

# 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 x
2 usages
1  def is_command(value):
2      """
3      Checks the value to be between 0 and 7
4      """
5      while True:
6          try:
7              value = int(value)
8              if 0 < value < 7:
9                  return value
10             value = input("Value should be between 0 and 7, input value: ")
11         except ValueError:
12             value = input("Invalid input, please enter a valid value: ")
13
```

```
menu.py x
2 usages
1  def menu():
2      """
3      Returns menu
4      """
5      print("\n1: Start Task1")
6      print("2: Start Task2")
7      print("3: Start Task3")
8      print("4: Start Task4")
9      print("5: Start Task5")
10     print("6: Exit")
11
```

```
requirements.txt x
1  PrettyTable==3.10.0
```

main.py ×

```
1  ✓ from errors import is_command
2    from menu import menu
3    import Task1
4    import Task2
5    import Task3
6    import Task4
7    import Task5
8
9
10   1 usage
11   ✓ def program():
12   ✓     while True:
13       menu()
14       command = is_command(input("\nEnter value: "))
15       if command == 1:
16           Task1.program()
17       if command == 2:
18           Task2.program()
19       if command == 3:
20           Task3.program()
21       if command == 4:
22           Task4.program()
23       if command == 5:
24           Task5.program()
25       if command == 6:
26           break
27
28   ▶ if __name__ == '__main__':
29       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  $\epsilon$ .

Предусмотреть максимальное количество итераций, равное 500.

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

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

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

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

Task1.py ×

```
1  # Program: Calculating the value of a function using a power series expansion of the function
2  # Version: 1.0
3  # Author: Troshko A.
4  # Date: 18.03.2024
5
6
7  import math
8  from prettytable import PrettyTable
9
10
11 table = PrettyTable()
12
13
14 1 usage
15 def is_size(value):
16     """
17     Checks the value to make sure it is greater than 0
18     """
19     while True:
20         try:
21             value = int(value)
22             if value > 0:
23                 return value
24             value = input("Value should be greater than 0, input value: ")
25         except ValueError:
26             value = input("Invalid input, please enter a valid value: ")
27
28 2 usages
29 def is_eps(value):
30     """
31     Checks the value to be between 0 and 1
32     """
33     while True:
34         try:
35             value = float(value)
36             if 0 < value < 1:
37                 return value
38             value = input("Value should be between 0 and 1, input value: ")
39         except ValueError:
40             value = input("Invalid input, please enter a valid value: ")
```

2 usages

```
42 def is_value(value):
43     """
44     Checks the value to make sure it is greater than 1
45     """
46     while True:
47         try:
48             value = float(value)
49             if value > 1:
50                 return value
51             value = input("Value should be greater than 1, input value: ")
52         except ValueError:
53             value = input("Invalid input, please enter a valid value: ")
54
55
```

1 usage

```
56 def is_command(value):
57     """
58     Checks the value to be between 0 and 4
59     """
60     while True:
61         try:
62             value = int(value)
63             if 0 < value < 4:
64                 return value
65             value = input("Value should be between 0 and 4, input value: ")
66         except ValueError:
67             value = input("Invalid input, please enter a valid value: ")
68
69
```

1 usage

```
70 def get_size_tuple():
71     """
72     Returns a number that will be the size of the list
73     """
74     return is_size(input("Enter size of list: "))
75
```

Task1.py ×

```
77  def generator(size: int):
78      """
79      Returns a sequence of numbers
80      """
81      for _ in range(size):
82          yield is_value(input("Enter values: "))
83
84
85      1 usage
86  def get_list(size: int):
87      """
88      Returns eps and generator
89      """
90      return is_eps(input("Enter eps: ")), tuple(generator(size))
91
92      1 usage
93  def get_values():
94      """
95      Returns eps and value
96      """
97      return is_eps(input("Enter eps: ")), is_value(input("Enter value: "))
98
99      1 usage
100  def get_taylor_series_math(value: int):
101      """
102      Returns the value of a function using a module math
103      """
104      return math.log((value + 1) / (value - 1))
```

Task1.py ×

```
106  ✓ def get_taylor_series(eps: float, value: int):
107  ✓     """
108      Returns the value of a function using eps
109      """
110      s = n = 0
111      a = value
112  ✓     while abs(a) > eps and n < 501:
113         s += a
114         a = 1 / ((2 * n + 1) * value ** (2 * n + 1))
115         n += 1
116     s -= value
117     return n, 2 * s
118
119
120  ✓ def add_value(eps: float, value):
121  ✓     """
122     Adds values to the table
123     """
124     n, s = get_taylor_series(eps, value)
125     smath = get_taylor_series_math(value)
126
127     table.field_names = ["x", "n", "F(x)", "Math F(x)", "eps"]
128     table.add_row([value, n, s, smath, eps])
129
130
131  ✓ def add_tuple(eps: float, *args):
132  ✓     """
133     Unpacks the tuple and calls the method 'add_value'
134     """
135     for value in args:
136         add_value(eps, value)
```

```
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		0.200666666666666677		0.20067069546215124		1e-05	
	20.0		3		0.1000833333333333752		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		0.058840500022933395		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

```
Task2.py ×
1  # Program: Organize a loop that takes integers and calculates the arithmetic mean of even numbers. End - input 1
2  # Version: 1.0
3  # Author: Troshko A.
4  # Date: 18.03.2024
5
6
7  1 usage
8  def is_size(value):
9      """
10     Checks the value to make sure it is greater than 0
11     """
12     while True:
13         try:
14             value = int(value)
15             if value > 0:
16                 return value
17             value = input("Value should be greater than 0, input value: ")
18         except ValueError:
19             value = input("Invalid input, please enter a valid value: ")
20
21  1 usage
22  def is_value(value):
23      """
24     Checks if a value is a number
25     """
26     while True:
27         try:
28             value = int(value)
29             return value
30         except ValueError:
31             value = input("Invalid input, please enter a valid value: ")
32
```

```
33  def is_command(value):
34      """
35      Checks the value to be between 0 and 4
36      """
37      while True:
38          try:
39              value = int(value)
40              if 0 < value < 4:
41                  return value
42              value = input("Value should be between 0 and 4, input value: ")
43          except ValueError:
44              value = input("Invalid input, please enter a valid value: ")
45
46      1 usage
47  def input_numbers():
48      """
49      Returns a list of numbers
50      """
51      numbers = list()
52      while True:
53          try:
54              number = int(input("Enter a number: "))
55              if number == 1:
56                  break
57              numbers.append(number)
58          except ValueError:
59              number = input("Invalid input, please enter a valid value: ")
60              if number == 1:
61                  break
62      return numbers
63
64      1 usage
65  def get_size():
66      """
67      Returns a number that will be the size of the list
68      """
69      return is_size(input("Enter a size of list: "))
```

```
72 def generator():
73     """
74     Returns a sequence of numbers
75     """
76     while True:
77         value = is_value(input("Enter a number: "))
78         yield value
79         if value == 1:
80             break
81
82     1 usage
83 def calculate_avg(args: list):
84     """
85     Returns the arithmetic mean of even numbers
86     """
87     try:
88         avg = [num for num in args if num % 2 == 0]
89         return sum(avg) / len(avg)
90     except ZeroDivisionError:
91         return None
92
93     2 usages
94 def output_avg(args):
95     """
96     Returns the result
97     """
98     print(f"\nAverage of even numbers: {calculate_avg(args)}\n")
99
100     1 usage
101 def menu():
102     """
103     Returns menu
104     """
105     print("\n1: Counting a avg for one series numbers")
106     print("2: Counting a avg for several series numbers")
107     print("3: Exit")
```

```

1 usage
110  def program():
111      """
112      Returns the context menu
113      """
114      while True:
115          menu()
116          command = is_command(input("\nEnter a value: "))
117          if command == 1:
118              new_list = input_numbers()
119              output_avg(new_list)
120          if command == 2:
121              size = get_size()
122              for _ in range(size):
123                  k = generator()
124                  output_avg(k)
125          if command == 3:
126              break
127

```

```

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 x
1  # Program: In a string entered from the keyboard, count the number of words starting with a lowercase letter
2  # Version: 1.0
3  # Author: Troshko A.
4  # Date: 18.03.2024
5
6
7  from Task2 import is_command
8
9
10 1 usage
11 def is_size(value):
12     """
13     Checks the value to make sure it is greater than 0
14     """
15     while True:
16         try:
17             value = int(value)
18             if value > 0:
19                 return value
20             value = input("Value should be greater than 0, input value: ")
21         except ValueError:
22             value = input("Invalid input, please enter a valid value: ")
23
24 2 usages
25 def initialize():
26     """
27     Returns the string
28     """
29     return input("Enter the string: ")
30
31 1 usage
32 def generator(size: int):
33     """
34     Returns a sequence of strings
35     """
36     for _ in range(size):
37         yield initialize()
```

```
39 def get_size():
40     """
41     Returns a number that will be the size of the list
42     """
43     return is_size(input("Enter the size: "))
44
45
46     2 usages
47 def calculate_words(string):
48     """
49     Returns the number of words starting with a lowercase letter
50     """
51     k = [word for word in string.split(" ") if 96 < ord(word[:][0]) < 123]
52     return len(k)
53
54     1 usage
55 def calculate_several_words(*string):
56     """
57     Returns the number of words starting with a lowercase letter for each string
58     """
59     for word in string:
60         k = calculate_words(word)
61         yield k
62
63     2 usages
64 def output_words(value):
65     """
66     Returns the result
67     """
68     print(f"Number of words starting with a lowercase letter: {value}")
69
70     1 usage
71 def output_several_words(*values):
72     """
73     Returns the result for each string
74     """
75     for word in values:
76         output_words(word)
```

```

78  def menu():
79      """
80      Returns menu
81      """
82      print("\n1: Counting a number of words starting with a lowercase letter for string")
83      print("2: Counting a number of words starting with a lowercase letter for strings")
84      print("3: Exit")
85
86
1 usage
87  def program():
88      """
89      Returns the context menu
90      """
91      while True:
92          menu()
93          command = is_command(input("\nEnter a value: "))
94          if command == 1:
95              string = initialize()
96              output_words(calculate_words(string))
97          if command == 2:
98              size = get_size()
99              strings = generator(size)
100             values = calculate_several_words(*strings)
101             output_several_words(*values)
102         if command == 3:
103             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 x
1 # Program: Determine the number of words in a line, find the longest word and its serial number, print every odd word
2 # Version: 1.0
3 # Author: Troshko A.
4 # Date: 18.03.2024
5
6
7 1 usage
8 def is_command(value):
9     """
10     Checks the value to be between 0 and 4
11     """
12     while True:
13         try:
14             value = int(value)
15             if 0 < value < 4:
16                 return value
17             value = input("Value should be between 0 and 4, input value: ")
18         except ValueError:
19             value = input("Invalid input, please enter a valid value: ")
20
21 1 usage
22 def is_size(value):
23     """
24     Checks the value to make sure it is greater than 0
25     """
26     while True:
27         try:
28             value = int(value)
29             if value > 0:
30                 return value
31             value = input("Value should be greater than 0, input value: ")
32         except ValueError:
33             value = input("Invalid input, please enter a valid value: ")
```

```
2 usages
35  ✓ def initialize():
36  ✓     """
37      Returns the string
38      """
39      return input("Enter the string: ")
40
41
1 usage
42  ✓ def get_size():
43  ✓     """
44      Returns a number that will be the size of the list
45      """
46      return is_size(input("Enter the size: "))
47
48
1 usage
49  ✓ def generator(size: int):
50  ✓     """
51      Returns a sequence of strings
52      """
53      for _ in range(size):
54          yield initialize()
55
56
1 usage
57  ✓ def greatest_word(string: str):
58  ✓     """
59      Returns the longest word and its ordinal number
60      """
61      great_word = max(string.split(" "), key=lambda x: len(x))
62      return great_word, string.split().index(great_word) + 1
63
64
1 usage
65  ✓ def odd_words(string: str):
66  ✓     """
67      Returns even words
68      """
69      return string.split(" ")[::2]
```

```
71 1 usage
72 def decorator_function(func):
73     """
74     Decorator for function 'num_words'
75     """
76     def wrapper(args):
77         result = func(args)
78         print(f"\nNumber of words in string: ", end='')
79         return result
80     return wrapper
81
82 1 usage
83 @decorator_function
84 def num_words(string):
85     """
86     Returns the number of words in the string
87     """
88     return len(string.split(" ")[:])
89
90 2 usages
91 def output(string: str):
92     """
93     Returns the result
94     """
95     print(f"{num_words(string)}")
96     word, index = greatest_word(string)
97     print(f"The greatest word is '{word}' with index number {index}")
98     print(f"Odd words: {" ".join(odd_words(string))}\n")
99
100 1 usage
101 def output_several_words(*strings: str):
102     """
103     Returns the result for each string
104     """
105     for string in strings:
106         output(string)
```

```

109  ✓ def menu():
110  ✓     """
111      Returns menu
112      """
113      print("\n1: Complete task for string")
114      print("2: Complete task for strings")
115      print("3: Exit")
116
117
118  ✓ def program():
119  ✓     """
120      Returns the context menu
121      """
122  ✓     while True:
123         menu()
124         command = is_command(input("\nEnter a value: "))
125  ✓         if command == 1:
126             string = initialize()
127             output(string)
128  ✓         if command == 2:
129             size = get_size()
130             strings = generator(size)
131             output_several_words(*strings)
132         if command == 3:
133             break
134

```

```

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 making
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 making

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 x
1  # Program: Find the sum of the negative elements of a list and
2  # the product of the elements located between the max and min elements
3  # Version: 1.0
4  # Author: Troshko A.
5  # Date: 18.03.2024
6
7
8  1 usage
9  def is_command(value):
10     """
11     Checks the value to be between 0 and 4
12     """
13     while True:
14         try:
15             value = int(value)
16             if 0 < value < 4:
17                 return value
18             value = input("Value should be between 0 and 4, input value: ")
19         except ValueError:
20             value = input("Invalid input, please enter a valid value: ")
21
22  2 usages
23  def is_size(value):
24     """
25     Checks the value to make sure it is greater than 0
26     """
27     while True:
28         try:
29             value = int(value)
30             if value > 0:
31                 return value
32             value = input("Value should be greater than 0, input value: ")
33         except ValueError:
34             value = input("Invalid input, please enter a valid value: ")
```

```
36  ✓ def is_value(value):
37  ✓     """
38      Checks if a value is a number
39      """
40  ✓     while True:
41  ✓         try:
42             value = float(value)
43             return value
44         except ValueError:
45             value = input("Invalid input, please enter a valid value: ")
46
47
48      2 usages
49  ✓ def get_list(size: int):
50  ✓     """
51      Returns the sequence of numbers
52      """
53      new_list = list()
54      for _ in range(size):
55          number = is_value(input("Enter a number: "))
56          new_list.append(number)
57      return new_list
58
59      2 usages
60  ✓ def get_size():
61  ✓     """
62      Returns a number that will be the size of the list
63      """
64      return is_size(input("Enter the size: "))
65
66      1 usage
67  ✓ def generator(size: int):
68  ✓     """
69      Returns a list of sequences
70      """
71      for _ in range(size):
72          size_list = is_size(input("Enter the size of list: "))
73          yield get_list(size_list)
```

Task5.py ×

```
75 def calculate_sum(numbers: list):
76     """
77     Returns the sum of negative numbers
78     """
79     return sum(num for num in numbers if num < 0)
80
81
82 1 usage
83 def calculate_product(numbers: list):
84     """
85     Returns the product of numbers between the maximum and minimum elements
86     """
87     min_index, max_index = numbers.index(min(numbers)), numbers.index(max(numbers))
88     result = 1
89     if max_index < min_index:
90         for i in range(max_index + 1, min_index):
91             result *= numbers[i]
92         return result
93     elif max_index > min_index:
94         for i in range(min_index + 1, max_index):
95             result *= numbers[i]
96     return result
97
98
99 2 usages
100 def output(numbers: list):
101     """
102     Returns the result
103     """
104     print(f"\nSum of negative numbers: {calculate_sum(numbers)}")
105     print(f"Product of numbers: {calculate_product(numbers)}")
```

```
107 def output_several_series(*lists: list):
108     """
109     Returns the result for each strings
110     """
111     for numbers in lists:
112         output(numbers)
113
114
115 1 usage
116 def menu():
117     """
118     Returns menu
119     """
120     print("\n1: Complete task for one series numbers")
121     print("2: Complete task for several series numbers")
122     print("3: Exit\n")
123
124 1 usage
125 def program():
126     """
127     Returns the context menu
128     """
129     while True:
130         menu()
131         command = is_command(input("\nEnter a value: "))
132         if command == 1:
133             size = get_size()
134             numbers = get_list(size)
135             output(numbers)
136         if command == 2:
137             size_series = get_size()
138             numbers = generator(size_series)
139             output_several_series(*numbers)
140         if command == 3:
141             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