**Форматирование через f-строки**

#### Как добавить отступы к F-строке, выравнивание.

>>> greetings = "hello"

# отступы справа и слева

>>> f"{greetings:>10}"

# ' hello'

>>> f"{greetings:<10}"

# 'hello '

# выравнивание чисел

>>> a = "1"

>>> b = "21"

>>> c = "321"

>>> d = "4321"

>>> print("\n".join((f"{a:>10}", f"{b:>10}", f"{c:>10}", f"{d:>10}")))

# 1

# 21

# 321

# 4321

# выравнивание по середине

>>> hello = "world"

>>> f"{hello:^11}"

# ' world '

**Форматирование числа с помощью запятых в качестве десятичного разделителя.**

>>> big\_num = 1234567890

>>> f"{big\_num:,}"

# '1,234,567,890'

>>> num = 2343552.6516251625

>>> f"{num:,.3f}"

# '2,343,552.652'

**Использование спецификатора формата даты.**

>>> import datetime

>>> today = datetime.datetime(year=2021, month=1, day=27)

>>> f`{today:%B %d, %Y}`

'January 27, 2021'

>>> now = datetime.datetime.now()

>>> ten\_days\_ago = now - datetime.timedelta(days=10)

>>> f'{ten\_days\_ago:%Y-%m-%d %H:%M:%S}'

# '2020-10-13 20:24:17'

>>> f'{now:%Y-%m-%d %H:%M:%S}'

# '2020-10-23 20:24:17'

<https://peps.python.org/pep-0498/>

# объявление переменных

print(f"{2 \* 2}") # вывод: 4

Также в {} можно вызвать функцию. Определим функцию greet() и вызовем ее в f-строке.

def greet(name):

return "Привет, " + name

# вызов функции с использованием f-строки

name = "Олег"

print(f"{greet(name)}")

Привет, Олег

Также можно вызвать определенные заранее методы.

# метод title, делает первую букву каждого слова заглавной

string = "pythonru это блог."

print(f"{string.title()}")

Pythonru Это Блог.

## **Вычисление выражений и вызов функций**

num = 7

print(f'{num} в квадрате равно {num \* num}')

>>> 7 в квадрате равно 49

Вызовем строковую функцию .upper(), превращающую все буквы строки в заглавные:

name = "спарта"

print(f"Это {name.upper()}!!!")

>>> Это СПАРТА!!!

## **Вывод кавычек и фигурных скобок**

print(f"Привет, \"{name}\", я 'Саша'")

>>> Привет, "Коля", я 'Саша'

## **Выравнивание и знаки после запятой**

В форматированных строках можно указать количество десятичных знаков после запятой, количество символов, выделенных для вывода значения, а также выровнять значение по левому или правому краю. Общий синтаксис выглядит так:

f'{value:{width}.{precision}}'

Значение, двоеточие, затем ширина строки в фигурных скобках, точка, требуемая точность в фигурных скобках.

Для начала давайте посмотрим, как задать точность вывода значения.

pi = 3.14159265

print(f'{pi:.2f}')

>>> 3.

Если в параметре precision указать 2f, как здесь, то значение выводится с двумя знаками после запятой. Буква f в данном случае означает fractional part, то есть дробную часть числа. Если оставить просто 2, то значение целиком — и целая, и дробная часть — будет занимать два знака (точка не считается).

print(f'{pi:.2}')

>>> 3.1

Параметр width, стоящий в фигурных скобках после двоеточия и перед точкой, определяет ширину строки, выделенной под значение, и выравнивание в ней.

print(f'{5:<5}₽') # ширина 5 символов, выравниванием влево <

>>> 5 ₽

print(f'{5:>5}₽') # ширина 5 символов, выравниванием вправо >

>>> 5₽

Если не указать направление выравнивания (< или >), то строка по умолчанию будет выравниваться по левому краю (<).

<https://skillbox.ru/media/base/formatirovannye-stroki-v-python-primery-ispolzovaniya/>