格式化输出——f-string

————since Python 3.6

f-string，亦称为格式化字符串常量（formatted string literals），是 Python3.6 新引入的一种字符串格式化方法，主要目的是使格式化字符串的操作更加简便。f-string 在形式上是以 f 或 F 修饰符引领的字符串（**f'xxx'**或 **F'xxx'**），以大括号{}标明被替换的字段。f-string 在本质上并不是字符串常量，而是一个在运行时运算求值的表达式。

f-string 在功能方面不逊于传统的%-formatting 语句和 str.format()函数，同时性能又优于二者，且使用起来也更加简洁明了，因此对于 Python3.6 及以后的版本，推荐使用 f-string 进行字符串格式化。

一、基本用法

f-string 是字符串前加“f”或者“F”，然后字符串内“{}”就有特殊意义了，里面填写变量名，运行时自动计算这个变量的值，代入字符串内输出。

反正一句话，不涉及格式化问题，就是在 f-string 中的大括号填写一句可执行的 Python 语句，运行时把它执行得到的值，替换这一个“{}”块。

```
name = "sky"
age = 18
print(f"My name is {name}, age is {age}.")
# 输出：My name is sky, age is 18.

fruits = {"apple":"red","banana":"yellow"}
s = F"The apple is {fruits['apple']},the banana is {fruits['banana']}"
# f-string 内的引号和整体的外部引号不能一致，否则会解析错误
print(s)
# 输出：The apple is red,the banana is yellow

s = "I love U"
print(f"the reverse is '{s[::-1]}'")
# 输出：the reverse is 'U evol I'
# 逆置还可以："".join(reversed(l))
```

二、f-string 的一些细节

1. 引号

有时候在大括号内写表达式的时候回用到引号，但是，大括号本身就是在 f-string 的引号内，如果是同样的引号的话，显然会造成解析错误的，所以应当注意，内部引号和边界引号一定不能相同，这样就能正常运行。

2. 大括号

f-string 中，大括号是有特殊含义的，它包裹的是可执行的语句，但是如果我们需要用到大括号怎么办呢？我们参照下面的例子：

```
print(f"the set is {{1}}")                    # 'the set is {1}'
print(f"{{{6}}}")                             # '{6}'，这里表达式是整数 6
print(f"{{{{6}}}}")                           # '{{6}}'，这里表达式是集合{6}
```

3. 反斜杠

这就涉及到转义的问题。在大括号外面仍然可以用反斜杠进行转义，大括号内部不允许出现反斜杠！若一定要用则应该把它赋值给变量，再通过变量传到大括号内。

4. 多行 f-string

```
s = f"ADC" \
      f"SUP"
r = f"""red
      blue"""
```

显然，三引号写法不必每行前加 f/F 标识 f-string

三、格式化

f-string 采用 {content:format} 设置字符串格式，其中 content 是替换并填入字符串的内容，可以是变量、表达式或函数等，format 是格式描述符。采用默认格式时不必指定 {:format}，如上面例子所示只写 {content} 即可。

语法：{**[content]:[填充字符][对齐][符号位][前缀][最小宽度][千位分隔符][.精度][类型]**}

（1）类型

符号	示例	结果	含义
d	f"age = ({26:4d})"	'age = (26)'	整数
f	f"{3.1415:.2f}"	'3.14'	浮点数
F	f"{math.nan:F}"	'NAN'	inf,nan 等转换为大写
%	f"{0.024937:.2%}"	'2.49%'	百分比格式
E/e	f"{314.15:.2e}"	'3.14e+02'	科学计数法
s	f"age = ({'26':4s})"	'age = (26)'	字符串
b	f"{36:b}"	'100100'	二进制整数
o	f"{36:o}"	'44'	八进制整数
X/x	f"{36:x}"	'24'	十六进制整数

（2）对齐与填充

^	居中
<	左对齐
>	右对齐

对齐符号左边是填充字符，注意：只能是一个字符！不写默认是空格填充

有填充字符必须有对齐符号，不然中间丢失对其符号会报错！

默认，字符串左对齐，数值右对齐

（3）符号位和前缀：针对数值，用于字符串会报错

+	显示正负号
-	负数显示负号，正数不显示
空格	正数显示空格，负数显示符号
#	仅用于显示前缀，如八进制、十六进制

（4）最小宽度

格式化后，该段格式化后整体的最小长度，不足长度会填充填充字符，如果替换的内容长度超过设定的最小长度，以实际长度为准。

（5）精度

作用于字符串时，是输出字符的个数，如果超过字符串长度则输出全部字符串内容；

作用于整数时会报错；

作用于浮点数时，表示输出小数位数，科学计数法表示的是底数的小数位数，默认 6 位小数，目前试验看来是四舍五入。

（6）千位分隔符

逗号或下划线，只作用于数值

```
>>> f"{1234356789.123456:^#20_.4f}"
' 1_234_356_789.1235 '
>>> f"{1234356789.123456:^#20,.4f}"
' 1,234,356,789.1235 '
```

四、日期时间的格式化

1. 格式化符号

格式字符太多，以下给出经常用得到的格式字符表示：

符号	含义	示例
%y	两位数的年份表示	99
%Y	四位数的年份表示	2019
%m	月份	12
%d	一个月的第几天	31
%F	返回年月日（-连接）	2019-05-22
%D	返回日月年（/连接）	05/22/19
%b	本地简化月份名称	Aug
%B	本地完整月份名称	August
%a	本地简化星期名称	Mon
%A	本地完整星期时间	Monday
%j	一年中的第几天	001-366
%p	本地 a.m.和 p.m.	AM/PM
%H	24 小时制	23
%I	12 小时制	05
%M	分钟，补足 2 位	09
%S	秒，补足 2 位	59
%f	微秒，补足 6 位	553777
%x	本地相应的日期表示	08/15/20
%X	本地相应的时间表示	00:00:00
%%	百分号	%

2. 一些示例

```
import datetime
d = datetime.datetime.now()

s = f"{d:%Y-%m-%d %A %p %H:%M:%S.%f}"
print(s)                                    # 2019-05-22 Wednesday PM 17:13:18.393931

s = f"{d:%F}"
print(s)                                    # 2019-05-22
s = f"{d:%D}"
print(s)                                    # 05/22/19

s = f"{d:%d %b %Y}"
print(s)                                    # 22 May 2019

s = f"{d:%X}"
print(s)                                    # 17:15:47
```

五、使用 lambda 表达式

f-string 大括号内也可填入 lambda 表达式，但 lambda 表达式的 ： 会被 f-string 误认为是表达式与格式描述符之间的分隔符，为避免歧义，需要将 lambda 表达式置于括号()内：

```
l = [1,2,3,4,5]
s = f"求列表每个数的平方:{(lambda x: [y*y for y in x])(l)}"
print(s)#  求列表每个数的平方:[1, 4, 9, 16, 25]
```

【注】lambda 表达式必须用括号括起来，不然会无法解析 f-string 内容，要调用时，方法为：**(lambda 函数)(*args,*kwargs)**

六、f-string 与对象

__str __()和__repr __()方法处理对象如何呈现为字符串，因此您需要确保在类定义中包含至少一个这些方法。如果必须选择一个，请使用__repr __()，因为它可以代替__str __()。

__str __()返回的字符串是对象的非正式字符串表示，应该可读。__repr __()返回的字符串是官方表示，应该是明确的。**调用 str()和 repr()比直接使用__str __()和__repr __()更好。**

默认情况下，f 字符串将使用__str __()方法，但如果包含**转换标志!r**，则可以确保它们使用__repr __()方法。

```
class Person(object):
    def __init__(self,name,age):
        self.__name = name
        self.__age = age
    def __str__(self):
        return f"我的名字是 {self.__name} ， 今年 {self.__age} 岁了!"
    def __repr__(self):
        return f"我的名字是 {self.__name} ， 今年 {self.__age} 岁了! Superise!"

p = Person("易",18)
print(f"{p}")                               # 我的名字是 易 ， 今年 18 岁了!
print(f"{p!r}")                             # 我的名字是 易 ， 今年 18 岁了! Superise!
```