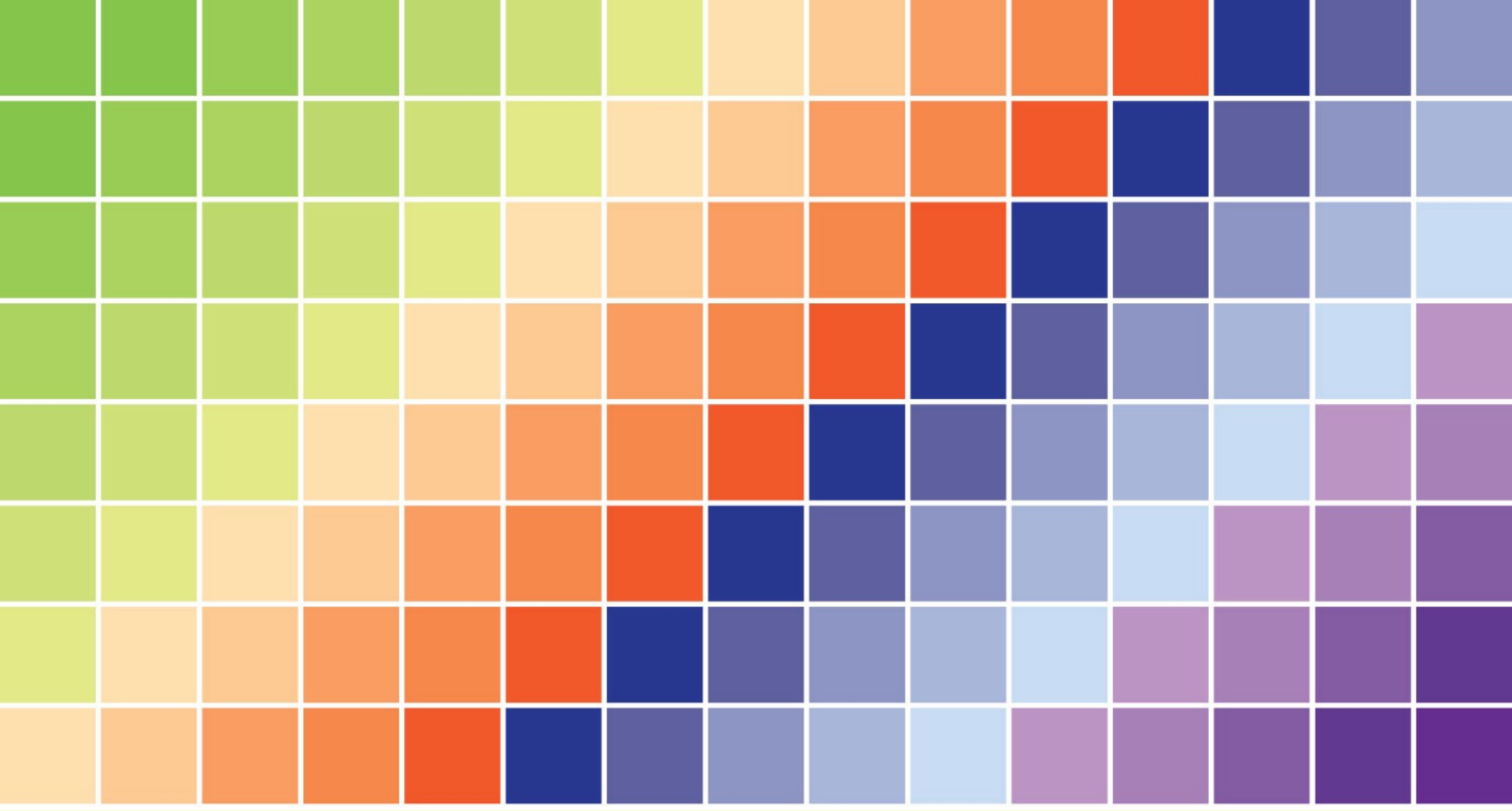


School of Computer Science

ФОРМАТИРОВАНИЕ СТРОК

ПРОГРАММИРОВАНИЕ НА PYTHON

Лекции для IT-школы

ФОРМАТИРОВАНИЕ СТРОК

Строки в Python – упорядоченная последовательность символов

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

При этом генерируется новая строка, содержащая все подстановки

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 = "Сергей"
>>> age = 46
>>> print("Меня зовут " + name + ". Мне
" + str(age) + " лет.")
>>> Меня зовут Сергей. Мне 46 лет.
```

%-ФОРМАТИРОВАНИЕ

- Этот способ пришёл из языка Си как аналог функции printf()

- Значения в строку передаются:
  a) через перечисление значений (кортеж) или
  b) с помощью словаря – в этом случае значения размещаются в соответствии с именами

```
>>> name = "Сергей"
>>> age = 46
>>> print("Меня зовут %s. Мне %d лет." % (name,
age))
>>> Меня зовут Сергей. Мне 46 лет.
>>> print("Меня зовут %(name)s. Мне %(age)d лет."
% {"name": name, "age": age})
>>> Меня зовут Сергей. Мне 46 лет.
```

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/formatirovanie-strok-operator.html

TEMPLATE STRINGS

- Поддерживает передачу значений по имени и использует $-синтаксис
- Не поддерживают спецификаторы формата
- Простой, но безопасный выбор для строк, введенных пользователем

```
>>> from string import Template
>>> name = "Сергей"
>>> age = 46
>>> templ_str = Template('Я $name. Мне $age лет.')
>>> print(templ_str.substitute(name=name, age=age))
>>> Я Сергей. Мне 46 лет.
```

STR.FORMAT(). Определение

- 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 = "Сергей"
>>> age = 46
>>> print("Я {}. Мне {} лет.".format(name, age))
>>> Я Сергей. Мне 46 лет.
>>>
>>> print("Я {name} Мне {age} лет.".format(age=age,
        name=name))
>>> Я Сергей. Мне 46 лет.
```

STR.FORMAT(). Синтаксис

- 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'"
```

STR.FORMAT(). Спецификация формата

- 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/formatirovanie-strok-metod-format.html

STR.FORMAT(). Спецификация формата в общем виде

:	*заполнитель*	*выравнивание*	*знак*	*#*	*0*	*ширина*	*.точность*	*тип*
	Любой символ, кроме }	< по левому краю; > по правому краю; ^ по центру = заполнять нулями пространство между знаком числа и первой значащей цифрой	+ всегда выводить знак; – знак выводится, только когда необходимо; “ ” пробел или знак «–»	префикс для целых чисел 0b, 0o, or 0x	Дополнение чисел нулями	Минимальная ширина поля	Максимальная ширина поля для строк; количество знаков после запятой для чисел с плавающей точкой	int b, c, d, n, o, x, X; floats e, E, f, g, G, n, %

STR.FORMAT(). Пример форматирования строк

```
>>> s = "The sword of truth"
>>> "{0}".format(s)     # форматирование по умолчанию
'The sword of truth'
>>> "{0:25}".format(s)  # минимальная ширина поля вывода 25
'The sword of truth '
>>> "{0:>25}".format(s) # выравнивание по правому краю, минимальная ширина 25
' The sword of truth'
>>> "{0:^25}".format(s) # выравнивание по центру, минимальная ширина 25
' The sword of truth '
>>> "{0:-^25}".format(s) # - заполнитель, по центру, минимальная ширина 25
'---The sword of truth----'
>>> "{0:.<25}".format(s) # . заполнитель, по левому краю, минимальная ширина 25
'The sword of truth.......'
>>> "{0:.10}".format(s) # максимальная ширина поля вывода 10
'The sword '
```

STR.FORMAT(). Пример форматирования чисел с дополнением нулями

Дополнение нулями слева можно реализовать двумя способами:

№1:

```
>>> "{0:0=12}".format(8749203) # 0 - символ-заполнитель, минимальная ширина 12
'000008749203'
>>> "{0:0=12}".format(-8749203)# 0 - символ-заполнитель, минимальная ширина 12
'-00008749203'
```

№2:

```
>>> "{0:012}".format(8749203)  # дополнение 0 и минимальная ширина 12
'000008749203'
>>> "{0:012}".format(-8749203) # дополнение 0 и минимальная ширина 12
'-00008749203'
```

F-STRINGS. Определение

- 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 = "Сергей"
>>> age = 46
>>> print(f"Я {name}. Мне {age} лет")
>>> Я Сергей. Мне 46 лет.
```

F-STRINGS. Возможности

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 x 5 / 2 = 25.0
```

F-STRINGS. Подстановки из последовательностей

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.
```

F-STRINGS. Вызовы методов и функций

Строчные литералы поддерживают:

- вызов методов объекта:

```
>>> name = "Сергей"
>>> print(f"Имя: {name.upper()}")
>>> Имя: СЕРГЕЙ
```

- Вызов функций:

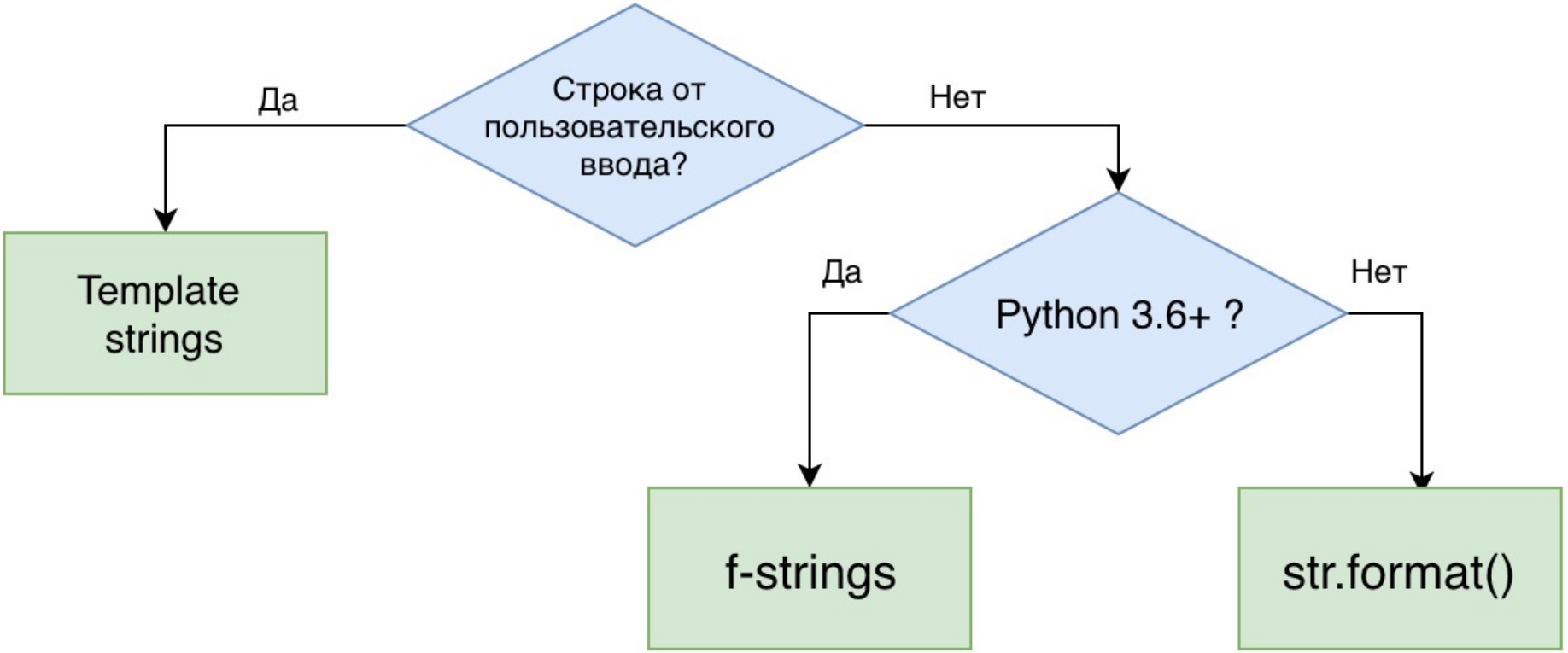
```
>>> print(f"13 / 3 = {round(13/3)}")
>>> 13 / 3 = 4
```

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.yyyy hh24:mi>.
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 метода **format** и с помощью **f-строк**.

Шаблон для выполнения – **advert.py**