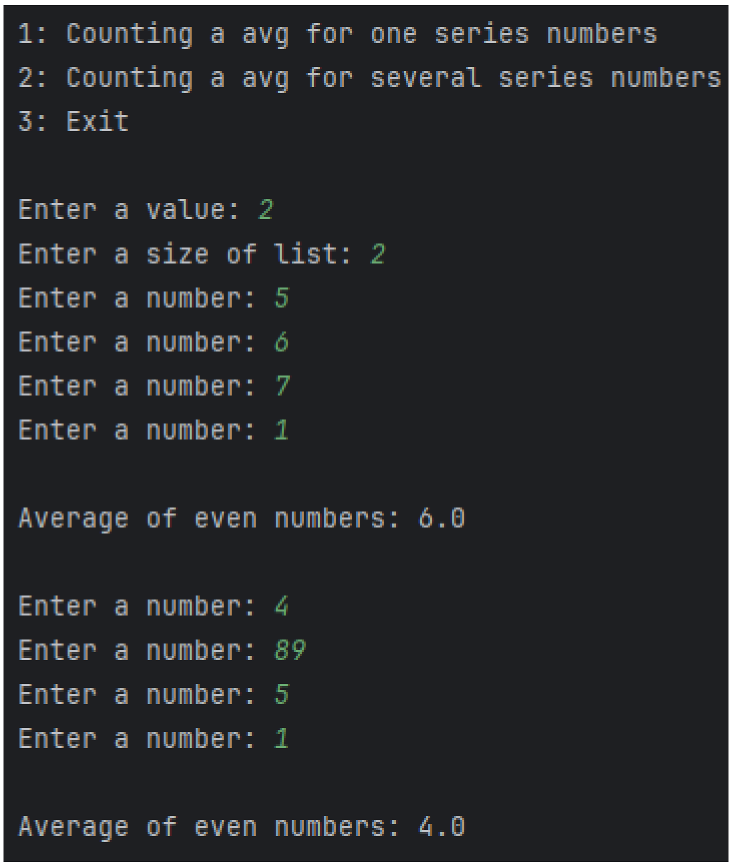
```python
def is_command(value):
    """
    Checks the value to be between 0 and 4
    """
    while True:
        try:
            value = int(value)
            if 0 < value < 4:
                return value
            value = input("Value should be between 0 and 4, input value: ")
        except ValueError:
            value = input("Invalid input, please enter a valid value: ")


1 usage
def input_numbers():
    """
    Returns a list of numbers
    """
    numbers = list()
    while True:
        try:
            number = int(input("Enter a number: "))
            if number == 1:
                break
            numbers.append(number)
        except ValueError:
            number = input("Invalid input, please enter a valid value: ")
            if number == 1:
                break
    return numbers


1 usage
def get_size():
    """
    Returns a number that will be the size of the list
    """
    return is_size(input("Enter a size of list: "))
```


Task2.py

```python
def generator():
    """
    Returns a sequence of numbers
    """
    while True:
        value = is_value(input("Enter a number: "))
        yield value
        if value == 1:
            break


1 usage
def calculate_avg(args: list):
    """
    Returns the arithmetic mean of even numbers
    """
    try:
        avg = [num for num in args if num % 2 == 0]
        return sum(avg) / len(avg)
    except ZeroDivisionError:
        return None


2 usages
def output_avg(args):
    """
    Returns the result
    """
    print(f"\nAverage of even numbers: {calculate_avg(args)}\n")


1 usage
def menu():
    """
    Returns menu
    """
    print("\n1: Counting a avg for one series numbers")
    print("2: Counting a avg for several series numbers")
    print("3: Exit")
```



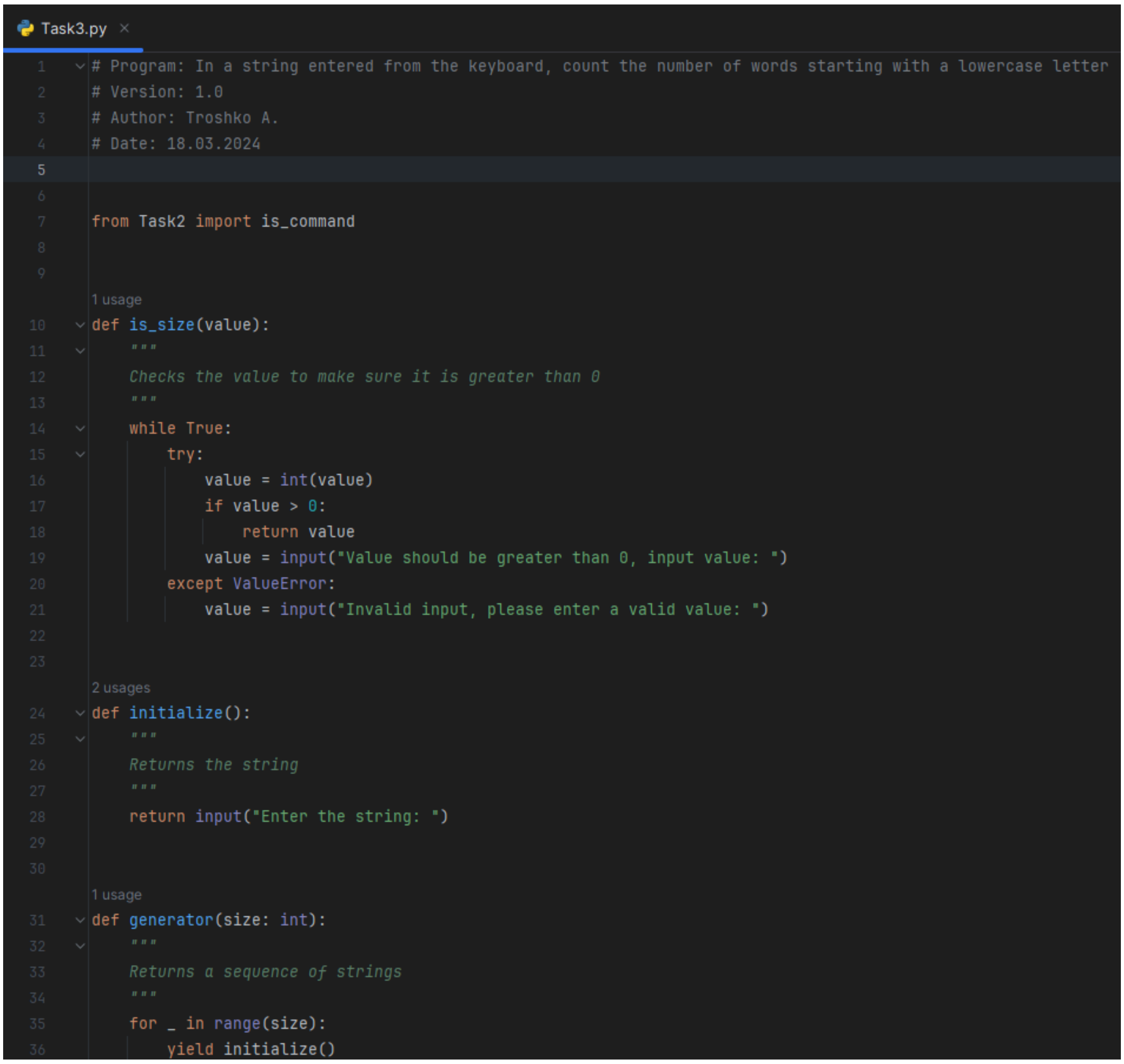
```
1 usage
def program():
    """
    Returns the context menu
    """
    while True:
        menu()
        command = is_command(input("\nEnter a value: "))
        if command == 1:
            new_list = input_numbers()
            output_avg(new_list)
        if command == 2:
            size = get_size()
            for _ in range(size):
                k = generator()
                output_avg(k)
        if command == 3:
            break

```



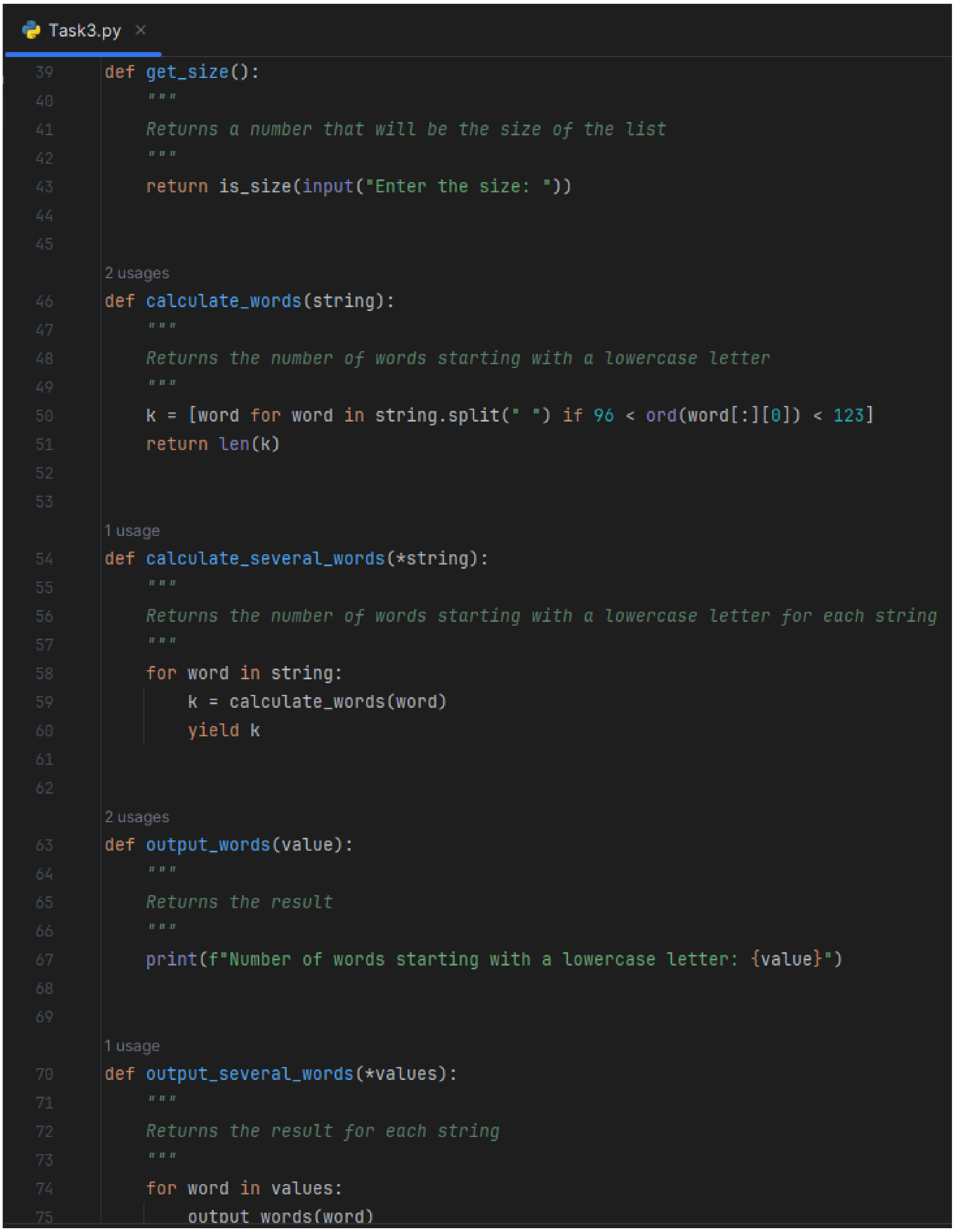
```
1: Counting a avg for one series numbers
2: Counting a avg for several series numbers
3: Exit

Enter a value: 2
Enter a size of list: 2
Enter a number: 5
Enter a number: 6
Enter a number: 7
Enter a number: 1

Average of even numbers: 6.0

Enter a number: 4
Enter a number: 89
Enter a number: 5
Enter a number: 1

Average of even numbers: 4.0
```

Задание 3. Не использовать регулярные выражения. В соответствии с заданием своего варианта составить программу для анализа текста, вводимого с клавиатуры.

Условие
В строке, вводимой с клавиатуры, подсчитать количество слов, начинающихся со строчной буквы

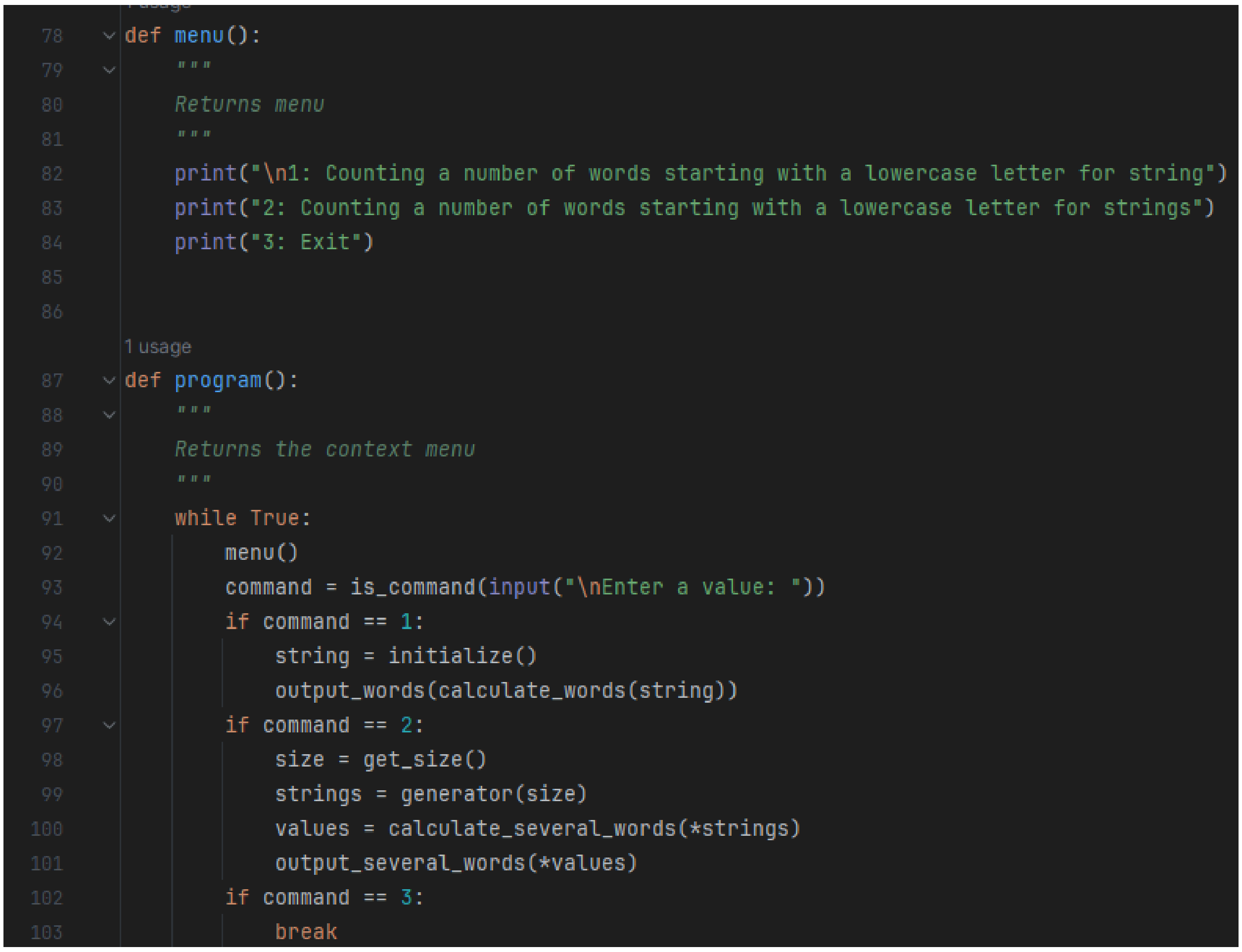

Task3.py

```python
# Program: In a string entered from the keyboard, count the number of words starting with a lowercase letter
# Version: 1.0
# Author: Troshko A.
# Date: 18.03.2024


from Task2 import is_command


1 usage
def is_size(value):
    """
    Checks the value to make sure it is greater than 0
    """
    while True:
        try:
            value = int(value)
            if value > 0:
                return value
            value = input("Value should be greater than 0, input value: ")
        except ValueError:
            value = input("Invalid input, please enter a valid value: ")


2 usages
def initialize():
    """
    Returns the string
    """
    return input("Enter the string: ")


1 usage
def generator(size: int):
    """
    Returns a sequence of strings
    """
    for _ in range(size):
        yield initialize()
```

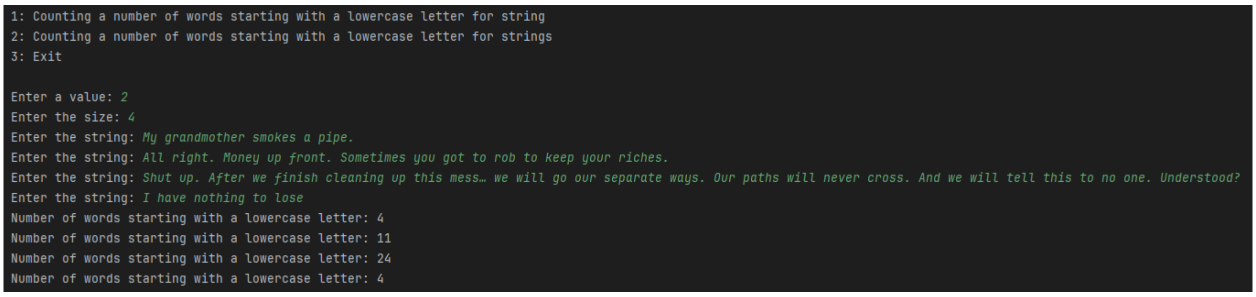


Task3.py ×

```
def get_size():
    """
    Returns a number that will be the size of the list
    """
    return is_size(input("Enter the size: "))


2 usages
def calculate_words(string):
    """
    Returns the number of words starting with a lowercase letter
    """
    k = [word for word in string.split(" ") if 96 < ord(word[:][0]) < 123]
    return len(k)


1 usage
def calculate_several_words(*string):
    """
    Returns the number of words starting with a lowercase letter for each string
    """
    for word in string:
        k = calculate_words(word)
        yield k


2 usages
def output_words(value):
    """
    Returns the result
    """
    print(f"Number of words starting with a lowercase letter: {value}")


1 usage
def output_several_words(*values):
    """
    Returns the result for each string
    """
    for word in values:
        output_words(word)
```



```python
def menu():
    """
    Returns menu
    """
    print("\n1: Counting a number of words starting with a lowercase letter for string")
    print("2: Counting a number of words starting with a lowercase letter for strings")
    print("3: Exit")


1 usage
def program():
    """
    Returns the context menu
    """
    while True:
        menu()
        command = is_command(input("\nEnter a value: "))
        if command == 1:
            string = initialize()
            output_words(calculate_words(string))
        if command == 2:
            size = get_size()
            strings = generator(size)
            values = calculate_several_words(*strings)
            output_several_words(*values)
        if command == 3:
            break
```



```
1: Counting a number of words starting with a lowercase letter for string
2: Counting a number of words starting with a lowercase letter for strings
3: Exit

Enter a value: 2
Enter the size: 4
Enter the string: My grandmother smokes a pipe.
Enter the string: All right. Money up front. Sometimes you got to rob to keep your riches.
Enter the string: Shut up. After we finish cleaning up this mess… we will go our separate ways. Our paths will never cross. And we will tell this to no one. Understood?
Enter the string: I have nothing to lose
Number of words starting with a lowercase letter: 4
Number of words starting with a lowercase letter: 11
Number of words starting with a lowercase letter: 24
Number of words starting with a lowercase letter: 4
```

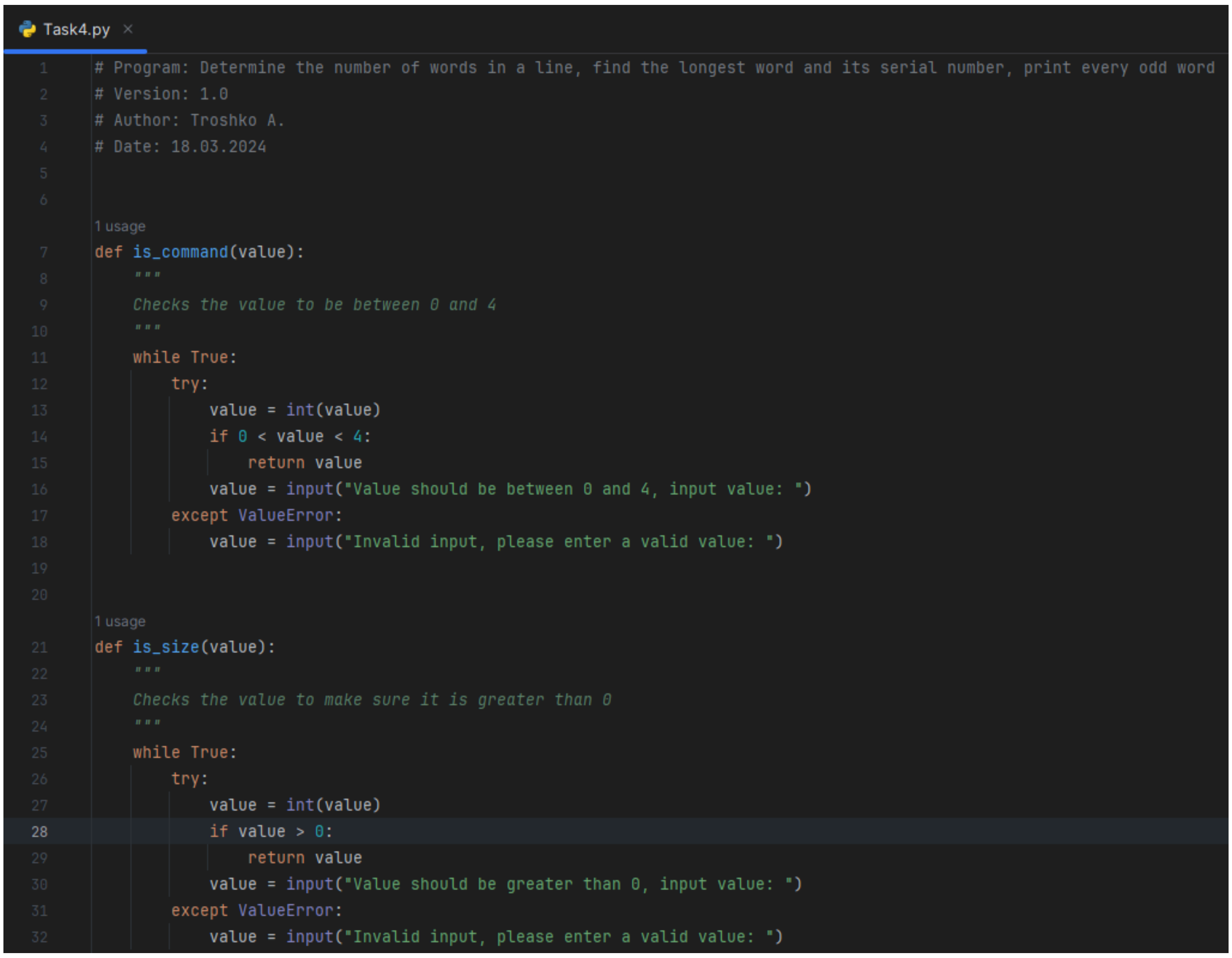
Задание 4. Не использовать регулярные выражения. Дана строка текста, в которой слова разделены пробелами и запятыми. В соответствии с заданием своего варианта составьте программу для анализа строки, инициализированной в коде программы:

«So she was considering in her own mind, as well as she could, for the hot day made her feel very sleepy and stupid, whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.»

Если не оговорено иное, то регистр букв при решении задачи не имеет значения.

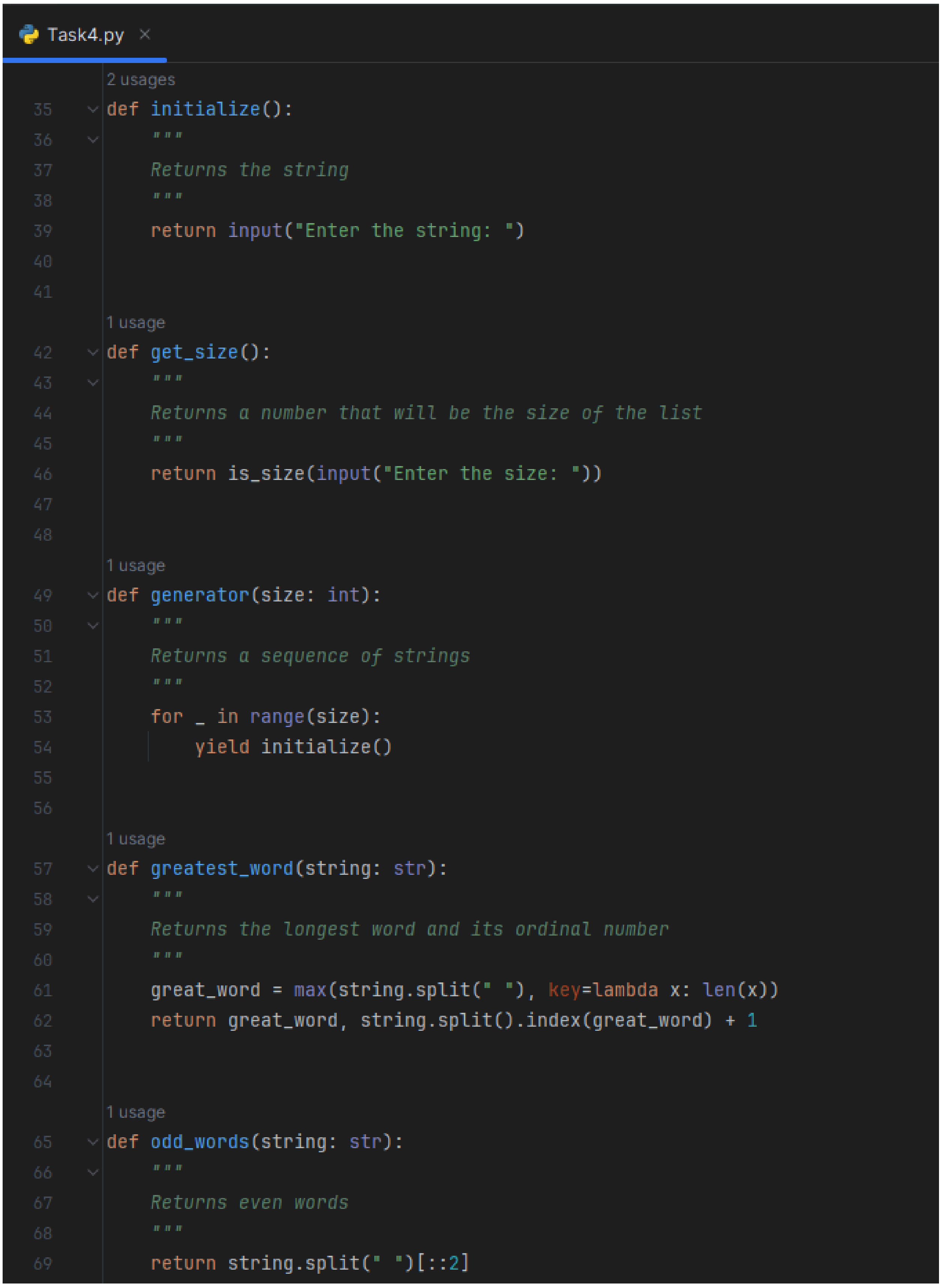
Условие

а) определить количество слов в строке;
б) найти самое длинное слово и его порядковый номер;
в) вывести каждое нечетное слово



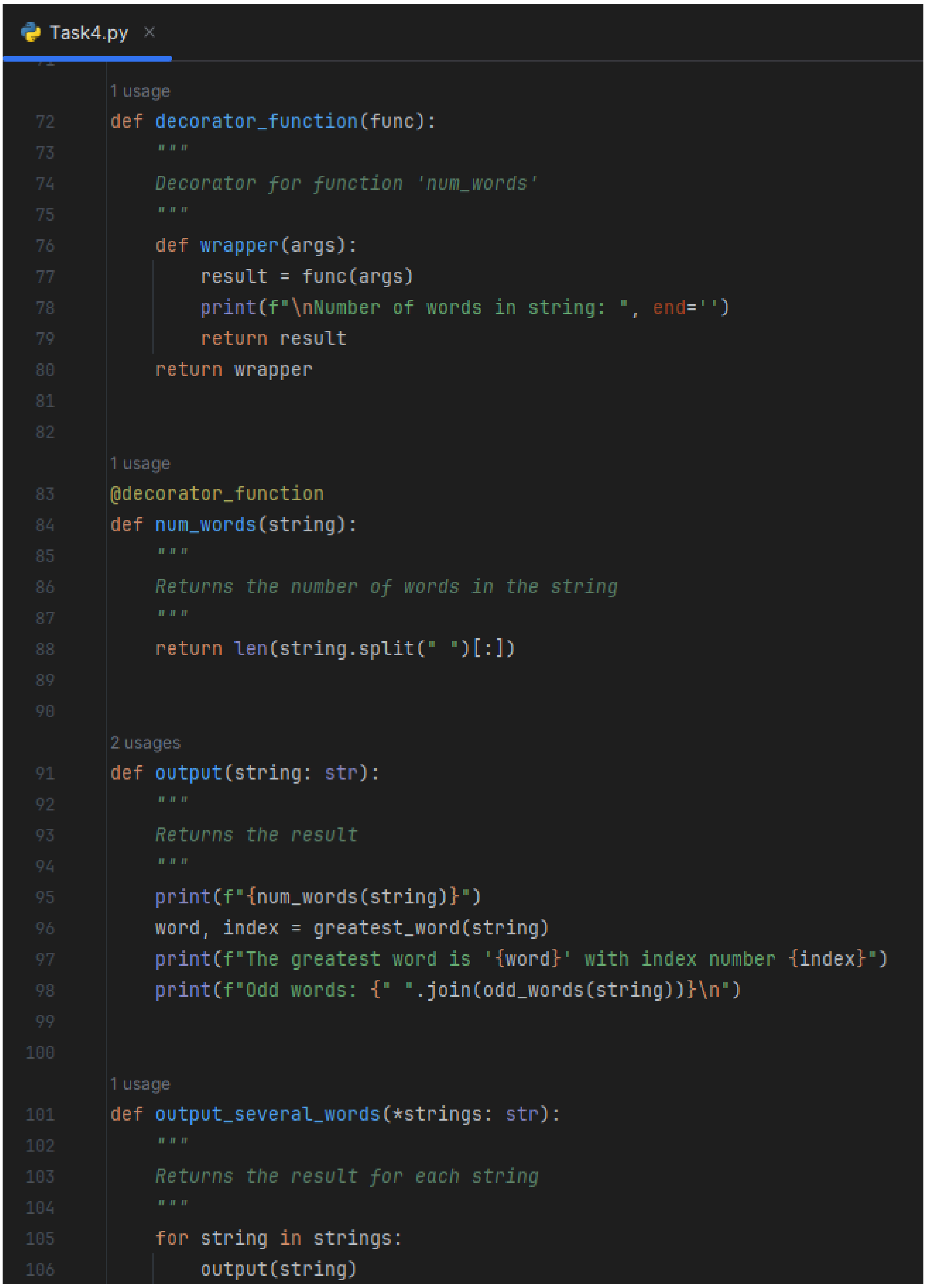
Task4.py

```
# Program: Determine the number of words in a line, find the longest word and its serial number, print every odd word
# Version: 1.0
# Author: Troshko A.
# Date: 18.03.2024


1 usage
def is_command(value):
    """
    Checks the value to be between 0 and 4
    """
    while True:
        try:
            value = int(value)
            if 0 < value < 4:
                return value
            value = input("Value should be between 0 and 4, input value: ")
        except ValueError:
            value = input("Invalid input, please enter a valid value: ")


1 usage
def is_size(value):
    """
    Checks the value to make sure it is greater than 0
    """
    while True:
        try:
            value = int(value)
            if value > 0:
                return value
            value = input("Value should be greater than 0, input value: ")
        except ValueError:
            value = input("Invalid input, please enter a valid value: ")
```



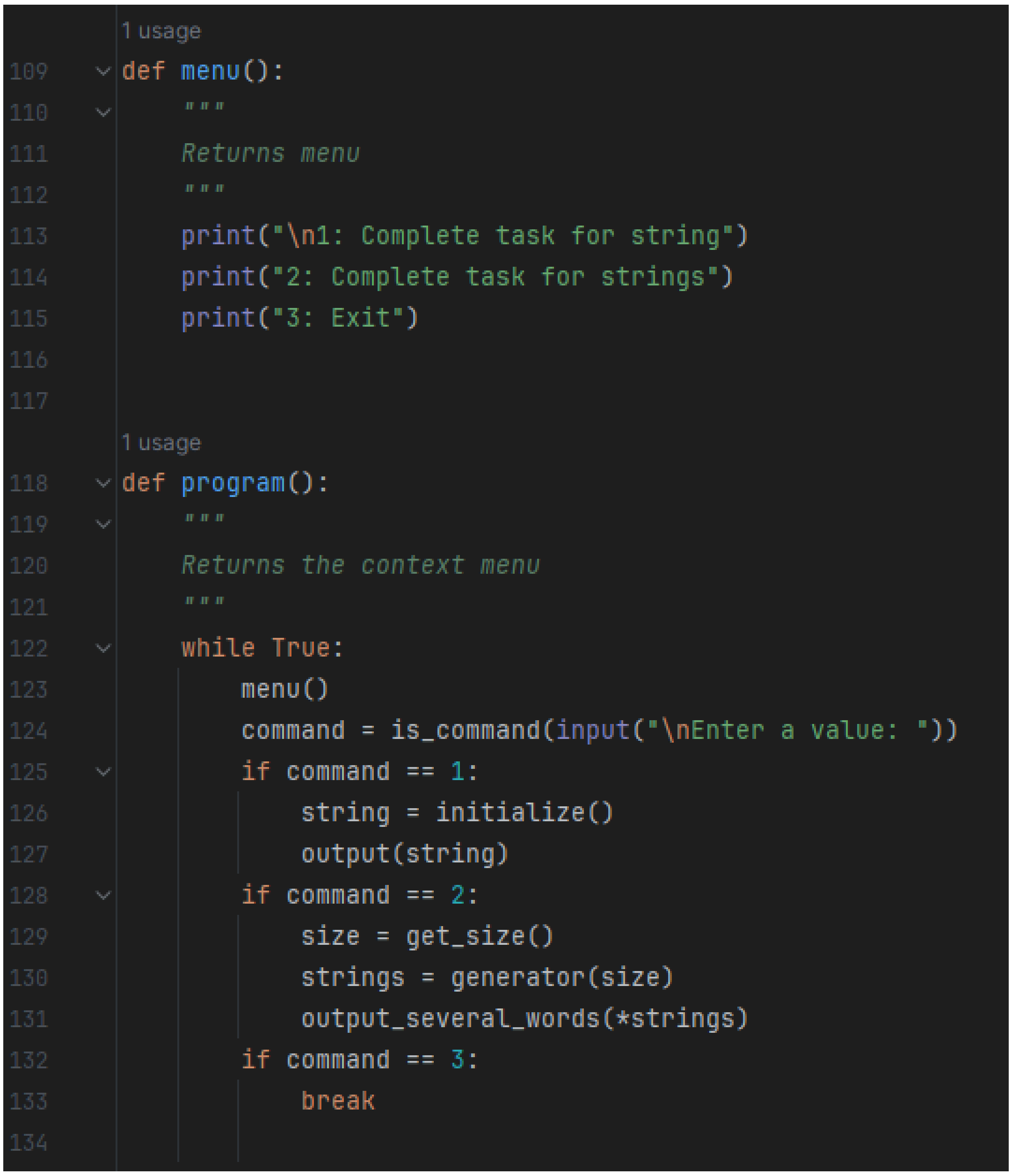
Task4.py ×

```python
2 usages
def initialize():
    """
    Returns the string
    """
    return input("Enter the string: ")


1 usage
def get_size():
    """
    Returns a number that will be the size of the list
    """
    return is_size(input("Enter the size: "))


1 usage
def generator(size: int):
    """
    Returns a sequence of strings
    """
    for _ in range(size):
        yield initialize()


1 usage
def greatest_word(string: str):
    """
    Returns the longest word and its ordinal number
    """
    great_word = max(string.split(" "), key=lambda x: len(x))
    return great_word, string.split().index(great_word) + 1


1 usage
def odd_words(string: str):
    """
    Returns even words
    """
    return string.split(" ")[::2]
```

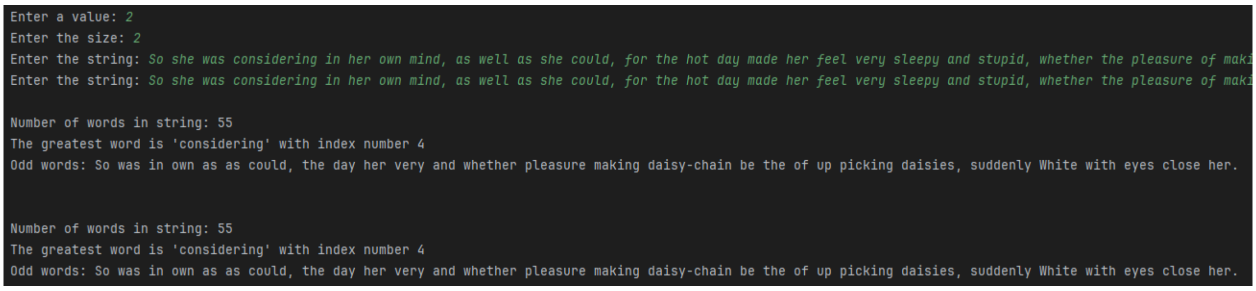


Task4.py

```
1 usage
def decorator_function(func):
    """
    Decorator for function 'num_words'
    """
    def wrapper(args):
        result = func(args)
        print(f"\nNumber of words in string: ", end='')
        return result
    return wrapper


1 usage
@decorator_function
def num_words(string):
    """
    Returns the number of words in the string
    """
    return len(string.split(" ")[:])


2 usages
def output(string: str):
    """
    Returns the result
    """
    print(f"{num_words(string)}")
    word, index = greatest_word(string)
    print(f"The greatest word is '{word}' with index number {index}")
    print(f"Odd words: {" ".join(odd_words(string))}\n")


1 usage
def output_several_words(*strings: str):
    """
    Returns the result for each string
    """
    for string in strings:
        output(string)
```



```
1 usage
def menu():
    """
    Returns menu
    """
    print("\n1: Complete task for string")
    print("2: Complete task for strings")
    print("3: Exit")


1 usage
def program():
    """
    Returns the context menu
    """
    while True:
        menu()
        command = is_command(input("\nEnter a value: "))
        if command == 1:
            string = initialize()
            output(string)
        if command == 2:
            size = get_size()
            strings = generator(size)
            output_several_words(*strings)
        if command == 3:
            break

```



```
Enter a value: 2
Enter the size: 2
Enter the string: So she was considering in her own mind, as well as she could, for the hot day made her feel very sleepy and stupid, whether the pleasure of maki
Enter the string: So she was considering in her own mind, as well as she could, for the hot day made her feel very sleepy and stupid, whether the pleasure of maki

Number of words in string: 55
The greatest word is 'considering' with index number 4
Odd words: So was in own as as could, the day her very and whether pleasure making daisy-chain be the of up picking daisies, suddenly White with eyes close her.


Number of words in string: 55
The greatest word is 'considering' with index number 4
Odd words: So was in own as as could, the day her very and whether pleasure making daisy-chain be the of up picking daisies, suddenly White with eyes close her.
```

Задание 5. В соответствии с заданием своего варианта составить программу для обработки вещественных списков. Программа должна содержать следующие базовые функции:

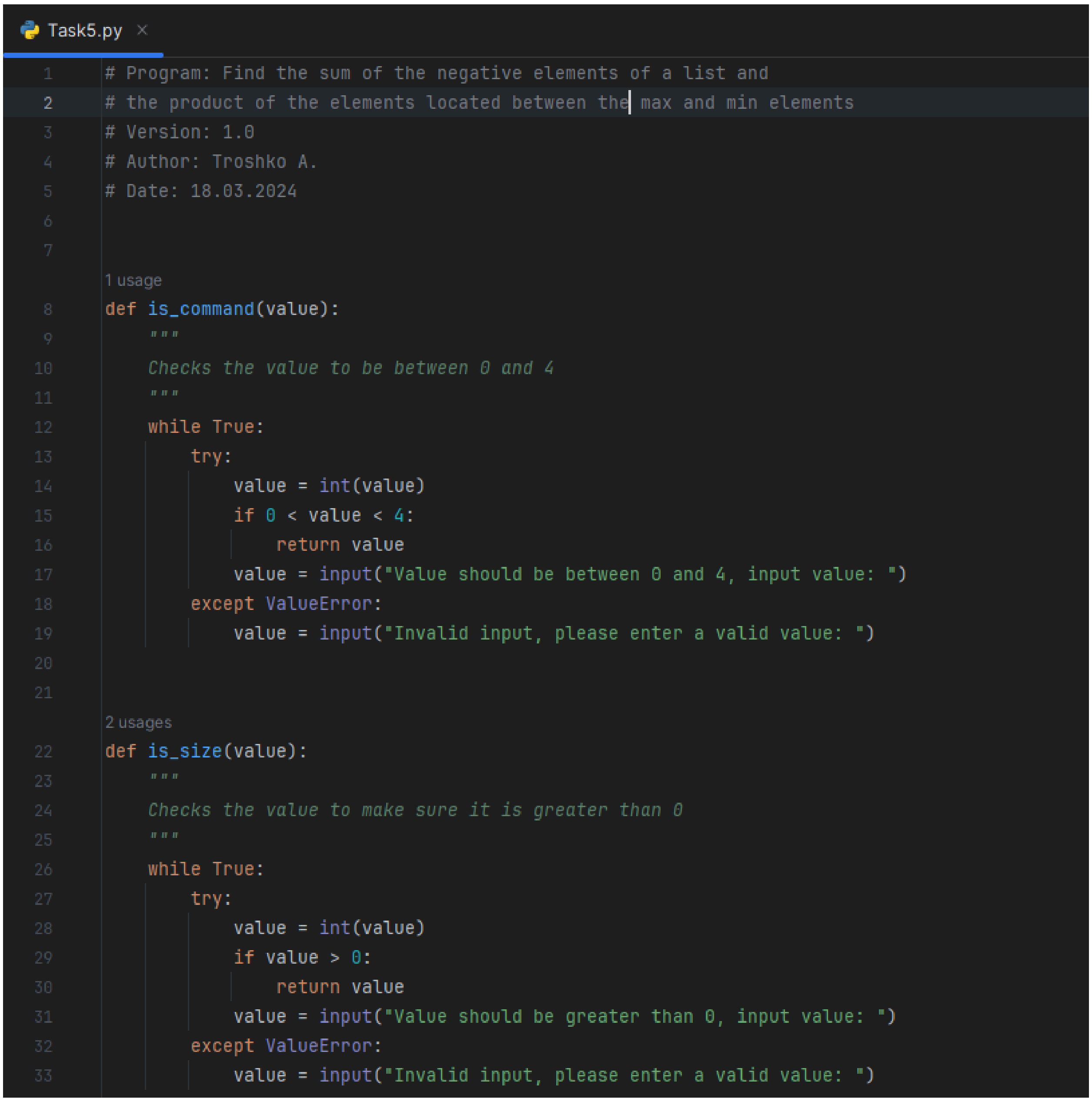
1) ввод элементов списка пользователем;

2) проверка корректности вводимых данных;

3) реализация основного задания с выводом результатов;

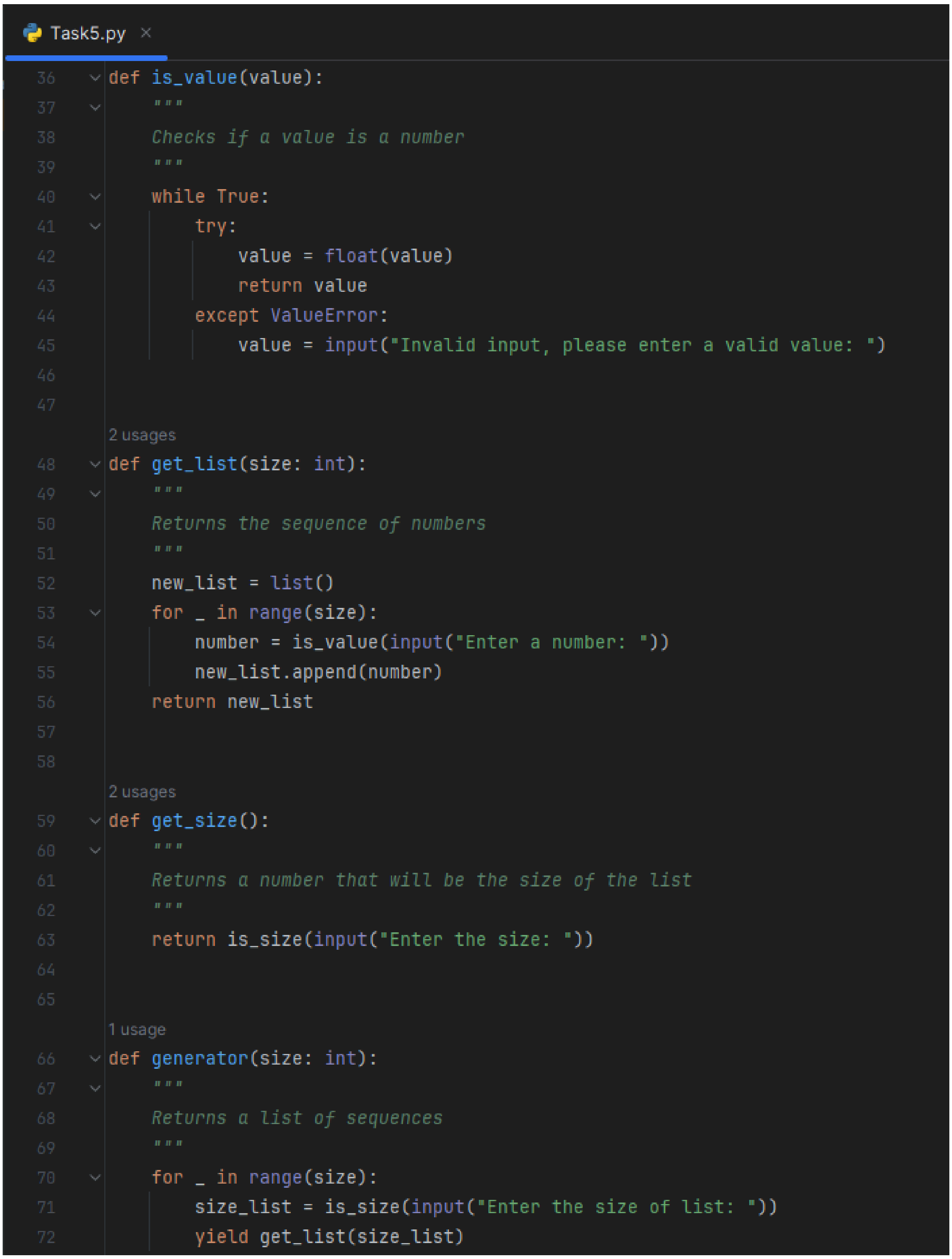
4) вывод списка на экран.

Условие
Найти сумму отрицательных элементов списка и произведение элементов, расположенных между максимальным и минимальным элементами

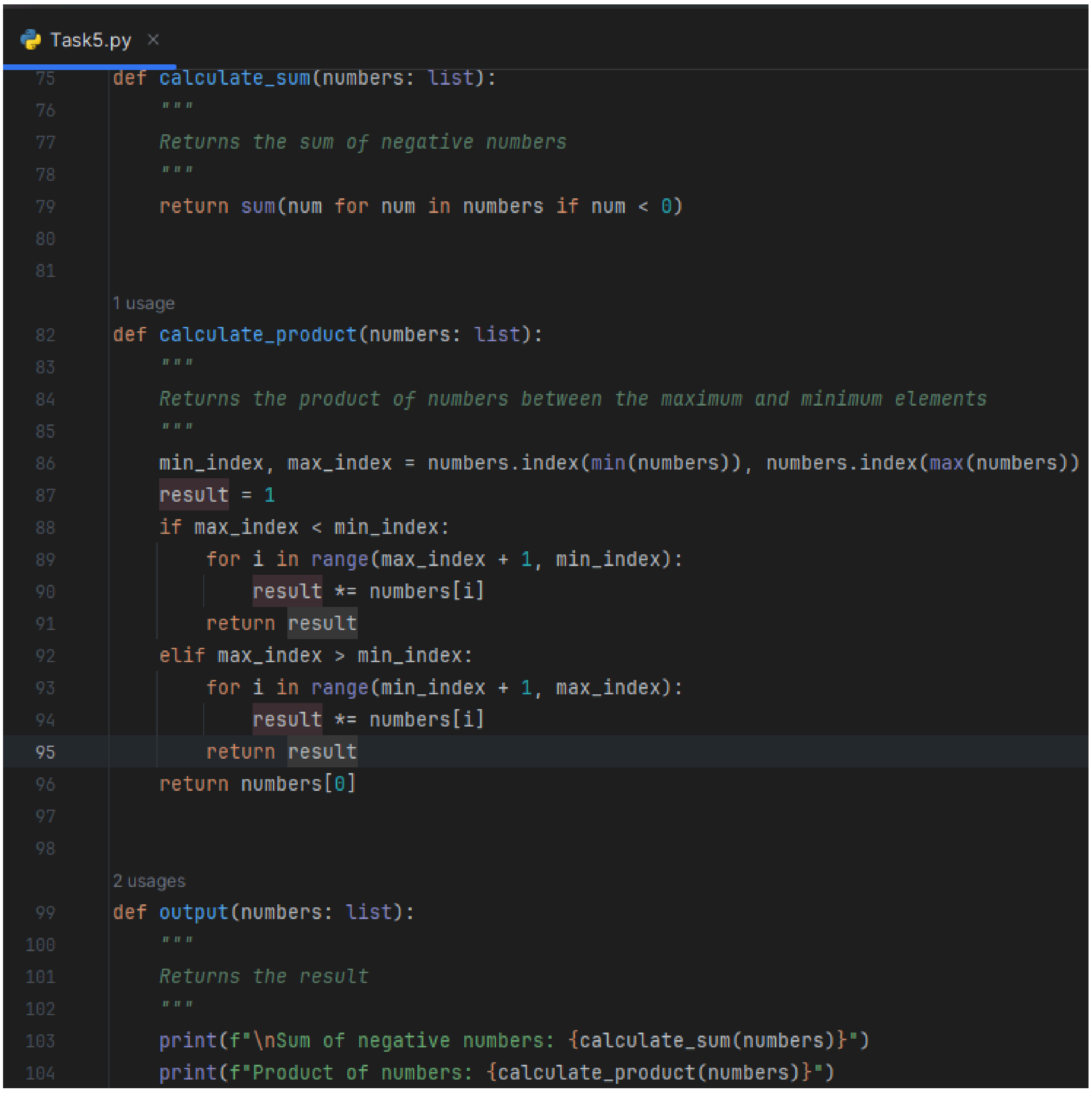


Task5.py

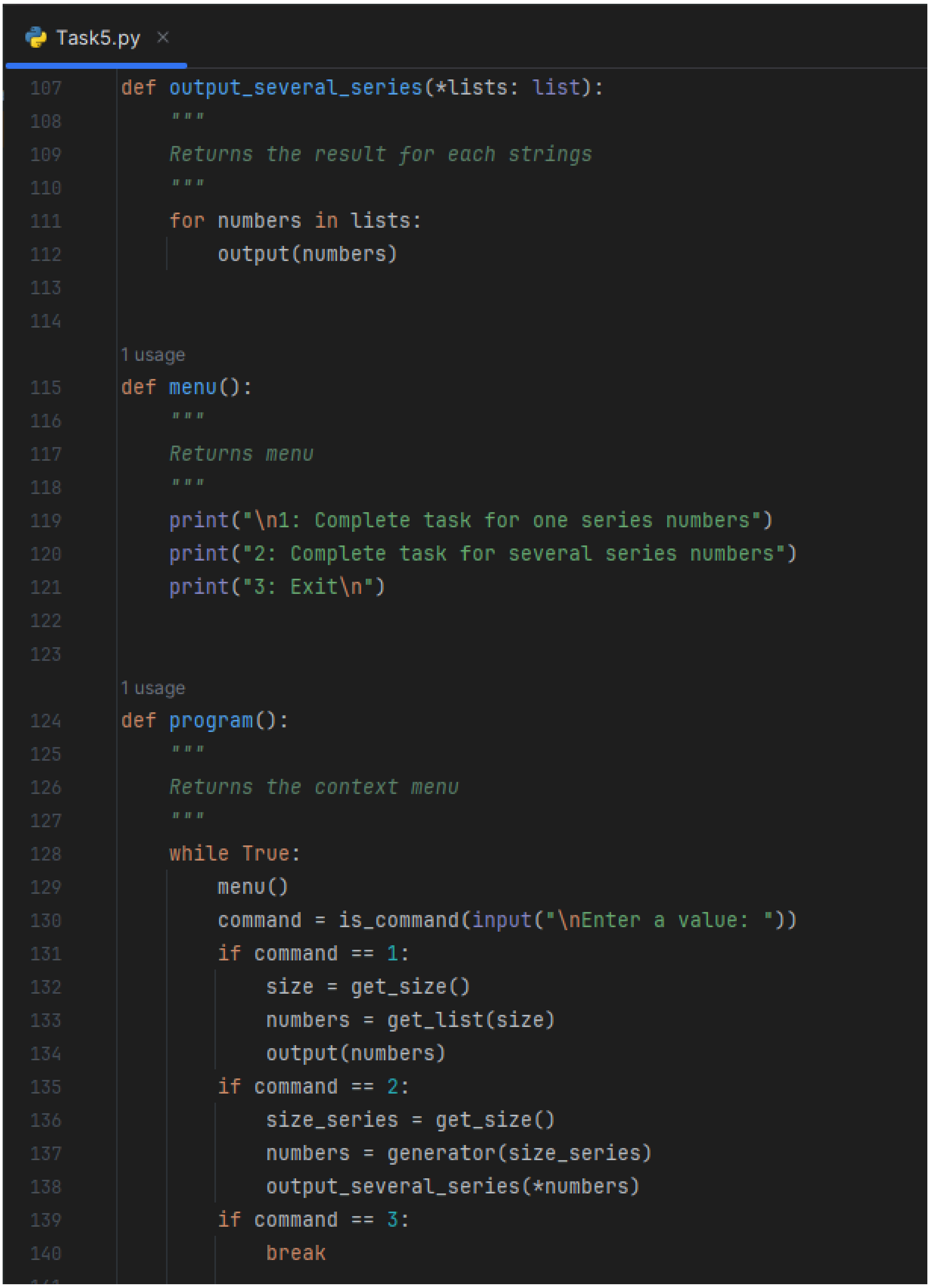
```python
# Program: Find the sum of the negative elements of a list and
# the product of the elements located between the max and min elements
# Version: 1.0
# Author: Troshko A.
# Date: 18.03.2024


1 usage
def is_command(value):
    """
    Checks the value to be between 0 and 4
    """
    while True:
        try:
            value = int(value)
            if 0 < value < 4:
                return value
            value = input("Value should be between 0 and 4, input value: ")
        except ValueError:
            value = input("Invalid input, please enter a valid value: ")


2 usages
def is_size(value):
    """
    Checks the value to make sure it is greater than 0
    """
    while True:
        try:
            value = int(value)
            if value > 0:
                return value
            value = input("Value should be greater than 0, input value: ")
        except ValueError:
            value = input("Invalid input, please enter a valid value: ")
```


Task5.py ×

```
def is_value(value):
    """
    Checks if a value is a number
    """
    while True:
        try:
            value = float(value)
            return value
        except ValueError:
            value = input("Invalid input, please enter a valid value: ")


2 usages
def get_list(size: int):
    """
    Returns the sequence of numbers
    """
    new_list = list()
    for _ in range(size):
        number = is_value(input("Enter a number: "))
        new_list.append(number)
    return new_list


2 usages
def get_size():
    """
    Returns a number that will be the size of the list
    """
    return is_size(input("Enter the size: "))


1 usage
def generator(size: int):
    """
    Returns a list of sequences
    """
    for _ in range(size):
        size_list = is_size(input("Enter the size of list: "))
        yield get_list(size_list)
```

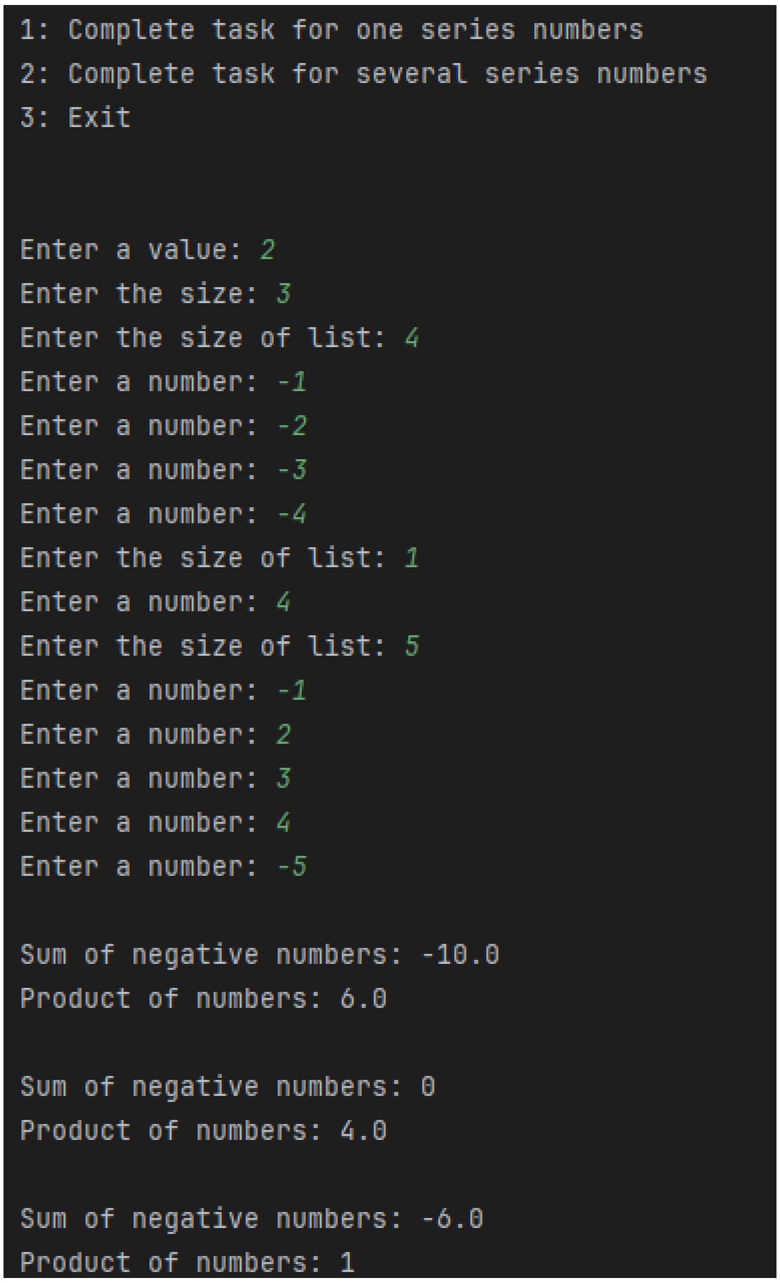

Task5.py

```
def calculate_sum(numbers: list):
    """
    Returns the sum of negative numbers
    """
    return sum(num for num in numbers if num < 0)


1 usage
def calculate_product(numbers: list):
    """
    Returns the product of numbers between the maximum and minimum elements
    """
    min_index, max_index = numbers.index(min(numbers)), numbers.index(max(numbers))
    result = 1
    if max_index < min_index:
        for i in range(max_index + 1, min_index):
            result *= numbers[i]
        return result
    elif max_index > min_index:
        for i in range(min_index + 1, max_index):
            result *= numbers[i]
        return result
    return numbers[0]


2 usages
def output(numbers: list):
    """
    Returns the result
    """
    print(f"\nSum of negative numbers: {calculate_sum(numbers)}")
    print(f"Product of numbers: {calculate_product(numbers)}")
```



Task5.py

```python
def output_several_series(*lists: list):
    """
    Returns the result for each strings
    """
    for numbers in lists:
        output(numbers)


1 usage
def menu():
    """
    Returns menu
    """
    print("\n1: Complete task for one series numbers")
    print("2: Complete task for several series numbers")
    print("3: Exit\n")


1 usage
def program():
    """
    Returns the context menu
    """
    while True:
        menu()
        command = is_command(input("\nEnter a value: "))
        if command == 1:
            size = get_size()
            numbers = get_list(size)
            output(numbers)
        if command == 2:
            size_series = get_size()
            numbers = generator(size_series)
            output_several_series(*numbers)
        if command == 3:
            break
```



```
1: Complete task for one series numbers
2: Complete task for several series numbers
3: Exit


Enter a value: 2
Enter the size: 3
Enter the size of list: 4
Enter a number: -1
Enter a number: -2
Enter a number: -3
Enter a number: -4
Enter the size of list: 1
Enter a number: 4
Enter the size of list: 5
Enter a number: -1
Enter a number: 2
Enter a number: 3
Enter a number: 4
Enter a number: -5

Sum of negative numbers: -10.0
Product of numbers: 6.0

Sum of negative numbers: 0
Product of numbers: 4.0

Sum of negative numbers: -6.0
Product of numbers: 1
```