中国大学 MOOC 课程

《Python 语言程序设计》

课后练习（第 3 周）

北京理工大学

Python 语言教学团队

【说明】

本文是中国大学 MOOC 课程《Python 语言程序设计》第 3 周的课后学习内容，预估学习完成时间约 30 分钟。

本周课后学习内容是 Python 语言中字符串的格式化方法。Python 提供两种字符串格式方法。一种类似 C 语言的格式化方法，使用%；另一种采用 format()方法，Python 推荐使用这种。

这里介绍 Python 推荐的 format()方法，相比 C 语言风格格式化方法，该方法能力更强、更直观、更容易格式化组合数据类型。

请同学们学习课后内容同时打开 IDLE，边学边练。

对于尚未安装 Python 运行环境的同学，请根据第 1 周课程内容介绍的步骤安装 Python 3.5.1 或者 Python 3.5.2 版本解释器，如果操作系统兼容性有问题，可以安装 Python 3.4 版本解释器。

【学习内容】

字符串类型格式化采用 format()方法，基本使用格式是：

<模板字符串>.format(<逗号分隔的参数>)

<模板字符串>由一系列的槽组成，用来控制修改字符串中嵌入值出现的位置，其基本思想是将 format()方法的<逗号分隔的参数>中的参数按照序号关系替换到<模板字符串>的槽中。

槽用大括号({})表示，如果大括号中没有序号，则按照出现顺序替换，如图 3.1 所示。

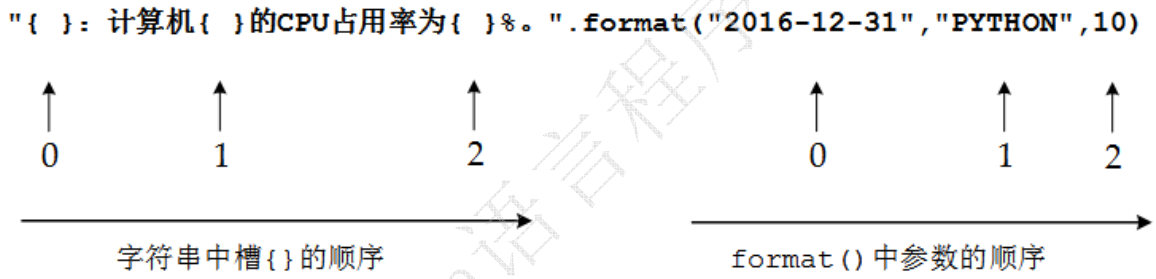

图 3.1: format()方法的槽顺序和参数顺序

如果大括号中指定了使用参数的序号，按照序号对应参数替换，如图 3.2 所示。调用 format()方法后会返回一个新的字符串，参数从 0 开始编号。

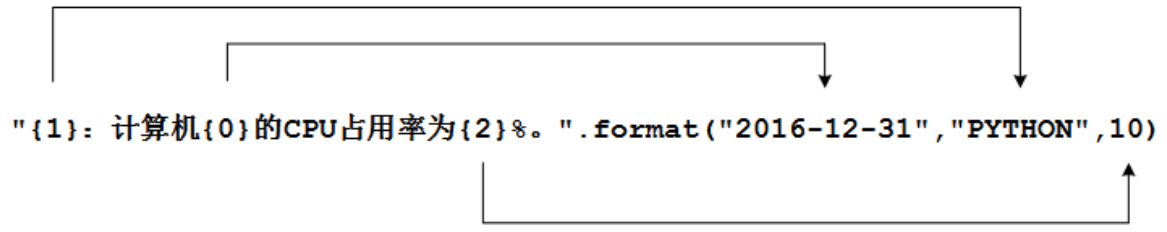

图 3.2: format()方法槽与参数的对应关系

例如：

```
>>>"{}：计算机{}的 CPU 占用率为{}%。".format("2016-12-31","PYTHON",10)
'2016-12-31：计算机 PYTHON 的 CPU 占用率为 10%。'
```

format()方法可以非常方便地连接不同类型的变量或内容，如果需要输出大括号，采用{{表示{，}}表示}，例如：

```
>>>"{}{}{}".format("圆周率是",3.1415926,"...")
'圆周率是 3.1415926...'
>>>"圆周率{{{1}{2}}}是{0}".format("无理数",3.1415926,"...")
'圆周率{3.1415926...}是无理数'
>>>s="圆周率{{{1}{2}}}是{0}"  #大括号本身是字符串的一部分
>>>s
'圆周率{{{1}{2}}}是{0}'
>>>s.format("无理数",3.1415926,"...")  #当调用 format()时解析大括号
'圆周率{3.1415926...}是无理数'
```

format()方法中<模板字符串>的槽除了包括参数序号，还可以包括格式控制信息。此时，槽的内部样式如下：

{<参数序号>: <格式控制标记>}

其中，<格式控制标记>用来控制参数显示时的格式，如图 3.3 所示。

:	<填充>	<对齐>	<宽度>	,	<.精度>	<类型>
	用于填充的单个字符	< 左对齐 > 右对齐 ^ 居中对齐	槽的设定输出宽度	数字的千位分隔符 适用于整数和浮点数	浮点数小数部分的精度 或 字符串的最大输出长度	整数类型 b, c, d, o, x, X 浮点数类型 e, E, f, %

图 3.3: 槽中格式控制标记的字段

<格式控制标记>包括：<填充><对齐><宽度>,<.精度><类型>6 个字段，这些字段都是可选的，可以组合使用，逐一介绍如下。

<填充>、<对齐>和<宽度>是 3 个相关字段。<宽度>指当前槽的设定输出字符宽度，如果该槽对应的 format()参数长度比<宽度>设定值大，则使用参数实际长度。如果该值的实际位数小于指定宽度，则位数将被默认以空格字符补充。<对齐>指参数在<宽度>内输出时的对齐方式，分别使用<、>和^三个符号表示左对齐、右对齐和居中对齐。<填充>指<宽度>内除了参数外的字符采用什么方式表示，默认采用空格，可以通过<填充>更换。例如：

```
>>>s = "PYTHON"
>>>"{0:30}".format(s)
'PYTHON                        '
>>>"{0:>30}".format(s)
'                        PYTHON'
>>>"{0:*^30}".format(s)
'************PYTHON************'
>>>"{0:-^30}".format(s)
'------------PYTHON------------'
>>>"{0:3}".format(s)
'PYTHON'
```

<格式控制标记>中逗号（,）用于显示数字的千位分隔符，例如：

```
>>>"{0:-^20,}".format(1234567890)
'---1,234,567,890----'
>>>"{0:-^20}".format(1234567890)  #对比输出
'-----1234567890-----'
>>>"{0:-^20,}".format(12345.67890)
'----12,345.6789-----'
```

<.精度>表示两个含义，由小数点（.）开头。对于浮点数，精度表示小数部分输出的有效位数。对于字符串，精度表示输出的最大长度。

```
>>>"{0:.2f}".format(12345.67890)
'12345.68'
>>>"{0:H^20.3f}".format(12345.67890)
'HHHHH12345.679HHHHHH'
>>>"{0:.4}".format("PYTHON")
'PYTH'
```

〈类型〉表示输出整数和浮点数类型的格式规则。对于整数类型，输出格式包括 6 种：

- b：输出整数的二进制方式；
- c：输出整数对应的 Unicode 字符；
- d：输出整数的十进制方式；
- o：输出整数的八进制方式；
- x：输出整数的小写十六进制方式；
- X：输出整数的大写十六进制方式；


```
>>>"{0:b},{0:c},{0:d},{0:o},{0:x},{0:X}".format(425)
'110101001,Σ,425,651,1a9,1A9'
```

对于浮点数类型，输出格式包括 4 种：

- e：输出浮点数对应的小写字母 e 的指数形式；
- E：输出浮点数对应的大写字母 E 的指数形式；
- f：输出浮点数的标准浮点形式；
- %：输出浮点数的百分形式。

浮点数输出时尽量使用<.精度>表示小数部分的宽度，有助于更好控制输出格式。

```
>>>"{0:e},{0:E},{0:f},{0:%}".format(3.14)
'3.140000e+00,3.140000E+00,3.140000,314.000000%'
>>>"{0:.2e},{0:.2E},{0:.2f},{0:.2%}".format(3.14)
'3.14e+00,3.14E+00,3.14,314.00%'
```

Python 有哪些数据类型？

（请自找介质默写）