Лекція 8

Форматування рядків

Форматування замість операцій над рядками

Замість з'єднання рядків за допомогою оператора + краще використовувати `форматування`.
Ця операція дозволяє:
- з'єднувати рядок з будь-яким іншим типом даних
- виконується швидше конкатенації.

ПРИМІТКА

У наступних версіях Python оператор форматування % може бути вилучений. Замість цього оператора в новому коді слід використовувати метод `format()`.

Синтаксис форматування

<Рядок спеціального формату> % <Значення>

Усередині параметра <Рядок спеціального формату> можуть бути зазначені специфікатори, що мають наступний синтаксис:

%[(<Ключ>)] [<Прапор>] [<Ширина>] [.<Точність>]<Тип перетворення>

1. Кількість специфікаторів усередині рядка повинна дорівнювати кількості елементів у параметрі <Значення>.
2. Один специфікатор ⇒ параметр <Значення> може містити одне значення
3. Кілька специфікаторів ⇒ перераховуємо значення через кому усередині круглих дужок, створюючи тим самим кортеж.

Приклад 1. Як задавати поле <значення>

```
>>> "%s" % 10                  # один елемент
'10'
>>> "%s-%s-%s"%(10,20,30)# декілька елементів
'10-20-30'
```

Параметри всередині специфікатора

Рядок спеціального формату має вигляд:

%[(<Ключ>)][<Прапор>][<Ширина>][.<Точність>]<Тип перетворення>

(<Ключ>) – ключ словника. Якщо заданий ключ, то в параметрі <Значення> необхідно вказати словник, а не кортеж.

Приклад 2. Використання ключа словника

```
>>>"%(name)s-%(year)s"%{"year":1978,"name":"Nik"}
'Nik - 1978'
```

"%(name)s-%(year)s" – рядок спеціального формату
  %(name)s – специфікатор для значення "name":"Nik"
  %(year)s – специфікатор для значення "year":1978
  % – ознака початку специфікатора
s – тип перетворення відповідного значення
(name), (year)- ключі

<Прапор> прапор перетворення

%[(<Ключ>)] [<Прапор>] [<Ширина>] [.<Точність>]<Тип перетворення>

Може містити наступні значення: #o, #x, #X, #.F
- для вісімкових значень додає на початок комбінацію символів 0o, для шістнадцяткових – 0x (якщо використовується тип x) або 0X (якщо використовується тип X), для дійсних чисел: #.F,#F

Приклад 3

```
>>> print("%#o %#o %#o" % (0o77, 10, 12))
0o77 0o12 0o14
>>> print("%#x %#x %#x" % (0xff, 10, 12))
0xff 0xa 0xc
>>>print("%#X %#X %#X" % (0xff, 10, 12))
0XFF 0XA 0XC
>>> print("%#.F %#.F" % (300, 300))
300. 300.
>>> print("%#F    %#F" % (300.12345678, 300.123))
300.123457    300.123000
```

Застосування значень прапора 0 та -

0 - задає наявність провідних нулів для числового значення:

Приклад 4

```
>>> "'%d'" % (3)# 5 - ширина поля
>>> "'%d' - '%05d'" % (3,3)# 5 - ширина поля
"'3' - '00003'"
```

- – задає вирівнювання по лівій границі області. За замовчуванням використовується вирівнювання по правій границі. Якщо прапор зазначений одночасно з прапором 0, то дія прапора 0 буде скасована.

Приклад 5

```
>>> "'%5d' - '%-5d'" % (3, 3)#5- ширина поля
"'    3' - '3    '"
>>>"'%05d' - '%-05d'" % (3, 3)
"'00003' - '3    '"
```

Застосування значень прапора «пробіл» і +

пробіл – вставляє пробіл перед додатним числом. Перед від'ємним числом буде стояти мінус.

Приклад 6

```
>>>"'% d' - '% d'" % (-3, 3)
"'-3' - ' 3'"
```

+ – задає обов'язковий вивід знака як для від'ємних, так і для додатних чисел. Якщо прапор + зазначений одночасно з прапором пробіл, то дія прапора пробіл буде скасована.

Приклад 7

```
>>> "'%+d' - '%+d'" % (-3, 3)
"'-3' - '+3'"
```

<Ширина> мінімальна ширина поля

%[(<Ключ>)][<Прапор>][<Ширина>][.<Точність>]<Тип перетворення>

Якщо рядок не вміщується в зазначену ширину, то значення ігнорується, і рядок виводиться повністю:

Приклад 8

```
>>> "'%10d' - '%-10d'" % (3, 3)
"'         3' - '3         '"
>>> "'%3s''%10s'" % ("string", "string")
"'string''    string'"
```

Замість значення можна вказати символ "*". У цьому випадку значення слід задати всередині кортежу:

Приклад 9

```
>>> "'%*s''%10s'" % (10, "string", "str")
"'    string''       str'"
```

<Точність>

кількість знаків після точки для дійсних чисел

%[(<Ключ>)] [<Прапор>] [<Ширина>] [.<Точність>]<Тип перетворення>

Перед цим параметром повинна стояти крапка.
Без крапки одержуємо 6 знаків після крапки.

Приклад 10.

```
>>> import math
>>>"%s %f %.2f" % (math.pi,math.pi,math.pi)
'3.141592653589793 3.141593 3.14'
```

Замість значення можна вказати символ «*». У цьому випадку значення слід задати всередині кортежу:

Приклад 11.

```
>>> "'%*.*f'" % (8, 5, math.pi)
"' 3.14159'"
>>>"'%.*f'" % (3, math.pi)
"' 3.142'"
```

<Тип перетворення> задає тип перетворення

%[(<Ключ>)][<Прапор>][<Ширина>][.<Точність>]<Тип перетворення>

Параметр є обов'язковим!

У параметрі <Тип перетворення> можуть бути зазначені наступні символи: **s, r, a, c, d, i, o, x, X, f, F, e, E, g, G**

Кожний із символів указує на виконання відповідних дій по перетворенню:

s – використовує функцію `str()`;

r – використовує функцію `repr()`;

a – використовує функцію `ascii()`;

c – перетворює код символу в символ;

d і **i** – повертають цілу частину числа;

o – вісімкове значення;

x, X – шістнадцяткове значення;

f, F – дійсне число в десятковій формі;

e, E – дійсне число в експонентній формі;

g, G – Еквівалентно `f, F` і `e, E`

Символи s і r у параметрі <Тип перетворення>

s – перетворює будь-який об'єкт у рядок за допомогою функції `str()`:

Приклад 12

```
>>> print("%s" % ("Звичайний рядок"))
Звичайний рядок
>>>print("%s %s %s" % (10, 10.52, [1, 2, 3]))
10 10.52 [1, 2, 3]
```

r – перетворює будь-який об'єкт у рядок за допомогою функції `repr()`.

Приклад13

```
>>> print ( "%r" % ("Звичайний рядок"))
'Звичайний рядок'
```

Символи a і c у параметрі <Тип перетворення>

a – перетворює об'єкт у рядок за допомогою функції `ascii()`:

Приклад 14

```
>>> print("%a" % ("рядок"))
'\u0440\u044f\u0434\u043e\u043a'
```

c – виводить одиночний символ або перетворює числове значення на символ. Як приклад виведемо числове значення й відповідний до цього значення символ:

Приклад 15

```
>>> for i in range(33, 127): print("%s => %c" % (i, i))
65 => A
```

Символи d, i і o у параметрі <Тип перетворення>

d i **i** – повертають цілу частину числа:

Приклад16

```
>>> print("%d %d %d" % ( 10, 25.6, -80))
10 25 -80
>>> print("%i %i %i" % ( 10, 25.6, -80))
10 25 -80
```

o – вісімкове значення:

Приклад17

```
>>> print("%o %o %o" % (0o77, 10, 10.5))
77 12 12
>>> print("%#o %#o %#o" % (0o77, 10, 10.5))
0o77 0o12 0o12
```

Символи x і X у параметрі <Тип перетворення>

x – шістнадцяткове значення в нижньому регістрі:

Приклад 18

```
>>> print("%x %x %x" % (0xff, 10, 10.5))
ff a a
>>> print("%#x %#x %#x" % (0xff, 10, 10.5))
0xff 0xa 0xa
```

X – шістнадцяткове значення у верхньому регістрі:

```
>>> print("%X %X %X" % (0xff, 10, 10.5))
FF A A
>>> print("%#X %#X %#X" % (0xff, 10, 10.5))
0XFF 0XA 0XA
```

Символи f і F у параметрі <Тип перетворення>

f і **F** – дійсне число в десятковій формі:

```
>>> print("%f  %f  %f" % (300,  18.65781452,  -12.5))
300.000000  18.657815 -12.500000

>>> print("%F  %F  %F" % (300,  18.65781452,  -12.5))
300.000000 18.657815 -12.500000

>>> print("%#.0F  %#.0F" % (300, 300))
300. 300.
```

Символи e і E у параметрі <Тип перетворення>

e – дійсне число в експонентній формі (буква «**e**» у нижньому регістрі):

Приклад 19

```
>>> print("%e %e" % (3000, 18657.81452))
3.000000e+03 1.865781e+04
```

E – дійсне число в експонентній формі (буква «**E**» у верхньому регістрі):

Приклад 20

```
>>> print("%E %E" % (3000, 18657.81452))
3.000000E+03 1.865781E+04
```

Символи g і G у параметрі <Тип перетворення>

g – еквівалентно **f** або **e** (вибирається більш короткий запис числа):

Приклад 21

```
>>> print("%g %g %g" % (0.086578, 0.000086578, 1.865E-005))
0.086578 8.6578e-05 1.865e-05
```

G – еквівалентно **f** або **E** (вибирається більш короткий запис числа):

```
>>>print("%G %G %G" % (0.086578, 0.000086578, 1.865E-005))
0.086578 8.6578E-05 1.865E-05
```

Вивід службових символів

Якщо усередині рядка необхідно використовувати символ відсотка (%), то цей символ слід подвоїти (%%), інакше буде виведене повідомлення про помилку:

Приклад 22

```
>>> print("% %s" % ("- це символ відсотка"))
# Помилка
Traceback (most recent call last):
File "<Input>", line 1, in <module>
print("% %s" % ("- це символ відсотка")) # Помилка
Typeerror:not all arguments converted during string
formatting
# Нормально
>>> print("%% %s" % ("- це символ відсотка"))
% - це символ відсотка
```

Досягти такого ж виводу можна так

```
>>> print("%s" % ("% - це символ відсотка"))
```

Метод форматування

`center ( <Ширина> [, <Символ> ])`

Здійснює вирівнювання рядка по центру усередині поля зазначеної ширини. Якщо другий параметр не зазначений, то справа і зліва від початкового рядка будуть додані пробіли.

Приклад 26

```
>>> s = "str"
>>> s.center(15), s.center(11, "-")
('     str     ', '----str----')
```

Приклад спільного використання методу `center` і прапора - для вирівнювання

Виконаємо вирівнювання трьох фрагментів шириною 15 символів. Перший фрагмент буде вирівняний по правому краю, другій – по лівому, а третій – по центру:

Приклад 27

```
>>> s = "str"
>>> "'%5s' '%-5s' '%s'" % (s,s,s.center(5))
"'  str' 'str  ' ' str '"
```

Якщо кількість символів у рядку перевищує ширину поля, то значення ширини ігнорується, і рядок повертається повністю:

Приклад 28

```
>>> s = "string"
>>> s.center(6), s.center(5)
( 'string', 'string')
```

Метод форматування

`ljust ( <Ширина> [, <Символ>] )`

Виконує вирівнювання рядка по лівому краю усередині поля зазначеної ширини. Якщо другий параметр не зазначений, то праворуч від початкового рядка будуть додані пробіли. Якщо кількість символів у рядку перевищує ширину поля, то значення ширини ігнорується, і рядок повертається повністю.

Приклад 29

```
>>> s = "string"
>>> s.ljust(15), s.ljust(15, "-")
('string         ', 'string---------')
>>> s.ljust(6), s.ljust(5)
( 'string', 'string' )
```

Метод форматування

`rjust(<Ширина>[, <Символ>])`

Виконує вирівнювання рядка по правому краю усередині поля зазначеної ширини. Якщо другий параметр не зазначений, то ліворуч від початкового рядка будуть додані пробіли. Якщо кількість символів у рядку перевищує ширину поля, то значення ширини ігнорується, і рядок повертається повністю.

Приклад 30

```
>>> s = "string"
>>> s.rjust(15), s.rjust(15, "-")
('          string', '----------string')
>>> s.rjust(6), s.rjust(5)
('string', 'string')
```

Метод форматування zfill(<Ширина>)

Виконує вирівнювання фрагмента по правому краю усередині поля зазначеної ширини. Ліворуч від фрагмента будуть додані нулі. Якщо кількість символів у рядку перевищує ширину поля, то значення ширини ігнорується, і рядок повертається повністю.

Приклад 31

```
>>> "5".zfill(20), "123456".zfill(5)
('00000000000000000005', '123456')

>>> s="Рядок"
>>> s.zfill(12), s.zfill(5)
('000000Рядок', 'Рядок')
```

Метод `format()`

Крім операції форматування, ми можемо використовувати для цієї ж мети метод `format()`. Він має наступний синтаксис:

```
<Рядок>={<Рядок спец.формату>}.format(*args, **kwargs)
```

У параметрі `<Рядок спеціального формату>` усередині фігурних дужок: `{ }` – вказуються специфікатори, що мають наступний синтаксис:

```
{[<Поле>][! <Функція>][ : <Формат>]}
```

{ [<Поле>] [! <Функція>] [: <Формат>] }

1. Усі символи, розташовані поза фігурними дужками, виводяться без перетворень.

2. Якщо всередині рядка необхідно використовувати фігурні дужки `{ }`, то ці символи слід подвоїти, інакше виконується виключення `Valueerror`.

Приклад 32

```
>>> print("{0}".format("Мінімальний набір параметрів"))
Мінімальний набір параметрів

>>> print("Символи {{i}} - {0}".format("спеціальні"))
Символи { } - спеціальні
```

Використання параметра <Поле>

1. Можна вказати індекс позиції (нумерація починається з нуля) або ключ.
2. Дозволено комбінувати позиційні й іменовані параметри. У цьому випадку в методі `format()` іменовані параметри вказуються в самому кінці.

Приклад 33

Указуємо індекс для параметрів:

```
>>> "{0} - {1} - {2}".format(10,12.3, "string")# Індекси
'10 - 12.3 - string'
```

Указуємо список у параметрі `*args`:

```
>>> arr = [10, 12.3, "string"]
>>> "{0} - {1} - {2}".format(*arr)# Індекси
'10 - 12.3 - string'
```

Указуємо ключ для елемента словника:

```
>>>"{model}-{color}".format(color="red", model ="BMW")#ключі
'BMW - red'
```

Указуємо словник у параметрі `**kwargs`:

```
>>> d = {"color": "red", "model": "BMW"}
>>> "{model}- {color}".format(**d) # Ключі
'BMW - red'
```

Змішана вказівка

```
>>>"{color} - {0}".format(2015, color="red")
# Комбінація
'red - 2015'
```

Об'єкт як параметр методу `format()`

1. Як параметр в методі `format()` можна вказати об'єкт.
2. Для доступу до елементів по індексу всередині рядка формату застосовуються квадратні дужки
3. Для доступу до атрибутів об'єкта використовується точкова нотація:

Приклад 34

```
>>> arr = [10, [12.3, "string"]]
>>> "{0[0]} - {0[1][0]} - {0[1][1]}".format(arr)
#{0[0]} - об'єкт 0, елемент 0
#{0[1][0]} - об'єкт 1, елемент 0
#{0[1][1]} - об'єкт 1, елемент 1
'10 - 12.3 - string'
# Індекси
>>>"{a[0]} - {a[1][1]}".format(a=arr)
'10 - string'
```

Коротка форма запису без параметра **<Поле>**

Існує також коротка форма запису, при якій параметр <Поле> не вказується. У цьому випадку дужки без зазначеного індексу нумеруються зліва направо, починаючи з нуля:

Приклад 35

```
>>>"{} - {} - {} - {n}".format ( 1, 2, 3, n=4)
# "{0} – {1} – {2} – {n}"
'1 – 2 – 3 – 4'

>>>"{} - {} - {n} - {}".format(1, 2, 3, n= 4)
# "{0} – {1} – {n} – {2}"
'1 – 2 – 4 – 3'
```

Використання параметра <Функція>

{[<Поле>][! <Функція>][: <Формат>]}

Задає функцію, за допомогою якої обробляються дані перед вставкою в рядок.

s – дані обробляються функцією str (),

r – функцією repr(),

a – функцією ascii ().

Якщо параметр не зазначений, то для перетворення даних у рядок використовується функція str ().

Приклад 36

```
>>> print("{0!s}".format("рядок"))#str()рядок

>>> print("{0!r}".format("рядок"))#repr()'рядок'

>>> print("{0!a}".format("рядок"))  # ascii()
''\u0440\u044f\u0434\u043e\u043a'
```

Використання параметра <Формат>

{[<Поле>][! <Функція>][: <Формат>]}

У параметрі <Формат> вказується значення, що має наступний синтаксис:

[[<Заповнювач>] <Вирівнювання>] [<Знак>] [#] [0]
[<Ширина>][,][.<Точність>][<Перетворення>]

Параметр <Ширина>

Задає мінімальну ширину поля. Якщо рядок не міститься в зазначеній ширині, то значення ігнорується, і рядок виводиться повністю:

Приклад 37

```
>>>"'{0:10}' '{1:3}'".format(3, "string")
"'         3' 'string'"
```

Передача параметра <Ширина>

Ширину поля можна передати як параметр в методі `format()`. У цьому випадку замість числа вказується індекс параметра всередині фігурних дужок:

Приклад 38

```
>>>"'{0:{1}}'".format(3,10) # 10-це ширина поля
"'         3'"
```

За замовчуванням значення всередині поля вирівнюється по правому краю.

Параметр <Вирівнювання>

Управляє вирівнюванням. Можна вказати наступні значення: **<, >,^,=**

< – по лівому краю;

> – по правому краю;

^ – по центру поля.

Приклад 39

```
>>>"'{0:<10}'{1:>10}' '{2:10}'".format(3,3,3)
"'3         '  '         3'  '       3       '"
```

= – знак числа вирівнюється по лівому краю, а число по правому краю

Приклад 40

```
>>>"'{0:=10}' '{1:=10}'".format(-3, 3)
"'-         3' '          3'"
```

<Вирівнювання> з заповненням нулями

1. Як видно з наведеного прикладу, простір між знаком і числом за замовчуванням заповнюється пробілами.
2. Знак додатного числа не вказується.
3. Щоб замість пробілів простір заповнювався нулями, необхідно вказати нуль перед шириною поля

Приклад 41

```
>>> "'{0:=010}' '{1:=010}'".format(-3, 3)
"'-000000003' '0000000003'"

>>> "'{0:=05}' '{1:=06}'".format(-123, 34)
"'-0123' '000034'"
```

Параметр <Заповнювач>

1. Такого ж ефекту можна досягтися, указавши нуль у параметрі <Заповнювач>.
2. У цьому параметрі допускаються інші символи, які будуть виводитися замість пробілів:

Приклад 42

```
>>>"'{0:0=10}' '{1:0=10}'".format(-3, 3)
"'-000000003' '0000000003'"

>>>"'{0:*<10}' '{1:+>10}' '{2:10}'".format(3, 3, 3)
"'3********* ' '+++++++++3' '....  3.....'"
```

Параметр <Знак>.

Припустимі значення цього параметра:

+, пробіл, відсутність параметрів

+ – задає обов'язковий вивід знака як для від'ємних, так і для додатних чисел;

параметри відсутні – вивід знака тільки для від'ємних чисел (значення за замовчуванням);

пробіл – вставляє перед додатним числом. Перед від'ємним числом буде стояти мінус.

Приклад 43

```
>>>"'{0:+}' '{1:+}' '{0:-}' '{1:-}'".format(3,-3)
"'+3' '-3' '3' '-3'"

>>>"'{0: }' '{1: }'".format(3,-3)# Пробіл
"' 3' '-3'"
```

Параметр <Перетворення>

Для цілих чисел у параметрі <Перетворення> можуть бути зазначені наступні опції: **b,c,d,n**

b – двійкове значення:

Приклад 44.

```
>>>"'{0:b}' '{0:#b}'".format(3)
"'11' '0b11'"
```

c – перетворює ціле число у відповідний символ:

Приклад 45.

```
>>>"'{0:c}'".format(100)
"'d'"
```

d – десяткове значення;

n – аналогічно опції **d**, але враховує настроювання локалі. Наприклад, виведемо велике число з поділом тисячних розрядів пробілом:

Приклад 46

```
>>> print("{0:,d}".format(1000000000))
100,000,000
```

o – вісімкое значення:

```
>>> "'{0:d}' '{0: o}' '{0:#o}'".format(511)
"'511' '777' ' 0o777'"
```

x – шістнадцяткове значення в нижньому регістрі:

```
>>> "'{0:x}' '{0:#x}'".format(255)
"'ff ' '0xff'"
```

X – шістнадцяткове значення у верхньому регістрі:

```
>>> '"{0:X}' ' {0:#X}'".format(255)
"'FF' '0XFF'"
```

Параметр <**Перетворення**>

Для дійсних чисел у параметрі <**Перетворення**> можуть бути зазначені наступні опції: **f**, **F**

f і **F** – дійсне число в десятковій формі:

Приклад 47

```
>>>"'{0:f}' '{1:f}' '{2:f}'".format(30, 18.6578145, -2.5)
"'30.000000' ' 18.657815' '-2.500000'"
```

Параметр <Точність>

За замовчуванням виведене число має шість знаків після коми. Задати інше кількість знаків після коми можна в параметрі <**Точність**>:

Приклад 48

```
>>>"'{0:7f}' '{1:.2f}"'.format(18.6578145, -2.5)
"'18.6578145' '-2.50'"
```

e – дійсне число в експонентній формі (буква e в нижньому регістрі):

Приклад 49

```
>>>"'{0:e}' '{1:e} '".format(3000,
18657.81452)
"'3.000000e+03' '1.865781e+04'"
```

E – дійсне число в експонентній формі (буква `E` в верхньому регістрі):

Приклад 50

```
>>> "'{0:E}' '{1:E}'".format(3000, 18657
.81452)
"'3.000000E+03' '1.865781E+04'"
```

Тут за замовчуванням кількість знаків після коми також рівно шести, але ми можемо вказати іншу величину цього параметра:

Приклад 51

```
>>> "'{0:.2e}' '{1:.2E}'".format(3000,
18657.81452)
"'3.00e+03' '1.87E+04'"
```

g - еквівалентно **f** або **e** (вибирається більш короткий запис числа):

Приклад 52

```
>>>"'{0:g}' '{1:g}'".format(0.086578, 0.000086578)
"'0.086578' '8.6578e-05'"
```

n - аналогічно опції **g**, але враховує настроювання локалі;

G - еквівалентно **f** або **E** (вибирається більш короткий запис числа):

Приклад 53

```
>>>"'{0:G}' '{1:G}'".format(0.086578, 0.000086578)
"'0.086578' '8. 6578E-05'"
```

`%` – множить число на 100 і додає символ відсотка в кінець. Значення відображається відповідно до опції **`f`**.

Приклад 54

```
>>>"'{0:%}' '{1:.4%}'".format(0.086578, 0.000086578)
"'8.657800%' '0.0087%'"
```