Форматирование строк с помощью f-строк

В Python 3.6 добавился новый вариант форматирования строк - f-строки или интерполяция строк. F-строки позволяют не только подставлять какие-то значения в шаблон, но и позволяют выполнять вызовы функций, методов и т.п.

Во многих ситуациях f-строки удобней и проще использовать, чем format, кроме того, f-строки работают быстрее, чем format и другие методы форматирования строк.

Синтаксис

F-строки - это литерал строки с буквой `f` перед ним. Внутри f-строки в паре фигурных скобок указываются имена переменных, которые надо подставить:

```
ip = '10.1.1.1'
mask = 24

f"IP: {ip}, mask: {mask}"
# 'IP: 10.1.1.1, mask: 24'
```

Аналогичный результат с format можно получить так: `"IP: {ip}, mask: {mask}".format(ip=ip, mask=mask)`.

Очень важное отличие f-строк от format: f-строки это выражение, которое выполняется, а не просто строка. То есть, в случае с ipython, как только мы написали выражение и нажали Enter, оно выполнилось и вместо выражений `{ip}` и `{mask}` подставились значения переменных.

Поэтому, например, нельзя сначала написать шаблон, а затем определить переменные, которые используются в шаблоне:

```
f"IP: {ip}, mask: {mask}"
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
```

```
last)
<ipython-input-1-e6f8e01ac9c4> in <module>()
----> 1 f"IP: {ip}, mask: {mask}"

NameError: name 'ip' is not defined
```

Кроме подстановки значений переменных, в фигурных скобках можно писать выражения:

```
octets = ['10', '1', '1', '1']
mask = 24

print(f"IP: {'.'.join(octets)}, mask: {mask}")
# 'IP: 10.1.1.1, mask: 24'
```

После двоеточия в f-строках можно указывать те же значения, что и при использовании format:

```
oct1, oct2, oct3, oct4 = [10, 1, 1, 1]

print(f'''
        IP address:
        {oct1:<8} {oct2:<8} {oct3:<8} {oct4:<8}
        {oct1:08b} {oct2:08b} {oct3:08b} {oct4:08b}''')

# IP address:
# 10       1        1        1
# 00001010 00000001 00000001 00000001
```

Особенности использования f-строк

При использовании f-строк нельзя сначала создать шаблон, а затем его использовать, как при использовании format.

F-строка сразу выполняется и в нее подставляются значения переменных, которые должны быть определены ранее:

```
ip = '10.1.1.1'
mask = 24

print(f"IP: {ip}, mask: {mask}")
# IP: 10.1.1.1, mask: 24
```

Если необходимо подставить другие значения, надо создать новые переменные (с теми же именами) и снова написать `f-строку`

При использовании f-строк в циклах, f-строку надо писать в теле цикла, чтобы она "подхватывала" новые значения переменных на каждой итерации:

```
ip_list = ['10.1.1.1/24', '10.2.2.2/24', '10.3.3.3/24']

for ip_address in ip_list:
  ip, mask = ip_address.split('/')
  print(f"IP: {ip}, mask: {mask}")

# IP: 10.1.1.1, mask: 24
# IP: 10.2.2.2, mask: 24
# IP: 10.3.3.3, mask: 24
```

Примеры f-строк

Базовая подстановка переменных:

```
intf_type = 'Gi'
intf_name = '0/3'

print(f'interface {intf_type}/{intf_name}')
# 'interface Gi0/3'
```

Выравнивание столбцами:

```
topology = [['sw1', 'Gi0/1', 'r1', 'Gi0/2'],
            ['sw1', 'Gi0/2', 'r2', 'Gi0/1'],
            ['sw1', 'Gi0/3', 'r3', 'Gi0/0'],
            ['sw1', 'Gi0/5', 'sw4', 'Gi0/2']]


for connection in topology:
  l_device, l_port, r_device, r_port = connection
  print(f'{l_device:10} {l_port:7} {r_device:10} {r_port:7}')

# sw1        Gi0/1   r1         Gi0/2
# sw1        Gi0/2   r2         Gi0/1
```

```
# sw1         Gi0/3    r3          Gi0/0
# sw1         Gi0/5    sw4         Gi0/2
```

Ширина столбцов может быть указана через переменную:

```
topology = [['sw1', 'Gi0/1', 'r1', 'Gi0/2'],
            ['sw1', 'Gi0/2', 'r2', 'Gi0/1'],
            ['sw1', 'Gi0/3', 'r3', 'Gi0/0'],
            ['sw1', 'Gi0/5', 'sw4', 'Gi0/2']]


width = 10

for connection in topology:
   l_device, l_port, r_device, r_port = connection

print(f'{l_device:{width}} {l_port:{width}} {r_device:{width}} {r_port:
{width}}')

# sw1         Gi0/1       r1          Gi0/2
# sw1         Gi0/2       r2          Gi0/1
# sw1         Gi0/3       r3          Gi0/0
# sw1         Gi0/5       sw4         Gi0/2
```

Работа со словарями

```
session_stats = {'done': 10, 'todo': 5}

if session_stats['todo']:
   print(f"Pomodoros done: {session_stats['done']}, TODO:
{session_stats['todo']}")
else:
   print(f"Good job! All {session_stats['done']} pomodoros done!")

# Pomodoros done: 10, TODO: 5
```

Вызов функции len внутри f-строки:

```
topology = [['sw1', 'Gi0/1', 'r1', 'Gi0/2'],
            ['sw1', 'Gi0/2', 'r2', 'Gi0/1'],
            ['sw1', 'Gi0/3', 'r3', 'Gi0/0'],
            ['sw1', 'Gi0/5', 'sw4', 'Gi0/2']]
```

```
print(f'Количество подключений в топологии: {len(topology)}')
Количество подключений в топологии: 4
```

Вызов метода upper внутри f-строки:

```
name = 'python'

print(f'Zen of {name.upper()}')
# Zen of PYTHON
```

Конвертация чисел в двоичный формат:

```
ip = '10.1.1.1'

oct1, oct2, oct3, oct4 = ip.split('.')

print(f'{int(oct1):08b} {int(oct2):08b} {int(oct3):08b}
{int(oct4):08b}')
# 00001010 00000001 00000001 00000001
```

Округления

```
num = 2.3123

print(f'{num:.1f}') # Limit to 1 decimal
print(f'{num:.2f}') # Limit to 2 decimals

# 2.3
# 2.31
```

Что использовать format или f-строки

Во многих случаях f-строки удобней использовать, так как шаблон выглядит понятней и компактней. Однако бывают случаи, когда метод format удобней. Например:

```
ip = [10, 1, 1, 1]
```

```
oct1, oct2, oct3, oct4 = ip
print(f'{oct1:08b} {oct2:08b} {oct3:08b} {oct4:08b}')

# 00001010 00000001 00000001 00000001

template = "{:08b} "*4

template.format(oct1, oct2, oct3, oct4)
# '00001010 00000001 00000001 00000001 '
```

Еще одна ситуация, когда format, как правило, удобней использовать: необходимость использовать в скрипте один и тот же шаблон много раз. F-строка выполнится первый раз и подставит текущие значения переменных и для использования шаблона еще раз, его надо заново писать. Это значит, что в скрипте будут находится копии одной и то же строки. В то же время format позволяет создать шаблон в одном месте и потом использовать его повторно, подставляя переменные по мере необходимости.

Это можно обойти создав функцию, но создавать функцию для вывода строки по шаблону далеко не всегда оправдано. Пример создания функции:

```
def show_me_ip(ip, mask):
   return f"IP: {ip}, mask: {mask}"


show_me_ip('10.1.1.1', 24)
# 'IP: 10.1.1.1, mask: 24'

show_me_ip('192.16.10.192', 28)
# 'IP: 192.16.10.192, mask: 28'
```

Теги: functions f-string python string

✎ Отредактировать эту страницу

Последнее обновление ***4 сент. 2023 г.*** *от* ***Stavis***