



Национальные проекты
России

МОДУЛЬ 1
ТЕМА 3

Занятие №2

«Строки и методы работы со строками»



Тема: Строки и методы работы со строками.

Цель занятия: Изучение строк как структурных единиц и методов по работе со строками на языке Python.

Глоссарий:

1. **Функция** - фрагмент программного кода, к которому можно обратиться из другого места программы.
2. **Строка** – это упорядоченная последовательность символов, которые используются для хранения и представления текстовой информации.

1. Какую роль выполняет функция **lambda** в языке *Python*?

2. Что такое **функция**?

3. Как можно передавать аргументы функции?

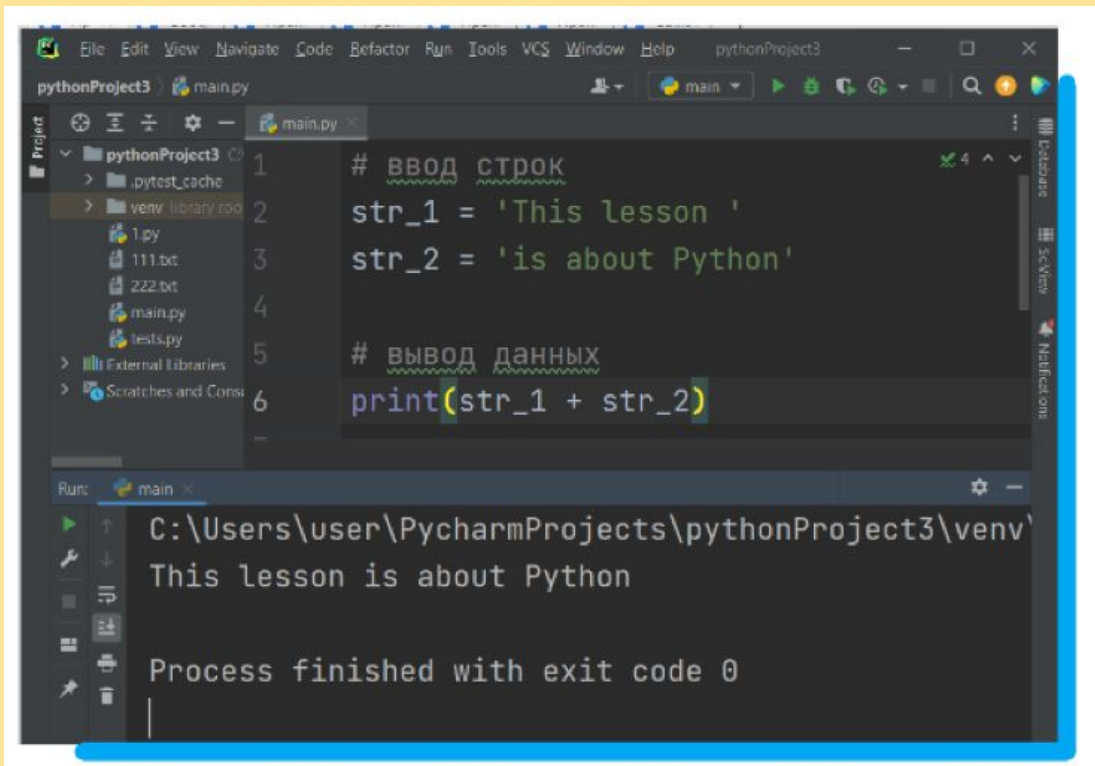
4. Какую роль выполняет функция **filter()** в языке *Python*?

5. Какое максимальное количество аргументов можно передать функции на языке *Python* за один раз?

Работа со строками в языке Python

Строка - набор однотипных элементов, которые объединены под одним именем. Со строками можно производить большое количество операций, рассмотрим самые основные. При помощи строк можно реализовывать различные математические, логические и прочие операции на объектно-ориентированном языке Python. Все строки описываются типом данных **str**. Если имеется некая последовательность элементов (цифр, символов или целых слов), то используется, как правило, тип данных **list**.

Сложение строк (конкатенация). При помощи данной операции можно складывать несколько строк и выводить их вместе:

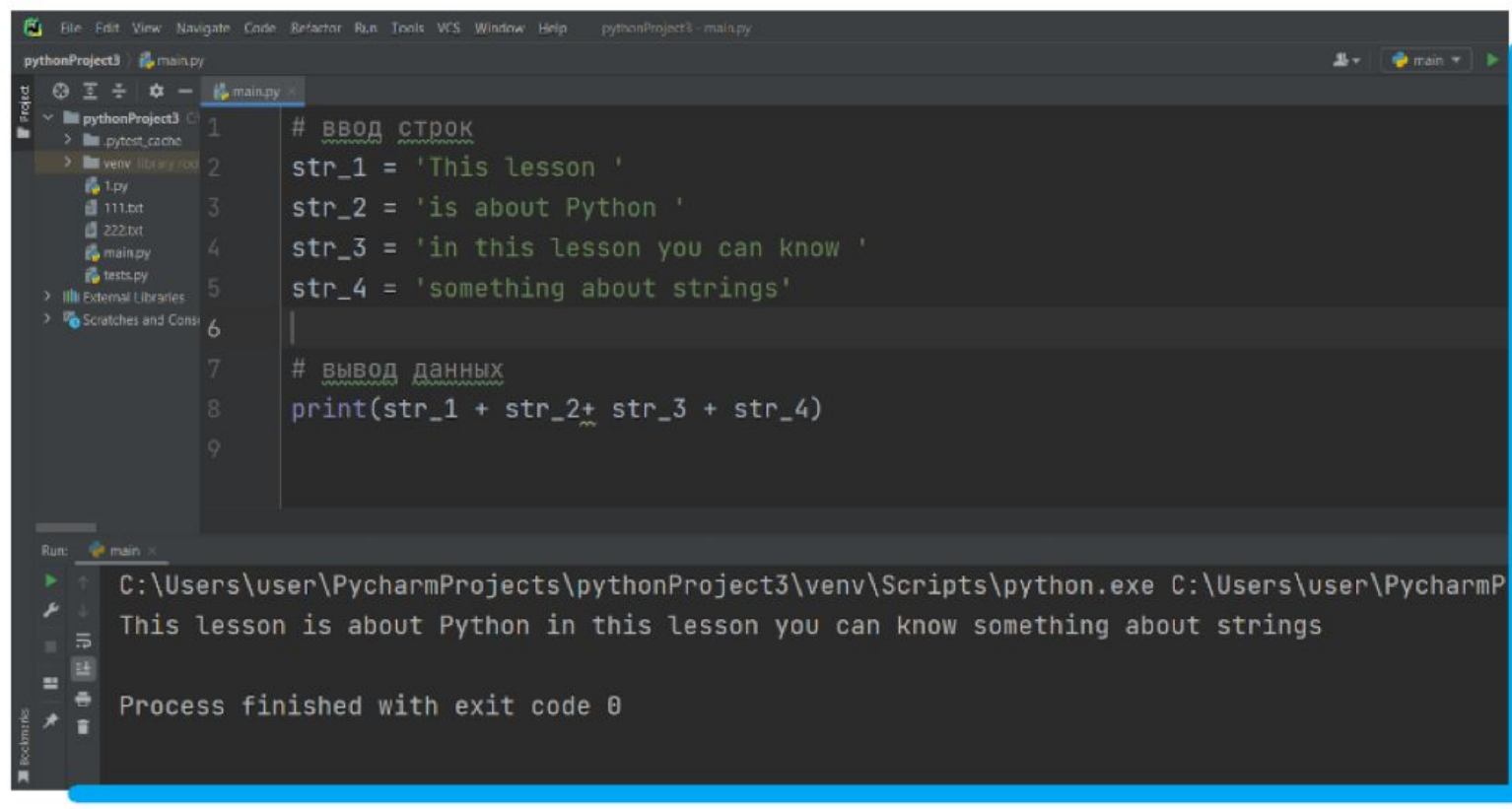
The image is a screenshot of the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The project name 'pythonProject3' is visible in the top right. The left sidebar shows the project structure with folders like 'pythonProject3', '.pytest_cache', and 'venv', and files like '1.py', '111.txt', '222.txt', 'main.py', and 'tests.py'. The main editor window displays the code in 'main.py' with line numbers 1 through 6. The code is in Russian and demonstrates string concatenation. The bottom panel shows the 'Run' output, indicating the program executed successfully and printed the concatenated string.

```
1 # ввод строк  
2 str_1 = 'This lesson '  
3 str_2 = 'is about Python'  
4  
5 # вывод данных  
6 print(str_1 + str_2)
```

Run: main ×

C:\Users\user\PycharmProjects\pythonProject3\venv
This lesson is about Python
Process finished with exit code 0

Данная операция может работать и с большим числом строк, чем две:

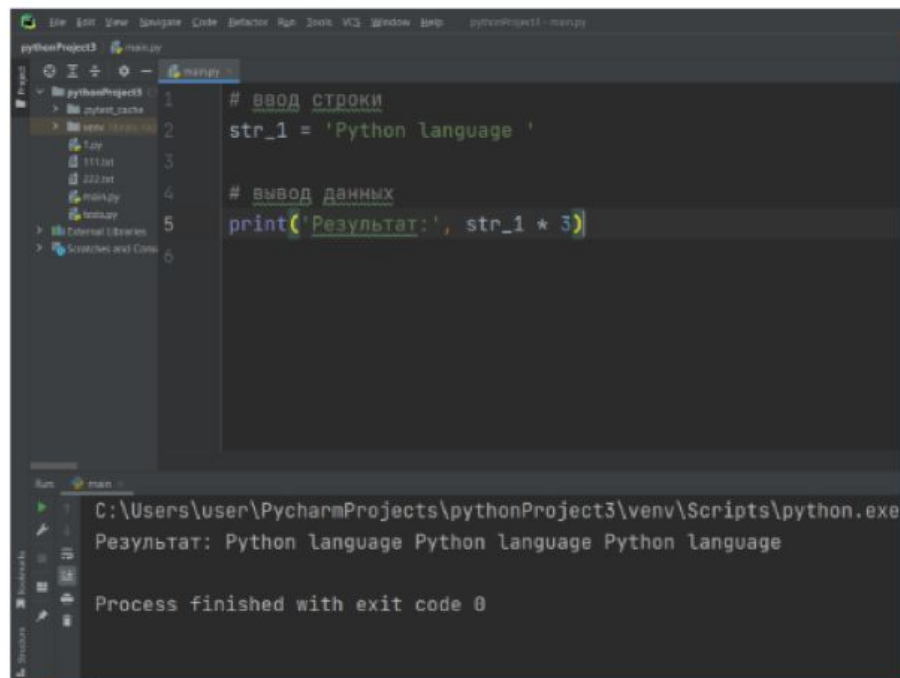


The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The project name is 'pythonProject3'. The left sidebar shows the project structure with folders like 'pythonProject3', 'pytest_cache', and 'venv', and files like '1.py', '111.txt', '222.txt', 'main.py', and 'tests.py'. The main editor window displays the code in 'main.py' with line numbers 1 through 9. The code defines four strings and prints their concatenation. The bottom panel shows the Run output, indicating the command used to execute the script and the resulting output string.

```
1 # ВВОД строк
2 str_1 = 'This lesson '
3 str_2 = 'is about Python '
4 str_3 = 'in this lesson you can know '
5 str_4 = 'something about strings'
6
7 # ВЫВОД ДАННЫХ
8 print(str_1 + str_2 + str_3 + str_4)
9
```

Run: C:\Users\user\PycharmProjects\pythonProject3\venv\Scripts\python.exe C:\Users\user\PycharmP
This lesson is about Python in this lesson you can know something about strings
Process finished with exit code 0

Дублирование строки. Строки можно дублировать при помощи специального оператора **'*'** и числа, которое будет указывать - сколько раз повторять вывод строки:



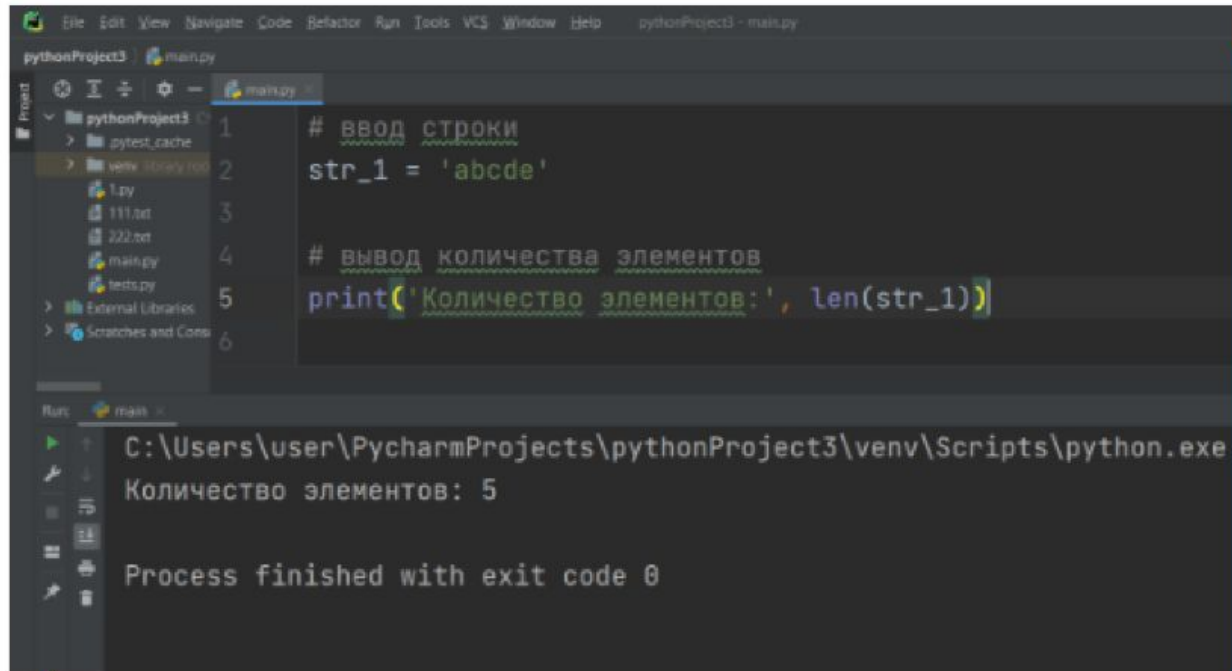
The screenshot shows the PyCharm IDE interface. The editor window displays a Python script with the following code:

```
1 # ВВОД СТРОКИ
2 str_1 = 'Python language '
3
4 # ВЫВОД ДАННЫХ
5 print('Результат:', str_1 * 3)
```

The Run window at the bottom shows the execution output:

```
C:\Users\user\PycharmProjects\pythonProject3\venv\Scripts\python.exe
Результат: Python language Python language Python language
Process finished with exit code 0
```


Нахождение длины строки. Данную операцию можно реализовать при помощи одного оператора **len()**:



The screenshot shows the PyCharm IDE interface. The main editor window displays a Python file named `main.py` with the following code:

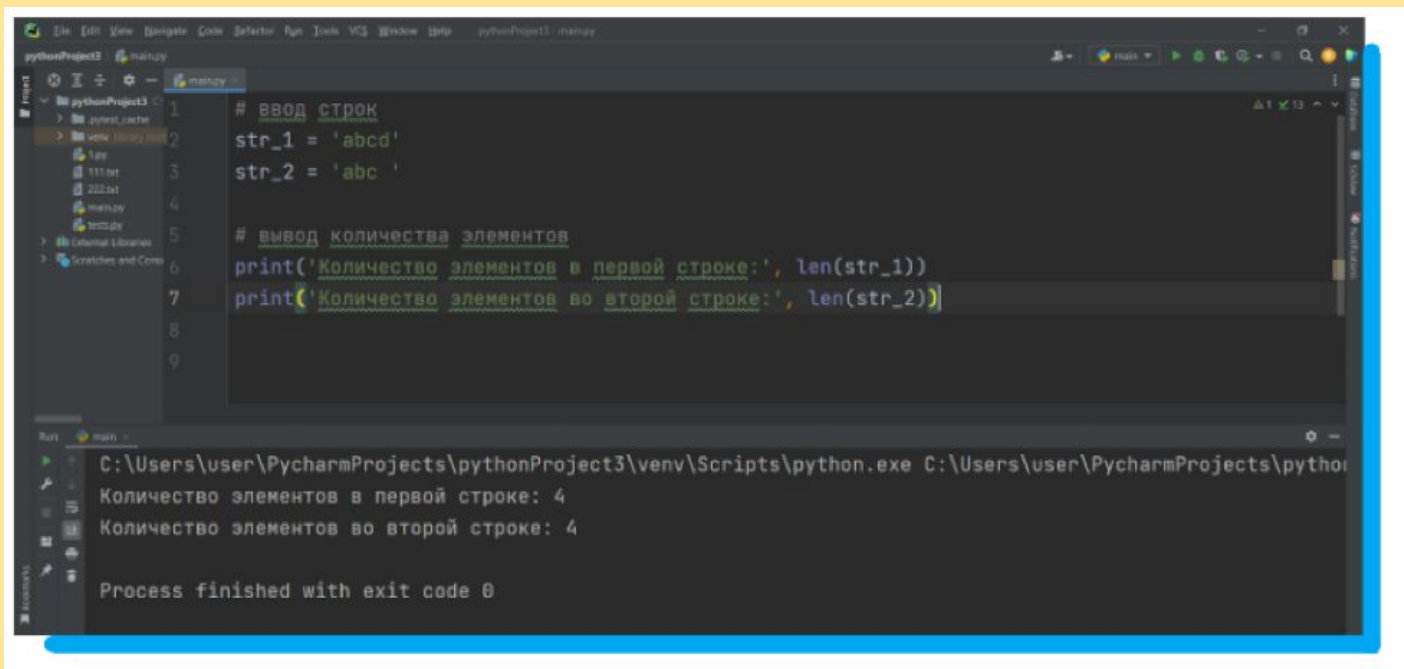
```
1 # ВВОД СТРОКИ
2 str_1 = 'abcde'
3
4 # ВЫВОД КОЛИЧЕСТВА ЭЛЕМЕНТОВ
5 print('Количество элементов:', len(str_1))
6
```

The left sidebar shows the project structure for `pythonProject3`, including `pytest_cache`, `venv`, `Scripts`, `lib`, `111.txt`, `222.txt`, `main.py`, and `tests.py`.

The bottom panel shows the output of the program execution:

```
C:\Users\user\PycharmProjects\pythonProject3\venv\Scripts\python.exe
Количество элементов: 5
Process finished with exit code 0
```

Если пользователь введет не число и не символ (например, знак пробела), то компилятор все равно посчитает его и количество элементов данной строки со строкой, где нет этого пробела, но на один элемент больше, будет одинаковым:



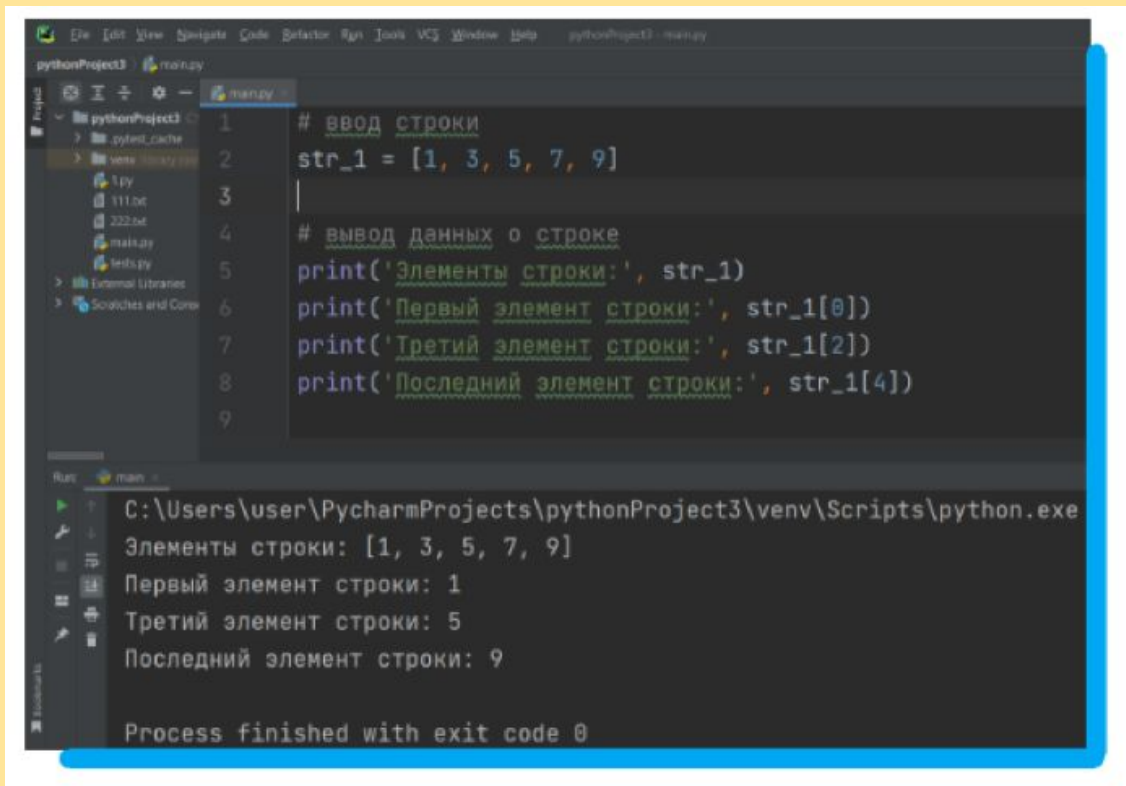
The screenshot shows the PyCharm IDE interface. The main editor window displays a Python script with the following code:

```
1 # ВВОД СТРОК
2 str_1 = 'abcd'
3 str_2 = 'abc '
4
5 # ВЫВОД КОЛИЧЕСТВА ЭЛЕМЕНТОВ
6 print('Количество элементов в первой строке:', len(str_1))
7 print('Количество элементов во второй строке:', len(str_2))
8
9
```

The Run window at the bottom shows the execution output:

```
C:\Users\user\PycharmProjects\pythonProject3\venv\Scripts\python.exe C:\Users\user\PycharmProjects\pythonProject3\main.py
Количество элементов в первой строке: 4
Количество элементов во второй строке: 4
Process finished with exit code 0
```

Доступ к элементу строки по индексу. К каждому элементу можно обращаться отдельно, указав его индекс:

The image is a screenshot of the PyCharm IDE interface. The top part shows the editor with a Python file named 'main.py'. The code defines a list 'str_1' with values [1, 3, 5, 7, 9] and then prints the list and its elements at indices 0, 2, and 4. The bottom part shows the 'Run' console with the output of the script. The output matches the printed statements in the code. The entire screenshot is framed with a thick blue border.

```
1 # ВВОД строки
2 str_1 = [1, 3, 5, 7, 9]
3
4 # Вывод данных о строке
5 print('Элементы строки:', str_1)
6 print('Первый элемент строки:', str_1[0])
7 print('Третий элемент строки:', str_1[2])
8 print('Последний элемент строки:', str_1[4])
9
```

Run main

C:\Users\user\PycharmProjects\pythonProject3\venv\Scripts\python.exe

Элементы строки: [1, 3, 5, 7, 9]

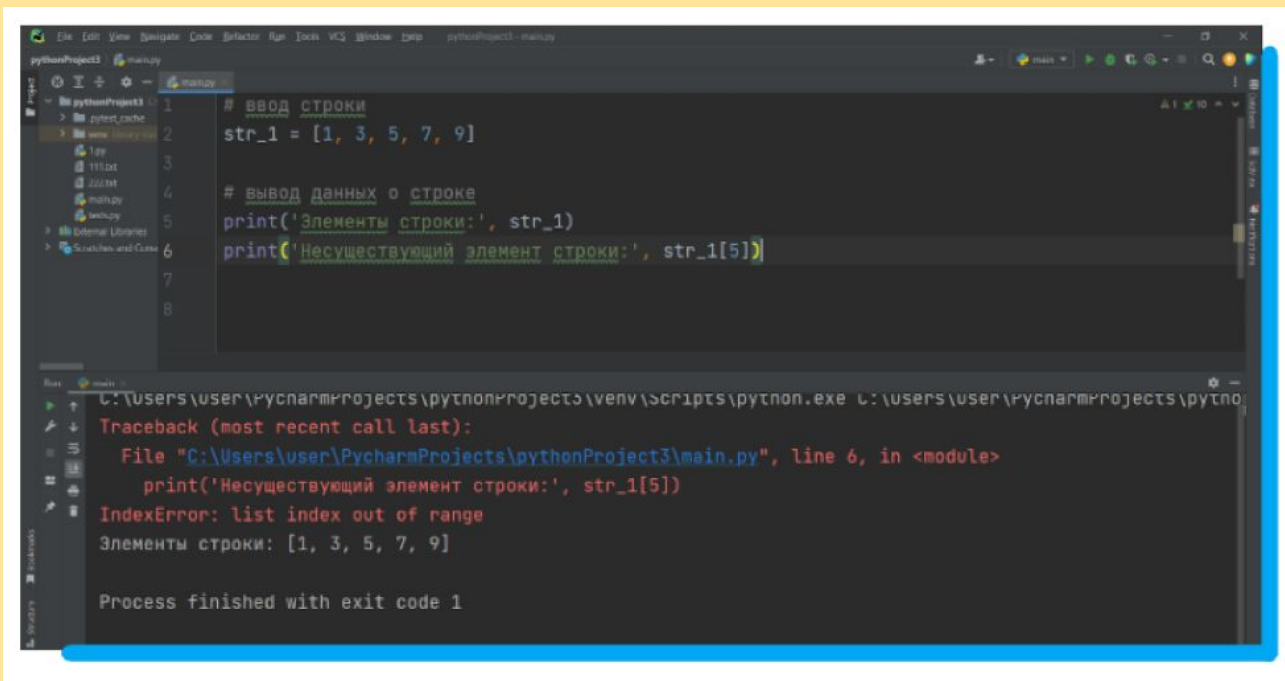
Первый элемент строки: 1

Третий элемент строки: 5

Последний элемент строки: 9

Process finished with exit code 0

Как можно заметить, элементы индекса начинаются с нуля. Если указать индекс, который не предусмотрен для данного списка, вывод консоли будет соответствовать размеру списка:

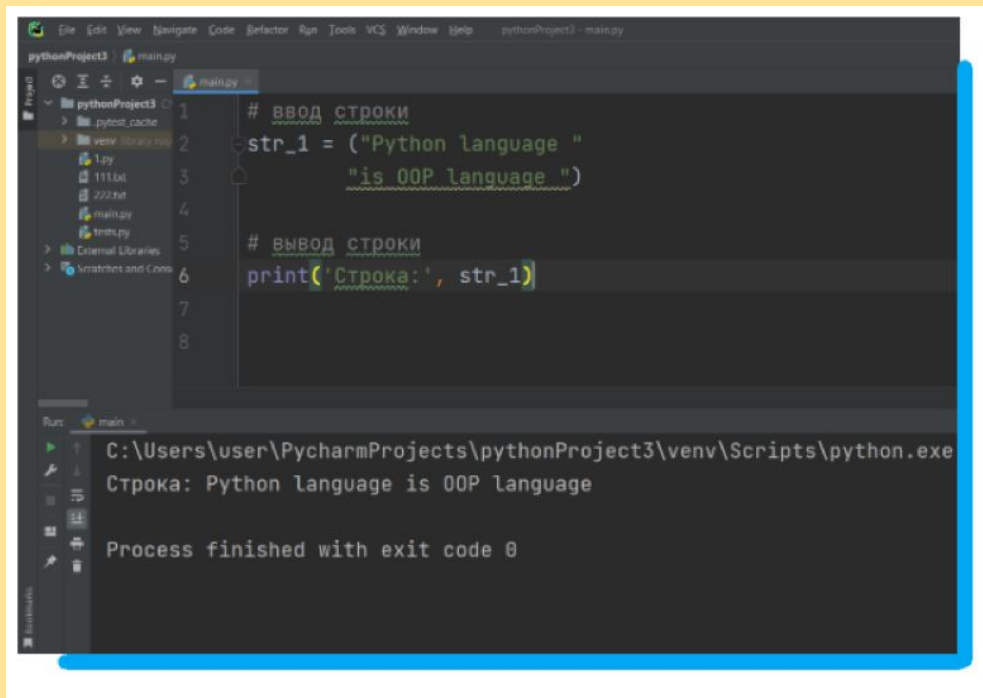


```
pythonProject3 main.py
1 # ввод строки
2 str_1 = [1, 3, 5, 7, 9]
3
4 # вывод данных о строке
5 print('Элементы строки:', str_1)
6 print('Несуществующий элемент строки:', str_1[5])
7
8

C:\Users\user\PycharmProjects\pythonProject3\venv\scripts\python.exe C:\Users\user\PycharmProjects\pythonProject3\main.py
Traceback (most recent call last):
  File "C:\Users\user\PycharmProjects\pythonProject3\main.py", line 6, in <module>
    print('Несуществующий элемент строки:', str_1[5])
IndexError: list index out of range
Элементы строки: [1, 3, 5, 7, 9]

Process finished with exit code 1
```

Длинную строку можно разбить на части и разместить их на разных строках кода. В данном случае, вся строка заключается в круглые скобки, а ее отдельные части помещаются в кавычки:

The image is a screenshot of the PyCharm IDE interface. The top pane shows a Python file named 'main.py' with the following code:

```
1 # ВВОД строки
2 str_1 = ("Python language "
3         "is OOP language ")
4
5 # ВЫВОД строки
6 print('Строка:', str_1)
7
8
```

The bottom pane shows the output of the program execution. The command prompt displays the path to the Python interpreter, the output of the print statement, and the exit code.

```
Run: main
C:\Users\user\PycharmProjects\pythonProject3\venv\Scripts\python.exe
Строка: Python language is OOP language

Process finished with exit code 0
```

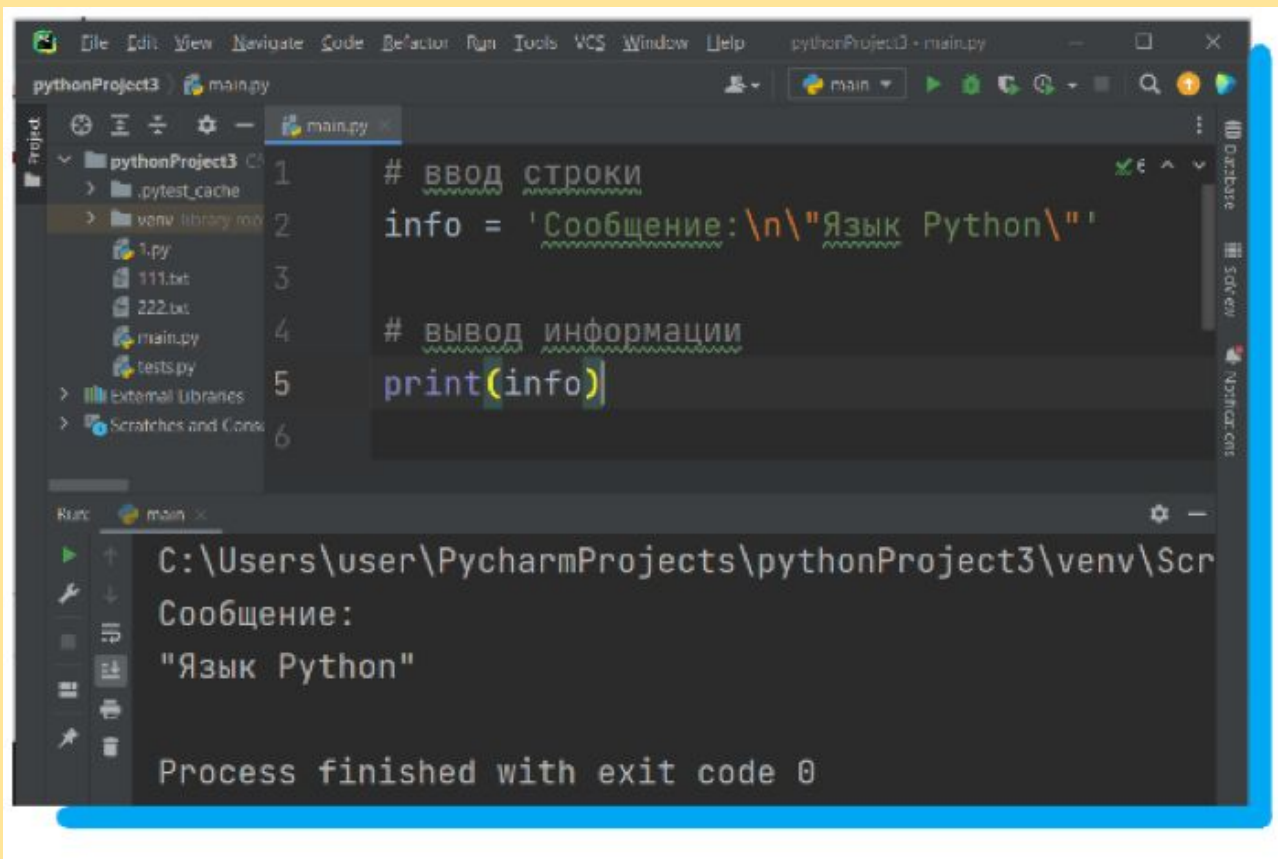
Управляющие последовательности в строке

Строка может содержать ряд специальных символов, которые управляют последовательностями или escape-последовательностями.

Перечислим самые применяемые управляющие последовательности:

- `\` - позволяет добавить знак слэша внутри строки;
- `\'` - позволяет добавить внутрь строки одинарную кавычку;
- `\''` - позволяет добавить внутрь строки двойную кавычку;
- `\n` - переход на новую строку (вертикальная табуляция);
- `\t` - сдвиг в правую сторону на 4 отступа (горизонтальная табуляция).

Применим символ `'\n'` для перехода новую строку:



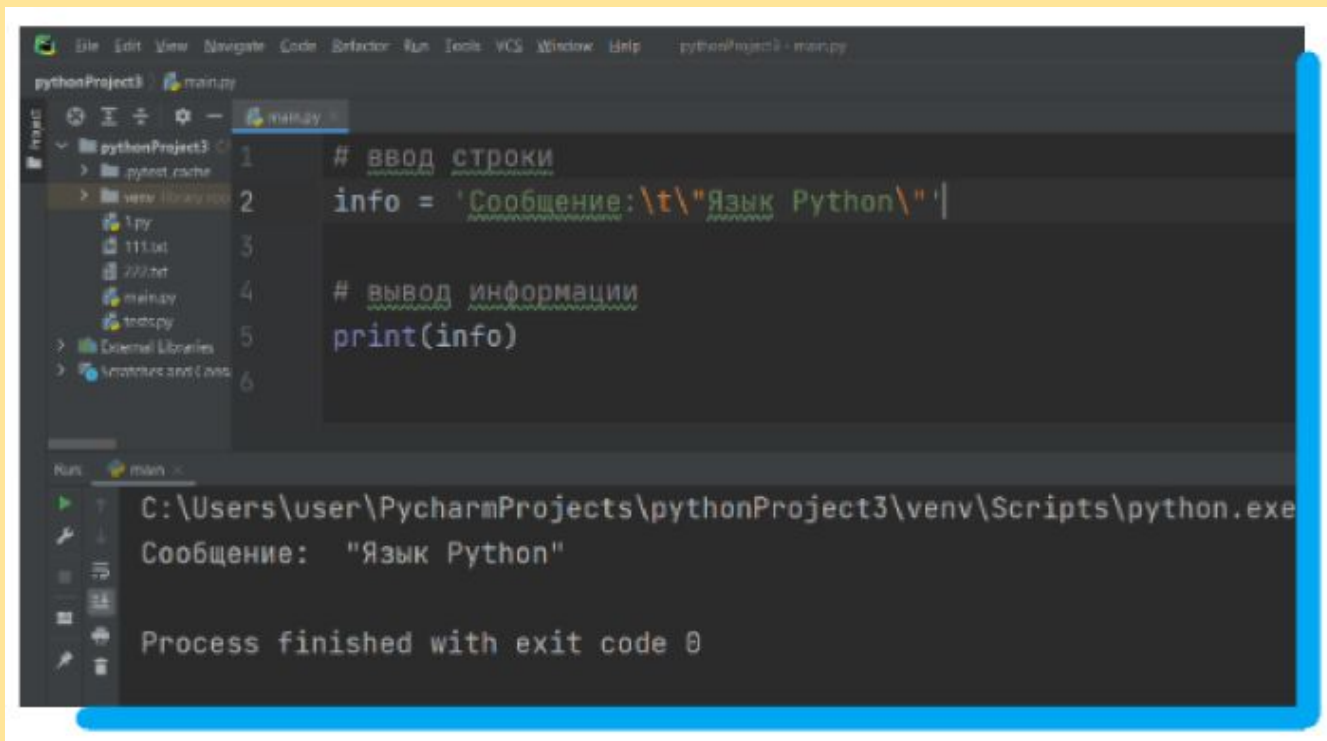
The screenshot displays the PyCharm IDE interface. The main editor window shows a Python file named `main.py` with the following code:

```
1 # ввод строки
2 info = 'Сообщение:\n"Язык Python"'
3
4 # вывод информации
5 print(info)
6
```

The code uses the `\n` escape sequence to create a new line in the output. The left sidebar shows the project structure, including the `venv` directory and the `main.py` file. The bottom panel shows the output of the program:

```
C:\Users\user\PycharmProjects\pythonProject3\venv\Scr
Сообщение:
"Язык Python"
Process finished with exit code 0
```

Горизонтальная табуляция “\t” работает аналогично, лишь сдвигает символы не на следующую строку, а лишь отступает вправо на 4 пробела:



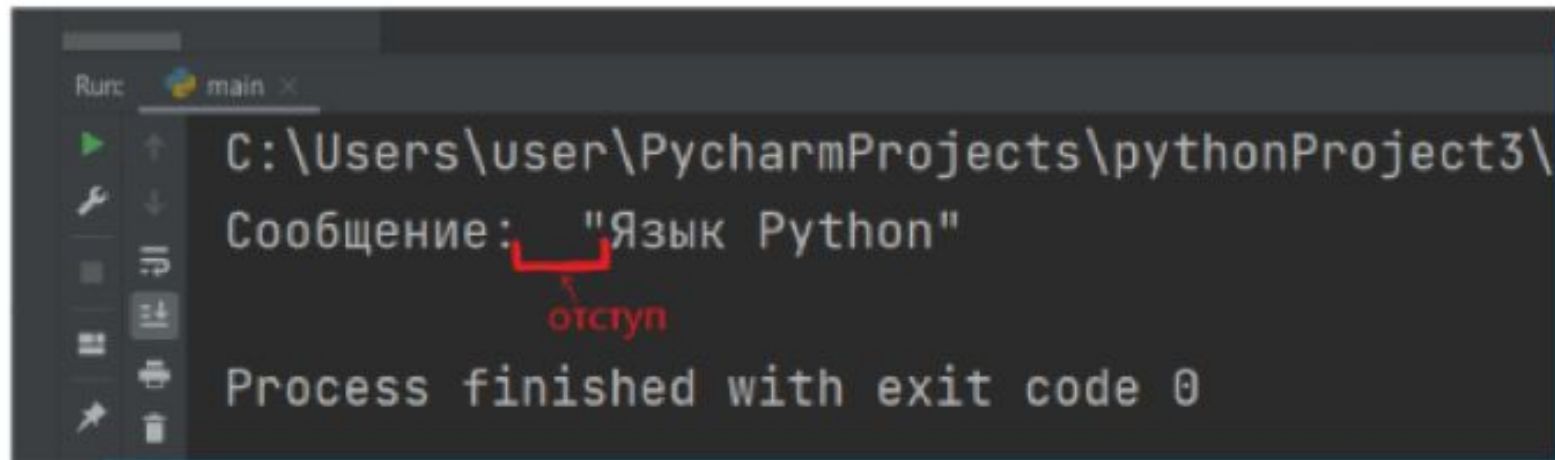
The screenshot displays the PyCharm IDE interface. The top toolbar includes menus like File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The left sidebar shows a project tree for 'pythonProject3' with files like '1.py', '111.txt', '222.txt', 'main.py', and 'test.py'. The main editor window shows a Python script with the following code:

```
1 # ввод строки
2 info = 'Сообщение:\t"Язык Python"'
3
4 # вывод информации
5 print(info)
6
```

Below the editor, the 'Run' console shows the execution path and output:

```
C:\Users\user\PycharmProjects\pythonProject3\venv\Scripts\python.exe
Сообщение: "Язык Python"
Process finished with exit code 0
```


Как можно заметить, в области вывода есть отступ:

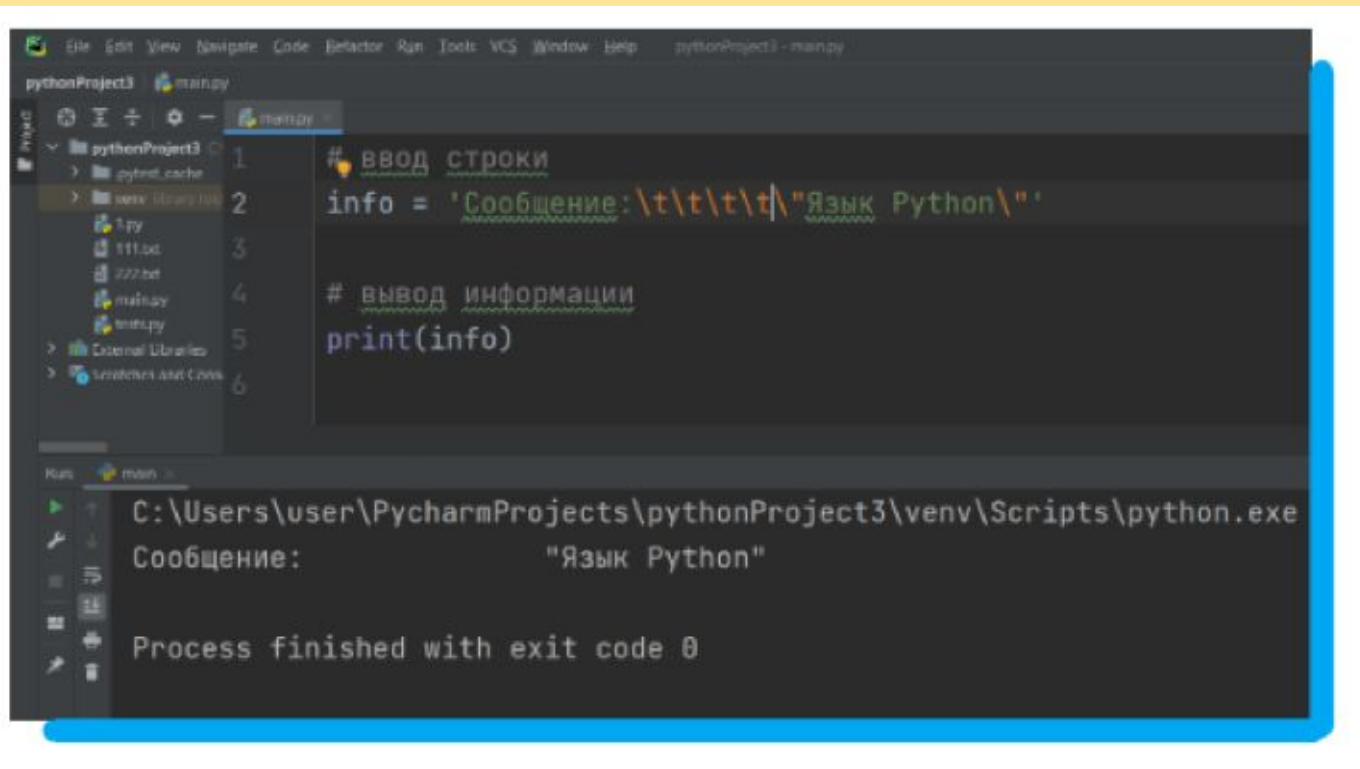


The screenshot shows the 'Run' console in PyCharm. The output text is as follows:

```
C:\Users\user\PycharmProjects\pythonProject3\  
Сообщение: "Язык Python"  
Process finished with exit code 0
```

A red bracket is drawn under the space between 'Сообщение:' and '"Язык Python"', with the word 'отступ' (indentation) written in red below it. The entire console window is highlighted with a blue border.

Данную последовательность, как и вертикальную, также возможно применять несколько раз:

The image is a screenshot of the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The title bar shows 'pythonProject3 - main.py'. On the left, the 'Project' tool window displays a file tree for 'pythonProject3', including files like '1.py', '111.txt', '222.txt', 'main.py', and 'temp.py'. The main editor window shows a Python script with the following code:

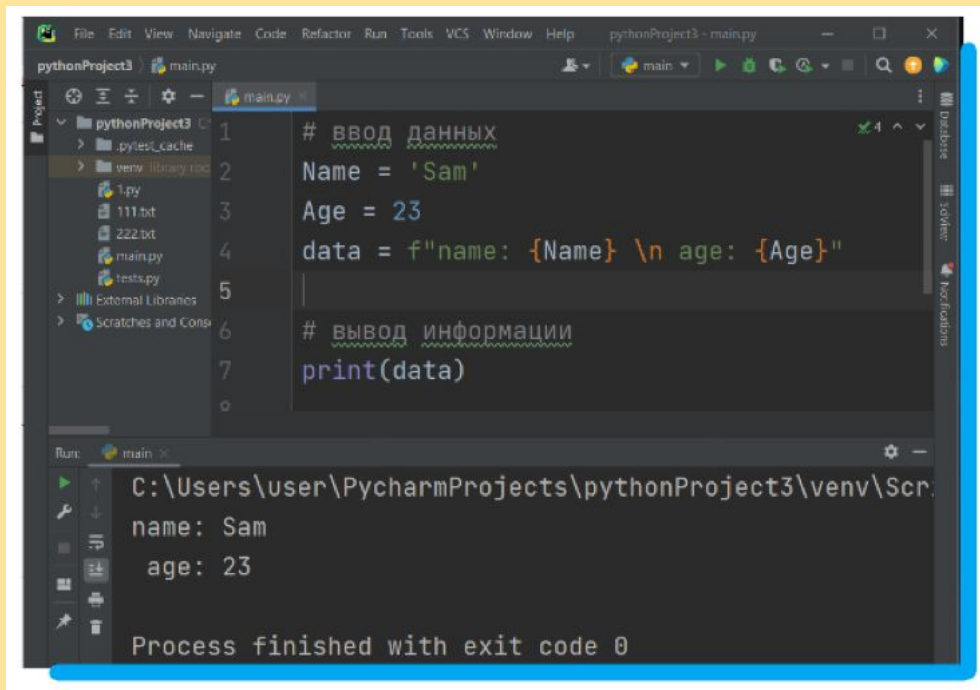
```
1 # ввод строки
2 info = 'Сообщение:\t\t\t\t"Язык Python"'
3
4 # вывод информации
5 print(info)
6
```

Below the editor, the 'Run' tool window is active, showing the execution command: 'C:\Users\user\PycharmProjects\pythonProject3\venv\Scripts\python.exe'. The output of the script is displayed as:

```
Сообщение:          "Язык Python"
```

At the bottom of the Run window, it states 'Process finished with exit code 0'. The entire screenshot is framed by a thick blue border on the right and bottom sides.

Вставка значений в строку. Объектно-ориентированный язык Python позволяет встраивать в строку значения других переменных. Приведем пример работы программы с вставкой значений в строку:

The image is a screenshot of the PyCharm IDE interface. The main editor window displays a Python file named 'main.py' with the following code:

```
1 # ввод данных
2 Name = 'Sam'
3 Age = 23
4 data = f"name: {Name} \n age: {Age}"
5
6 # вывод информации
7 print(data)
```

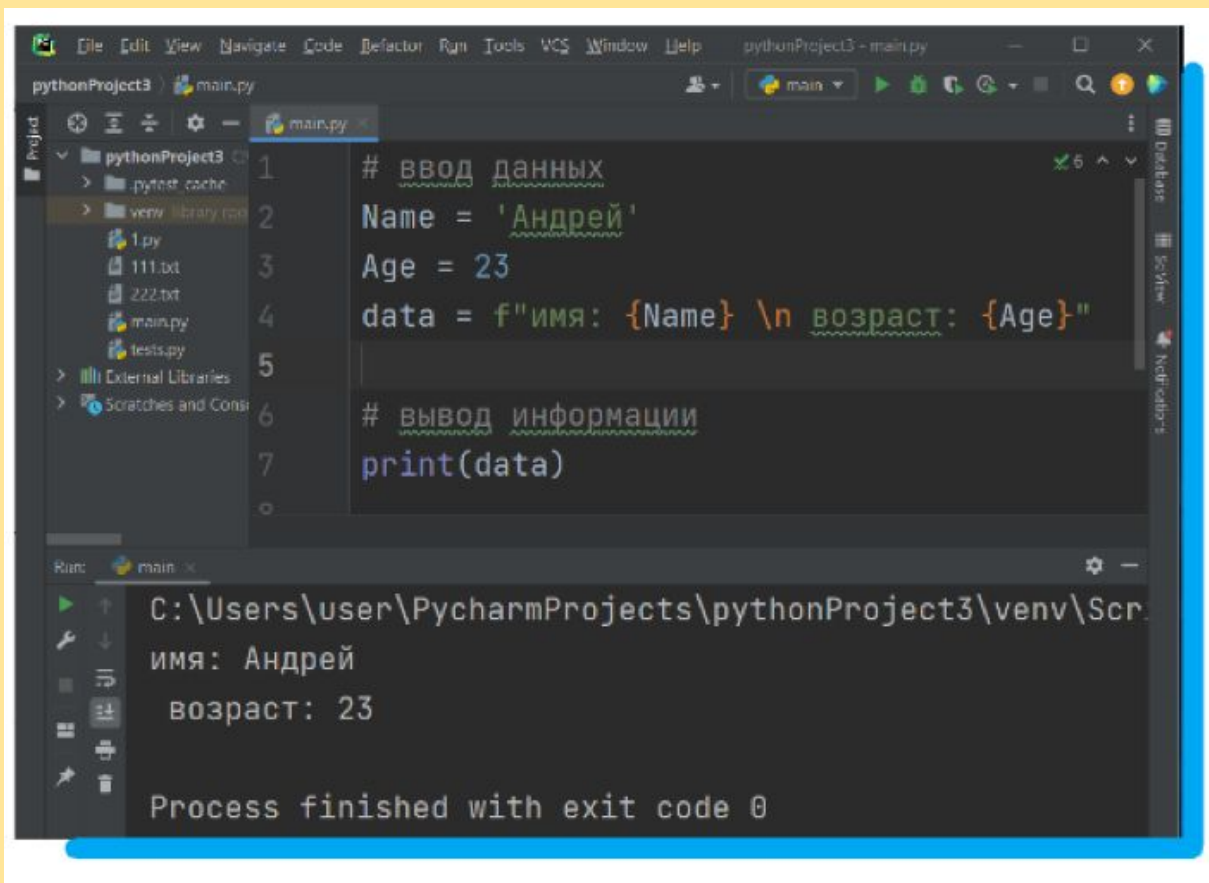
The code uses f-strings to format the output. The left sidebar shows the project structure for 'pythonProject3', including files like '1.py', '111.txt', '222.txt', 'main.py', and 'tests.py'. The bottom 'Run' console shows the execution output:

```
C:\Users\user\PycharmProjects\pythonProject3\venv\Scr
name: Sam
age: 23

Process finished with exit code 0
```

The console output matches the formatted string defined in the code, demonstrating the successful insertion of variable values into the string.

В данной конструкции можно использовать и русский язык:



The screenshot displays the PyCharm IDE interface. The main editor window shows a Python file named `main.py` with the following code:

```
1 # ввод данных
2 Name = 'Андрей'
3 Age = 23
4 data = f"имя: {Name} \n возраст: {Age}"
5
6 # вывод информации
7 print(data)
```

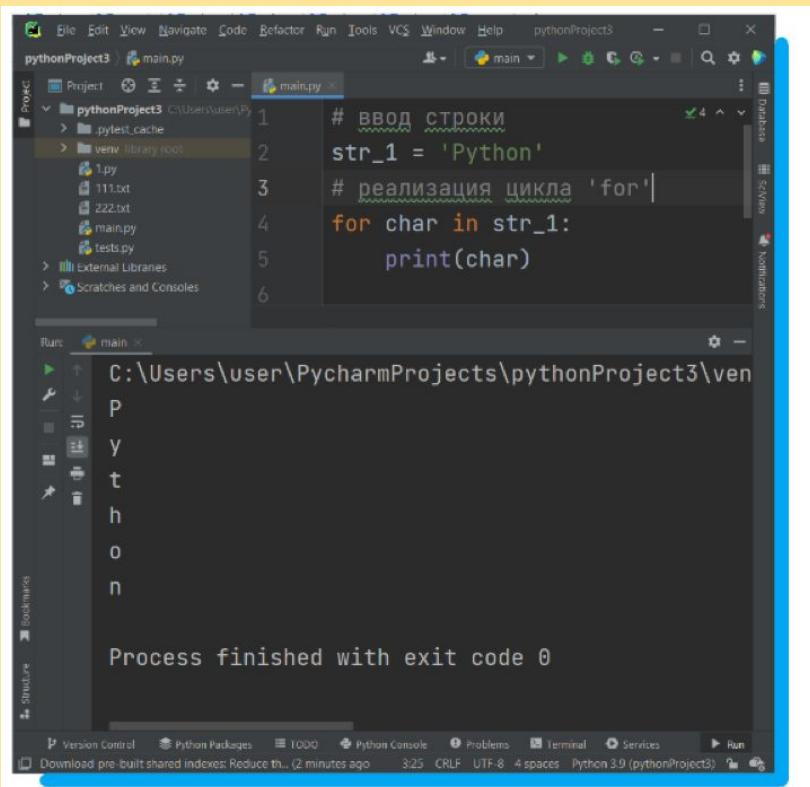
The left sidebar shows the project structure for `pythonProject3`, including a `venv` directory and several files like `1.py`, `111.txt`, `222.txt`, `main.py`, and `tests.py`.

The bottom panel shows the output of the program execution:

```
C:\Users\user\PycharmProjects\pythonProject3\venv\Scr
имя: Андрей
возраст: 23

Process finished with exit code 0
```

Перебор строки. С помощью цикла **for** можно перебирать все символы строки, приведем примеры с использованием данного цикла:



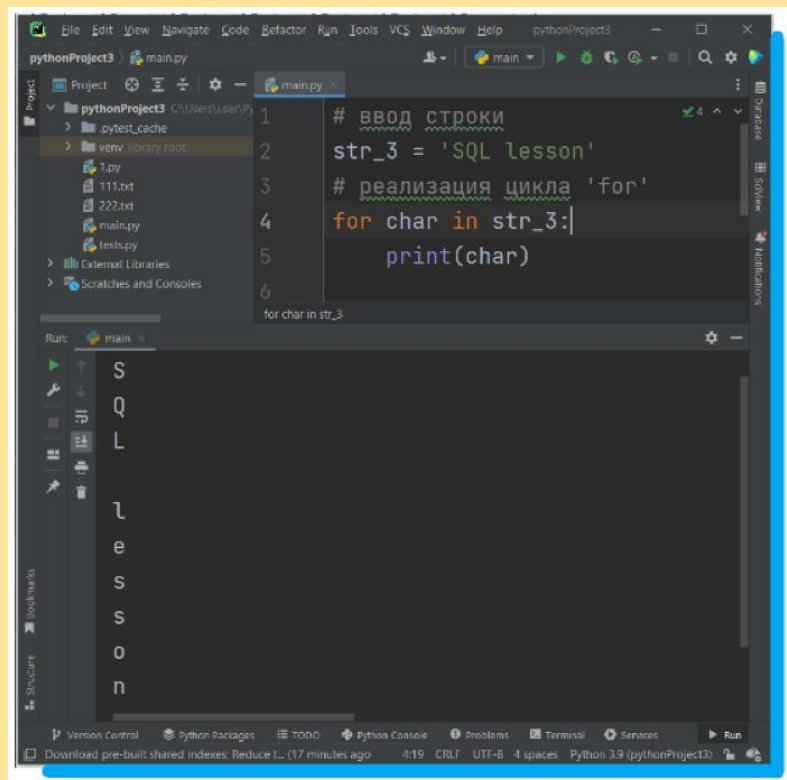
The screenshot shows the PyCharm IDE with a file named `main.py`. The code in the editor is as follows:

```
1 # ВВОД строки
2 str_1 = 'Python'
3 # реализация цикла 'for'
4 for char in str_1:
5     print(char)
6
```

The Run console at the bottom displays the output of the program:

```
C:\Users\user\PycharmProjects\pythonProject3\venv
P
y
t
h
o
n

Process finished with exit code 0
```



The screenshot shows the PyCharm IDE with a file named `main.py`. The code in the editor is as follows:

```
1 # ВВОД строки
2 str_3 = 'SQL lesson'
3 # реализация цикла 'for'
4 for char in str_3:
5     print(char)
6
```

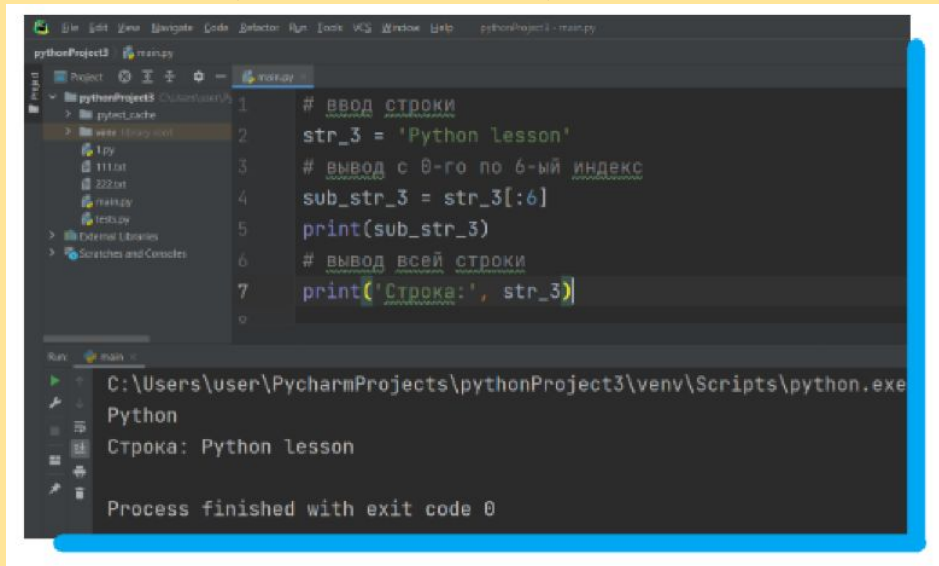
The Run console at the bottom displays the output of the program:

```
S
Q
L

l
e
s
s
o
n
```

Получение подстроки. При необходимости, возможно получить из строки не только отдельные символы, но и подстроку. Для этого можно использовать следующий синтаксис:

- **string[:end]** - извлекается последовательность символов, начиная с 0-го индекса по индекс end (не включая его):

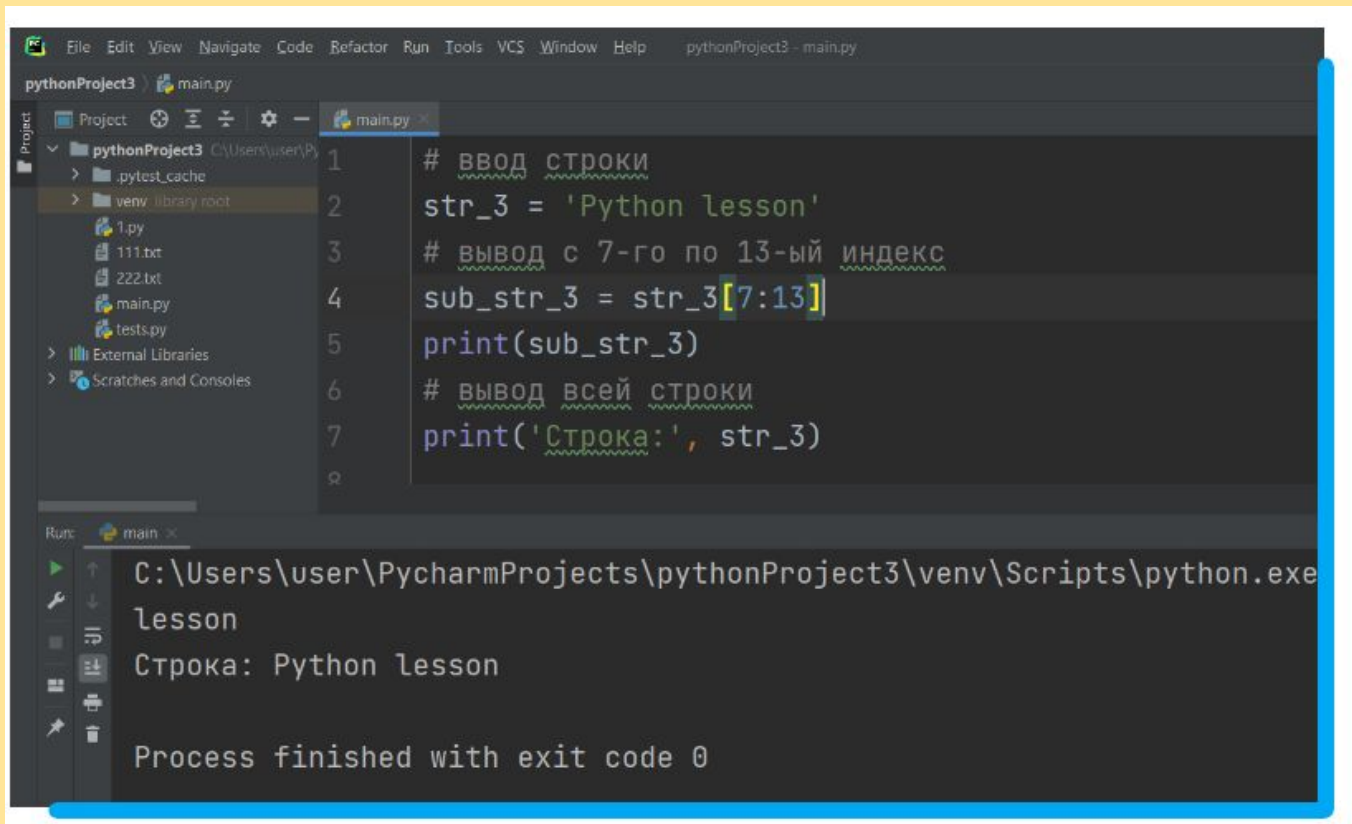
A screenshot of the PyCharm IDE interface. The main editor window shows a Python script with the following code:

```
1 # Ввод строки
2 str_3 = 'Python lesson'
3 # вывод с 0-го по 6-ый индекс
4 sub_str_3 = str_3[:6]
5 print(sub_str_3)
6 # вывод всей строки
7 print('Строка:', str_3)
```

The left sidebar shows the project structure with files like 1.py, 111.txt, 222.txt, test.py, and test2.py. The bottom panel shows the Run console output:

```
C:\Users\user\PycharmProjects\pythonProject3\venv\Scripts\python.exe
Python
Строка: Python lesson
Process finished with exit code 0
```

- **string [start:end]** - извлекается последовательность символов, начиная с индекса start по end (не включая):

The image is a screenshot of the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The main editor window displays a file named 'main.py' with the following Python code:

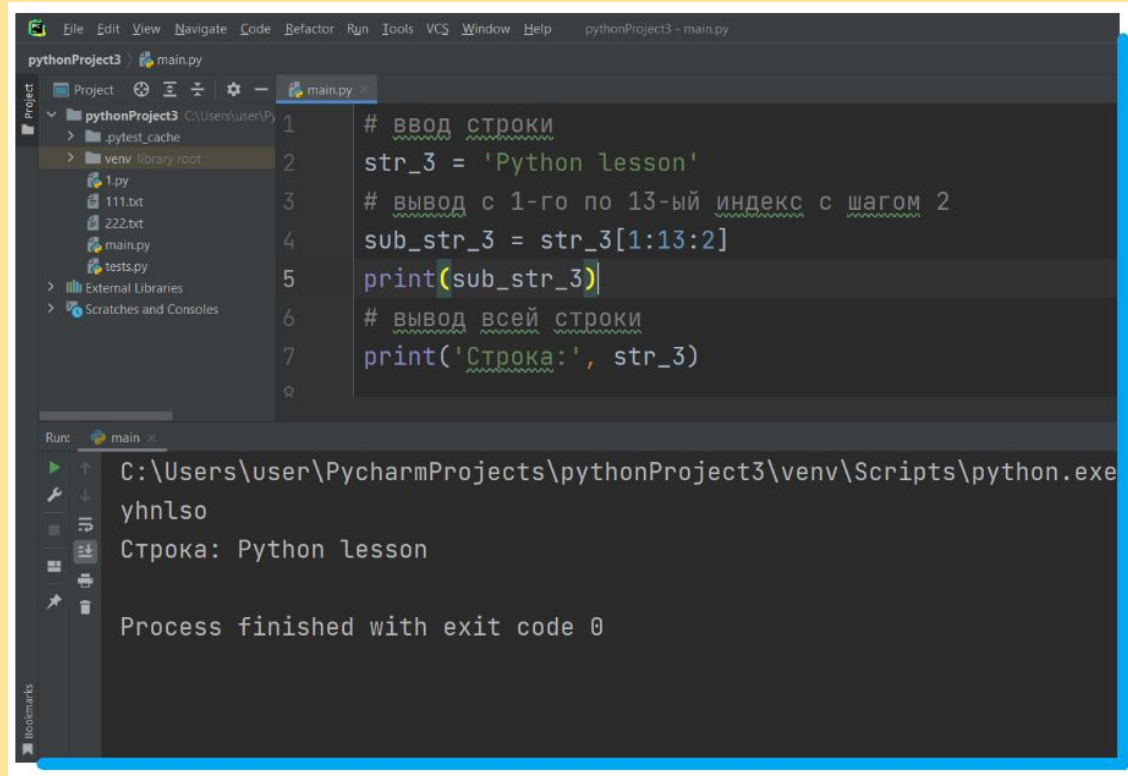
```
1 # ВВОД строки
2 str_3 = 'Python lesson'
3 # вывод с 7-го по 13-ый индекс
4 sub_str_3 = str_3[7:13]
5 print(sub_str_3)
6 # вывод всей строки
7 print('Строка:', str_3)
8
```

The code uses comments in Russian to describe each step: line 1 is '# ВВОД строки' (input string), line 2 is the string assignment, line 3 is '# вывод с 7-го по 13-ый индекс' (output from 7th to 13th index), line 4 is the slicing operation, line 5 is the print statement for the slice, line 6 is '# вывод всей строки' (output of the whole string), and line 7 is the print statement for the full string. The left sidebar shows the project structure for 'pythonProject3', including files like '1.py', '111.txt', '222.txt', 'main.py', and 'tests.py'. At the bottom, the 'Run' console shows the execution output:

```
C:\Users\user\PycharmProjects\pythonProject3\venv\Scripts\python.exe
lesson
Строка: Python lesson
Process finished with exit code 0
```

The output shows the string 'lesson' on the first line and 'Строка: Python lesson' on the second line, followed by the message 'Process finished with exit code 0'.

- **string [start:end:step]** - извлекается последовательность символов, начиная с индекса start по индекс end (не включая) через шаг step:



The screenshot shows the PyCharm IDE interface. The left sidebar displays the project structure for 'pythonProject3', including files like '1.py', '111.txt', '222.txt', 'main.py', and 'tests.py'. The main editor window shows the content of 'main.py' with the following Python code:

```
1 # ВВОД строки
2 str_3 = 'Python lesson'
3 # вывод с 1-го по 13-ый индекс с шагом 2
4 sub_str_3 = str_3[1:13:2]
5 print(sub_str_3)
6 # вывод всей строки
7 print('Строка:', str_3)
```

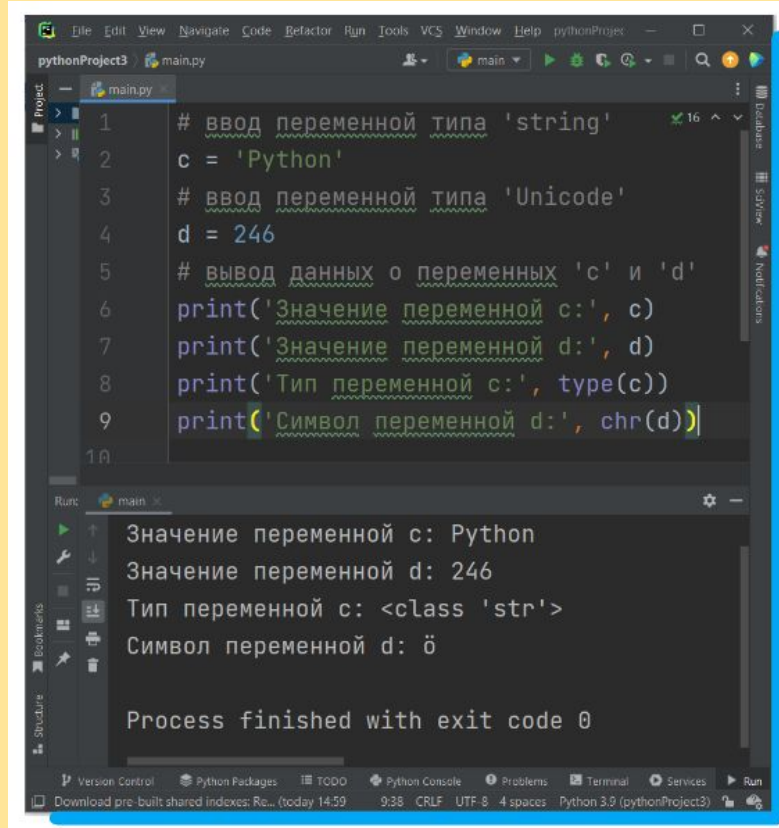
Below the editor, the 'Run' console shows the execution output for the 'main' configuration:

```
C:\Users\user\PycharmProjects\pythonProject3\venv\Scripts\python.exe
python3
Строка: Python lesson
Process finished with exit code 0
```


Подробнее о типах данных `string` и `Unicode`

Типы данных `string` и тип `Unicode`. В объектно-ориентированном языке программирования Python строки бывают двух типов: обычные и `Unicode`-строки. **Строка представляет собой последовательности символов.**

Строки - константы можно легко задать в программе с помощью строковых литералов:



The screenshot shows an IDE window titled 'pythonProject3' with a file named 'main.py'. The code in the editor is as follows:

```
1 # ввод переменной типа 'string'
2 c = 'Python'
3 # ввод переменной типа 'Unicode'
4 d = 246
5 # вывод данных о переменных 'c' и 'd'
6 print('Значение переменной c:', c)
7 print('Значение переменной d:', d)
8 print('Тип переменной c:', type(c))
9 print('Символ переменной d:', chr(d))
```

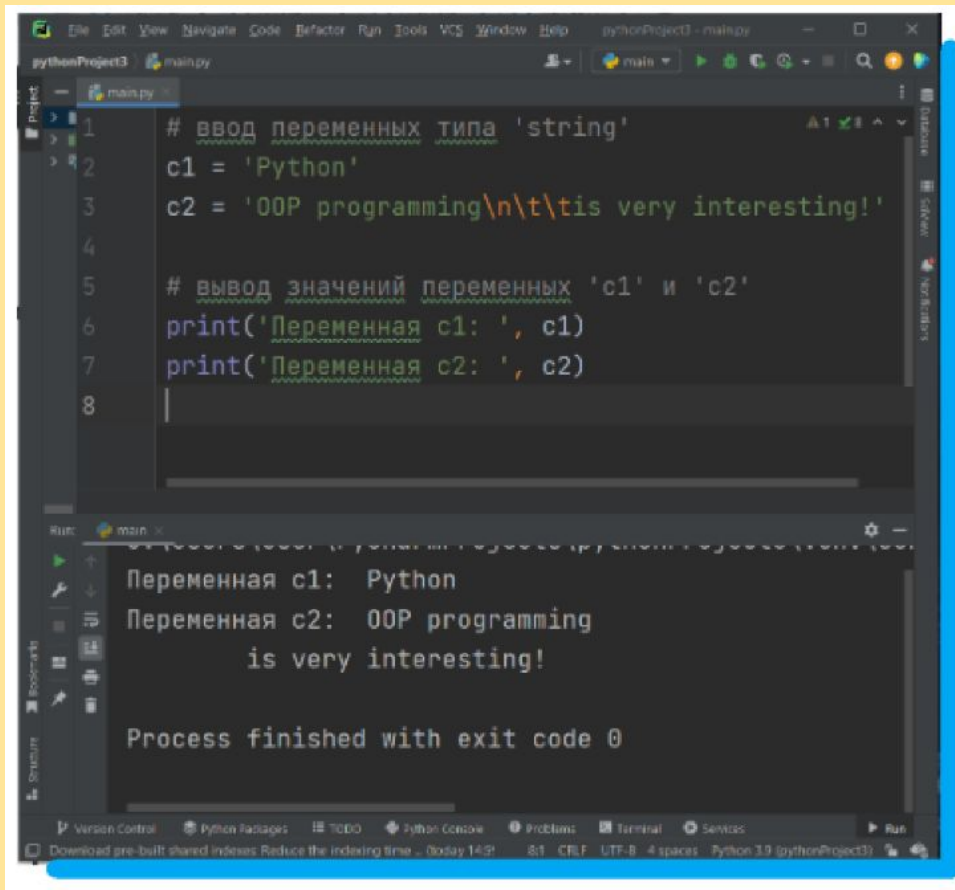
Below the editor, the 'Run' console displays the output of the program:

```
Значение переменной c: Python
Значение переменной d: 246
Тип переменной c: <class 'str'>
Символ переменной d: ö
Process finished with exit code 0
```

The status bar at the bottom indicates the file encoding is UTF-8, the line length is 4 spaces, and the Python version is 3.9.

Переменной можно передавать несколько строк, для этого нужно использовать оператор разделения ‘\n’:

В данной программе используется оператор ‘\t’ (горизонтальная табуляция), который позволяет сдвинуть начало строки на фиксированное количество отступов. Оператор перехода ‘\n’ позволяет сдвинуть вывод на следующую строку.



```
# Ввод переменных типа 'string'
c1 = 'Python'
c2 = 'OOP programming\n\t\tis very interesting!'

# вывод значений переменных 'c1' и 'c2'
print('Переменная c1: ', c1)
print('Переменная c2: ', c2)
```

Переменная c1: Python
Переменная c2: OOP programming
 is very interesting!

Process finished with exit code 0

Доступ к определенному элементу можно получить при помощи указания индекса элемента. В языке Python первый элемент списка имеет индекс [0], а не [1]. Посмотрим на схему обозначения индексов в языке Python:

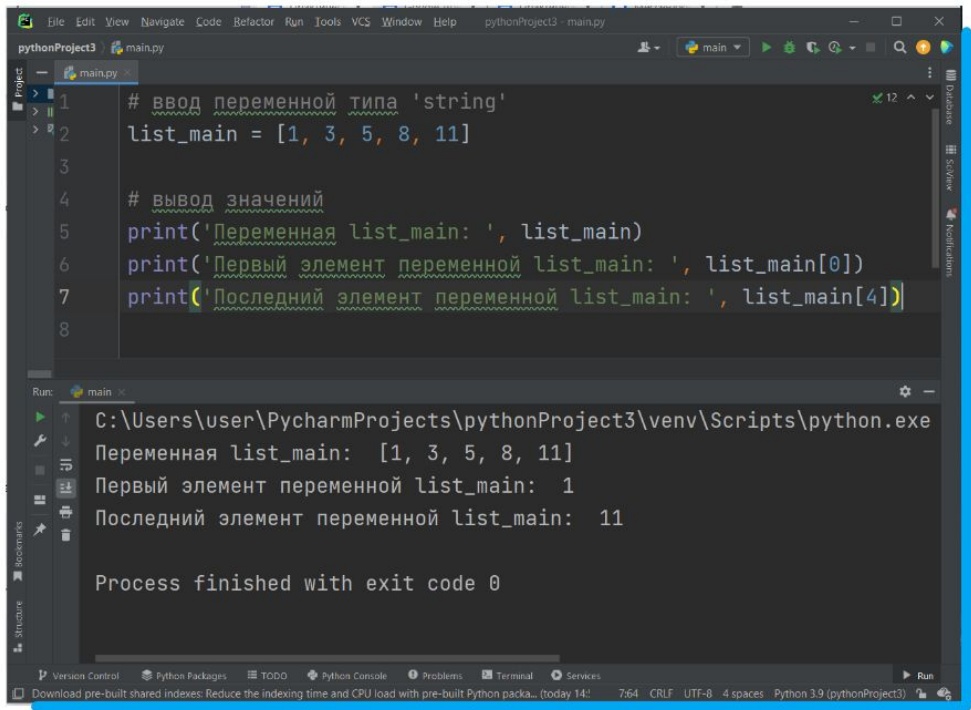


A hand-drawn diagram on a chalkboard background illustrating list indexing in Python. It consists of a table with two rows and five columns. The first row is labeled 'Имя списка' (List Name) in green and contains the values 3, 5, 11, and 13 in red. The second row is labeled 'Индексы' (Indices) in green and contains the values 0, 1, 2, and 3 in blue. A red box labeled 'Элементы списка' (List Elements) in green spans the top of the table, covering the first row and the first four columns of the second row.

Элементы списка				
Имя списка	3	5	11	13
Индексы	0	1	2	3

Посмотрим на пример программы, в которой пользователь хочет получить доступ к отдельным элементам списка:

Как можно заметить, когда мы указываем индекс [4], консоль выводит последний элемент списка 'list_main' со значением 11.



The screenshot shows the PyCharm IDE interface. The top pane displays a Python script named `main.py` with the following code:

```
1 # ввод переменной типа 'string'
2 list_main = [1, 3, 5, 8, 11]
3
4 # вывод значений
5 print('Переменная list_main: ', list_main)
6 print('Первый элемент переменной list_main: ', list_main[0])
7 print('Последний элемент переменной list_main: ', list_main[4])
8
```

The bottom pane shows the output of the script execution:

```
C:\Users\user\PycharmProjects\pythonProject3\venv\Scripts\python.exe
Переменная list_main: [1, 3, 5, 8, 11]
Первый элемент переменной list_main: 1
Последний элемент переменной list_main: 11

Process finished with exit code 0
```

The status bar at the bottom indicates the file encoding is UTF-8, the line length is 4 spaces, and the Python version is 3.9.

ПРАКТИЧЕСКИЕ ЗАДАЧИ



Задача 1.

Дан следующий список чисел: **my_list** = [1, 4, 6, 10, 12, 15, 17, 19].

Используя функцию **filter()**, отобрать из данного списка только нечетные числа.

Решение.

Напишем код для решения данной практической задачи и посмотрим на вывод консоли:

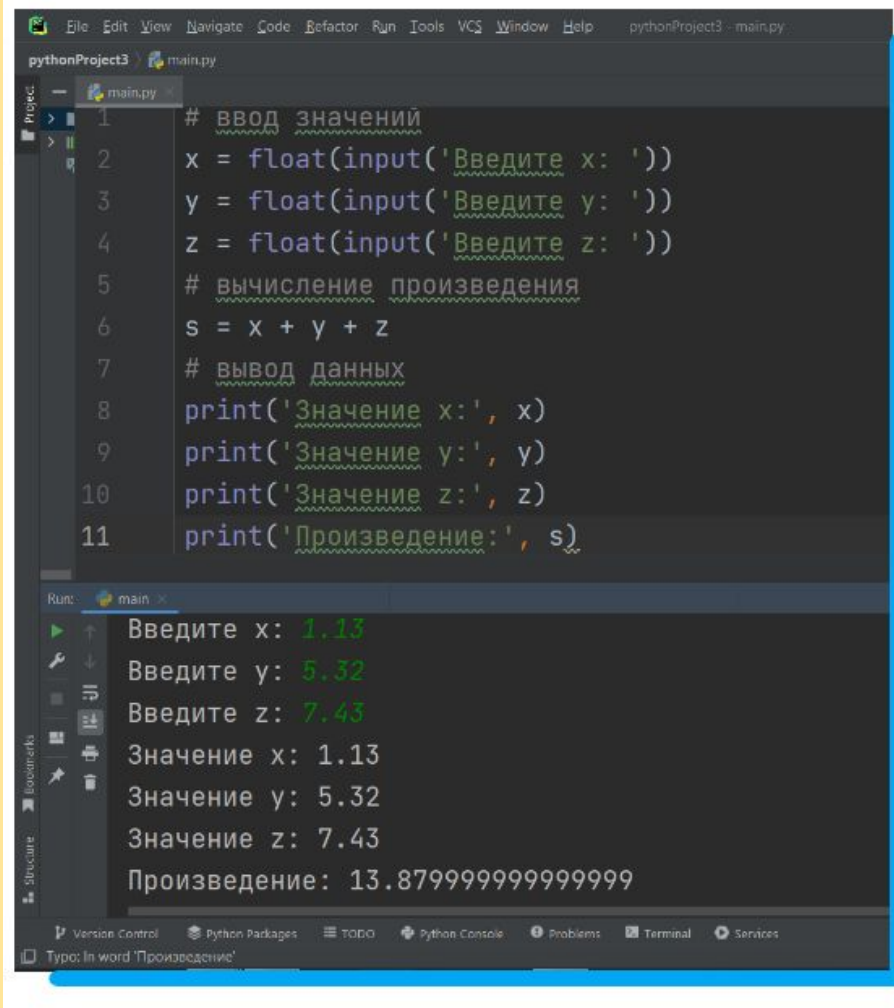
<div>main.py</div> <div><div>1 my_list = [1, 4, 6, 10, 12, 15, 17, 19]</div><div>2 new_list = list(filter(lambda x: (x%2 == 1), my_list))</div><div>3 print('Нечетные числа: ', new_list)</div><div>4</div></div>	<div>Shell</div> <div>Clear</div> <div>Нечетные числа: [1, 15, 17, 19]</div> <div>> </div>
---	--

Задача 2.

Создать программу в среде PyCharm, в которой вводятся с клавиатуры три переменные с типом данных **float**: 'x', 'y' и 'z'. Затем в переменную 's' записывается произведений значений данных переменных. Вывести на консоль значения введенных переменных и произведение.

Решение.

Напишем код для решения данной практической задачи и посмотрим на вывод консоли:



The screenshot shows an IDE window titled 'pythonProject3 - main.py'. The editor displays a Python script with 11 lines of code. The code prompts the user to input three floating-point numbers (x, y, z), calculates their sum (s), and prints the results. The Run console at the bottom shows the execution of the program with the following output:

```
# ввод значений
1 x = float(input('Введите x: '))
2 y = float(input('Введите y: '))
3 z = float(input('Введите z: '))
4
5 # вычисление произведения
6 s = x + y + z
7
8 # вывод данных
9 print('Значение x:', x)
10 print('Значение y:', y)
11 print('Значение z:', z)
12 print('Произведение:', s)
```

Run: main

```
Введите x: 1.13
Введите y: 5.32
Введите z: 7.43
Значение x: 1.13
Значение y: 5.32
Значение z: 7.43
Произведение: 13.879999999999999
```

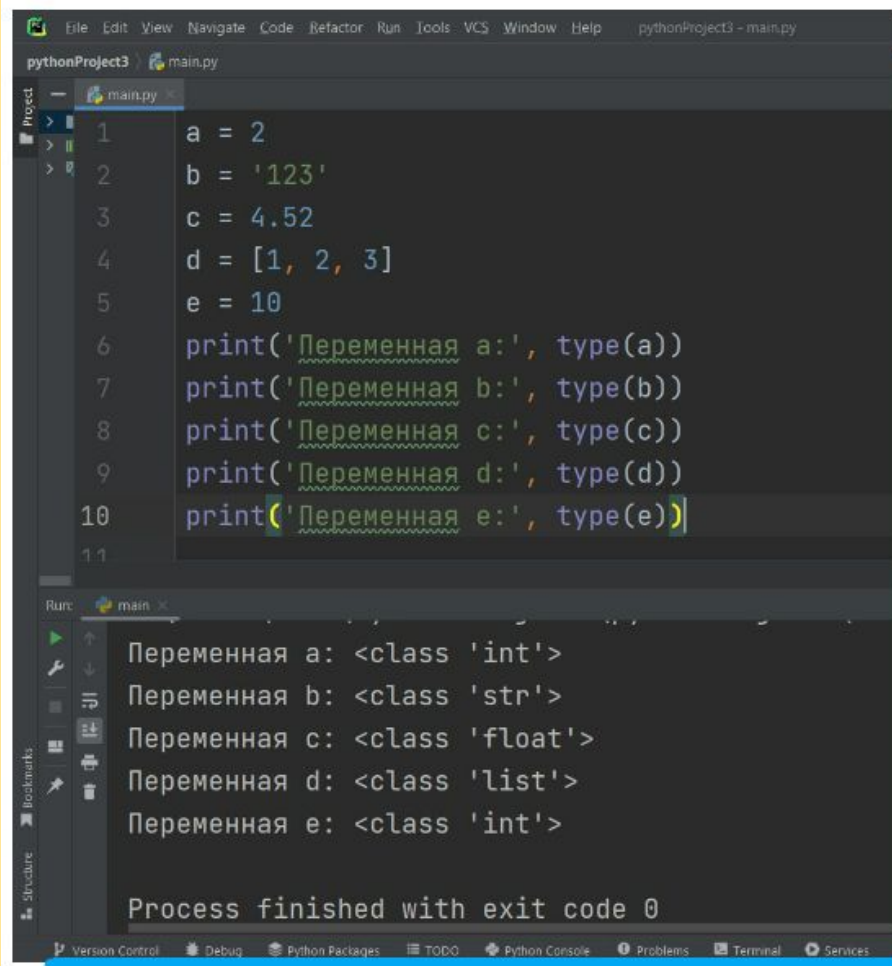
The bottom status bar of the IDE shows 'Type: In word "Произведение"'.

Задача 3.

Создайте 5 переменных и узнайте их тип с помощью `type()`.

Решение.

Напишем код для решения данной практической задачи и посмотрим на вывод консоли:



The screenshot shows an IDE window titled 'pythonProject3 - main.py'. The editor displays a Python script in 'main.py' with the following code:

```
1 a = 2
2 b = '123'
3 c = 4.52
4 d = [1, 2, 3]
5 e = 10
6 print('Переменная a:', type(a))
7 print('Переменная b:', type(b))
8 print('Переменная c:', type(c))
9 print('Переменная d:', type(d))
10 print('Переменная e:', type(e))
```

Below the editor, the 'Run' console shows the output of the script:

```
Переменная a: <class 'int'>
Переменная b: <class 'str'>
Переменная c: <class 'float'>
Переменная d: <class 'list'>
Переменная e: <class 'int'>

Process finished with exit code 0
```

The IDE interface includes a menu bar (File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help) and a sidebar with icons for Project, Run, Structure, and Bookmarks.

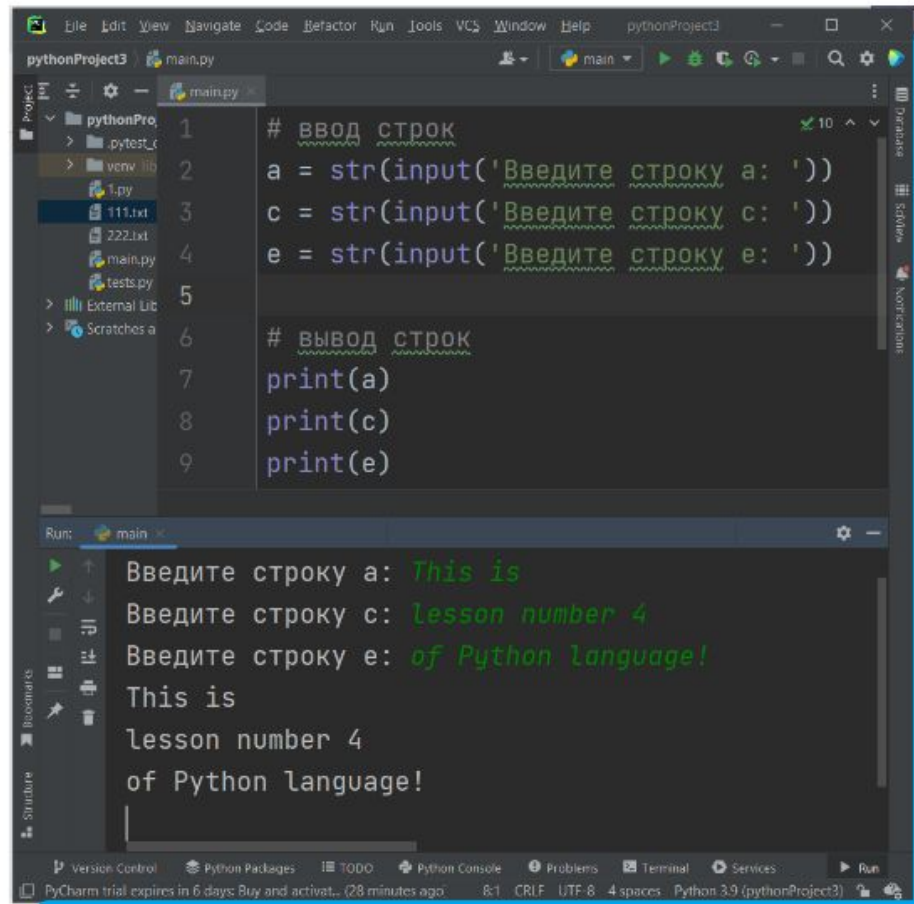
Задача 4.

Создать программу, в которой пользователь вводит три строки из символов.

Необходимо вывести данные строки отдельно на консоль.

Решение.

Напишем код для решения данной практической задачи и посмотрим на вывод консоли:



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

```
1 # ВВОД СТРОК
2 a = str(input('Введите строку a: '))
3 c = str(input('Введите строку c: '))
4 e = str(input('Введите строку e: '))
5
6 # ВЫВОД СТРОК
7 print(a)
8 print(c)
9 print(e)
```

The Run window at the bottom shows the execution output:

```
Введите строку a: This is
Введите строку c: lesson number 4
Введите строку e: of Python language!
This is
lesson number 4
of Python language!
```

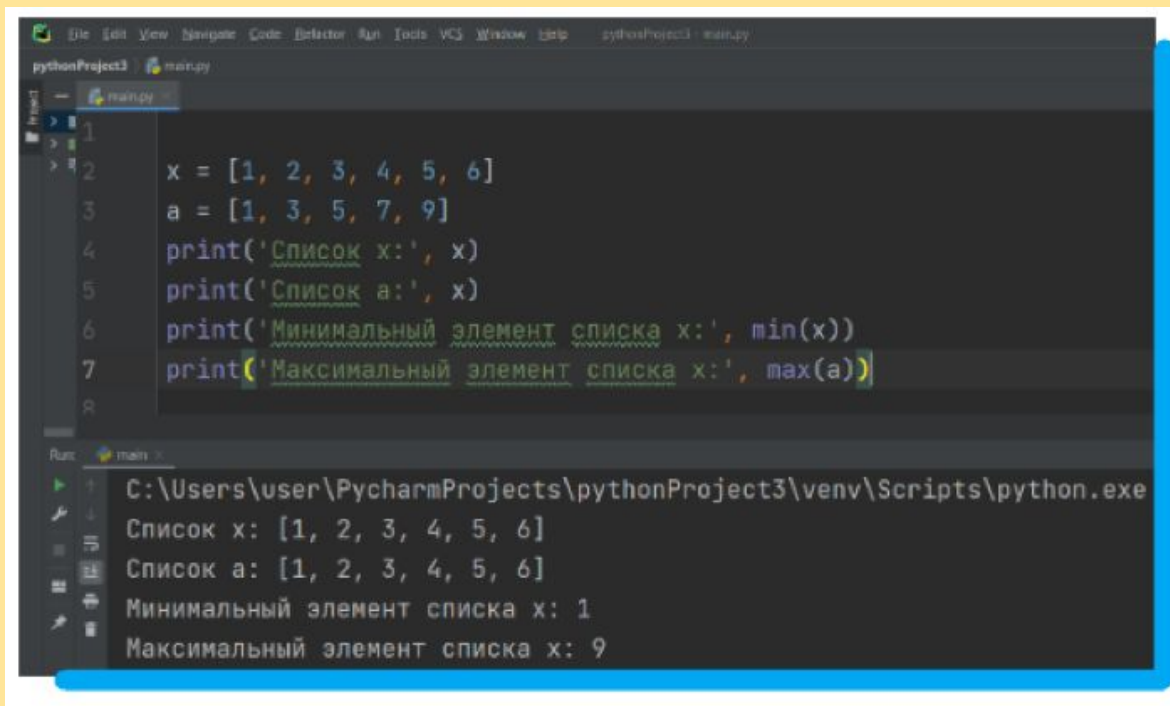
The status bar at the bottom indicates the file encoding is UTF-8, 4 spaces, and the Python version is 3.9 (pythonProject3).

Задача 5.

Создать программу, в которой вводятся списки **'x' = [1, 2, 3, 4, 5, 6]** и **'a' = [1, 3, 5, 7, 9]**. Необходимо найти максимальный элемент в списке **'a'** и минимальный элемент в списке **'x'**. Данные значения вывести на консоль.

Решение.

Напишем код для решения данной практической задачи и посмотрим на вывод консоли:



The screenshot displays the PyCharm IDE interface. The top pane shows a Python file named `main.py` with the following code:

```
1  
2 x = [1, 2, 3, 4, 5, 6]  
3 a = [1, 3, 5, 7, 9]  
4 print('Список x:', x)  
5 print('Список a:', x)  
6 print('Минимальный элемент списка x:', min(x))  
7 print('Максимальный элемент списка x:', max(a))  
8
```

The bottom pane shows the console output of the program:

```
C:\Users\user\PycharmProjects\pythonProject3\venv\Scripts\python.exe  
Список x: [1, 2, 3, 4, 5, 6]  
Список a: [1, 2, 3, 4, 5, 6]  
Минимальный элемент списка x: 1  
Максимальный элемент списка x: 9
```


1. Какую роль выполняет функция max(x) в языке <i>Python</i> ?
2. Что такое строка ?
3. Как можно передавать аргументы строке?
4. Какую роль выполняет функция print() в языке <i>Python</i> ?
5. Какое минимальное количество аргументов можно передать функции на языке <i>Python</i> за один раз?