

ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ

КАФЕДРА АВТОМАТИЗАЦИЯ ПРОИЗВОДСТВЕННЫХ ПРОЦЕССОВ

ЛЕКЦИЯ №7

# Работа со строками и текстом

СОСТАВИТЕЛЬ:            КАНД. ТЕХН. НАУК    БЫКАДОР В.С.

# Форматирование вывода

Для форматированного вывода текстовых данных в консоль в Python имеется несколько вариантов.

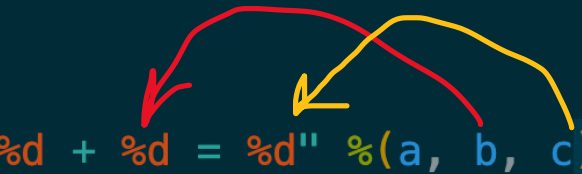


# ОСНОВЫ ВЫВОДА ДАННЫХ В КОНСОЛЬ

Базовое форматирование через %.

«Си-стиль».

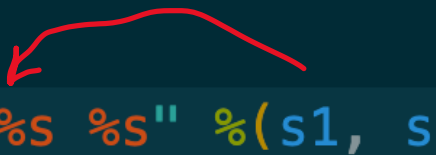
```
1
2 a = 1
3 b = 3
4 c = a + b
5
6 print("c = %d + %d = %d" %(a, b, c))
```



ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛАДКИ    ТЕРМИНАЛ

- (venv) vityalybykador@Air-Vitaly test % /Users/vityalybykador/PROJECTS/test/for\_lectures.py  
c = 1 + 3 = 4
- (venv) vityalybykador@Air-Vitaly test %

```
1
2 s1 = "Привет!"
3 s2 = "Толик"
4
5 print("%s %s" %(s1, s2))
```

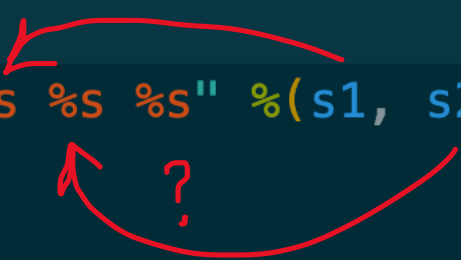


ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛАДКИ

- (venv) vityalybykador@Air-Vitaly test % /Users/vityalybykador/PROJECTS/test/for\_lectures.py  
Привет! Толик
- (venv) vityalybykador@Air-Vitaly test %

# ОСНОВЫ ВЫВОДА ДАННЫХ В КОНСОЛЬ

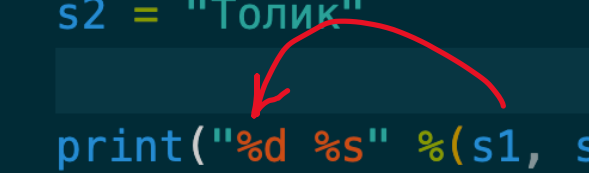
```
1
2 s1 = "Привет!"
3 s2 = "Толик"
4
5 print("%s %s %s" %(s1, s2))
```



ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛАДКИ

```
⊗ (venv) vityalybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for_lectures.py
Traceback (most recent call last):
  File "/Users/vitalybykador/PROJECTS/test/for_lectures.py", line 5, in <module>
    print("%s %s %s" %(s1, s2))
TypeError: not enough arguments for format string
○ (venv) vityalybykador@Air-Vitaly test %
```

```
1
2 s1 = "Привет!"
3 s2 = "Толик"
4
5 print("%d %s" %(s1, s2))
```



Шаблон ожидает число.

ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛАДКИ    ТЕРМИНАЛ

```
⊗ (venv) vityalybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for_lectures.py
Traceback (most recent call last):
  File "/Users/vitalybykador/PROJECTS/test/for_lectures.py", line 5, in <module>
    print("%d %s" %(s1, s2))
TypeError: %d format: a real number is required, not str
○ (venv) vityalybykador@Air-Vitaly test %
```

# ОСНОВЫ ВЫВОДА ДАННЫХ В КОНСОЛЬ

Базовое форматирование через `str()`.

«Pascal-стиль».

```
1
2  s1 = "Привет!"
3  s2 = "Толик"
4
5  print(s1 + s2)
6
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

- (venv) vityalybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for\_lectures.py  
Привет!Толик
- (venv) vityalybykador@Air-Vitaly test %

# ОСНОВЫ ВЫВОДА ДАННЫХ В КОНСОЛЬ

Базовое форматирование через `str()`.

«Pascal-стиль».

```
1
2  a = 1
3  b = 3
4  c = a + b
5
6  print("c = " + str(a) + " + " + str(b) + " = " + str(c))
7
8
9
10
11
```



Преобразование числа в строку.

ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛАДКИ    ТЕРМИНАЛ    JUPYTER

```
● (venv) vitalitybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/venv/bin/
s/vitalybykador/PROJECTS/test/for_lectures.py
c = 1 + 3 = 4
○ (venv) vitalitybykador@Air-Vitaly test %
```

# ОСНОВЫ ВЫВОДА ДАННЫХ В КОНСОЛЬ

Базовое форматирование через **f-строки**.

«Python-стиль».

```
1
2  s1 = "Привет!"
3  s2 = "Толик"
4
5  print(f'{s1} {s2}')
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

- (venv) vityalybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for\_lectures.py  
Привет! Толик
- (venv) vityalybykador@Air-Vitaly test % █

# ОСНОВЫ ВЫВОДА ДАННЫХ В КОНСОЛЬ

Базовое форматирование через **f-строки**.

«Python-стиль».

```
1  import datetime
2
3  a = 1
4  b = 2
5  c = a + b
6
7  dat = datetime.datetime.now()
8
9
10 print(f'c = {a} + {b} = {c} : {dat}')
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

ТЕРМИНАЛ

- (venv) vityalybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for\_lectures.py  
c = 1 + 2 = 3 : 2022-11-02 16:06:28.635989
- (venv) vityalybykador@Air-Vitaly test % █



# Основы вывода данных в консоль

## ЮНИКОД - СИМВОЛЫ

```
1
2 print('\u2211')
3
4
```

ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ

```
● (venv) vityalybykador@Air-Vita
s/vityalybykador/PROJECTS/test
Σ
○ (venv) vityalybykador@Air-Vita
```

`\u` – указывает на то, что следующие **4-ре!** числа будут кодом символа в Юникоде в шестнадцатеричной системе счисления.


**2211** – код символа в Юникоде.

# ОСНОВЫ ВЫВОДА ДАННЫХ В КОНСОЛЬ

## ЮНИКОД - СИМВОЛЫ

```
1  
2 print('\u26F5')  
3  
4
```

ПРОБЛЕМЫ      ВЫХОДНЫЕ ДАННЫЕ

- (venv) vityalybykador@Air-Vital  
s/vityalybykador/PROJECTS/test/  
○  (venv) vityalybykador@Air-Vital



# Форматирования функцией `format()`

Общий формат записи

`'{} {} {} ... {}' .format('str1', 'str2', 'str3', ..., 'strN')`

↑  
Форматирование  
для строки `'str1'`

↑  
Форматирование  
для строки `'str2'`

↑  
Форматирование  
для строки `'str3'`

↑  
Форматирование  
для строки `'strN'`

# Примеры форматирования функцией format()

```
1
2 s = '{}'.format('word')
3
4 print(s)
```

ПРОБЛЕМЫ      ВЫХОДНЫЕ ДАННЫЕ      КОНСОЛЬ ОТЛА

- (venv) vityalybykador@Air-Vitaly test % /Users/vityalybykador/PROJECTS/test/for\_lectures.py  
word
- (venv) vityalybykador@Air-Vitaly test % █

```
1
2 s = '{:>10}'.format('word')
3
4 print(s)
```

ПРОБЛЕМЫ      ВЫХОДНЫЕ ДАННЫЕ      КОНСОЛЬ ОТЛА

- (venv) vityalybykador@Air-Vitaly test % /Users/vityalybykador/PROJECTS/test/for\_lectures.py  
word
- (venv) vityalybykador@Air-Vitaly test % /Users/vityalybykador/PROJECTS/test/for\_lectures.py  
----- word
- (venv) vityalybykador@Air-Vitaly test % █

# Примеры форматирования функцией format()

```
1
2 s = '{:10}{}'.format('word', 'hello')
3
4 print(s)
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ

- (venv) vityalybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for\_lectures.py  
word\_\_\_\_\_hello
- (venv) vityalybykador@Air-Vitaly test %

```
1
2 s = '{:_<10}{}'.format('word', 'hello')
3
4 print(s)
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

ТЕРМИНАЛ

- (venv) vityalybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for\_lectures.py  
word\_\_\_\_\_hello
- (venv) vityalybykador@Air-Vitaly test %

# Примеры форматирования функцией format()

```
1  lst_cities = ['Москва', 'Нижний Новгород', 'Омск']
2  lst_populations = [13010, 1226, 1126]
3
4  print('')
5
6  for city, population in zip(lst_cities, lst_populations):
7      |    print(f'{city} {population}')
8
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

ТЕРМИНАЛ

JUPYTER

- (venv) vityalybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/venv/bin/python3 s/vitalybykador/PROJECTS/test/for\_lectures.py

Москва 13010

Нижний Новгород 1226

Омск 1126

- (venv) vityalybykador@Air-Vitaly test % █

# Примеры форматирования функцией format()

```
1 lst_cities = ['Москва', 'Нижний Новгород', 'Омск']
2 lst_populations = [13010, 1226, 1126]
3
4 print('')
5
6 for city, population in zip(lst_cities, lst_populations):
7     | print('{:20.16}{}'.format(city, population))
8
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

ТЕРМИНАЛ

JUPYTER

● (venv) vitalitybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/venv/bin/s/vitalybykador/PROJECTS/test/for\_lectures.py

Москва	13010
Нижний Новгород	1226
Омск	1126

○ (venv) vitalitybykador@Air-Vitaly test % █

# Примеры форматирования функцией format()

```
print('{:20.16}{}'.format(city, population))
```

Москва	13010
Нижний Новгород	1226
Омск	1126

16

20



# Примеры форматирования функцией format()

```
7 print('{:20.3f}'.format(city, population))
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

ТЕРМИНАЛ

JUPYTER

- (venv) vitalitybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for\_lectures.py

Москва	13010
Нижний Новгород	1226
Омск	1126

- (venv) vitalitybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for\_lectures.py

Москв	13010
Нижни	1226
Омск	1126

- (venv) vitalitybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for\_lectures.py

Мос	13010
Ниж	1226
Омс	1126

- (venv) vitalitybykador@Air-Vitaly test %

# Примеры форматирования функцией format()

```
1 lst_cities = ['Москва', 'Нижний Новгород', 'Омск']
2 lst_populations = [13010, 1226, 1126]
3
4 print('')
5
6 for city, population in zip(lst_cities, lst_populations):
7     print('{:20.16}{:5d}'.format(city, population))
8
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

ТЕРМИНАЛ

JUPYTER

```
• (venv) vityalybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/venv/bin/p
s/vitalybykador/PROJECTS/test/for_lectures.py
```

Москва	13010
Нижний Новгород	1226
Омск	1126

```
• (venv) vityalybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/venv/bin/p
s/vitalybykador/PROJECTS/test/for_lectures.py
```

Москва	13010
Нижний Новгород	1226
Омск	1126

```
• (venv) vityalybykador@Air-Vitaly test %
```

# Примеры форматирования функцией format()

```
3
4 print('{}'.format(34))
5
6
```

ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛ

- (venv) vityalybykador@Air-Vitaly test % /User  
s/vitalybykador/PROJECTS/test/for\_lectures.p

34

- (venv) vityalybykador@Air-Vitaly test % █

```
3
4 print('{:10d}'.format(34))
5
6
```

ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛ

- (venv) vityalybykador@Air-Vitaly test % /User  
s/vitalybykador/PROJECTS/test/for\_lectures.p

34

- (venv) vityalybykador@Air-Vitaly test % █

# Примеры форматирования функцией format()


```
3  
4 print('{:+d}'.format(34))  
5  
6
```



ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛ

• (venv) vityalybykador@Air-Vitaly test % /Users/vityalybykador/PROJECTS/test/for\_lectures

+34




```
3  
4 print('{:=+5d}'.format(34))  
5  
6
```



ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛ

• (venv) vityalybykador@Air-Vitaly test % /Users/vityalybykador/PROJECTS/test/for\_lectures.py

+ 34



○ (venv) vityalybykador@Air-Vitaly test %

# Примеры форматирования функцией format()

```
3  
4 print('{:=+05d}'.format(34))  
5  
6
```



ПРОБЛЕМЫ

- (venv) vit  
s/vitalyby

+0034

3

```
4 print('{:=+010.1f}'.format(34.14532))  
5  
6
```

ПРОБЛЕМЫ

ВЫХОДН

- (venv) vitalitybykador  
s/vitalybykador/PROJ

+0000034.1

```
4 print('{:=+010.3f}'.format(34.14532))  
5  
6
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

ТЕРМ

- (venv) vitalitybykador@Air-Vitaly test % /Users/vitalybyka  
s/vitalybykador/PROJECTS/test/for\_lectures.py

+00034.145

# Некоторые полезные функции для работы со строками

```
3 s = 'привет'
4 print(s.upper())
5
```

ПРОБЛЕМЫ      ВЫХОДНЫЕ ДАННЫЕ

- (venv) vitalitybykador@Air-Vitaly test  
s/vitalybykador/PROJECTS/test/for\_le

ПРИВЕТ

```
3 s = 'HELLO'
4 print(s.lower())
5
```

ПРОБЛЕМЫ      ВЫХОДНЫЕ ДАННЫЕ

- (venv) vitalitybykador@Air-Vitaly test  
s/vitalybykador/PROJECTS/test/for\_le

hello

```
3 s = 'HELLO'
4 print(s.capitalize())
5
```

ПРОБЛЕМЫ      ВЫХОДНЫЕ ДАННЫЕ      КОН

- (venv) vitalitybykador@Air-Vitaly test  
s/vitalybykador/PROJECTS/test/for\_le

Hello

# Некоторые полезные функции для работы со строками

Разделение строки на составляющие методом `split()` объекта строки.

```
3 s = 'Иванов;Петр;Аристархович;59'
4 print(s)
5 result = s.split(';')
6
7 print()
8 for s_i in result:
9     print(s_i)
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

• (venv) vitalitybykador@Air-Vitaly test % /Users/vitaly

Иванов;Петр;Аристархович;59

Иванов  
Петр  
Аристархович  
59

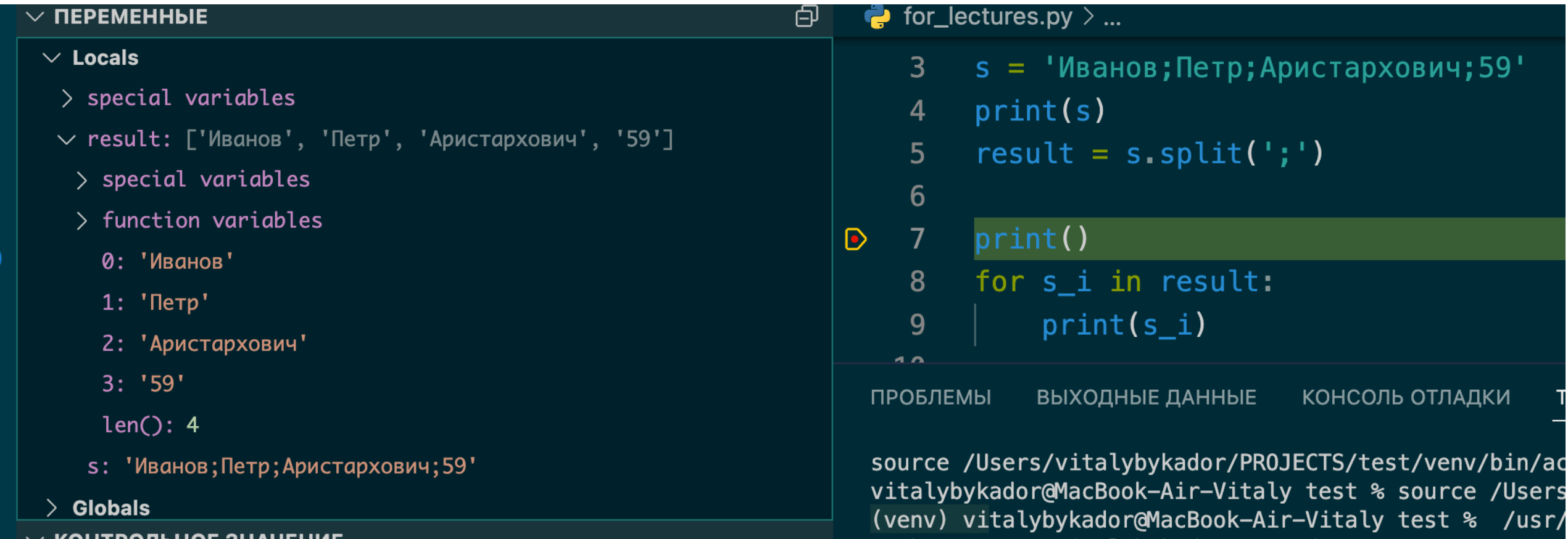
Исходная строка

Символ-разделитель  
(может быть любой  
символ не  
встречающийся в  
основном тексте)

Каждое слово исходного  
текста как элемент  
списка.

# Некоторые полезные функции для работы со строками

Разделение строки на составляющие методом `split()` объекта строки.



The screenshot shows a Python IDE with two main panels. The left panel, titled 'ПЕРЕМЕННЫЕ' (Variables), displays the 'Locals' section. It shows a variable 'result' containing a list of strings: ['Иванов', 'Петр', 'Аристархович', '59']. Below this, the 'Globals' section is partially visible. The right panel shows a Python script named 'for\_lectures.py'. The script defines a string 's' with the value 'Иванов;Петр;Аристархович;59', prints it, splits it by semicolons into the 'result' list, and then iterates over the list to print each element. The current execution state shows the 'print()' call on line 7 being executed, indicated by a yellow arrow cursor. At the bottom of the right panel, there is a terminal window showing the command prompt and the execution of the script.

```
for_lectures.py > ...  
3 s = 'Иванов;Петр;Аристархович;59'  
4 print(s)  
5 result = s.split(';')  
6  
7 print()  
8 for s_i in result:  
9     print(s_i)  
10
```

ПЕРЕМЕННЫЕ

Locals

- > special variables
- > result: ['Иванов', 'Петр', 'Аристархович', '59']
  - > special variables
  - > function variables
  - 0: 'Иванов'
  - 1: 'Петр'
  - 2: 'Аристархович'
  - 3: '59'
  - len(): 4
- s: 'Иванов;Петр;Аристархович;59'
- > Globals

ПРОБЛЕМЫ Выходные данные КОНСОЛЬ ОТЛАДКИ

```
source /Users/vitalybykador/PROJECTS/test/venv/bin/activate  
vitalybykador@MacBook-Air-Vitaly test % source /Users/vitalybykador/PROJECTS/test/venv/bin/activate  
(venv) vitalybykador@MacBook-Air-Vitaly test % python for_lectures.py
```

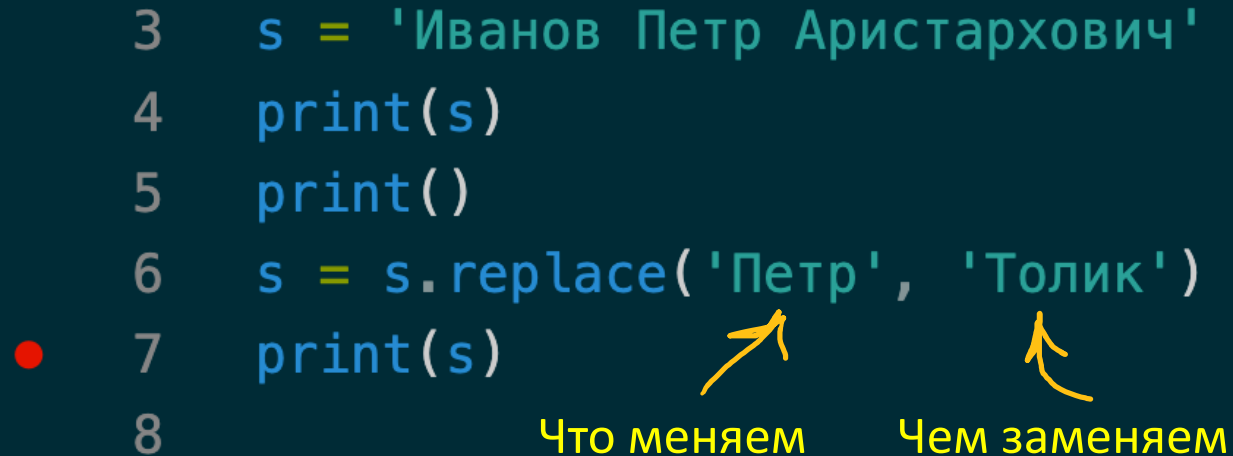


# Некоторые полезные функции для работы со строками

Замена текста методом `replace()` объекта строки.

```
3 s = 'Иванов Петр Аристархович'
4 print(s)
5 print()
6 s = s.replace('Петр', 'Толик')
7 print(s)
8
```

Что меняем      Чем заменяем



ПРОБЛЕМЫ


ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

```
• (venv) vitybykador@Air-Vitaly test % /Users/vita
s.py
```

Иванов Петр Аристархович

Иванов Толик Аристархович



```
○ (venv) vitybykador@Air-Vitaly test % █
```

# Некоторые полезные функции для работы со строками

Поиск слова «текст» в предложении методом `find()` объекта строки.

```
3 s = '''Завершённость высказывания связана со смысловой цельностью текста.  
4 Показателем законченности текста является возможность подобрать к нему заголовок,  
5 отражающий его содержание.''' # https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82  
6  
7 print(s.find('текст'))  
8
```

Искомое слово/текст.

ПРОБЛЕМЫ Выходные данные КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ JUPYTER

• (venv) vitalitybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/venv/bin/python /Users/vitalybykador/PROJECTS/test/fo

59 ← Позиция, с которой начинается слово «текст».

```
5 отражающий его содержание.''' # https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82  
6  
7 print(s.find('Привет'))  
8
```

ПРОБЛЕМЫ Выходные данные КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ JUPYTER

• (venv) vitalitybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test

-1

← Слово не найдено.

# Основы регулярных выражений в Python

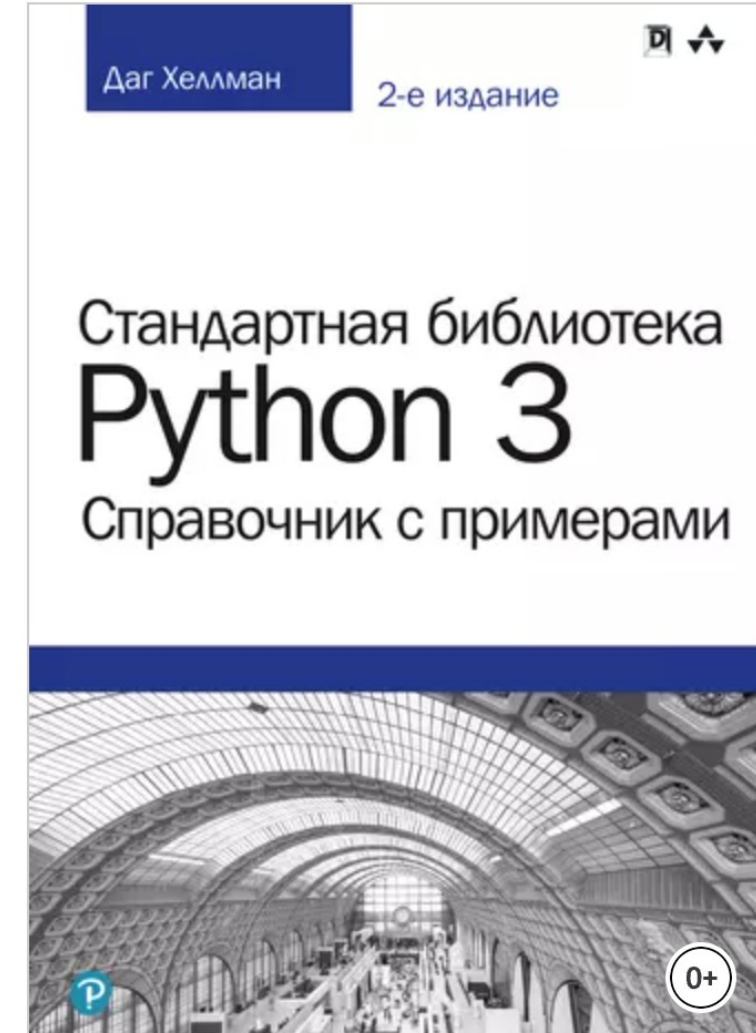
Используя регулярные выражения можно создавать сложные шаблоны поиска каких-либо строк в текстах.

Сами по себе регулярные выражение представляют собой специфичный микро язык программирования для обработки строк, которые можно отдельно изучать. Во многих языках программирования регулярные выражения представлены в той или иной степени полноты.

Книга о регулярных  
выражениях вообще



Книги о регулярных выражениях в языке  
**Python** и о многом другом, что можно  
программировать в **Python**



# Основы регулярных выражений в Python

**Пример.** Пусть есть строка из которой необходимо получить номер телефона. Известен формат записи номера телефона, но должно быть абсолютно безразлично какие цифры записаны, регулярное выражение должно находить любой номер телефона в строке, который удовлетворяет формату записи.

```
'мой телефон +7(955)24-31-544'
```

# Основы регулярных выражений в Python

Составим шаблон для поиска номера телефона используя специальный символ `\d` обозначающий любой цифровой символ от **0** до **9**.

The diagram illustrates the mapping from a specific phone number to its corresponding regular expression pattern. The phone number `+7(955)24-31-544` is shown in a dark teal box. Red arrows point from each part of the number to its equivalent in the regular expression pattern `'\+\d\(\d\d\d\) \d\d-\d\d-\d\d\d'` shown in a lighter teal box below. Specifically, the arrow from `+` points to `\+`, from `7` to `\d`, from `(` to `\(`, from `9` to `\d`, from `5` to `\d`, from `5` to `\d`, from `)` to `\)`, from  to , from `2` to `\d`, from `4` to `\d`, from `-` to `-`, from `3` to `\d`, from `1` to `\d`, from `-` to `-`, from `5` to `\d`, from `4` to `\d`, and from `4` to `\d`. The regular expression pattern is enclosed in single quotes.

```
+7(955)24-31-544
```

```
'\+\d\(\d\d\d\) \d\d-\d\d-\d\d\d'
```

# Основы регулярных выражений в Python

```
1 import re
2
3 source = 'мой телефон +7(955)24-31-544'
4 pattern = '\+\d+(\d\d\d)\d\d-\d\d-\d\d\d'
5
6 m = re.search(pattern, source)
7 if m is not None:
8     print(m.group())
9 else:
10    print('НЕТ СООТВЕТСТВИЙ!')
11
```

Библиотека для работы с регулярными выражениями.

Специальные метод (их несколько).

Обработка результата поиска.

ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛАДКИ    ТЕРМИНАЛ

```
• (venv) vitalitybykador@Air-Vitaly test % /Users/vitalybykador/PROJ
s.py
+7(955)24-31-544
• (venv) vitalitybykador@Air-Vitaly test %
```

# Основы регулярных выражений в Python

Усовершенствуем шаблон, используя возможности языка регулярных выражений.

The diagram illustrates the transformation of a regular expression template. The top line shows the original template: `'\+\d\(\d\d\d\) \d\d-\d\d-\d\d\d'`. Red curly braces are placed under the sequences `\d\d\d`, `\d\d`, `\d\d`, and `\d\d\d`. Red arrows point from each brace to the corresponding quantified version in the bottom line: `'\+\d\(\d{3}\) \d{2}-\d{2}-\d{3}'`. In the bottom line, the quantified parts `{3}` and `{2}` are highlighted in orange, while the backslashes and other characters are in teal.

```
'\+\d\(\d\d\d\) \d\d-\d\d-\d\d\d'
```

↓ ↓ ↓ ↓

```
'\+\d\(\d{3}\) \d{2}-\d{2}-\d{3}'
```



# Основы регулярных выражений в Python

```
1  import re
2
3  source = 'мой телефон +7(955)24-31-544'
4  pattern = '\+\d+(\d{3})\d{2}-\d{2}-\d{3}'
5
6  m = re.search(pattern, source)
7  if m is not None:
8      |   print(m.group())
9  else:
10     |   print('НЕТ СООТВЕТСТВИЙ!')
11
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

ТЕРМИНАЛ

- (venv) vityalybykador@Air-Vitaly test % /Users/vitalybykador/PRO  
s.py  
+7(955)24-31-544
- (venv) vityalybykador@Air-Vitaly test % █

# Основы регулярных выражений в Python

Усложним исходную строку.

```
source = '''мой телефон +7(955)24-31-544, телефон Лизы -> +7(906)77-24-008,  
телефон Толика: +7(916)05-55-327'''
```

И найдем в этой строке все телефоны.

# Основы регулярных выражений в Python

```
1  import re
2
3  source = '''мой телефон +7(955)24-31-544, телефон Лизы -> +7(906)77-24-008,
4  телефон Толика: +7(916)05-55-327'''
5
6  pattern = '\+\d\(\d{3}\)\d{2}-\d{2}-\d{3}'
7
8  m = re.search(pattern, source)
9  if m is not None:
10 |     print(m.group())
11 else:
12 |     print('НЕТ СООТВЕТСТВИЙ!')
13
```

ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛАДКИ    ТЕРМИНАЛ    JUPYTER

```
● (venv) vitalitybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/venv/bin/python /Users/vitalybykador/s.py
+7(955)24-31-544
○ (venv) vitalitybykador@Air-Vitaly test %
```

Найден только первый телефон :((.

Нужно менять метод поиска, так как метод **search()** ищет только до первого совпадения с шаблоном.

# Основы регулярных выражений в Python

```
1 import re
2
3 source = '''мой телефон +7(955)24-31-544, телефон Лизы -> +7(906)77-24-008,
4 телефон Толика: +7(916)05-55-327'''
5
6 pattern = '\+\d+(\d{3})\d{2}-\d{2}-\d{3}'
7
8 m = re.findall(pattern, source)
9 if m is not None:
10     print(m)
11 else:
12     print('НЕТ СООТВЕТСТВИЙ!')
```

Другой метод поиска шаблона в тексте.

Теперь результат возвращается в списке.

ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛАДКИ    ТЕРМИНАЛ    JUPYTER

```
• (venv) vityalybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/venv/bin/python /Users/vitalyby
s.py
['+7(955)24-31-544', '+7(906)77-24-008', '+7(916)05-55-327']
○ (venv) vityalybykador@Air-Vitaly test %
```

# Основы регулярных выражений в Python

В регулярных выражениях существуют шаблоны и для других видов символов, специальные конструкции для указания правил поиска подстрок в текстах. Регулярные выражения – это отдельный язык программирования.

# Основы регулярных выражений в Python

[https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D1%8B%D0%B5\\_%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F)

Символ	Возможный эквивалент <sup>[9]</sup>	Соответствие
<code>\d</code>	<code>[0-9]</code>	Цифровой символ
<code>\D</code>	<code>[^0-9]</code>	Нецифровой символ
<code>\s</code>	<code>[\f\n\r\t\v]</code>	Пробельный символ
<code>\S</code>	<code>[^\f\n\r\t\v]</code>	Непробельный символ Пример: Выражение вида <code>^\S.*</code> ил
<code>\w<sup>[10]</sup></code>	<code>[A-Za-z0-9_]</code>	Буквенный или цифровой симво Пример: Выражение вида <code>\w+</code> будет
<code>\W<sup>[11]</sup></code>	<code>[^A-Za-z0-9_]</code>	Любой символ, кроме буквенно

Представление	Позиция	Пример	Соответствие
<code>^</code>	Начало текста (или строки при модификаторе <code>?m</code> )	<code>^a</code>	<code>aa aa</code>
<code>\$</code>	Конец текста (или строки при модификаторе <code>?m</code> )	<code>a\$</code>	<code>aaa aa</code>
<code>\b</code>	Граница слова	<code>a\b</code>	<code>aa</code> <code>aa</code>

Представление	Число повторений	Эквивалент	Пример	Соответствие
<code>?</code>	Ноль или одно	<code>{0,1}</code>	<code>colou?r</code>	<code>color</code> , <code>colour</code>
<code>*</code>	Ноль или более	<code>{0,}</code>	<code>colou*r</code>	<code>color</code> , <code>colour</code> , <code>colouur</code> и т. д.
<code>+</code>	Одно или более	<code>{1,}</code>	<code>colou+r</code>	<code>colour</code> , <code>colouur</code> и т. д. (но не <code>color</code> )

Представление	Вид просмотра	Пример
<code>(?=шаблон)</code>	Позитивный просмотр вперёд	Людovic ( <code>?=XVI</code> )
<code>(?!шаблон)</code>	Негативный просмотр вперёд (с отрицанием)	Людovic ( <code>?!XVI</code> )

Представление	Число повторений	Пример	Соответствие
<code>{n}</code>	Ровно <i>n</i> раз	<code>colou{3}r</code>	<code>colouuur</code>
<code>{m,n}</code>	От <i>m</i> до <i>n</i> включительно	<code>colou{2,4}r</code>	<code>colouur</code> , <code>colouuur</code> , <code>colouuuur</code>
<code>{m,}</code>	Не менее <i>m</i>	<code>colou{2,}r</code>	<code>colouur</code> , <code>colouuur</code> , <code>colouuuur</code> и т. д.
<code>{,n}</code>	Не более <i>n</i>	<code>colou{,3}r</code>	<code>color</code> , <code>colour</code> , <code>colouur</code> , <code>colouuur</code>

# Основы регулярных выражений в Python

Анализатор структуры ВКР

Сведения о структуре ВКР [redacted] ант.pdf

Правильная структура документа		стр. 66
Введение		стр. 5
Экономика	$\Delta = -3$	стр. 45
Экология и БЖД	$\Delta = -7$	стр. 52
Заключение		стр. 55
Использованные источники		стр. 56

Закреть

# Основы регулярных выражений в Python

```
"ECOLOGY": {  
    "regexp": ["\\d\\..*\\s*Безопасно\\w*\\s*и\\s*экологично\\w*\\s*[работ\\w*|проект\\w*]"],
```

```
"ECONOMICS": {  
    "regexp": [ "\\d\\..*\\s*Экономическ\\w*\\s*обоснован\\w*\\s*[работ\\w*|проект\\w*]|",  
                "\\d\\..*\\s*Техн\\w*-экономическ\\w*\\s*обосн\\w*|",  
                "\\d\\..*\\s*Эконом\\w*\\s*расч\\w*\\s*эффект\\w*"],
```

```
"REFERENCES": {  
    "regexp": [ "\\s*Перече\\w*\\s*использ[\\w*\\s]*ресурс\\w*|",  
                "\\s*Перече\\w*\\s*использ\\w*\\s*информац\\w*\\s*ресур\\w*|",  
                "\\s*Спис\\w*\\s*использ\\w*\\s*информац\\w*\\s*ресур\\w*|",  
                "\\s*Спис\\w*\\s*информац\\w*\\s*ресур\\w*|",  
                "\\s*Спис\\w*\\s*использ\\w*\\s*источник\\w*"],
```