

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО
ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего
образования**

«Северо-Кавказский федеральный

университет» Кафедра

инфокоммуникаций

**Отчет по лабораторной работе №7
«Работа со списками в языке Python»**

по дисциплине «Основы программной инженерии»

Выполнила:

Первых Дарья Александровна,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:

Доцент кафедры инфокоммуникаций,
Воронкин Роман Александрович

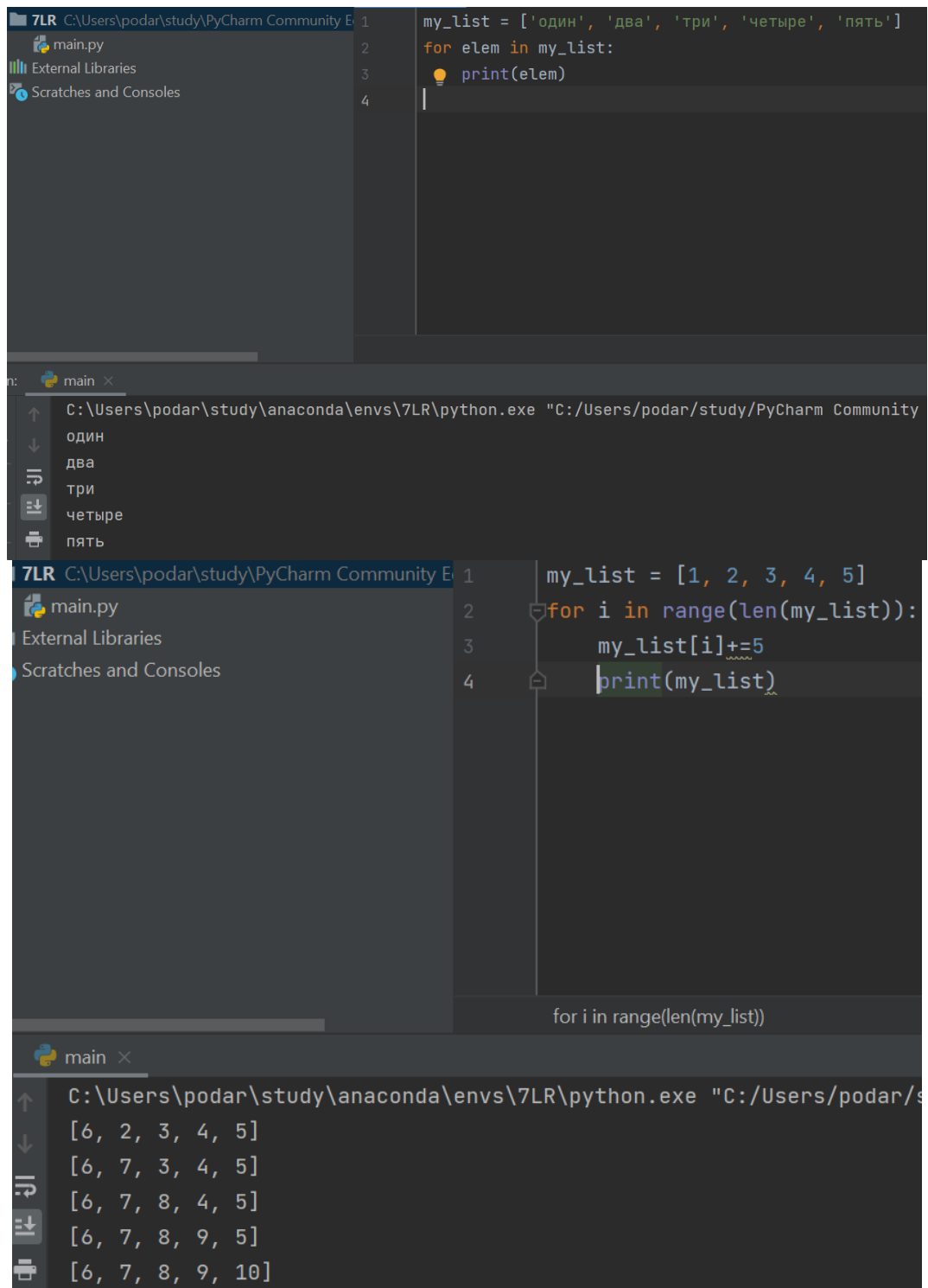
Ставрополь 2021

ВЫПОЛНЕНИЕ

```
my_list = [1, 2, 3, 4, 5]
my_list = ['один', 'два', 'три', 'четыре', 'пять']
my_list = ['один', 10, 2.25, [5, 15], 'пять']

third_elem = my_list[2]
```

Рисунок 1 – Пример создания списка



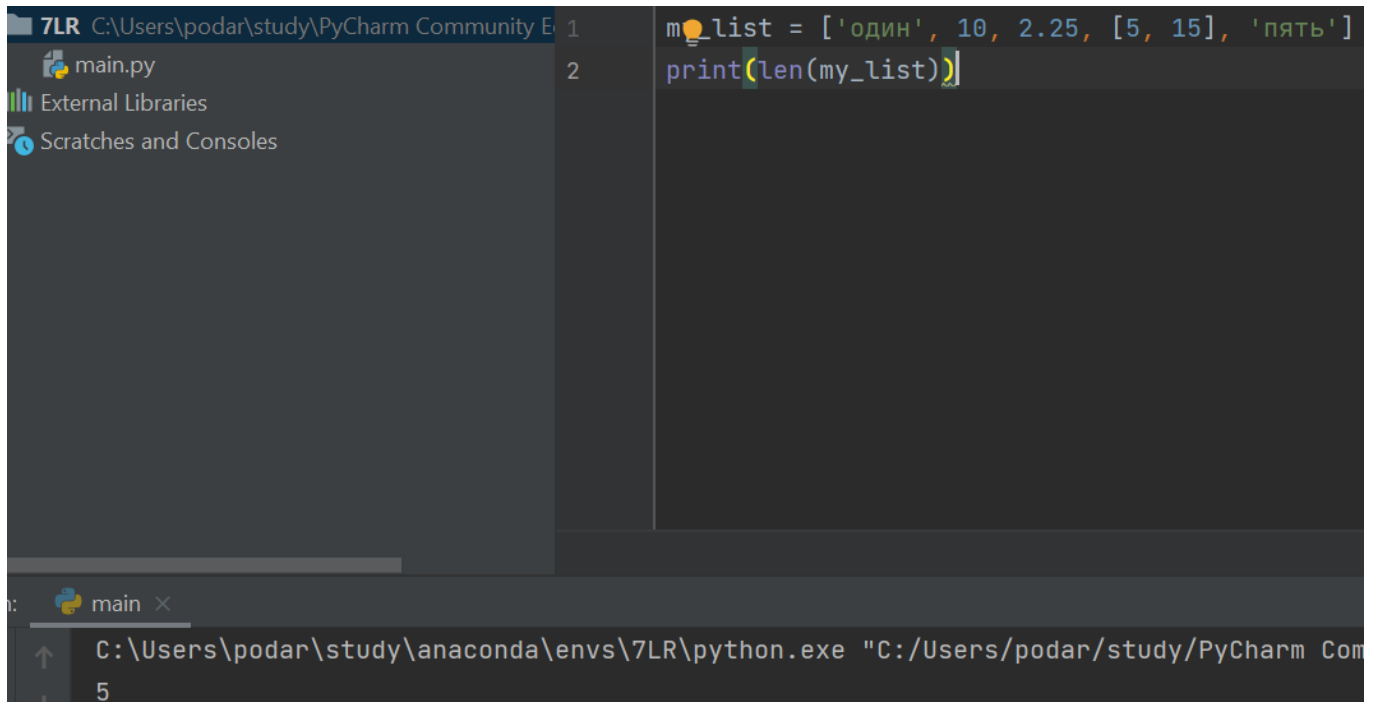


Рисунок 2 – Пример прохода (итерации) по списку

```
>>> a = [10, 20, 30, 40]  
>>> for i, item in enumerate(a):  
...     print(f"({i}, {item})")  
...  
(0, 10)  
(1, 20)  
(2, 30)  
(3, 40)  
>>>
```

Рисунок 3 – Пример прохода (итерации) по списку

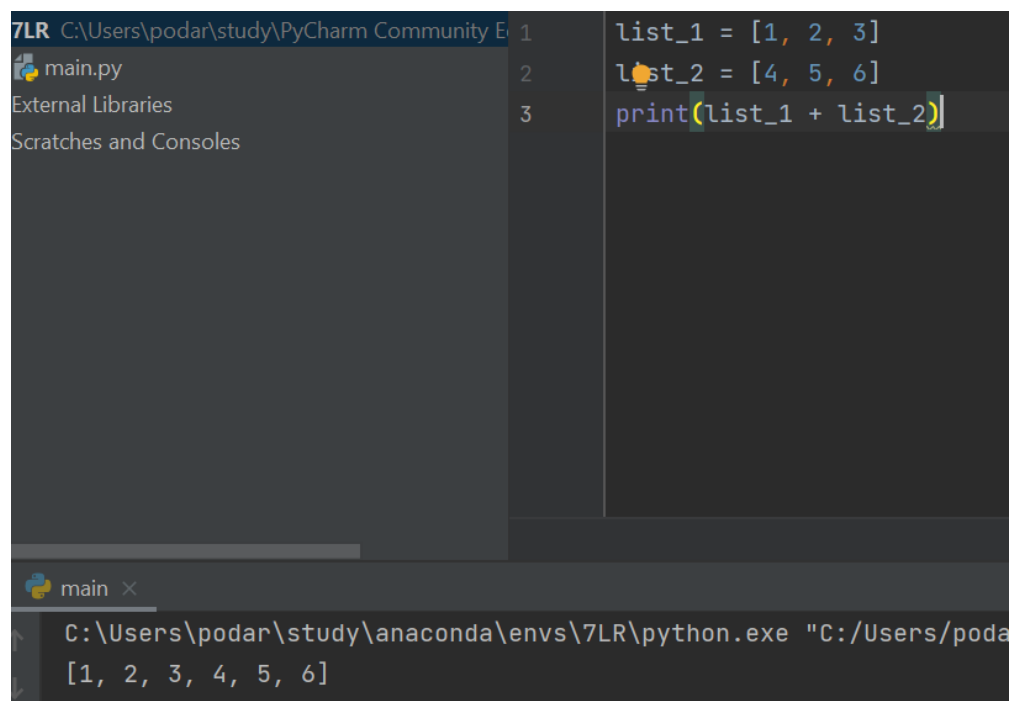


Рисунок 4 – Пример операции сложения со списками

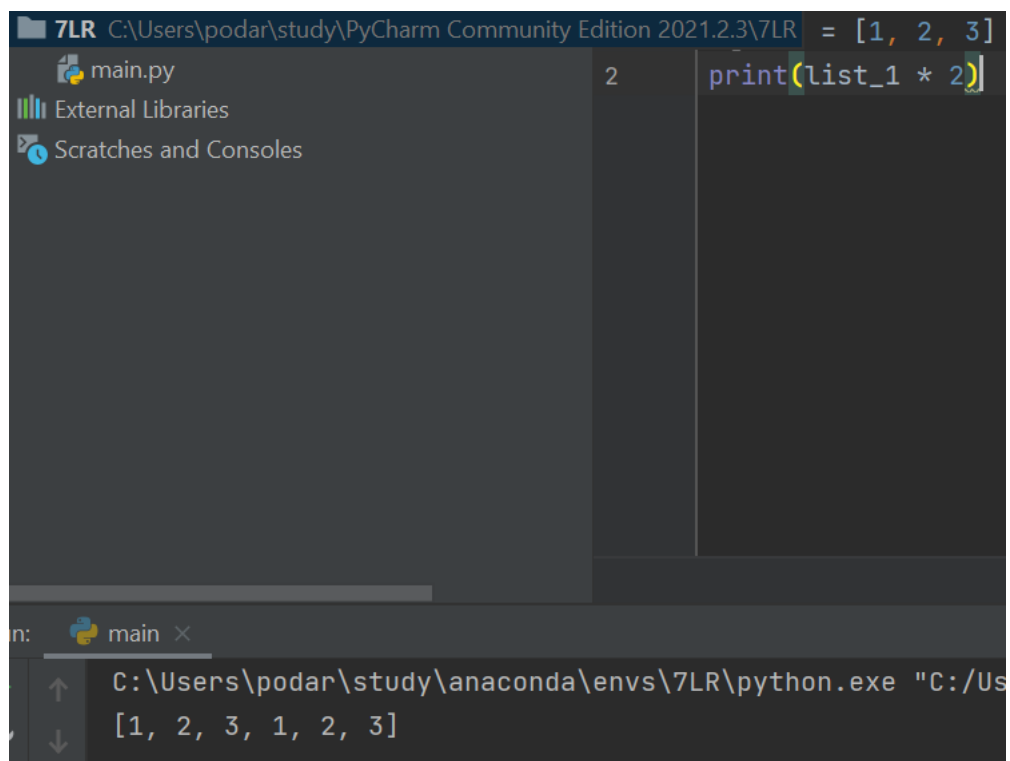


Рисунок 5 – Пример операции умножения

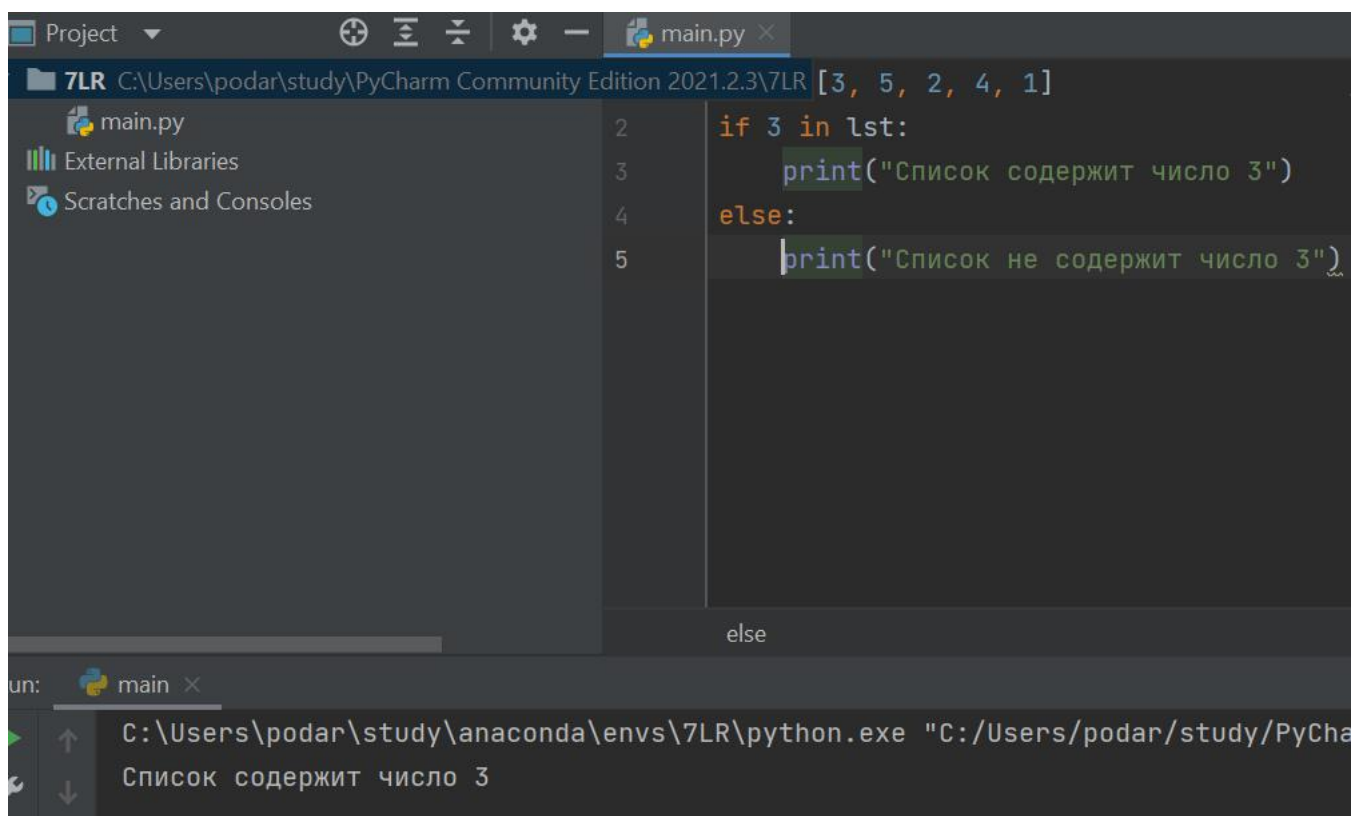
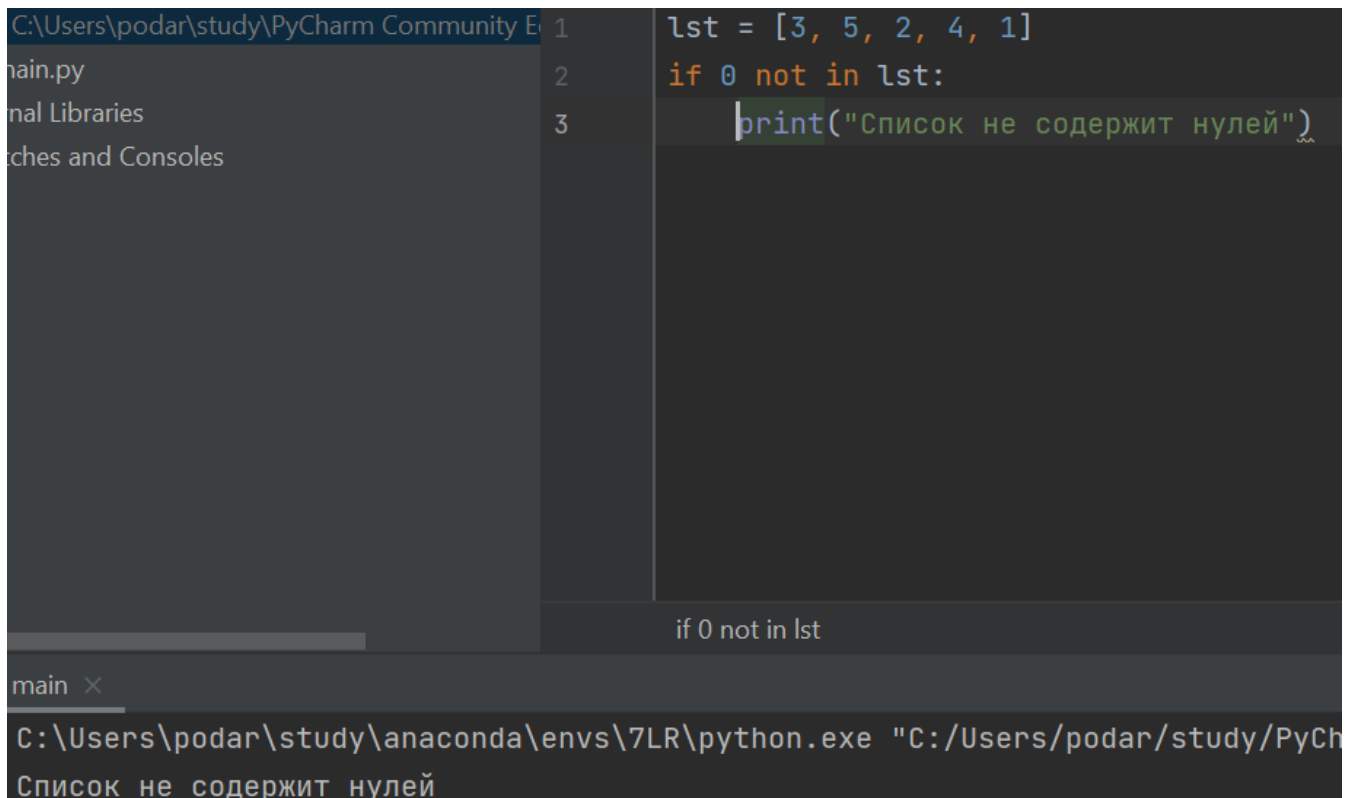


Рисунок 6 – Пример поиска элемента в списке



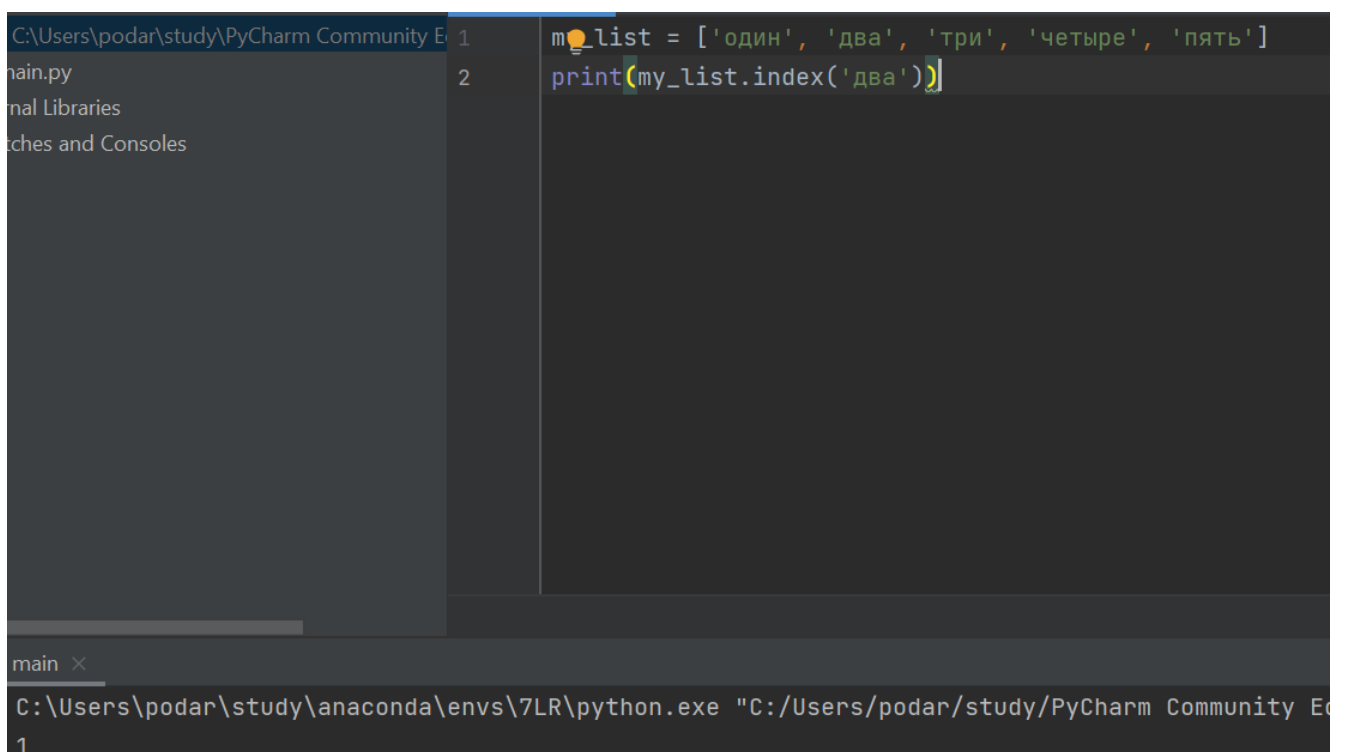
```
C:\Users\podar\study\PyCharm Community Edition>
main.py
External Libraries
Run and Debug
Run and Consoles

1 lst = [3, 5, 2, 4, 1]
2 if 0 not in lst:
3     print("Список не содержит нулей")

if 0 not in lst

main x
C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/study/PyCharm Community Edition/main.py"
Список не содержит нулей
```

Рисунок 7 – Пример поиска элемента в списке с помощью оператора not

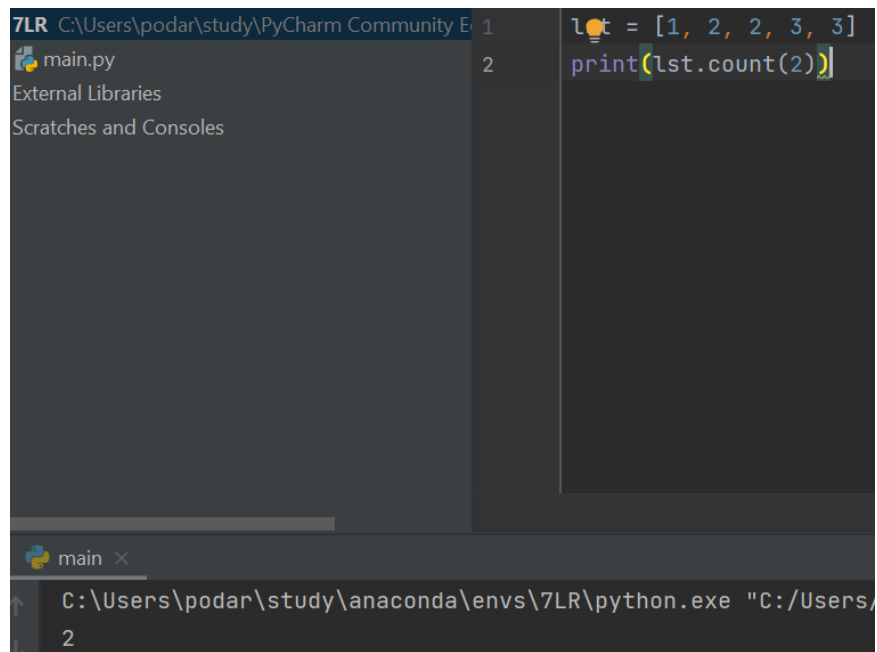


```
C:\Users\podar\study\PyCharm Community Edition>
main.py
External Libraries
Run and Debug
Run and Consoles

1 my_list = ['один', 'два', 'три', 'четыре', 'пять']
2 print(my_list.index('два'))

main x
C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/study/PyCharm Community Edition/main.py"
1
```

Рисунок 8 – Пример индекса элемента в списке



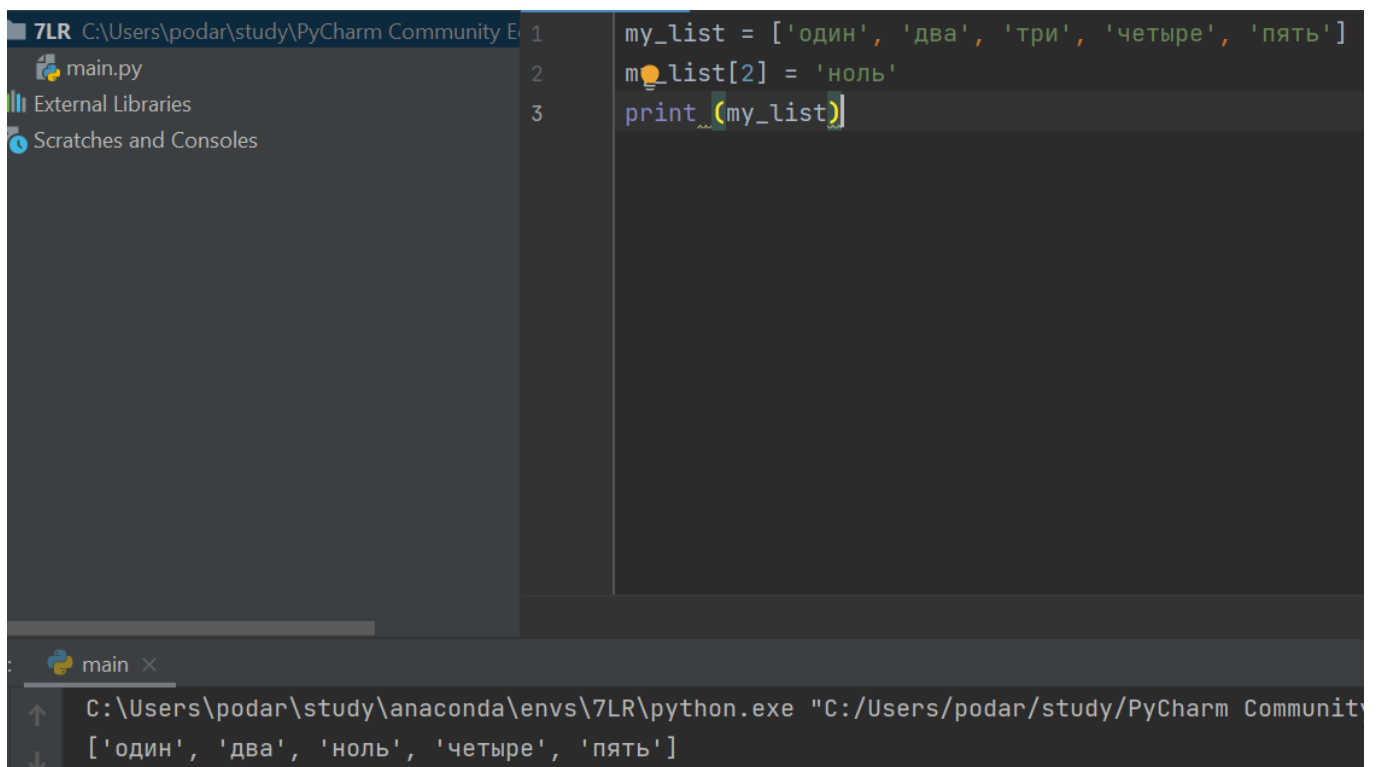
```
7LR C:\Users\podar\study\PyCharm Community E 1 lst = [1, 2, 2, 3, 3]
main.py 2 print(lst.count(2))
External Libraries
Scratches and Consoles
```

main ×

C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/

2

Рисунок 9 – Пример вывода числа вхождений элемента в список



```
7LR C:\Users\podar\study\PyCharm Community E 1 my_list = ['один', 'два', 'три', 'четыре', 'пять']
main.py 2 my_list[2] = 'ноль'
External Libraries 3 print(my_list)
Scratches and Consoles
```

main ×

C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/study/PyCharm Communit

['один', 'два', 'ноль', 'четыре', 'пять']

Рисунок 10 – Пример изменения списка

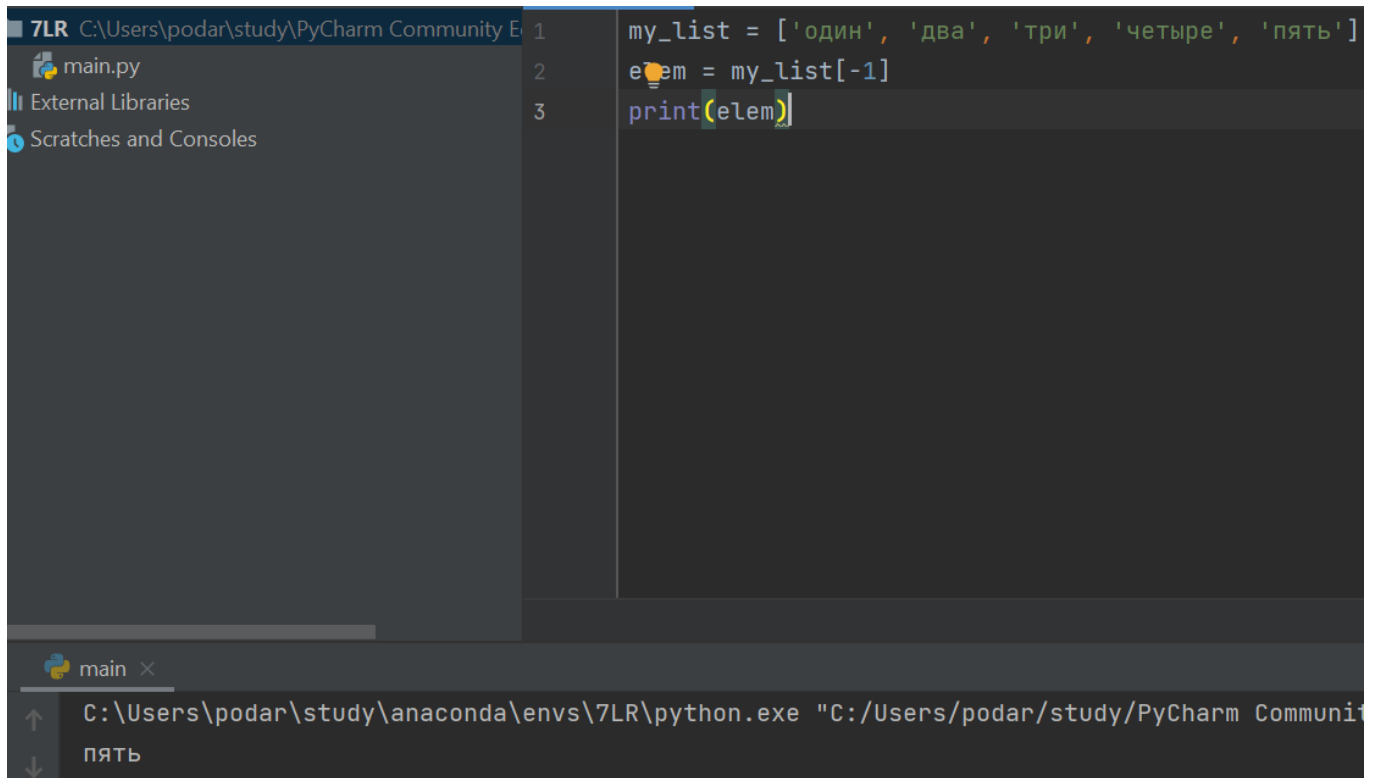


Рисунок 11 – Пример вывода элемента списка с отрицательным индексом

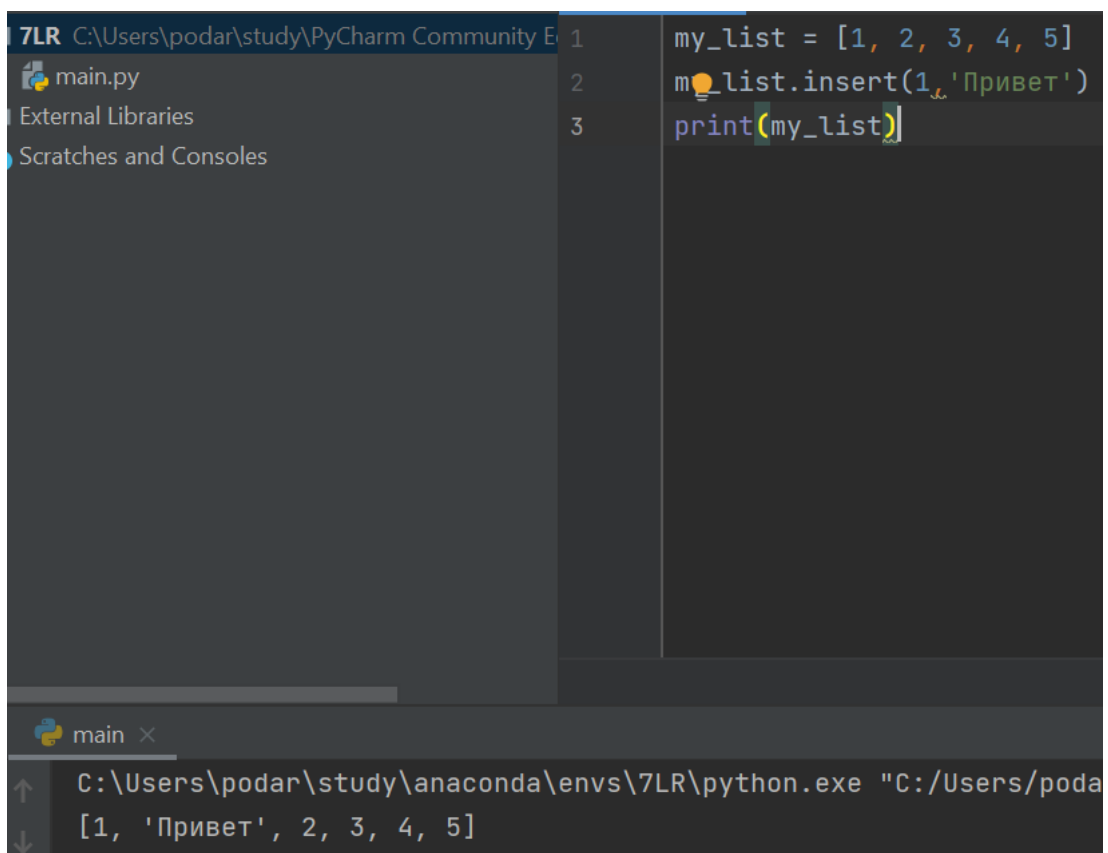
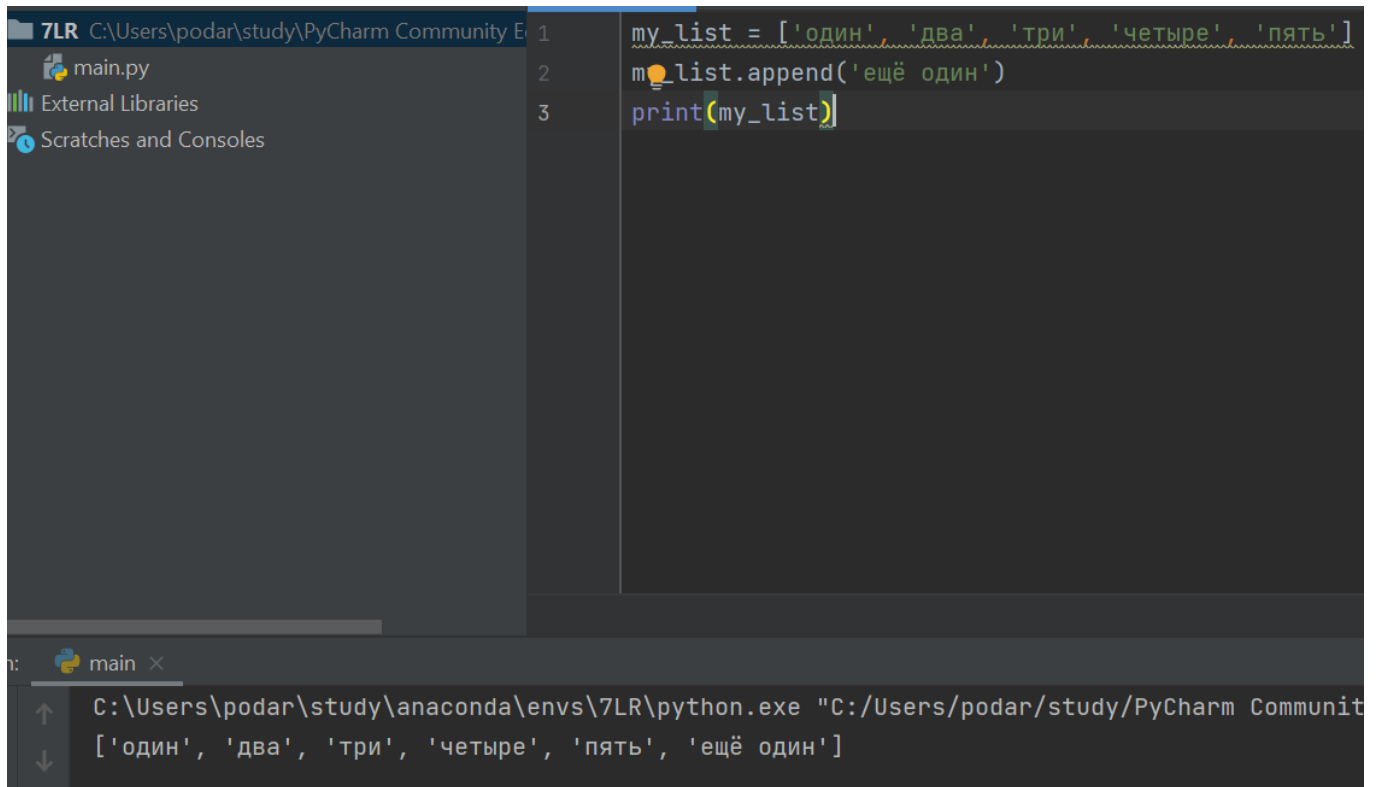


Рисунок 12 – Пример вставки элемента в список



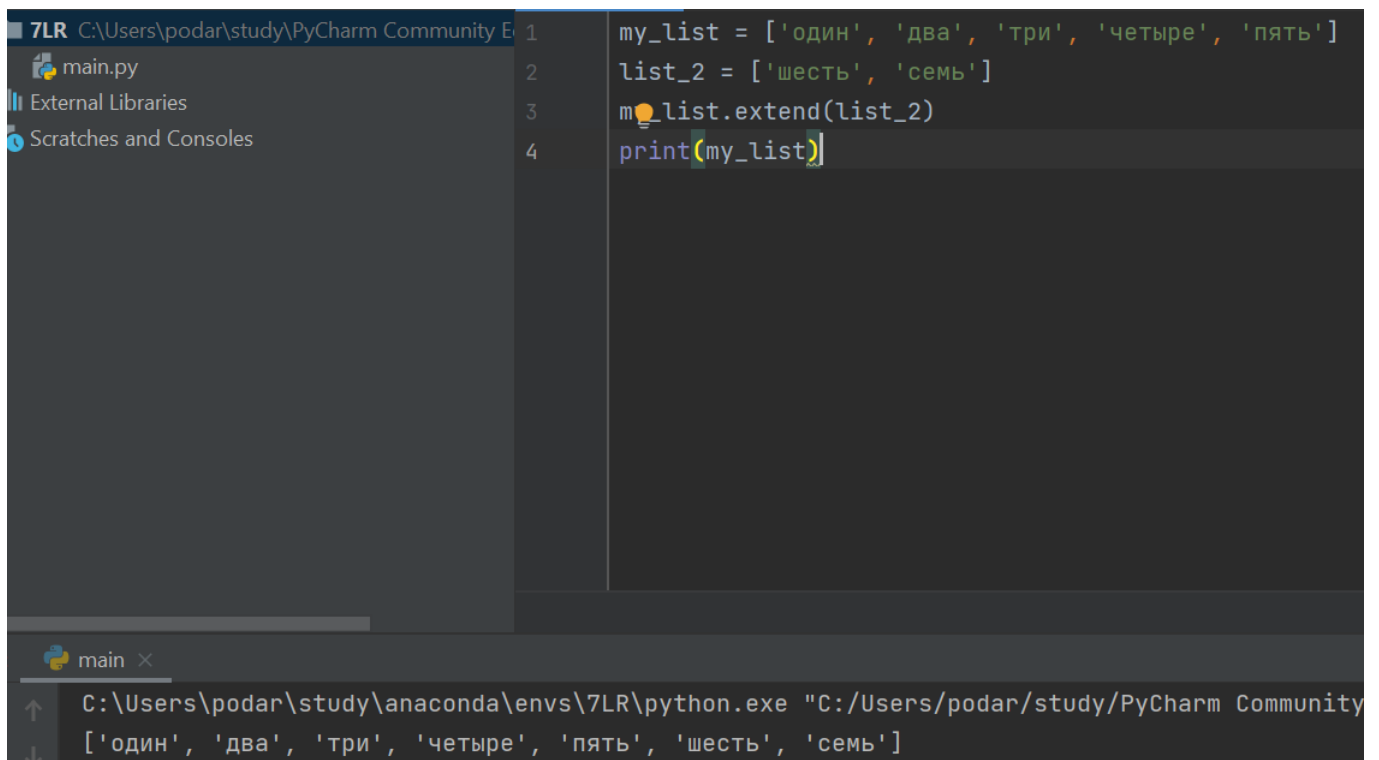
The screenshot shows the PyCharm IDE interface. The left sidebar contains a project view with 'main.py' selected. The main editor window displays the following Python code:

```
1 my_list = ['один', 'два', 'три', 'четыре', 'пять']
2 my_list.append('ещё один')
3 print(my_list)
```

Below the editor, the console window shows the output of the program:

```
main x
C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/study/PyCharm Community
['один', 'два', 'три', 'четыре', 'пять', 'ещё один']
```

Рисунок 13 – Пример добавления элемента в список



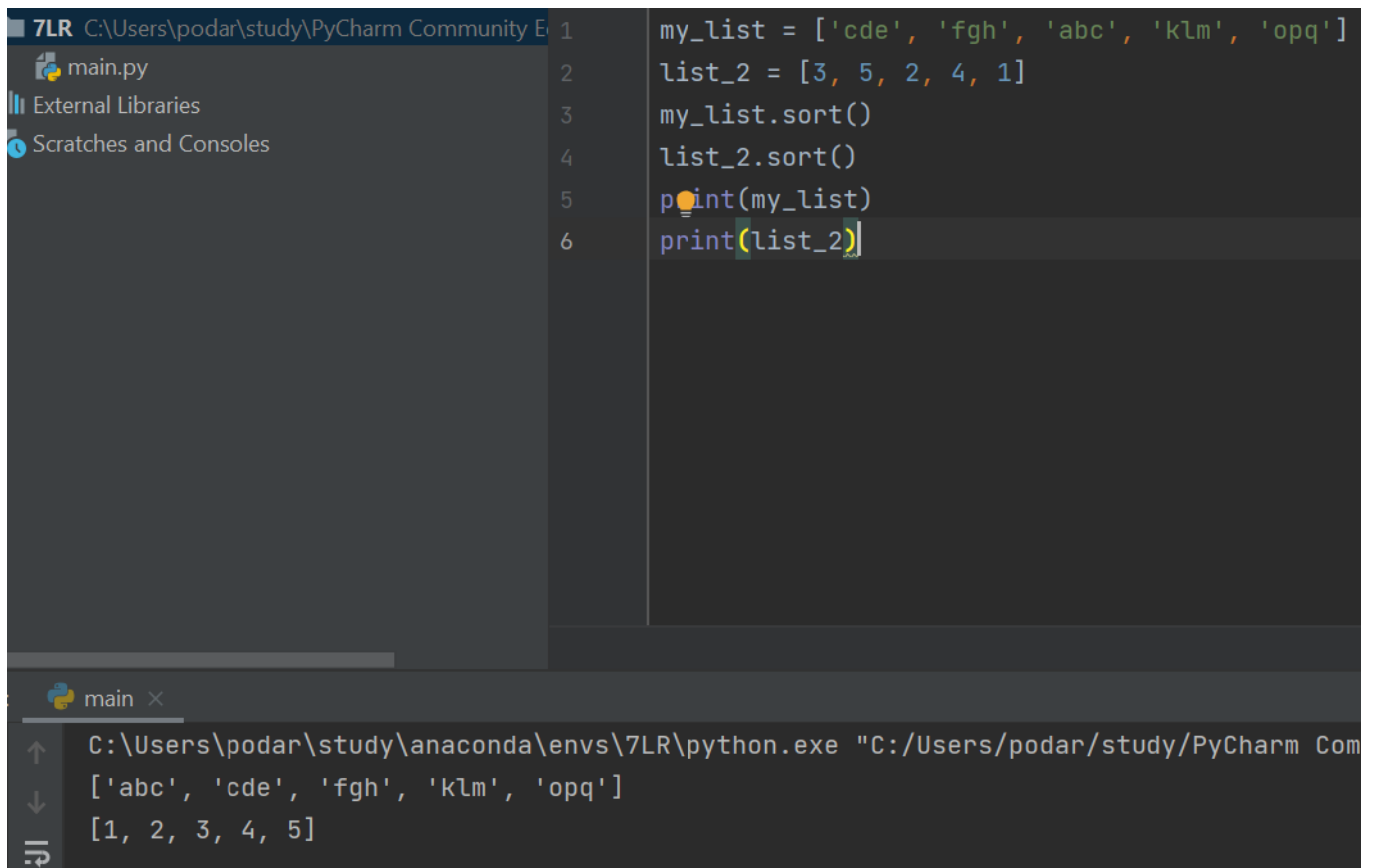
The screenshot shows the PyCharm IDE interface. The left sidebar contains a project view with 'main.py' selected. The main editor window displays the following Python code:

```
1 my_list = ['один', 'два', 'три', 'четыре', 'пять']
2 list_2 = ['шесть', 'семь']
3 my_list.extend(list_2)
4 print(my_list)
```

Below the editor, the console window shows the output of the program:

```
main x
C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/study/PyCharm Community
['один', 'два', 'три', 'четыре', 'пять', 'шесть', 'семь']
```

Рисунок 14 – Пример добавления элементов в список, объединяя два списка



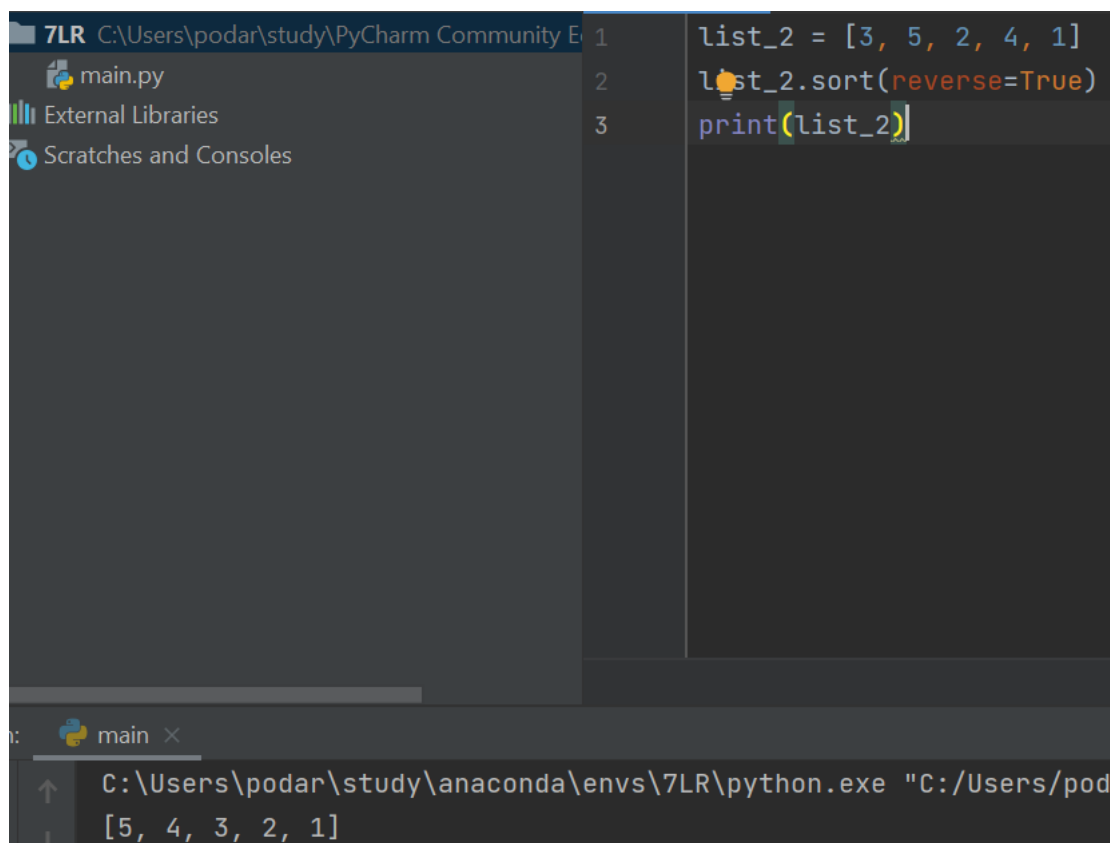
The screenshot shows the PyCharm IDE interface. The left sidebar contains a project tree with 'main.py' selected. The main editor window displays a Python script with the following code:

```
1 my_list = ['cde', 'fgh', 'abc', 'klm', 'opq']
2 list_2 = [3, 5, 2, 4, 1]
3 my_list.sort()
4 list_2.sort()
5 print(my_list)
6 print(list_2)
```

The bottom console window shows the output of the script:

```
main x
C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/study/PyCharm Com
['abc', 'cde', 'fgh', 'klm', 'opq']
[1, 2, 3, 4, 5]
```

Рисунок 15 – Пример сортировки элементов



The screenshot shows the PyCharm IDE interface. The left sidebar contains a project tree with 'main.py' selected. The main editor window displays a Python script with the following code:

```
1 list_2 = [3, 5, 2, 4, 1]
2 list_2.sort(reverse=True)
3 print(list_2)
```

The bottom console window shows the output of the script:

```
main x
C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar
[5, 4, 3, 2, 1]
```

Рисунок 16 – Пример сортировки чисел по убыванию

```
1 my_list = [1, 2, 3, 4, 5]
2 my_list.reverse()
3 print(my_list)
```

main

C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/...

[5, 4, 3, 2, 1]

Рисунок 17 – Пример перевёрнутого списка

```
1 my_list = ['один', 'два', 'три', 'четыре', 'пять']
2 removed = my_list.pop(2)
3 print(my_list)
4 print(removed)
```

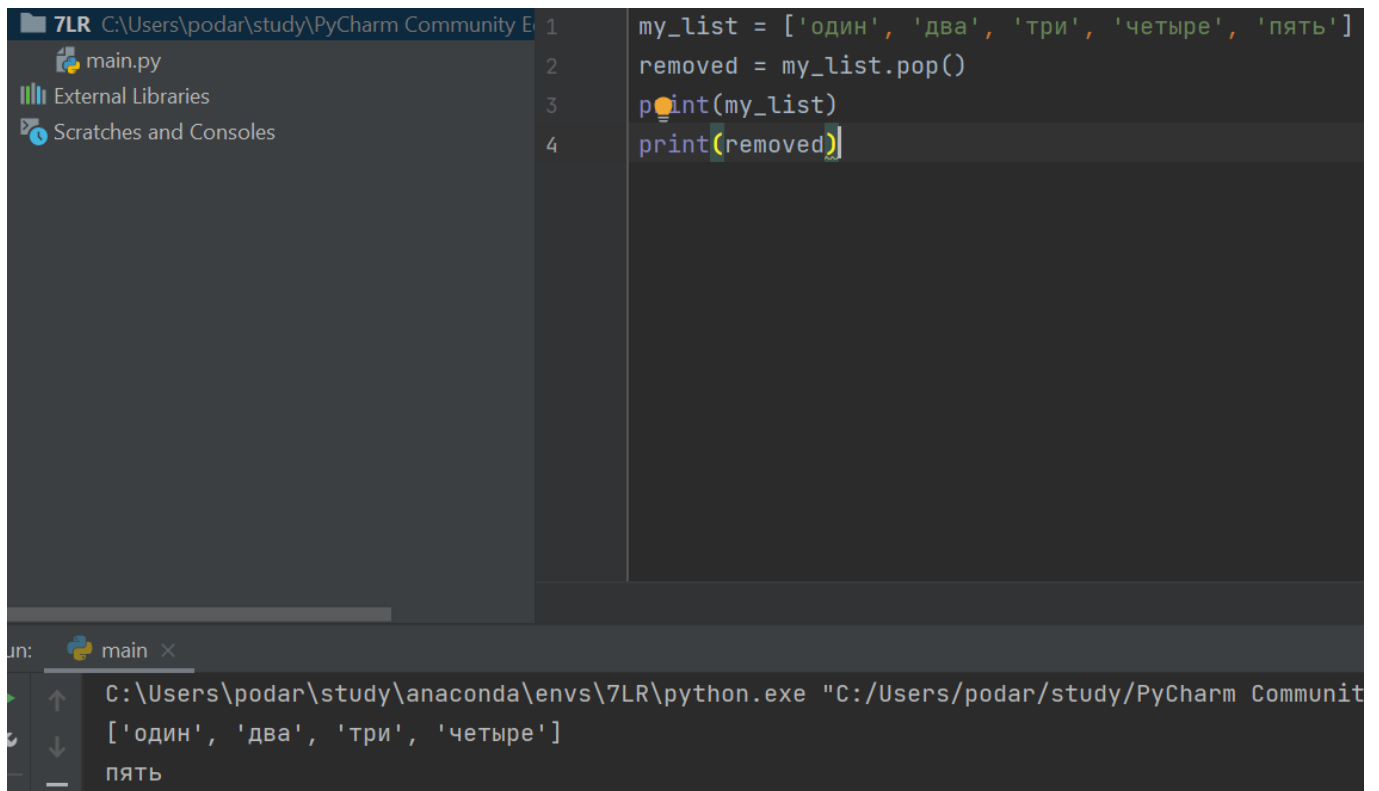
main

C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/study/PyCharm Communi...

['один', 'два', 'четыре', 'пять']

три

Рисунок 18 – Пример удаление элементов из списка



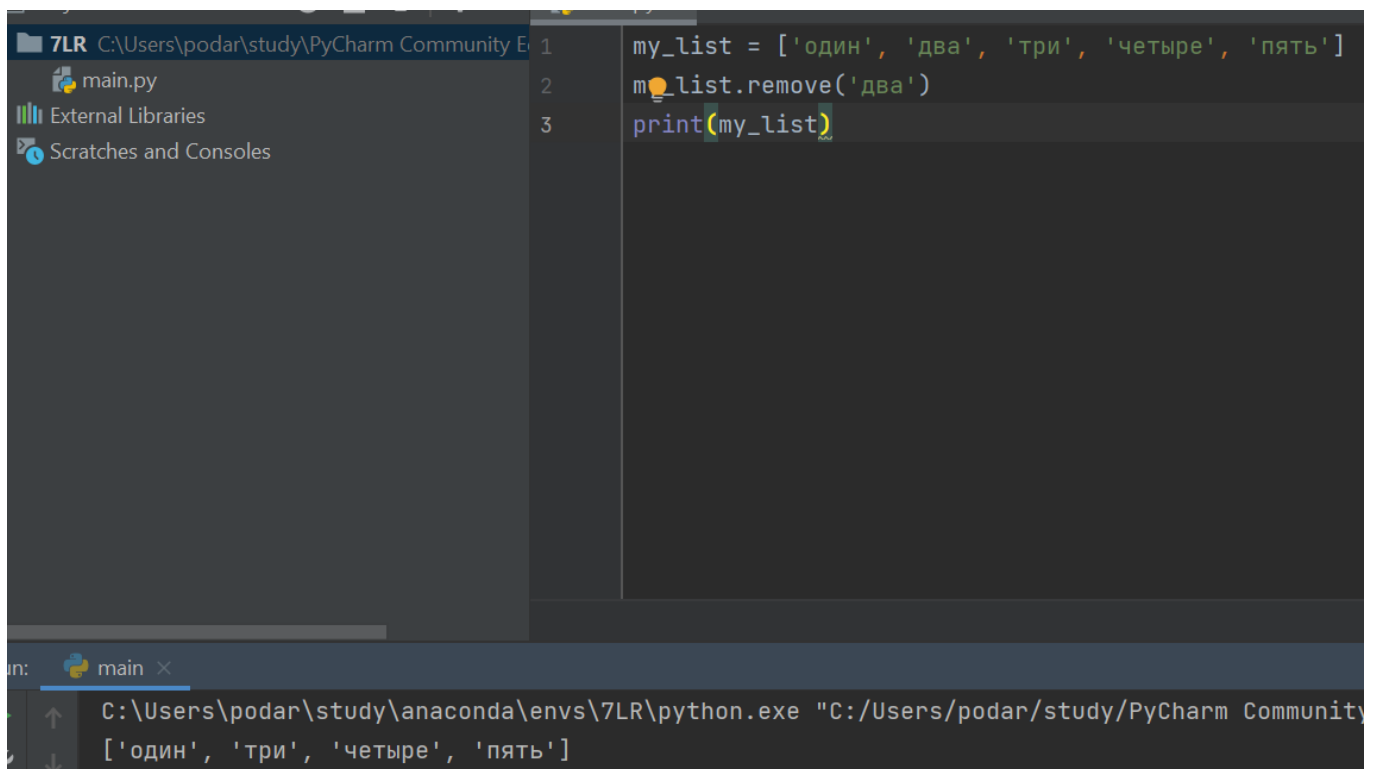
The screenshot shows the PyCharm IDE interface. On the left, the 'Project' tool window displays a file named 'main.py'. The main editor window shows the following Python code:

```
1 my_list = ['один', 'два', 'три', 'четыре', 'пять']
2 removed = my_list.pop()
3 print(my_list)
4 print(removed)
```

Below the editor, the 'Run' tool window is open, showing the command executed: `C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/study/PyCharm Community`. The output of the program is displayed as:

```
['один', 'два', 'три', 'четыре']
пять
```

Рисунок 19 – Пример удаления элемента не указывая индекс



The screenshot shows the PyCharm IDE interface. On the left, the 'Project' tool window displays a file named 'main.py'. The main editor window shows the following Python code:

```
1 my_list = ['один', 'два', 'три', 'четыре', 'пять']
2 my_list.remove('два')
3 print(my_list)
```

Below the editor, the 'Run' tool window is open, showing the command executed: `C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/study/PyCharm Community`. The output of the program is displayed as:

```
['один', 'три', 'четыре', 'пять']
```

Рисунок 20 – Пример удаления элемента с помощью метода remove

The screenshot shows the PyCharm IDE interface. The left sidebar contains a project tree with 'main.py' selected. The main editor window displays the following Python code:

```
1 my_list = ['один', 'два', 'три', 'четыре', 'пять']
2 del my_list[2]
3 print(my_list)
```

Below the editor, the console window shows the output of the program:

```
main x
C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/study/PyCharm Community
['один', 'два', 'четыре', 'пять']
```

Рисунок 21 – Пример удаления элемента с помощью оператора del

The screenshot shows the PyCharm IDE interface. The left sidebar contains a project tree with 'main.py' selected. The main editor window displays the following Python code:

```
1 my_list = ['один', 'два', 'три', 'четыре', 'пять']
2 del my_list[1:3]
3 print(my_list)
```

Below the editor, the console window shows the output of the program:

```
main x
C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/study/PyCharm Community
['один', 'четыре', 'пять']
```

Рисунок 22 – Пример удаления нескольких элементов с помощью оператора среза

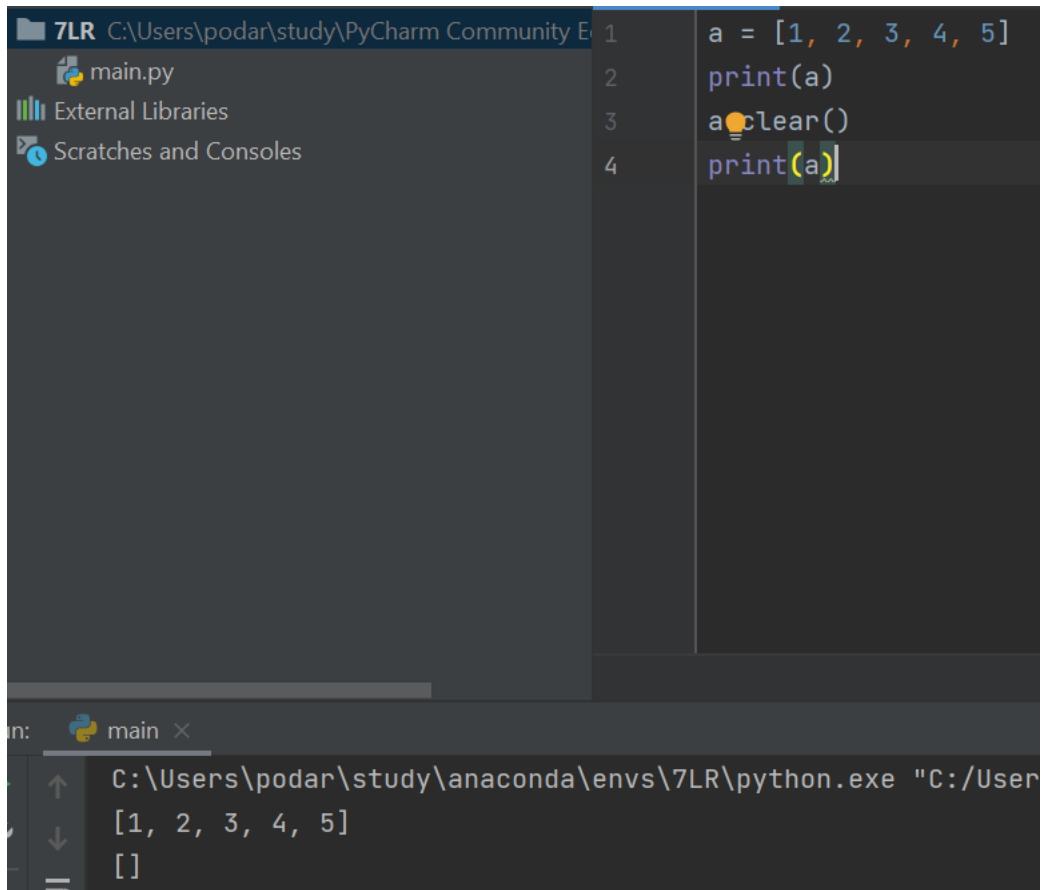


Рисунок 23 – Пример удаления всех элементов из списка

```
>>> n = int(input())
7
>>> a=[]
>>> for i in range(n):
...     a.append(i)
...
>>> print(a)
[0, 1, 2, 3, 4, 5, 6]
>>>
```

Рисунок 24 – Классический пример создания последовательности

```
>>> n = int(input())
7
>>> a = [i for i in range(n)]
>>> print(a)
[0, 1, 2, 3, 4, 5, 6]
>>>
```

Рисунок 25 – Пример работы list comprehensions

```
>>> a = [i for i in range(int(input()))]
7
>>> print(a)
[0, 1, 2, 3, 4, 5, 6]
>>>
```

Рисунок 26 – Пример работы list comprehensions, если больше не нужно использовать n

```
>>> a = [1, 2, 3, 4, 5, 6, 7]
>>> b = []
>>> for i in a:
...     b.append(i**2)
...
>>> print('a = {}\nb = {}'.format(a, b))
a = [1, 2, 3, 4, 5, 6, 7]
b = [1, 4, 9, 16, 25, 36, 49]
```

Рисунок 27 – Пример работы list comprehensions как обработчика списков

```
>>> a = [1, 2, 3, 4, 5, 6, 7]
>>> b = list(map(lambda x: x**2, a))
>>> print('a = {}\nb = {}'.format(a, b))
a = [1, 2, 3, 4, 5, 6, 7]
b = [1, 4, 9, 16, 25, 36, 49]
```

Рисунок 28 – Пример работы list comprehensions с использованием map

```
>>> a = [1, 2, 3, 4, 5, 6, 7]
>>> b = [i**2 for i in a]
>>> print('a = {}\nb = {}'.format(a, b))
a = [1, 2, 3, 4, 5, 6, 7]
b = [1, 4, 9, 16, 25, 36, 49]
```

Рисунок 29 – Пример работы list comprehensions и решением задачи через списковое включение

```
>>> a = [1, 2, 3, 4, 5, 6, 7]
>>> b = []
>>> for i in a:
...     if i%2 == 0:
...         b.append(i)
...
>>> print('a = {}\nb = {}'.format(a, b))
a = [1, 2, 3, 4, 5, 6, 7]
b = [2, 4, 6]
```

Рисунок 30 – Пример построения на базе существующего списка нового, состоящего только из чётных чисел

```
>>> a = [1, 2, 3, 4, 5, 6, 7]
>>> b = list(filter(lambda x: x % 2 == 0, a))
>>> print('a = {}\nb = {}'.format(a, b))
a = [1, 2, 3, 4, 5, 6, 7]
b = [2, 4, 6]
```

Рисунок 31 – Пример решения задачи с использованием filter

```
>>> a = [1, 2, 3, 4, 5, 6, 7]
>>> b = [i for i in a if i % 2 == 0]
>>> print('a = {}\nb = {}'.format(a, b))
a = [1, 2, 3, 4, 5, 6, 7]
b = [2, 4, 6]
>>>
```

Рисунок 32 – Пример решения задачи через списковое включение

```
>>> a = [i for i in range(10)]
>>> a[:]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> a[0:5]
[0, 1, 2, 3, 4]
>>> a[2:7]
[2, 3, 4, 5, 6]
>>> a[:2]
[0, 1]
>>> a[::2]
[0, 2, 4, 6, 8]
>>> a[1:8:2]
[1, 3, 5, 7]
```

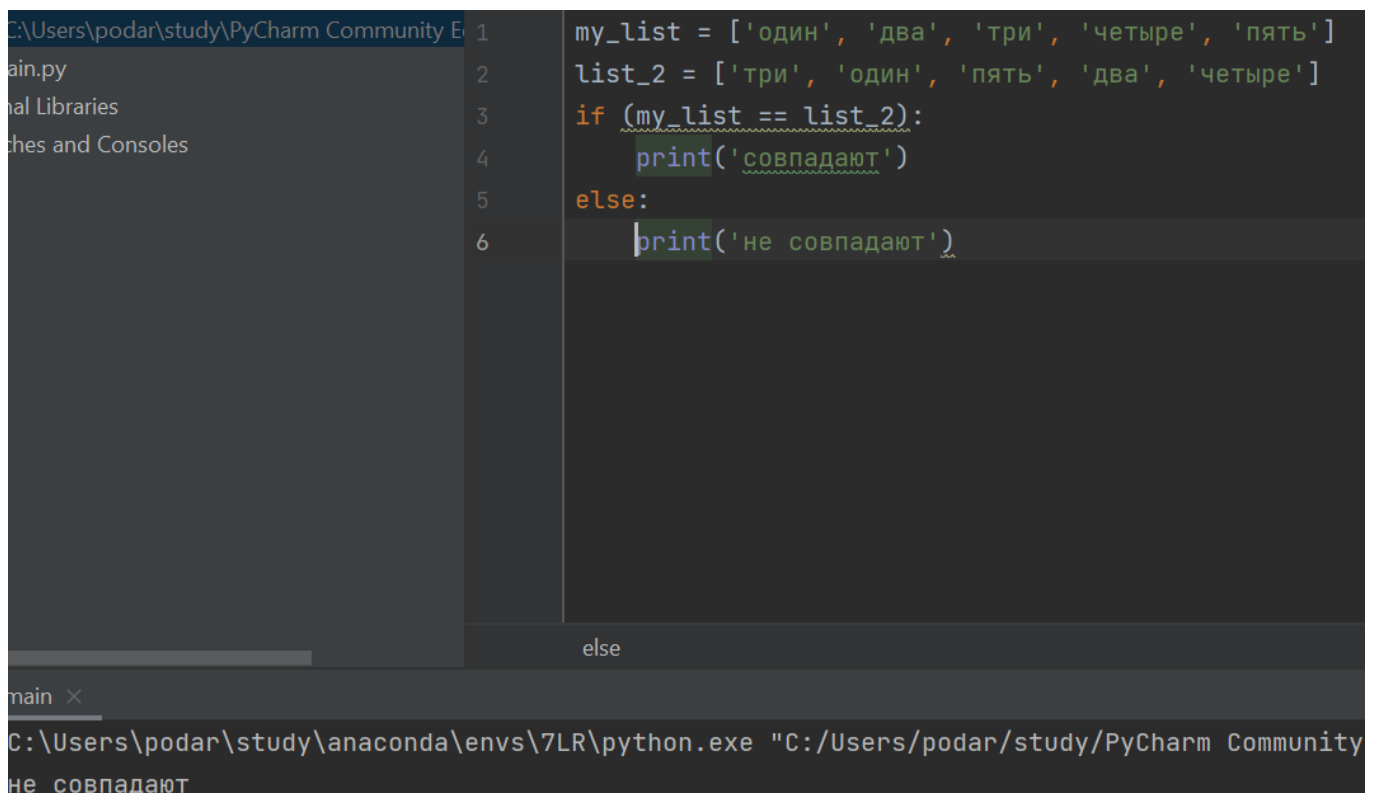
Рисунок 33 – Пример работы слайсов и срезов

```
>>> s = slice(0, 5, 1)
>>> a[s]
[0, 1, 2, 3, 4]
>>> s = slice(1, 8, 2)
>>> a[s]
[1, 3, 5, 7]
```

Рисунок 34 – Пример работы слайса

```
>>> my_list = [5, 3, 2, 4, 1]
>>> print(len(my_list))
5
>>> print(min(my_list))
1
>>> print(max(my_list))
5
>>> print(sum(my_list))
15
```

Рисунок 35 – Пример работы функций агрегации



```
C:\Users\podar\study\PyCharm Community Edition
main.py
Python Libraries
Python Files and Consoles

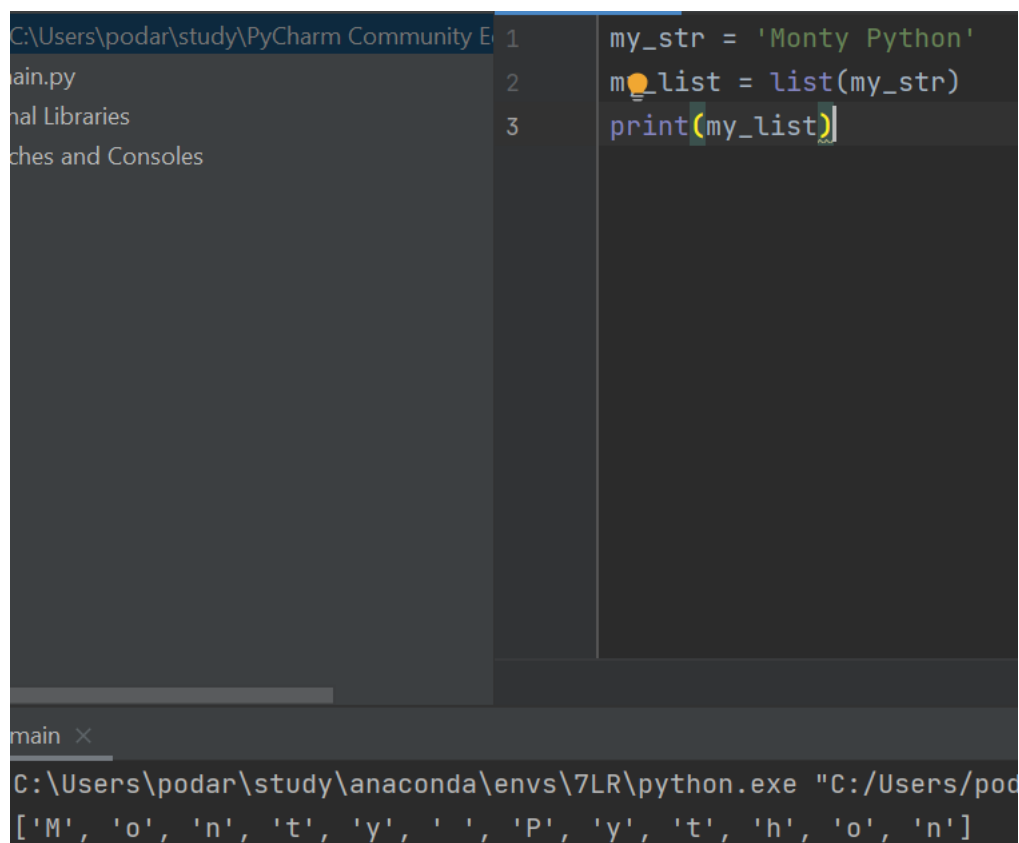
1 my_list = ['один', 'два', 'три', 'четыре', 'пять']
2 list_2 = ['три', 'один', 'пять', 'два', 'четыре']
3 if (my_list == list_2):
4     print('совпадают')
5 else:
6     print('не совпадают')
```

main ×

C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/study/PyCharm Community Edition/main.py"

не совпадают

Рисунок 36 – Пример сравнения списков



```
C:\Users\podar\study\PyCharm Community Edition
main.py
Python Libraries
Python Files and Consoles

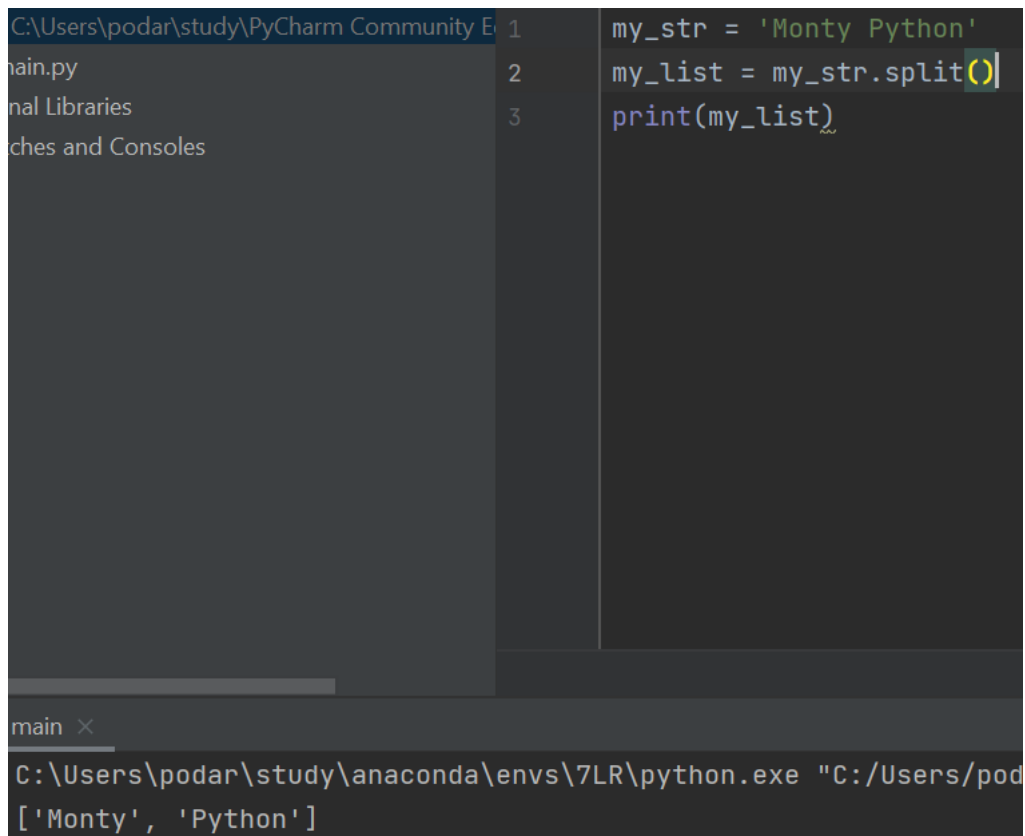
1 my_str = 'Monty Python'
2 my_list = list(my_str)
3 print(my_list)
```

main ×

C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/study/PyCharm Community Edition/main.py"

['M', 'o', 'n', 't', 'y', ' ', 'P', 'y', 't', 'h', 'o', 'n']

Рисунок 37 – Пример конвертации строки в набор символов с использованием функции list

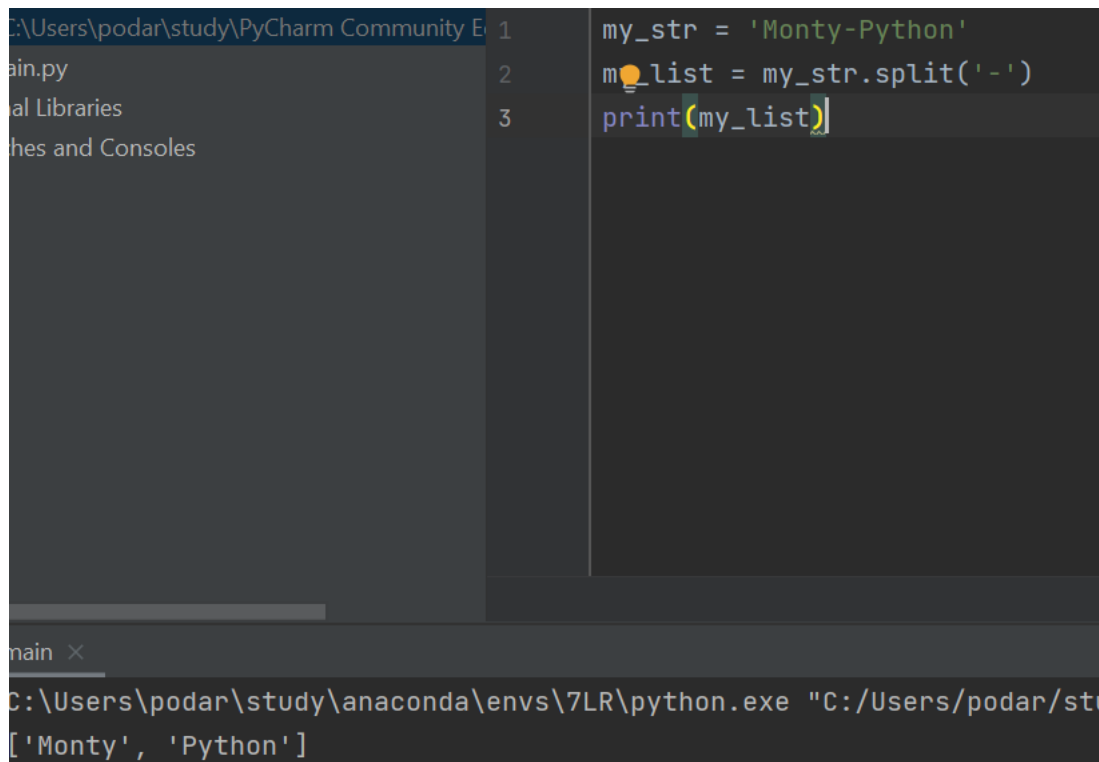


```
C:\Users\podar\study\PyCharm Community Edition 1
main.py
Python Libraries
Python Files and Consoles

1 my_str = 'Monty Python'
2 my_list = my_str.split()
3 print(my_list)

main x
C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/stu
['Monty', 'Python']
```

Рисунок 38 – Пример использования метода split для разбиения строки на слова

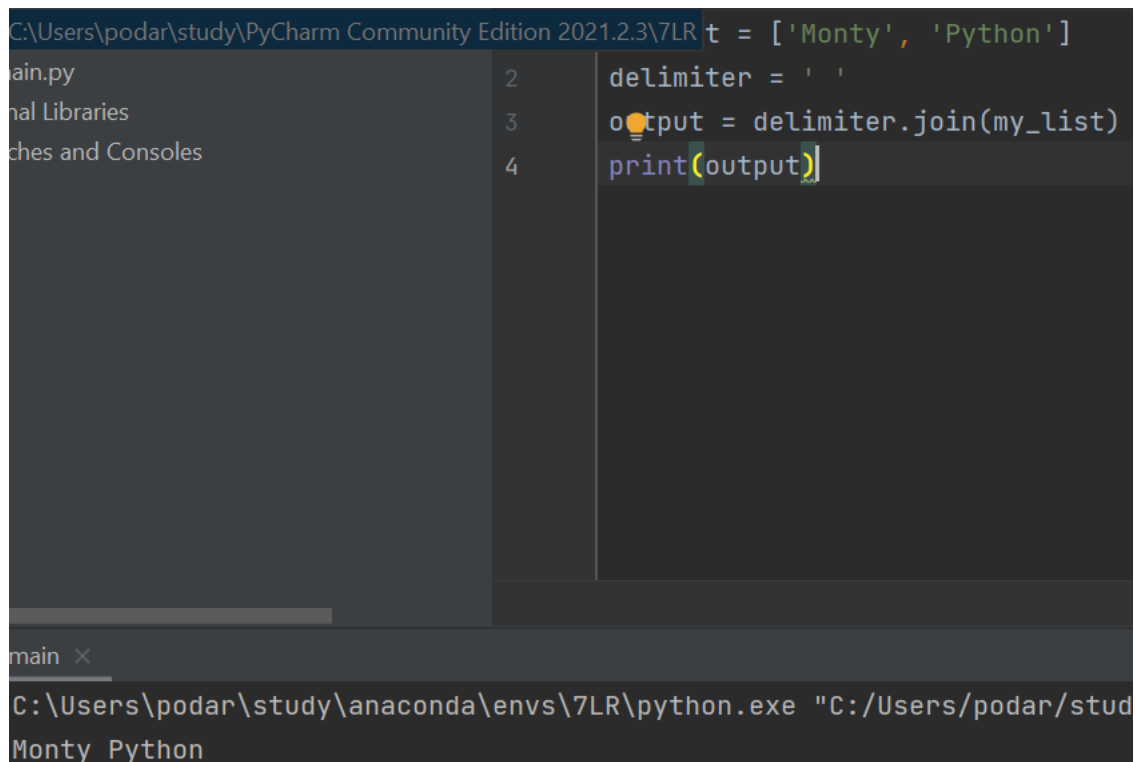


```
C:\Users\podar\study\PyCharm Community Edition 1
main.py
Python Libraries
Python Files and Consoles

1 my_str = 'Monty-Python'
2 my_list = my_str.split('-')
3 print(my_list)

main x
C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/stu
['Monty', 'Python']
```

Рисунок 39 – Пример использования метода split для разбиения строки на слова, где символом разбиения служит знак «-»

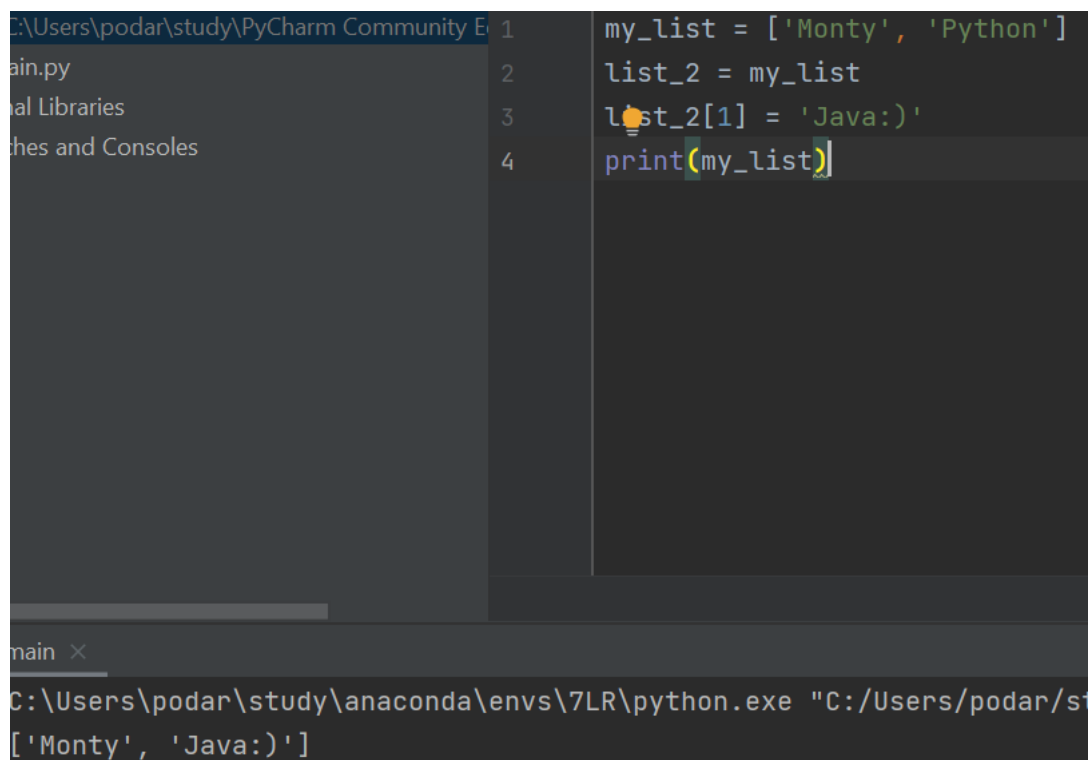


The screenshot shows the PyCharm IDE with a Python file named `main.py`. The code defines a list `my_list = ['Monty', 'Python']`, sets a `delimiter = ' '`, and then uses `output = delimiter.join(my_list)` to join the list elements. Finally, `print(output)` is used to display the result. The console output at the bottom shows the command `C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/stud` and the output `Monty Python`.

```
C:\Users\podar\study\PyCharm Community Edition 2021.2.3\7LR t = ['Monty', 'Python']
main.py
2 delimiter = ' '
3 output = delimiter.join(my_list)
4 print(output)

main x
C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/stud
Monty Python
```

Рисунок 40 – Пример объединения элементов списка в строку



The screenshot shows the PyCharm IDE with a Python file named `main.py`. The code defines a list `my_list = ['Monty', 'Python']`, creates a new list `list_2 = my_list`, and then modifies the second element of `list_2` to `'Java:)'` using `list_2[1] = 'Java:)'`. Finally, `print(my_list)` is used to display the result. The console output at the bottom shows the command `C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/s` and the output `['Monty', 'Java:)]`.

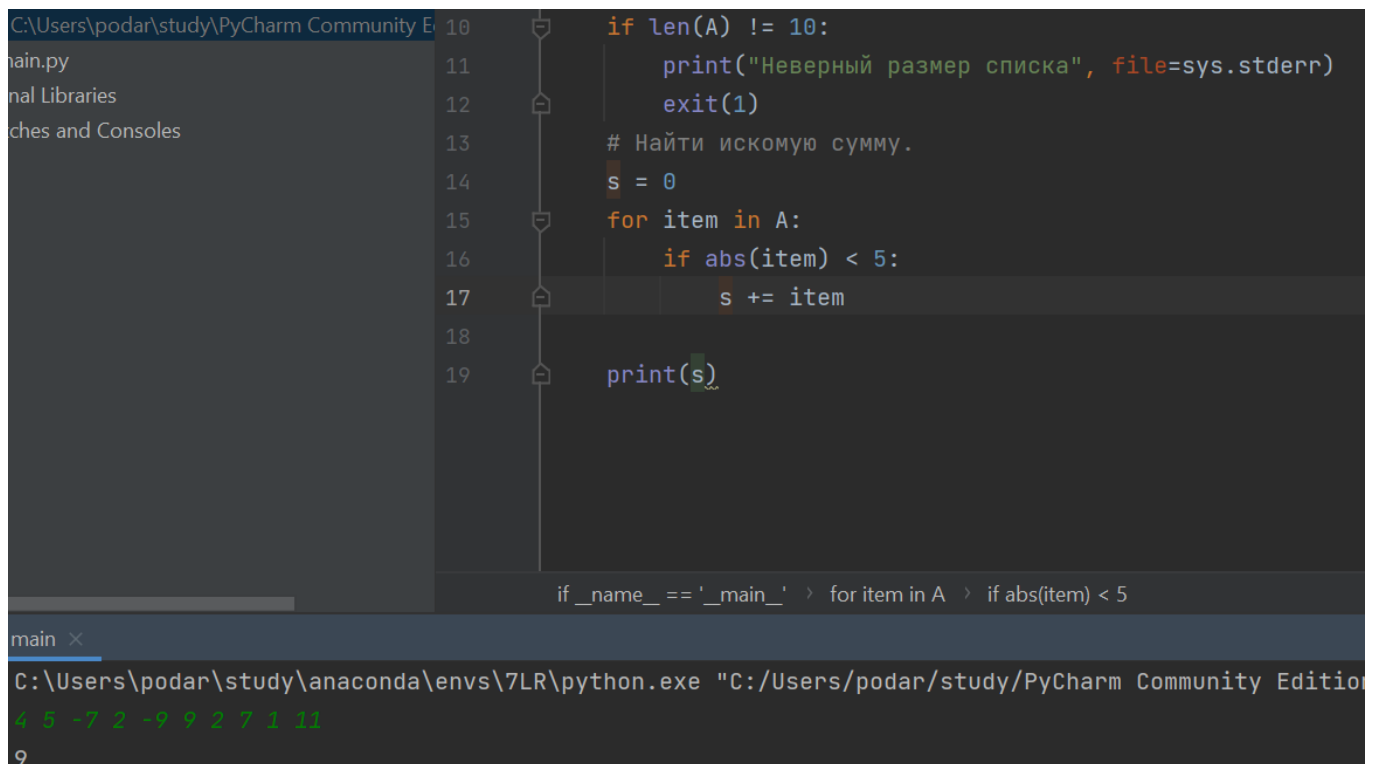
```
C:\Users\podar\study\PyCharm Community E 1 my_list = ['Monty', 'Python']
main.py
2 list_2 = my_list
3 list_2[1] = 'Java:)'
4 print(my_list)

main x
C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/s
['Monty', 'Java:)]
```

Рисунок 41 – Пример объединения элементов списка в строку

```
>>> a = [1, 2, 3, 4, 5]
>>> b = a.copy()
>>> b
[1, 2, 3, 4, 5]
>>> a == b
True
>>> a is b
False
>>> a is not b
True
```

Рисунок 42 – Пример создания копии списка с использованием метода copy



The screenshot shows the PyCharm IDE interface. The main editor window displays a Python script with the following code:

```
10 if len(A) != 10:
11     print("Неверный размер списка", file=sys.stderr)
12     exit(1)
13 # Найти искомую сумму.
14 s = 0
15 for item in A:
16     if abs(item) < 5:
17         s += item
18
19 print(s)
```

The left sidebar shows the project structure with files like 'main.py'. The bottom panel shows the execution output for the script 'main.py'.

Execution Command: C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/study/PyCharm Community Edition"

Output:

```
4 5 -7 2 -9 9 2 7 1 11
9
```

Рисунок 43 – Пример решения задачи вывода и суммы элементов списка

```
import sys

if __name__ == '__main__':
    # Ввести список одной строкой.
    A = list(map(int, input().split()))
    # Проверить количество элементов списка.
    if len(A) != 10:
        print("Неверный размер списка", file=sys.stderr)
        exit(1)

    # Найти искомую сумму.
    s = sum([a for a in A if abs(a) < 5])
    print(s)
```

if __name__ == '__main__' > if len(A) != 10

main ×

C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/po

3 -7 8 9 2 27 12 2 9 11

7

Рисунок 44 – Пример решения задачи вывода и суммы элементов списка, с помощью списковых включений

```

7      # Ввести список одной строкой.
8      a = list(map(int, input().split()))
9      # Если список пуст, завершить программу.
10     if not a:
11         print("Заданный список пуст", file=sys.stderr)
12         exit(1)
13
14     # Определить индексы минимального и максимального элементов.
15     a_min = a_max = a[0]
16     i_min = i_max = 0
17     for i, item in enumerate(a):
18         if item < a_min:
19             i_min, a_min = i, item
20
21         if item >= a_max:
22             i_max, a_max = i, item
23
24     # Проверить индексы и обменять их местами.
25     if i_min > i_max:
26         i_min, i_max = i_max, i_min
27
28     if __name__ == '__main__':
29
30

```

main x

C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/study/

12 7 5 2

2

Рисунок 45 – Пример нахождения количества элементов между максимальных и минимальных элементов списка

Индивидуальное задание 1.

Ввести список А из 10 элементов, найти квадраты элементов кратных 4 и их количество.

Преобразованный массив вывести.

```
#!/usr/bin/env python3
```

```
# -*- coding: utf-8 -*-
```

```
import sys
```

```
if __name__ == '__main__':
```

```
    a = list(map(int, input().split()))
```

```
    if len(a) != 10:
```

```
        print("Неверный размер списка", file=sys.stderr)
```

```
        exit(1)
```

```
s = 0
```

```

b = []
for item in a:
    if (item % 4 == 0):
        s += 1
        b.append(item**2)
    else :
        b.append(item)

print('количество возведённых в квадрат',s,'\nb ={ }'.format(b))

```

The screenshot shows a Python IDE with a file named `1zd.py`. The code in the editor is as follows:

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == '__main__':
    a = list(map(int, input().split()))
    if len(a) != 10:
        print("Неверный размер списка", file=sys.stderr)
        exit(1)

    s = 0
    b = []
    for item in a:
        if (item % 4 == 0):
            s += 1
            b.append(item**2)
        else:
            b.append(item)

    print('количество возведённых в квадрат', s, '\nb = { }'.format(b))

```

The output window shows the following results:

```

C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/study/PyCharm/1zd.py"
1 2 3 4 4 4 5 1 1 1
количество возведённых в квадрат 3
b = [1, 2, 3, 16, 16, 16, 5, 1, 1, 1]

```

Рисунок 46 – Результат решения

Индивидуальное задание 2.

В списке, состоящем из вещественных элементов, вычислить:

1. количество элементов списка, меньших C ;
2. сумму целых частей элементов списка, расположенных после последнего отрицательного элемента.

Преобразовать список таким образом, чтобы сначала располагались все элементы, отличающиеся от максимального не более чем на 20%, а потом - все остальные.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

```
if __name__ == '__main__':
    a = list(map(float, input("Введите элементы массива: ").split()))
    c = float(input("Введите число C: "))

    s = 0
    for item in a:
        if item < c:
            s += 1

    print ('Количество элементов меньше C: ', s)

    sum = 0
    a.reverse()
    for item in a:
        if item < 0:
            break
        sum += int(item)

    print(f"Сумма элементов, после последнего отрицательного: {sum}")

    max_a = max(a)
    b = [num for num in a if num >= 0.8*max_a]
    c = [num for num in a if num < 0.8*max_a]
    print(f"Отсортированный массив: {b+c}")
```

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == '__main__':
    a = list(map(float, input("Введите элементы массива: ").split()))
    c = float(input("Введите число C: "))

    s = 0
    for item in a:
        if item < c:
            s += 1

    print('Количество элементов меньше C: ', s)

    sum = 0
    a.reverse()

```

```
if __name__ == '__main__':
```

main ×

```

C:\Users\podar\study\anaconda\envs\7LR\python.exe "C:/Users/podar/study/PyCharm Communi
Введите элементы массива: 4 5 7 8.1 56 -9 6 7.1 2 6.3 8.1 9.1
Введите число C: 5.1
Количество элементов меньше C: 4
Сумма элементов, после последнего отрицательного: 38
Отсортированный массив: [56.0, 9.1, 8.1, 6.3, 2.0, 7.1, 6.0, -9.0, 8.1, 7.0, 5.0, 4.0]

```

Рисунок 46 - Результат решения

Ответы на контрольные вопросы

1. Что такое списки в языке Python?

Список (list) - структура данных для хранения объектов различных типов.

2. Как осуществляется создание списка в Python?

Для создания списка нужно заключить элементы в квадратные скобки.

3. Как организовано хранение списков в оперативной памяти?

Список является изменяемым типом данных. При его создании в памяти резервируется область, которую можно условно назвать некоторым “контейнером”, в котором хранятся ссылки на другие элементы данных в памяти. В отличие от таких типов данных как число или строка, содержимое “контейнера” списка можно менять.

4. Каким образом можно перебрать все элементы списка?

Элементы можно перебрать с помощью цикла for.

5. Какие существуют арифметические операции со списками?

Списки можно складывать и умножать.

6. Как проверить есть ли элемент в списке?

Проверить есть ли элемент в списке можно с помощью цикла if.

7. Как определить число вхождений заданного элемента в списке?

Чтобы определить число вхождений заданного элемента в списке, нужно воспользоваться методом count().

8. Как осуществляется добавление (вставка) элемента в список?

Метод insert можно использовать, чтобы вставить элемент в список. Метод append можно использовать для добавления элемента в список.

9. Как выполнить сортировку списка?

Для сортировки списка нужно использовать метод sort.

10. Как удалить один или несколько элементов из списка?

Удалить элемент можно, написав его индекс в методе `pop`. Элемент можно удалить с помощью метода `remove`.

Оператор `del` можно также использовать для удаления элемента. Можно удалить все элементы из списка с помощью метода `clear`.

11. Что такое списковое включение и как с его помощью осуществлять обработку списков?

List Comprehensions чаще всего на русский язык переводят как абстракция списков или списковое включение, является частью синтаксиса языка, которая предоставляет простой способ построения списков.

В языке Python есть две очень мощные функции для работы с коллекциями: `map` и `filter`. Они позволяют использовать функциональный стиль программирования, не прибегая к помощи циклов, для работы с такими типами как `list`, `tuple`, `set`, `dict` и т.п. Списковое включение позволяет обойтись без этих функций.

12. Как осуществляется доступ к элементам списков с помощью срезов?

Созданный список:

```
>>> a = [i for i in range(10)]
```

Доступ к его элементам:

```
>>> # Получить копию списка
>>> a[:]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> # Получить первые пять элементов списка
>>> a[0:5]
[0, 1, 2, 3, 4]

>>> # Получить элементы с 3-го по 7-ой
>>> a[2:7]
[2, 3, 4, 5, 6]

>>> # Взять из списка элементы с шагом 2
>>> a[::2]
[0, 2, 4, 6, 8]

>>> # Взять из списка элементы со 2-го по 8-ой с шагом 2
>>> a[1:8:2]
[1, 3, 5, 7]
```

13. Какие существуют функции агрегации для работы со списками?

Для работы со списками Python предоставляет следующие функции: `len(L)` - получить число элементов в списке `L` .

`min(L)` - получить минимальный элемент списка `L` . `max(L)` - получить максимальный элемент списка `L` .

`sum(L)` - получить сумму элементов списка `L` , если список `L` содержит только числовые значения.

14. Как создать копию списка?

- 1) Создать псевдоним. Создать новый список и присвоить значения имеющегося.
- 2) Создать копию, используя метод `copy`.

15. Самостоятельно изучите функцию `sorted` языка Python. В чем её отличие от метода `sort` списков?

Функция `sorted` возвращает новый отсортированный список, который получен из итерируемого объекта, который был передан как аргумент. Функция также поддерживает дополнительные параметры, которые позволяют управлять сортировкой.

```
In [1]: list_of_words = ['one', 'two', 'list', '', 'dict']

In [2]: sorted(list_of_words)
Out[2]: ['', 'dict', 'list', 'one', 'two']
```

