

Основы языка программирования Python¹

Строки
(класс `str`)

Способы создания строк

Однострочные (single line) строки

foo = "Hello"

bar = 'Привет'

print("type(foo):", type(foo))

print("type(bar):", type(bar))

type(foo): <class 'str'>

type(bar): <class 'str'>

Многострочные (multiline) строки

```
foo = """Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua."""
```

```
bar = '''Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua.'''
```

```
print("type(foo):", type(foo))  
print("type(bar):", type(bar))
```

```
print()  
print("foo:", foo)  
print()  
print("bar:", bar)
```

```
type(foo): <class 'str'>  
type(bar): <class 'str'>
```

```
foo: Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua.
```

```
bar: Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua.
```

Отступы и многострочный текст

if True:

```
foo = """Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua."""
```

```
bar = """Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua."""
```

```
print("foo:", foo)  
print()  
print("bar:", bar)
```

```
foo: Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua.
```

```
bar: Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua.
```

```
foo = "Этот текст содержит 'одинарные' кавычки"  
bar = 'Этот текст содержит "двойные" кавычки'  
spam = 'Этот текст содержит \'одинарные\' кавычки'  
eggs = "Этот текст содержит \"двойные\" кавычки"
```

```
print("foo: ", foo)  
print("bar: ", bar)  
print("spam:", spam)  
print("eggs:", eggs)
```

```
foo: Этот текст содержит 'одинарные' кавычки  
bar: Этот текст содержит "двойные" кавычки  
spam: Этот текст содержит 'одинарные' кавычки  
eggs: Этот текст содержит "двойные" кавычки
```

Использование кавычек в тексте

```
foo = '''Этот текст содержит 'одинарные' кавычки'''  
bar = """Этот текст содержит "двойные" кавычки"""  
  
print("foo: ", foo)  
print("bar: ", bar)
```

```
foo:  Этот текст содержит 'одинарные' кавычки  
bar:  Этот текст содержит "двойные" кавычки
```

Использование обратных слешей в тексте

```
foo = "Этот текст содержит обратные \слеси\\"  
bar = "Этот текст содержит обратные \\слеси\\"  
  
print("foo: ", foo)  
print("bar: ", bar)
```

```
foo:  Этот текст содержит обратные \слеси\  
bar:  Этот текст содержит обратные \слеси\
```



```
foo = "Lorem ipsum dolor sit amet,\nconsectetur adipiscing elit"  
bar = "Lorem ipsum dolor sit amet,\tconsectetur adipiscing elit"  
  
print("foo: ", foo)  
print()  
print("bar: ", bar)
```

```
foo: Lorem ipsum dolor sit amet,  
consectetur adipiscing elit
```

```
bar: Lorem ipsum dolor sit amet,      consectetur adipiscing elit
```

Использование "сырых" (raw) строк

```
foo = "Этот текст\\tсодержит \\\\ слишком много \\\\ обратных слешей\\n"  
bar = r'Этот текст\\tсодержит \\ слишком много \\ обратных слешей\\n'  
  
print("foo: ", foo)  
print("bar: ", bar)
```

```
foo:  Этот текст\\tсодержит \\ слишком много \\ обратных слешей\\n  
bar:  Этот текст\\tсодержит \\ слишком много \\ обратных слешей\\n
```

Вставка символов Unicode по коду

```
foo = "Это символ Unicode - \u03b4"  
bar = "Это символ Unicode - \U0001f923"  
  
print("foo:", foo)  
print("bar:", bar)
```

foo: Это символ Unicode - δ
bar: Это символ Unicode - 🍷

Вставка символов Unicode по названию

```
foo = "Это символ Unicode - \N{Greek Small Letter Delta}"  
bar = "Это символ Unicode - \N{rolling on the floor laughing}"  
  
print("foo:", foo)  
print("bar:", bar)
```

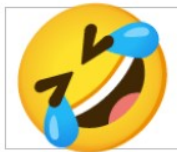
foo: Это символ Unicode - δ
bar: Это символ Unicode - 🤪

Вставка символов Unicode непосредственно в текст

```
foo = "Это символ Unicode - δ"  
bar = "Это символ Unicode - 🌀"  
  
print("foo:", foo)  
print("bar:", bar)
```

```
foo: Это символ Unicode - δ  
bar: Это символ Unicode - 🌀
```

Unicode Character “😂” (U+1F923)



| | |
|------------------------|--|
| Name: | Rolling On the Floor Laughing ^[1] |
| Unicode Version: | 9.0 (June 2016) ^[2] |
| Block: | Supplemental Symbols and Pictographs, U+1F900 - U+1F9FF ^[3] |
| Plane: | Supplementary Multilingual Plane, U+10000 - U+1FFFF ^[3] |
| Script: | Code for undetermined script (Zyyy) ^[4] |
| Category: | Other Symbol (So) ^[1] |
| Bidirectional Class: | Other Neutral (ON) ^[1] |
| Combining Class: | Not Reordered (0) ^[1] |
| Character is Mirrored: | No ^[1] |
| HTML Entity: | 🤣 🤣 |
| UTF-8 Encoding: | 0xF0 0x9F 0xA4 0xA3 |
| UTF-16 Encoding: | 0xD83E 0xDD23 |
| UTF-32 Encoding: | 0x0001F923 |

```
foo = 10
foo_str = str(foo)

bar = 20.5
bar_str = str(bar)

spam = 15 - 5j
spam_str = str(spam)

eggs = {"key_1": 10, "key_2": 20, "key_3": 30}
eggs_str = str(eggs)

baz = [10, 20, 42]
baz_str = str(baz)
```

```
print("foo_str: ", foo_str)
print("bar_str: ", bar_str)
print("spam_str:", spam_str)
print("eggs_str:", eggs_str)
print("baz_str: ", baz_str)
```

```
foo_str: 10
bar_str: 20.5
spam_str: (15-5j)
eggs_str: {'key_1': 10, 'key_2': 20, 'key_3': 30}
baz_str: [10, 20, 42]
```

```
foo = 10  
foo_str = str(foo)  
foo_repr = repr(foo)
```

```
bar = 20.5  
bar_str = str(bar)  
bar_repr = repr(bar)
```

```
baz = [10, 20, 42]  
baz_str = str(baz)  
baz_repr = repr(baz)
```

```
print("foo_str: ", foo_str)  
print("foo_repr: ", foo_repr)
```

```
print("bar_str: ", bar_str)  
print("bar_repr: ", bar_repr)
```

```
print("baz_str: ", baz_str)  
print("baz_repr: ", baz_repr)
```

```
foo_str: 10  
foo_repr: 10  
bar_str: 20.5  
bar_repr: 20.5  
baz_str: [10, 20, 42]  
baz_repr: [10, 20, 42]
```


Сравнение результатов работы функций
str() и repr() для более сложных объектов

```
import datetime
```

```
date = datetime.datetime(2024, 3, 10, 18, 40, 2)
```

```
print("str(date): ", str(date))
```

```
print("repr(date):", repr(date))
```

```
print("date:      ", date)
```

```
str(date):  2024-03-10 18:40:02
```

```
repr(date): datetime.datetime(2024, 3, 10, 18, 40, 2)
```

```
date:      2024-03-10 18:40:02
```

Действия со строками

Получение длины строки

```
foo = "Hello"  
bar = "Привет"  
spam = "你好"  
baz = "\U0001f923"  
  
print("len(foo): ", len(foo))  
print("len(bar): ", len(bar))  
print("len(spam):", len(spam))  
print("len(baz): ", len(baz))
```

```
len(foo):  5  
len(bar):  6  
len(spam): 2  
len(baz):  1
```

Получение символов по индексу

```
foo = "Привет \U0001f923"
```

```
print("foo[0]: ", foo[0])
```

```
print("foo[3]: ", foo[3])
```

```
print("foo[-1]:", foo[-1])
```

foo[0]: П

foo[3]: В

foo[-1]: 🍷

Строка - неизменяемый объект

foo = "Привет"

Ошибка!

foo[0] = "X"

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'str' object does not support item assignment

Получение подстроки

```
spam = "Lorem ipsum dolor sit amet, consectetur adipiscing elit"
```

```
print("spam[2:10]:      ", spam[2:10])
print("spam[2:14:2]:    ", spam[2:14:2])
print("spam[9:1:-1]:     ", spam[9:1:-1])
print("spam[:5]:         ", spam[:5])
print("spam[-4:]:        ", spam[-4:])
print("spam[:]:          ", spam[:])
print("spam[::-1]:       ", spam[::-1])
```

```
spam[2:10]:      rem ipsu
spam[2:14:2]:    rmismd
spam[9:1:-1]:    uspi mer
spam[:5]:        Lorem
spam[-4:]:       elit
spam[:]:         Lorem ipsum dolor sit amet, consectetur adipiscing elit
spam[::-1]:      tile gnic sipida rutetcesnoc ,tema tis rolod muspi merol
```

Использование операторов in / not in

```
spam = "Lorem ipsum dolor sit amet, consectetur adipiscing elit"  
foo = "ipsum"  
bar = "hello"
```

```
print("foo in spam:      ", foo in spam)  
print("bar not in spam:", bar not in spam)
```

```
foo in spam:      True  
bar not in spam: True
```

```
print(dir(str))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__',  
 '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__',  
 '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__',  
 '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__',  
 '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__',  
 '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count',  
 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum',  
 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric',  
 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans',  
 'partition', 'removeprefix', 'removesuffix', 'replace', 'rfind', 'rindex', 'rjust',  
 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title',  
 'translate', 'upper', 'zfill']
```


Использование методов startswith() / endswith()

```
foo = "images/picture.png"  
bar = "data/application.exe"
```

```
print("foo.startswith('images'):", foo.startswith('images'))  
print("bar.startswith('images'):", bar.startswith('images'))  
print("foo.endswith('.png'):", foo.endswith('.png'))  
print("bar.endswith('.png'):", bar.endswith('.png'))
```

```
foo.startswith('images'): True  
bar.startswith('images'): False  
foo.endswith('.png'): True  
bar.endswith('.png'): False
```

```
str.find(sub[, start[, end]])
```

Return the lowest index in the string where substring *sub* is found within the slice `s[start:end]`.

Optional arguments *start* and *end* are interpreted as in slice notation. Return `-1` if *sub* is not found.

Использование метода find()

```
foo = "Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet"
```

```
print("foo.find('ipsum'):          ", foo.find('ipsum'))
print("foo.find('Ipsum'):          ", foo.find('Ipsum'))
print("foo.find('ipsum', 7):       ", foo.find('ipsum', 7))
print("foo.find('ipsum', 35, 60): ", foo.find('ipsum', 35, 60))
```

```
foo.find('ipsum'):          6
foo.find('Ipsum'):          -1
foo.find('ipsum', 7):       34
foo.find('ipsum', 35, 60): -1
```

Нахождение всех позиций заданной подстроки

```
foo = "Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet"  
sub = "ipsum"
```

```
result = []  
pos = 0
```

```
while pos != -1:  
    pos = foo.find(sub, pos)  
    if pos != -1:  
        result.append(pos)  
        pos += 1
```

```
print("result:", result)
```

result: [6, 34, 62]

Нахождение всех позиций заданной подстроки
Использование оператора :=

```
foo = "Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet"  
sub = "ipsum"
```

```
result = []  
pos = 0
```

```
while (pos := foo.find(sub, pos)) != -1:  
    result.append(pos)  
    pos += 1
```

```
print("result:", result)
```

result: [6, 34, 62]

```
foo = "lorem IPSUM dolor SIT amet"
```

```
foo_lower = foo.lower()
```

```
foo_upper = foo.upper()
```

```
foo_cap = foo.capitalize()
```

```
print("foo:      ", foo)
```

```
print("foo_lower: ", foo_lower)
```

```
print("foo_upper: ", foo_upper)
```

```
print("foo_cap:   ", foo_cap)
```

```
foo:      lorem IPSUM dolor SIT amet
```

```
foo_lower: lorem ipsum dolor sit amet
```

```
foo_upper: LOREM IPSUM DOLOR SIT AMET
```

```
foo_cap:   Lorem ipsum dolor sit amet
```

Использование методов `strip()` / `lstrip()` / `rstrip()`

```
foo = "    Lorem ipsum    "
```

```
foo_strip = foo.strip()
```

```
foo_lstrip = foo.lstrip()
```

```
foo_rstrip = foo.rstrip()
```

```
print("foo:      |", foo, "|", sep="")
```

```
print("foo_strip: |", foo_strip, "|", sep="")
```

```
print("foo_lstrip: |", foo_lstrip, "|", sep="")
```

```
print("foo_rstrip: |", foo_rstrip, "|", sep="")
```

```
foo:      |    Lorem ipsum    |
```

```
foo_strip: |Lorem ipsum|
```

```
foo_lstrip: |Lorem ipsum    |
```

```
foo_rstrip: |    Lorem ipsum|
```

Использование методов `removeprefix()` / `removesuffix()`

```
foo = "images/picture.png"
bar = "data/application.exe"

print("foo:", foo)
print("bar:", bar)
print()
print("foo.removeprefix('images/'): ", foo.removeprefix('images/'))
print("foo.removesuffix('.png'): ", foo.removesuffix('.png'))
print()
print("bar.removeprefix('images/'): ", bar.removeprefix('images/'))
print("bar.removesuffix('.png'): ", bar.removesuffix('.png'))
```

```
foo: images/picture.png
bar: data/application.exe
```

```
foo.removeprefix('images/'): picture.png
foo.removesuffix('.png'): images/picture
```

```
bar.removeprefix('images/'): data/application.exe
bar.removesuffix('.png'): data/application.exe
```

`str.replace(old, new[, count])`

Return a copy of the string with all occurrences of substring *old* replaced by *new*. If the optional argument *count* is given, only the first *count* occurrences are replaced.

Использование метода replace()

```
foo = "Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet"  
old = "ipsum"  
new = "*****"
```

```
bar = foo.replace(old, new)  
baz = foo.replace(old, new, 1)
```

```
print("foo: ", foo)  
print("bar: ", bar)  
print("baz: ", baz)
```

```
foo:  Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet  
bar:  Lorem ***** dolor sit amet, Lorem ***** dolor sit amet, Lorem ***** dolor sit amet  
baz:  Lorem ***** dolor sit amet, Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet
```


Использование метода split()

```
foo = "a=10, b=20, c=30"  
bar = foo.split(", ")  
  
print("bar:", bar)
```

bar: ['a=10', 'b=20', 'c=30']

Использование метода join()

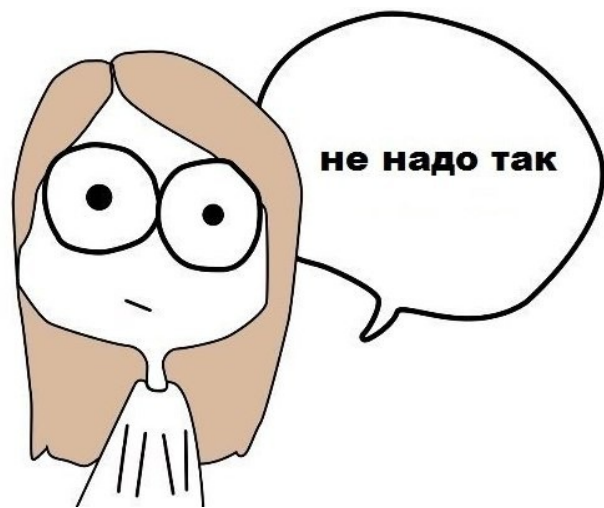
```
bar = ["a=10", "b=20", "c=30"]
```

```
foo = ", ".join(bar)
```

```
print("foo:", foo)
```

```
foo: a=10, b=20, c=30
```

Форматирование строк с использованием оператора "%"



Плохой способ форматирования строк

```
foo = 42  
bar = 12.5  
eggs = "hello"
```

```
spam = "f = " + str(foo) + "; b = " + str(bar) + "; e = " + str(eggs) + ". "  
print("spam:", spam)
```

```
spam: f = 42; b = 12.5; e = hello.
```

Использование оператора "%" для форматирования строк

```
foo = 42
bar = 12.5
eggs = "hello"

spam = "f = %d; b = %f; e = %s." % (foo, bar, eggs)
print("spam:", spam)
```

```
spam: f = 42; b = 12.500000; e = hello.
```

| Символ форматирования | Формат |
|-----------------------|--|
| d или i | Целое число в десятичной форме |
| o | Целое число в восьмеричной форме |
| x или X | Целое число в шестнадцатеричной форме |
| e или E | Число с плавающей точкой в экспоненциальной форме |
| f или F | Число с плавающей точкой |
| g или G | Число с плавающей точкой. Формат подбирается автоматически |
| c | Одиночный символ |
| r | Строка. Для преобразования используется repr() |
| s | Строка. Для преобразования используется str() |
| a | Строка. Для преобразования используется ascii() |
| % | %% заменяется на знак процента |

Различные представления целых чисел

```
foo = 42
```

```
spam = """%%d: %d
```

```
%%o: %o
```

```
%%x: %x
```

```
%%f: %f
```

```
%%e: %e
```

```
%%g: %g
```

```
""" % (foo, foo, foo, foo, foo, foo)
```

```
print(spam)
```

```
%d: 42
```

```
%o: 52
```

```
%x: 2a
```

```
%f: 42.000000
```

```
%e: 4.200000e+01
```

```
%g: 42
```

```
spam = "" | %14f | : | %14f |  
| %-14f | : | %-14f |  
| %14.3f | : | %14.3f |  
| %-14.3f | : | %-14.3f |  
| %+ -14.3f | : | %+ -14.3f |  
| %% -14.3f | : | % -14.3f |  
| %%014.3f | : | %014.3f |  
"" % ((foo,) * 7)
```

```
print(spam)
```

| | |
|------------|----------------|
| %14f : | 42.500000 |
| -14f : | 42.500000 |
| %14.3f : | 42.500 |
| -14.3f : | 42.500 |
| %+-14.3f : | +42.500 |
| -14.3f : | 42.500 |
| %014.3f : | 0000000042.500 |

Использование именованных параметров

```
foo = 42
bar = 12.5
eggs = "hello"

spam = ("f = %(f)d; b = %(b)f; e = %(e)s."
        % {"e": eggs, "f": foo, "b": bar})
print("spam:", spam)
```

```
spam: f = 42; b = 12.500000; e = hello.
```

```
from numpy import linspace
from numpy.random import rand
```

```
freq = linspace(10e9, 12e9, 11)
col1 = rand(len(freq)) * 2 - 1
col2 = rand(len(freq)) * 2 - 1
```

```
for f, c1, c2 in zip(freq, col1, col2):
    print("%-15.3e %- 12.3f %- 12.3f" % (f, c1, c2))
```

| | | |
|-----------|--------|--------|
| 1.000e+10 | -0.942 | -0.425 |
| 1.020e+10 | 0.430 | -0.547 |
| 1.040e+10 | 0.050 | 0.563 |
| 1.060e+10 | 0.025 | -0.402 |
| 1.080e+10 | 0.452 | 0.173 |
| 1.100e+10 | 0.939 | 0.413 |
| 1.120e+10 | -0.850 | -0.859 |
| 1.140e+10 | 0.053 | -0.635 |
| 1.160e+10 | -0.043 | 0.249 |
| 1.180e+10 | 0.954 | 0.713 |
| 1.200e+10 | 0.347 | 0.999 |

Форматирование строк с использованием метода `str.format()`

Использование метода str.format() для форматирования строк

```
foo = 42
bar = 12.5
eggs = "hello"

spam = "f = {}; b = {}; e = {}".format(foo, bar, eggs)
print("spam:", spam)
```

```
spam: f = 42; b = 12.5; e = hello.
```

Использование нумерованных параметров

```
foo = 42
bar = 12.5
eggs = "hello"

spam = ("f = {0}; b = {1}; e = {2}; foo = {0}."
        .format(foo, bar, eggs))

print("spam:", spam)
```

```
spam: f = 42; b = 12.5; e = hello; foo = 42.
```

Использование именованных параметров

```
foo = 42
bar = 12.5
eggs = "hello"

spam = ("f = {foo}; b = {bar}; e = {spam}."
        .format(foo=foo, spam=eggs, bar=bar))

print("spam:", spam)
```

```
spam: f = 42; b = 12.5; e = hello.
```

Различные представления целых чисел

```
foo = 42
```

```
spam = """{{foo}}: {foo}  
{{foo:d}}: {foo:d}  
{{foo:b}}: {foo:b}  
{{foo:o}}: {foo:o}  
{{foo:x}}: {foo:x}  
{{foo:X}}: {foo:X}  
""".format(foo=foo)
```

```
print(spam)
```

```
{foo}: 42  
{foo:d}: 42  
{foo:b}: 101010  
{foo:o}: 52  
{foo:x}: 2a  
{foo:X}: 2A
```

Различные представления чисел с плавающей точкой

```
foo = 0.425
```

```
spam = """{{foo}}: {foo}
```

```
{{foo:g}}: {foo:g}
```

```
{{foo:f}}: {foo:f}
```

```
{{foo:e}}: {foo:e}
```

```
{{foo:E}}: {foo:E}
```

```
{{foo:%}}: {foo:%}
```

```
""".format(foo=foo)
```

```
print(spam)
```

```
{foo}: 0.425
```

```
{foo:g}: 0.425
```

```
{foo:f}: 0.425000
```

```
{foo:e}: 4.250000e-01
```

```
{foo:E}: 4.250000E-01
```

```
{foo:%}: 42.500000%
```



```
foo = 42.5
```

```
spam = ""|{{foo:14f}}| :      |{{foo:14f}}|  
|{{foo:<14f}}| :      |{{foo:<14f}}|  
|{{foo:14.3f}}| :      |{{foo:14.3f}}|  
|{{foo:<14.3f}}| :      |{{foo:<14.3f}}|  
|{{foo:^14.3f}}| :      |{{foo:^14.3f}}|  
|{{foo:<+14.3f}}| :      |{{foo:<+14.3f}}|  
|{{foo:< 14.3f}}| :      |{{foo:< 14.3f}}|  
|{{foo:<014.3f}}| :      |{{foo:<014.3f}}|  
|{{foo:014.3f}}| :      |{{foo:014.3f}}|  
|{{foo:_<14.3f}}| :      |{{foo:_<14.3f}}|  
|{{foo:_>14.3f}}| :      |{{foo:_>14.3f}}|  
|{{foo:_^14.3f}}| :      |{{foo:_^14.3f}}|  
"".format(foo=foo)
```

```
print(spam)
```

| | |
|--------------------|-----------------|
| {{foo:14f}} : | 42.500000 |
| {{foo:<14f}} : | 42.500000 |
| {{foo:14.3f}} : | 42.500 |
| {{foo:<14.3f}} : | 42.500 |
| {{foo:^14.3f}} : | 42.500 |
| {{foo:<+14.3f}} : | +42.500 |
| {{foo:< 14.3f}} : | 42.500 |
| {{foo:<014.3f}} : | 42.500000000000 |
| {{foo:014.3f}} : | 0000000042.500 |
| {{foo:_<14.3f}} : | 42.500_____ |
| {{foo:_>14.3f}} : | ____42.500 |
| {{foo:_^14.3f}} : | ____42.500____ |

Использование индексации

```
foo = [10.1, 20.2, 40.3, 60.4]
```

```
bar = "foo[0] = {0[0]:.3f}; foo[1] = {0[1]:.3f}".format(foo)
```

```
baz = "foo[0] = {foo[0]:.3f}; foo[1] = {foo[1]:.3f}".format(foo=foo)
```

```
print("bar:", bar)
```

```
print("baz:", baz)
```

```
bar: foo[0] = 10.100; foo[1] = 20.200
```

```
baz: foo[0] = 10.100; foo[1] = 20.200
```

Использование индексации

```
foo = [10.1, 20.2, 40.3, 60.4]
```

Ошибка! Отрицательные индексы не работают

```
bar = "foo[-1] = {foo[-1]}".format(foo=foo)
```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: list indices must be integers or slices, not str

```
from numpy import linspace
from numpy.random import rand
```

```
freq = linspace(10e9, 12e9, 11)
col1 = rand(len(freq)) * 2 - 1
col2 = rand(len(freq)) * 2 - 1
```

```
for f, c1, c2 in zip(freq, col1, col2):
    print("{freq:<15.3e}{col1:< 12.3f}{col2:< 12.3f}"
          .format(freq=f, col1=c1, col2=c2))
```

| | | |
|-----------|--------|--------|
| 1.000e+10 | -0.996 | -0.922 |
| 1.020e+10 | -0.736 | -0.269 |
| 1.040e+10 | 0.197 | -0.591 |
| 1.060e+10 | -0.359 | 0.396 |
| 1.080e+10 | 0.223 | 0.648 |
| 1.100e+10 | -0.816 | 0.834 |
| 1.120e+10 | 0.823 | -0.553 |
| 1.140e+10 | 0.637 | 0.948 |
| 1.160e+10 | -0.925 | 0.814 |
| 1.180e+10 | -0.252 | -0.216 |
| 1.200e+10 | -0.513 | 0.437 |

Форматирование строк с использованием f-строк

```
# Использование f-строк  
foo = f"2 x 2 = {2 * 2}"  
  
print("foo:", foo)
```

foo: 2 x 2 = 4

Использование f-строк

```
foo = 42  
bar = 12.5  
eggs = "hello"
```

```
spam = f"foo = {foo}; bar = {bar}; eggs = {eggs}."  
print("spam:", spam)
```

```
spam: foo = 42; bar = 12.5; eggs = hello.
```

Различные представления целых чисел

```
foo = 42
```

```
spam = f"""{{foo}}: {foo}
{{foo:d}}: {foo:d}
{{foo:b}}: {foo:b}
{{foo:o}}: {foo:o}
{{foo:x}}: {foo:x}
{{foo:X}}: {foo:X}
"""
```

```
print(spam)
```

```
{foo}: 42
{foo:d}: 42
{foo:b}: 101010
{foo:o}: 52
{foo:x}: 2a
{foo:X}: 2A
```


Различные представления чисел с плавающей точкой

```
foo = 0.425
```

```
spam = f"""{{foo}}: {foo}
{{foo:g}}: {foo:g}
{{foo:f}}: {foo:f}
{{foo:e}}: {foo:e}
{{foo:E}}: {foo:E}
{{foo:%}}: {foo:%}
"""
```

```
print(spam)
```

```
{foo}: 0.425
{foo:g}: 0.425
{foo:f}: 0.425000
{foo:e}: 4.250000e-01
{foo:E}: 4.250000E-01
{foo:%}: 42.500000%
```

Использование дополнительных параметров форматирования

```
foo = 42.5
```

```
spam = f"""|{{foo:14f}}| :      |{{foo:14f}}|
|{{foo:<14f}}| :      |{{foo:<14f}}|
|{{foo:14.3f}}| :      |{{foo:14.3f}}|
|{{foo:<14.3f}}| :      |{{foo:<14.3f}}|
|{{foo:^14.3f}}| :      |{{foo:^14.3f}}|
|{{foo:<+14.3f}}| :      |{{foo:<+14.3f}}|
|{{foo:< 14.3f}}| :      |{{foo:< 14.3f}}|
|{{foo:<014.3f}}| :      |{{foo:<014.3f}}|
|{{foo:014.3f}}| :      |{{foo:014.3f}}|
|{{foo:_<14.3f}}| :      |{{foo:_<14.3f}}|
|{{foo:_>14.3f}}| :      |{{foo:_>14.3f}}|
|{{foo:_^14.3f}}| :      |{{foo:_^14.3f}}|
"""
```

```
print(spam)
```

| | | | |
|--------------------|--|-----------------|--|
| {{foo:14f}} : | | 42.500000 | |
| {{foo:<14f}} : | | 42.500000 | |
| {{foo:14.3f}} : | | 42.500 | |
| {{foo:<14.3f}} : | | 42.500 | |
| {{foo:^14.3f}} : | | 42.500 | |
| {{foo:<+14.3f}} : | | +42.500 | |
| {{foo:< 14.3f}} : | | 42.500 | |
| {{foo:<014.3f}} : | | 42.500000000000 | |
| {{foo:014.3f}} : | | 0000000042.500 | |
| {{foo:_<14.3f}} : | | 42.500 | |
| {{foo:_>14.3f}} : | | 42.500 | |
| {{foo:_^14.3f}} : | | 42.500 | |

```
# Использование индексации  
# можно использовать отрицательные индексы  
foo = [10.1, 20.2, 40.3, 60.4]  
  
bar = f"foo[0] = {foo[0]:.3f}; foo[-1] = {foo[-1]:.3f}"  
  
print("bar:", bar)
```

```
bar: foo[0] = 10.100; foo[-1] = 60.400
```

Доступ к элементам словаря

```
foo = {"key1": 20.5, "key2": 42, "key3": None}
```

```
bar = f"""foo['key1'] = {foo['key1']}
```

```
foo['key2'] = {foo['key2']}
```

```
foo['key3'] = {foo['key3']}"""
```

```
print(f"bar: {bar}")
```

```
bar: foo['key1'] = 20.5
```

```
foo['key2'] = 42
```

```
foo['key3'] = None
```

Использование выражений в f-строках

```
freq = 10.5e9  
c = 3e8
```

```
foo = f"Частота: {freq / 1e9:.3f} ГГц. Длина волны: {c / freq * 1e3:.3f} мм"  
print(foo)
```

Частота: 10.500 ГГц. Длина волны: 28.571 мм

Вызов функций в f-строках

```
from math import log10  
gain = 10e3
```

```
foo = f"Коэффициент усиления: {gain:.2e} или {10 * log10(gain):.3f} дБ"  
print(foo)
```

Коэффициент усиления: 1.00e+04 или 40.000 дБ

Использование методов класса

```
hello = "Hello"
```

```
foo = f"hello.lower(): {hello.lower()}\nhello.upper(): {hello.upper()}"  
print(foo)
```

```
hello.lower(): hello
```

```
hello.upper(): HELLO
```

Использование методов класса

```
foo = ["hello", "world", "spam"]
```

```
bar = f"foo: {' '.join(foo)}"
```

```
print(bar)
```

```
foo: hello, world, spam
```


Использование методов класса

```
foo = [10, 20, 30]
```

```
foo_str = [str(item) for item in foo]
```

```
bar = f"foo: {' '.join(foo_str)}"
```

```
print(bar)
```

```
foo: 10, 20, 30
```

```
foo = [10, 20, 30]
```

```
bar = f"foo: {' '.join([str(item) for item in foo])}"  
print(bar)
```

```
foo: 10, 20, 30
```

```
# Использование режима отладки
```

```
foo = 10
```

```
bar = "hello"
```

```
baz = 20.5
```

```
eggs = f"{foo}\n{bar}\n{baz:10.3f}"
```

```
spam = f"{foo=}\n{bar=}\n{baz=:10.3f}"
```

```
print(eggs)
```

```
print()
```

```
print(spam)
```

```
10
```

```
hello
```

```
20.500
```

```
foo=10
```

```
bar='hello'
```

```
baz= 20.500
```

```
from numpy import linspace
from numpy.random import rand
```

```
freq = linspace(10e9, 12e9, 11)
col1 = rand(len(freq)) * 2 - 1
col2 = rand(len(freq)) * 2 - 1
```

```
for f, c1, c2 in zip(freq, col1, col2):
    print(f"{f:<15.3e}{c1:< 12.3f}{c2:< 12.3f}")
```

| | | |
|-----------|--------|--------|
| 1.000e+10 | -0.445 | -0.671 |
| 1.020e+10 | -0.774 | -0.909 |
| 1.040e+10 | 0.160 | 0.971 |
| 1.060e+10 | -0.646 | 0.558 |
| 1.080e+10 | 0.131 | -0.070 |
| 1.100e+10 | 0.176 | 0.415 |
| 1.120e+10 | 0.168 | -0.433 |
| 1.140e+10 | 0.648 | -0.940 |
| 1.160e+10 | 0.032 | -0.819 |
| 1.180e+10 | 0.331 | -0.635 |
| 1.200e+10 | -0.116 | -0.617 |

```
from numpy import linspace
from numpy.random import rand
```

```
width = 17
prec = 4
```

```
freq = linspace(10e9, 12e9, 11)
col1 = rand(len(freq)) * 2 - 1
col2 = rand(len(freq)) * 2 - 1
```

```
for f, c1, c2 in zip(freq, col1, col2):
    print(f"{f:<{width + 3}.{prec}e}{c1:< {width}.{prec}f}{c2: .{prec}f}")
```

| | | |
|------------|---------|---------|
| 1.0000e+10 | 0.3853 | 0.1419 |
| 1.0200e+10 | -0.1974 | -0.9864 |
| 1.0400e+10 | -0.3081 | 0.9513 |
| 1.0600e+10 | 0.3721 | -0.7087 |
| 1.0800e+10 | -0.4048 | -0.0484 |
| 1.1000e+10 | 0.8782 | -0.4207 |
| 1.1200e+10 | -0.4168 | 0.5628 |
| 1.1400e+10 | -0.5999 | -0.5726 |
| 1.1600e+10 | -0.3414 | 0.6707 |
| 1.1800e+10 | 0.8510 | 0.3094 |
| 1.2000e+10 | -0.3867 | -0.6827 |