

*School of
Computer
Science*

ФОРМАТИРОВАНИЕ СТРОК

ПРОГРАММИРОВАНИЕ НА PYTHON
Лекции для ИТ-школы



ФОРМАТИРОВАНИЕ СТРОК

Строки в Python – упорядоченная последовательность символов.

К какому типу данных относятся строки – изменяемому или неизменяемому?

Форматирование строк – подстановка какого-либо шаблона в определённые позиции текста.



СПОСОБЫ ФОРМАТИРОВАНИЯ СТРОК

В Python существует 5 способов форматирования строк:

1. Конкатенация
2. %-форматирование
3. Шаблонные строки (Template)
4. str.format()
5. f-строки



КОНКАТЕНАЦИЯ

- Грубый способ форматирования, в котором строки склеиваются с помощью операции конкатенации.
- К моменту конкатенации строки уже должны быть в нужном формате.

```
>>> name = "Александра"  
>>> age = 22  
>>> print("Меня зовут " + name + ". Мне  
" + str(age) + " года.")  
>>> Меня зовут Александра. Мне 22 года.
```



%-ФОРМАТИРОВАНИЕ

- Значения в строку передаются через списки или с помощью словаря (в этом случае значения размещаются в соответствии с именами).
- Пришёл из языка Си как аналог функции printf.
- Аргумент для подстановки – кортеж со значениями.

```
>>> name = "Александра"
>>> age = 22
>>> print("Меня зовут %s. Мне %d года." % (name,
age) )
>>> Меня зовут Александра. Мне 22 года.
>>> print("Меня зовут %(name)s. Мне %(age)d лет."
% {"name": name, "age": age})
>>> Меня зовут Александра. Мне 22 года.
```



СПЕЦИФИКАТОРЫ ПРЕОБРАЗОВАНИЯ

Формат	Для чего используется
%d	Десятичное число
%x	Число в шестнадцатеричной системе счисления (буквы в нижнем регистре)
%f	Число с плавающей точкой (обычный формат)
%e	Число с плавающей точкой с экспонентой (экспонента в нижнем регистре)
%s	Строка (как обычно воспринимается пользователем)

Подробнее см.

<https://pythonworld.ru/osnovy/formatirovaniye-strok-operator.html>



TEMPLATE STRINGS

- Поддерживает передачу значений по имени и использует \$-синтаксис.
- Появился в Python 2.4, как замена %-форматированию.
- Не позволяют форматировать спецификаторы.

```
>>> from string import Template  
>>> name = "Александра"  
>>> age = 22  
>>> s = Template('Меня зовут $name. Мне $age лет.')  
>>> print(s.substitute(name=name, age=age))  
>>> Меня зовут Александра. Мне 22 года.
```



STR.FORMAT()

- Возвращает отформатированную версию строки, заменяя идентификаторы в фигурных скобках. Идентификаторы могут быть позиционными, числовыми индексами, ключами словарей, именами переменных.
- Появился в Python 3 в качестве замены %-форматированию.

```
>>> name = "Александра"
>>> age = 22
>>> print("Меня зовут {}. Мне {} лет.".format(name,
age))
>>> Меня зовут Александра. Мне 22 года.
>>> print("Меня зовут {name} Мне {age}
лет.".format(age=age, name=name))
>>> Меня зовут Александра. Мне 22 года.
```



- Синтаксис

поле замены ::= "{" [имя поля] ["!" преобразование] [":" спецификация] "}"

имя поля ::= arg_name ("." имя атрибута | "[" индекс "]")

преобразование ::= "r" (внутреннее представление) | "s" (человеческое представление)

спецификация ::= см далее

```
>>> "Units destroyed: {players[0]}".format(players = [1, 2, 3])
'Units destroyed: 1'
>>> "Units destroyed: {players[0]!r}").format(players = ['1', '2', '3'])
"Units destroyed: '1'"
```



- Спецификация

спецификация ::= [[fill]align][sign][width][.precision][type]

заполнитель ::= символ кроме '{' или '}'

выравнивание ::= "<" | ">" | "=" | "^"

знак ::= "+" | "-" | " "

ширина ::= integer

точность ::= integer

тип ::= "b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" | "o" | "s" | "x" | "X" | "%"

Подробнее см.

<https://pythonworld.ru/osnovy/formatirovaniye-strok-metod-format.html>



- Появился в Python 3.6.
- f-строки берут значения переменных, которые есть в текущей области видимости, и подставляют их в строку. В самой строке нужно указать имя этой переменной в фигурных скобках, а перед строкой поставить префикс f.

```
>>> name = "Александра"  
>>> age = 22  
>>> print("Меня зовут {name}. Мне {age}  
лет")  
>>> Меня зовут Александра. Мне 22 года.
```



Строчные литералы поддерживают:

- существующий синтаксис формата строк метода str.format():

```
>>> print(f"Значение числа pi: {pi:.2f}")
```

```
>>> Значение числа pi: 3.14
```

- базовые арифметические операции прямо в строках:

```
>>> x = 10
```

```
>>> y = 5
```

```
>>> print(f"{x} {x} {y} / 2 = {x * y / 2}")
```

```
>>> 10 10 5 / 2 = 25.0
```



Строчные литералы поддерживают:

- обращение к значениям списков по индексу :

```
>>> planets = ["Меркурий", "Венера", "Земля", "Марс"]  
>>> print(f"Мы живём на планете {planets[2]}")  
>>> Мы живём на планете Земля
```

- и элементам словаря по имени:

```
>>> planet = {"name": "Земля", "radius": 6378000}  
>>> print(f"Планета {planet['name']}. Радиус  
{planet['radius']/1000} км.")  
>>> Планета Земля. Радиус 6378.0 км.
```



Строчные литералы поддерживают:

- вызов методов объекта:

```
>>> name = "Александра"  
>>> print(f"Имя: {name.upper()}"")  
>>> Имя: АЛЕКСАНДРА
```

- Вызов функций:

```
>>> print(f"13 / 3 = {round(13/3)}")  
>>> 13 / 3 = 4
```



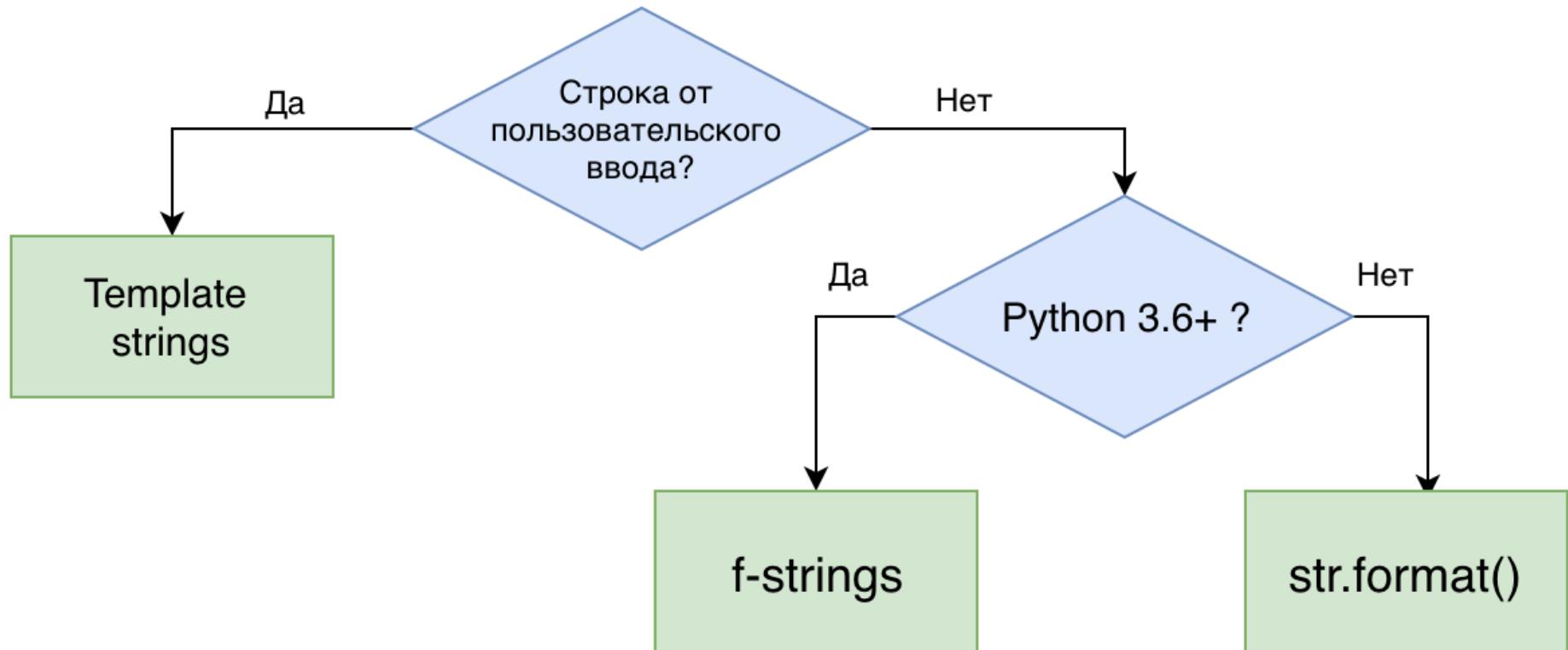
МНОГОСТРОЧНЫЕ F-STRINGS

Необходимо разместить префикс f перед каждой строкой

```
>>> name = "Eric"  
  
>>> profession = "comedian"  
  
>>> message = (  
    f"Hi {name}. "  
    f"You are a {profession}. ")  
  
>>> print(message)  
  
>>> Hi Eric. You are a comedian.
```



ЧТО ИСПОЛЬЗОВАТЬ?





ПРАКТИЧЕСКОЕ ЗАДАНИЕ

Создать заготовку объявления, внутри которого есть ряд изменяющихся параметров: имена людей, названия и время событий, стоимость билета:

Уважаемый (ая), <имя>!

Приглашаем Вас на <НАЗВАНИЕ СОБЫТИЯ>.

Дата и время события: <дата и время в формате dd.mm.yy hh:mm>.

Стоимость билета: <стоимость с точностью до двух знаков после запятой> руб.

- Вывести пять объявлений.
 - Для параметров создать списки, подстановку элементов в объявление осуществлять в цикле.
 - В дате и времени две позиции выбрать случайным образом.
 - Название события привести к верхнему регистру.
-
- Выполнить задание с помощью функции **format** и с помощью **f-строк**.

Шаблон для выполнения – **advert.py**