|  |
| --- |
| **Форматирование строк. Заполнители. (https://pythonworld.ru/osnovy/formatirovanie-strok-metod-format.html)** |
| **Стандартный метод форматирования**  **“. . .%s . . .%d” % (string, double)** | |
| **name="Dmitriy"**  **vozrast= 31**  **email= "dmitriyx@bk.ru"** | |
| **text = "Privet, menya zovut %s, mne %s let. Moy E-mail dlya svyazi:%s" %(name,vozrast,email)** | |
| **“%(n)d %(x)s” % {“n”:1, “x”:”spam”} -**В данном случае (n) и (x) в строке формата ссылаются на ключи в словаре в пра­вой части выражения и служат для извлечения соответствующих им значе­ний. Этот прием часто используется в программах, создающих код разметки HTML или XML | |
| **Print(name % vozrast)**  **Print(name % email)** | |
| **reply = “Greetings...Hello %(name)s! Your age squared is %(age)s»** *# Шаблон с замещаемыми спецификаторами формата*  **values = {‘name’: ‘Bob’, ‘age’: 40}**  *# Подготовка фактических значений*  **>>> print reply % values**  *# Подстановка значений* | |
| **Общая структура целей преобразований выглядит следующим образом:**  ***% [ (имя ключа) ] [флаги] [ширина] [. точность] код типа***  Между % и символом кода типа можно делать любое из описанных ниже действий:  • указывать *имя ключа* для индексации словаря, используемого в правой части выражения;  • перечислять *флаги,* которые указывают аспекты вроде выравнивания влево (-), знак числа (+), пусто перед положительными числами и — для отрицательных чисел (пробел) и дополнение нулями (0);  • задавать общую минимальную *ширину* поля для подставляемого текста;  • устанавливать количество цифр, отображаемых после десятичной точки *(точность),* для чисел с плавающей точкой.  х = 1234  res = 'integers: . . .%d. . .%-6d. . .%06d’ % (x, x, x)  **'integers: ... 1234 ... 1234 ...001234'**  ***'%е | %f I %g' % (x, x, x)***  ***'1.234568e+00 I 1.234568 | 1.23457’***  **'%-6.2f | %05.2f I %+06.1f' % (x, x, x)**  **’1.23 | 01.23 | +001.2’**  **’%s' % x, str(x)**  **('1.23456789’, ’1.23456789’)** | |
| **Форматирование вызовом метода строки**  Фигурные скобки внутри строки, на которой метод вызывается, обозначают цели подстановки и аргументы, вставляемые по позиции (например, {1}), по ключевому слову (скажем, {food}) или по относительной позиции в Python 2.7, 3.1 и более поздних версиях ({}).  ***# По позициям***  **template = ’{0}, {1} and {2}’**  **template.format('spam', ’ham’, ’eggs')**  **’spam, ham and eggs'**  ***# По ключевым словам***  **template = ’{motto}f {pork} and {food}'**  **template.format(motto=’spam', pork='ham', food='eggs')**  **'spam, ham and eggs'**  ***# По позициям и ключевым словам***  **template = ' {motto}, {0} and {food} '**  **template.format('ham', motto='spam', food='eggs')**  **'spam, ham and eggs'**  ***По относительным позициям***  **template = ’ {}, {} and {} ' *#***  **template.format('spam', 'ham', 'eggs')**  **'spam, ham and eggs'** | |
| **text1 = "Privet, menya zovut {1}, mne {2} let. Moy E-mail dlya svyazi:{0}".format(name,vozrast,email)** | |
| **text2 = "Privet, menya zovut {n}, mne {age:15.5f} let. Moy E-mail dlya svyazi:{mail}".format(n=name,age=vozrast,mail=email)** | |
| **text3 = f"Privet, menya zovut {name}, mne {vozrast} let. Moy E-mail dlya svyazi:{email}"** | |
| **import sys**  ’Му **{l[kind]}** runs **{0.platform}**’.format(**sys**, {**’kind’**: ’laptop’})  ***’My laptop runs Win32’***  ’My **{map [kind]}** runs **{sys.platform}**'.format(**sys=sys**, map={**'kind’**: ’laptop’})  ***'My laptop runs Win32'*** | |
| *{имя\_\_поля компонент ! флаг\_преобразования :спецификатор\_формата}*  *• имя\_поля —* необязательное число или ключевое слово, идентифицирующее аргумент, которое может быть опущено для применения относительной нумерации аргументов  *• компонент —* строка из нуля или большего числа ссылок .имя либо [индекс], используемых для извлечения атрибутов и индексированных значений аргумента, которая может быть опущена для применения всего значения аргумента.  *• флаг\_преобразования,* если присутствует, начинается с символа !, за которым следует г, s или а для вызова на значении встроенных функций герг, str или ascii соответственно.  *• спецификатор\_формата,* если присутствует, начинается с символа : за которым следует текст, указывающий, каким образом значение должно быть представлено, в том числе такие детали, как ширина поля, выравнивание, дополнение, десятичная точность и т.д., и в конце необязательный код типа данных.  Часть *спецификатор-формат а* после символа двоеточия имеет собственный расширенный формат и формально описывается так (квадратные скобки обозначают необязательные компоненты и записываться не должны):  *[[заполнение] выравнивание]* [знак] [#] [0] *[ширина]* [,] [.точность] *[код\_типа]*  Здесь *заполнение* может быть любым заполняющим символом кроме { или }; *выравнивание* — <,>,= или Л соответственно для выравнивания влево, выравнивания вправо, дополнения символом знака или выравнивания по центру; *знак* может быть *+ , - или* пробелом; а вариант , (запятая) требует запятой для разделителя тысяч, начиная с версии Python 2.7 и 3.1. Компоненты *ширина и точность* почти такие же, как в выражении %; кроме того, *спецификатор\_\_формата* может также содержать вложенные строки формата {} только с именами полей, чтобы получать значения из списка аргументов динамически (что очень похоже на \* в выражениях форматирования). | |
| **’ {0:10} = {1:10} ’ .format(’spam’ , 123.4567)**  {0:10} означает первый позиционный аргумент в поле шириной 10 символов,  **’spam = 123.4567'**  **’{0:>10} = {1:<10}'.format(’spam’, 123.4567)**  {1:<10} — второй позиционный аргумент, выровненный влево в поле шириной 10 символов,  **’ spam = 123.4567 ’**  **'{0.platform:>10} - {1[kind]:<10}’ .format(sys, diet(kind= ’laptop'))**  {0.platform:>10} — атрибут platform первого аргумента, выровненный вправо в поле шириной 10 символов  **’ Win32 = laptop** | |
| **’{0:е}, {1:.3е}, {2:g}’.format(3.14159, 3.14159, 3.14159)**  **’3.141590е+00, 3.142е+00, 3.14159’**  **’{0:f}, {l:.2f}, {2:06.2f}’.format(3.14159, 3.14159, 3.14159)**  **'3.141590, 3.14, 003.14’** | |
| ***# Шестнадцатеричный, восьмеричный, двоичный***  **’{0:Х}, {1:о}, {2:Ь}’.format(255, 255, 255)**  **’ FF, 377, 11111111'**  ***# Другие преобразования в/из двоичного***  **bin(255), int (' 11111111 ’ , 2), 0Ы1111111**  **( ’0Ы1111111’ , 255, 255)**  ***Другие преобразования в/из шестнадцатеричного***  **hex (255), int(’FF’, 16), OxFF #**  **(’Oxff’, 255, 255)**  ***# Другие преобразования в/из восьмеричного, Python З.Х***  **oct(255), int('377’, 8), 0о377**  **(’ 0о377 ', 255, 255)** | |
|  | |
|  | |
| **Таблица 7.4. Коды типов в форматировании строк**  **Код Описание**  s Строка (или строка str (X) любого объекта)  г То же, что и s, но применяется герг, а не str  с Символ (int или str)  d Десятичное число (целое число с основанием 10)  i Целое число  и То же, что и d (устарел: больше не является беззнаковым)  о Восьмеричное целое число (с основанием 8)  х Шестнадцатеричное целое число (с основанием 16)  х То же, что и х, но в верхнем регистре  е Число с плавающей точкой со степенью, в нижнем регистре  Е То же, что и е, но в верхнем регистре  f Десятичное число с плавающей точкой  F То же, что и f, но в верхнем регистре  g Число с плавающей точкой е или f  G Число с плавающей точкой Е или F  % Литерал % (записывается как %%) | |