

# Python 语言程序设计:Python 语法基础元素 2

李宽

likuan@dgut.edu.cn

东莞理工学院

2019.10



## 1 字符串类型

- 字符串类型的 4 种表示
- 字符串类型的引号处理
- 字符串类型之切片
- 字符串类型之转义符
- 字符串类型之操作符
- 字符串类型之操作函数
- 字符串类型之方法
- 字符串类型之格式化

## 2 教学实例：打印三角形

## 1 字符串类型

- 字符串类型的 4 种表示
- 字符串类型的引号处理
- 字符串类型之切片
- 字符串类型之转义符
- 字符串类型之操作符
- 字符串类型之操作函数
- 字符串类型之方法
- 字符串类型之格式化

## 2 教学实例：打印三角形

# 字符串的表示

字符串有 2 类共 4 种表示方法

- 由一对单引号或双引号表示，表示单行字符串

---

```
1 "请输入带有符号的温度值："
2 'C'
```

---

- 由一对三单引号或三双引号表示，表示多行字符串

---

```
1 ''' Python
   语言 '''
2
3
4 """ Python
   语言 """
5
```

---

**说明：**多行字符串和注释的区别

**复习：**索引举例:str-4kinds.py

# 字符串: 引号处理

**问题 1:** 如果希望在字符串中包含双引号或单引号呢?

---

```
1 '这里有个双引号 (")'  
2 "这里有个单引号 (')"
```

---

**问题 2:** 如果希望在字符串中既包括单引号又包括双引号呢?

---

```
1 ''' 这里既有单引号 (') 又有双引号 (")'''
```

---

某种程度上解释了为什么 Python 提供 4 种字符串表示方式

范例:str-quotes.py

# 字符串: 高级切片

切片: 返回字符串中一段字符串  $\langle \text{字符串} \rangle [M: N]$   
半开半闭, M 缺失表示从开头, N 缺失表示至结尾

**切片的高级用法**: 使用  $[M: N: K]$  根据步长 K 对字符串切片

---

```
1 "○一二三四五六七八九十"[1:8:2] # 结果是 " 一三五七 "
```

---

```
2 "○一二三四五六七八九十"[::-1] # 结果是 " 十九八七六五四三二一○ "
```

---

举例: str-slice.py str-WeekDayName.py

# 字符串: 转义符

转义符表达特定字符的**本义**

---

1 "这里有个双引号 (\")" # 结果为 这里有个双引号 (")

---

表达一些不可打印的含义

- "\b" 回退
- "\n" 换行 (光标移动到下行首)
- "\r" 回车 (光标移动到本行首)

# 字符串: 转义符

| 转义字符 | 描述                           |
|------|------------------------------|
| \    | (在行尾时) 续行符                   |
| \\   | 反斜杠符号                        |
| \'   | 单引号                          |
| \"   | 双引号                          |
| \a   | 响铃                           |
| \b   | 退格 (Backspace)               |
| \n   | 换行                           |
| \v   | 纵向制表符                        |
| \t   | 横向制表符                        |
| \r   | 回车                           |
| \f   | 换页                           |
| \oyy | 八进制数 yy 代表的字符, 例如: \o12 代表换行 |
| \xyy | 十进制数 yy 代表的字符, 例如: \x0a 代表换行 |

举例: str-trans.py



# 字符串: 操作符

| 操作符及使用            | 描述                                                               |
|-------------------|------------------------------------------------------------------|
| $x + y$           | 连接两个字符串 $x$ 和 $y$                                                |
| $n * x$ 或 $x * n$ | 复制 $n$ 次字符串 $x$                                                  |
| $x \text{ in } s$ | 如果 $x$ 是 $s$ 的子串, 返回 <code>True</code> , 否则返回 <code>False</code> |

举例:     `str-op.py`

# 字符串: 操作函数

| 函数及使用               | 描述                                    |
|---------------------|---------------------------------------|
| <code>len(x)</code> | 返回字符串 <code>x</code> 的长度              |
| <code>str(x)</code> | 返回任意类型 <code>x</code> 所对应的字符串形式       |
| <code>hex(x)</code> | 整数 <code>x</code> 的十六进制小写形式字符串        |
| <code>oct(x)</code> | 整数 <code>x</code> 的八进制小写形式字符串         |
| <code>chr(u)</code> | <code>x</code> 为Unicode编码, 返回其对应的字符   |
| <code>ord(x)</code> | <code>x</code> 为字符, 返回其对应的 Unicode 编码 |

Python 字符串编码方式: 每个字符都是 Unicode 编码字符<sup>1</sup>

举例: `str-func.py`

<sup>1</sup>ASCII 码, Unicode 码的相关内容请网络检索

# 字符串: 方法

“方法”特指 `<a>.<b>()` 风格中的函数 `<b>()`

方法本身也是函数, 但与 `<a>` 有关

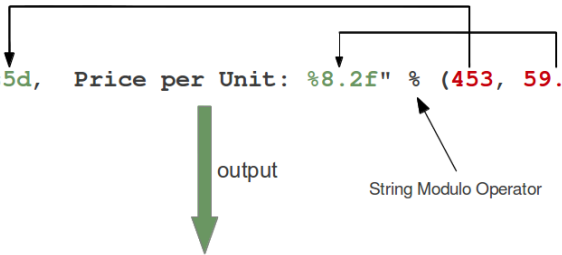
Python 中一切皆对象<sup>2</sup>: 包括数字和字符串, 存在操作方法

| 方法及使用                                      | 描述                                                          |
|--------------------------------------------|-------------------------------------------------------------|
| <code>str.lower()</code>                   | 返回字符串副本, 全部字符小写                                             |
| <code>str.upper()</code>                   | 返回字符串副本, 全部字符大写                                             |
| <code>str.split(sep=None)</code>           | 以 <code>sep</code> 为分割符, 切分成列表                              |
| <code>str.count(sub)</code>                | 返回子串 <code>sub</code> 在 <code>str</code> 中出现的次数             |
| <code>str.replace(old, new)</code>         | 返回字符串副本, <code>old</code> 子串被替换为 <code>new</code>           |
| <code>str.center(width[, fillchar])</code> | 根据宽度 <code>width</code> 居中, <code>fillchar</code> 可选        |
| <code>str.strip(chars)</code>              | 去掉左侧和右侧 <code>chars</code> 中列出的字符                           |
| <code>str.join(iter)</code>                | 在 <code>iter</code> 变量除最后元素外,<br>每个元素后增加一个 <code>str</code> |

# 字符串：格式化

回顾上次课程学习：

```
print("Art: %5d, Price per Unit: %8.2f" % (453, 59.058))
```



output

String Modulo Operator

```
Art:    453, Price per Unit:    59.06
```

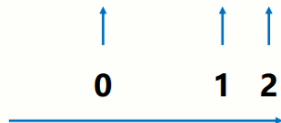
# 字符串：格式化

字符串格式化还可使用`.format()`方法，用法如下：  
< 模板字符串 >.format(< 逗号分隔的参数 >)

`"{ } : 计算机{ } 的CPU占用率为{ } %".format("2018-10-10", "C", 10)`



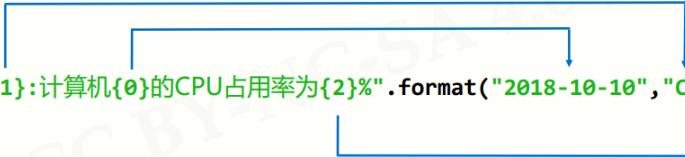
字符串中槽{}的默认顺序



`format()`中参数的顺序

# 字符串：格式化

参数顺序可以自己定义



```
"{1}:计算机{0}的CPU占用率为{2}%" .format("2018-10-10", "C", 10)
```

<https://blog.csdn.net/MessiNine>

# 字符串：格式化

## 槽内部解析

| :        | <填充>          | <对齐>                     | <宽度>         | <,>          | <.精度>                           | <类型>                                            |
|----------|---------------|--------------------------|--------------|--------------|---------------------------------|-------------------------------------------------|
| 引导<br>符号 | 用于填充的<br>单个字符 | < 左对齐<br>> 右对齐<br>^ 居中对齐 | 槽设定的输<br>出宽度 | 数字的千位<br>分隔符 | 浮点数小数<br>精度 或 字<br>符串最大输<br>出长度 | 整数类型<br>b, c, d, o, x, X<br>浮点数类型<br>e, E, f, % |

[https://blog.csdn.net/qq\\_41919999](https://blog.csdn.net/qq_41919999)

## 举例 str-format.py

## 1 字符串类型

- 字符串类型的 4 种表示
- 字符串类型的引号处理
- 字符串类型之切片
- 字符串类型之转义符
- 字符串类型之操作符
- 字符串类型之操作函数
- 字符串类型之方法
- 字符串类型之格式化

## 2 教学实例：打印三角形



# 数学实例：打印三角形

读入一个整数  $N$ ， $N$  是奇数，输出由星号字符组成的等边三角形

```
1 import math
2
3 tmpStr=input("请输入打印三角形的行数")
4 lines=int(tmpStr)
5 max_line_width=2*lines + 1
6
7 for i in range(0,lines):
8     star_len= 2*i + 1
9     line_str=star_len*"*"
10    print(line_str.center(max_line_width))
```

# 数学实例：打印三角形

新增要求：逐行打印

```
1 import math
2 import time
3
4 tmpStr=input("请输入打印三角形的行数")
5 lines=int(tmpStr)
6 max_line_width=2*lines + 1
7
8 for i in range(0,lines):
9     star_len= 2*i + 1
10    line_str=star_len*"*"
11    print(line_str.center(max_line_width))
12    time.sleep(1)
```

# 数学实例：打印三角形

新增要求：打印进度条

```
1 import math
2 import time
3
4 tmpStr=input("请输入打印三角形的行数")
5 lines=int(tmpStr)
6 max_line_width=2*lines + 1
7
8 print('开始绘制')
9 start=time.perf_counter()
10 for i in range(0,lines):
11     star_len= 2*i + 1
12     line_str=star_len*"*"
13     print('\r'+line_str.center(max_line_width))
14
15     # 以下用于更新进度条信息
16     a=(i/lines)*100
17     b=i*"*"
18     c=(lines-i)*"."
19     dur=time.perf_counter()-start
20
21     print("\r{:.0f}%[{}-->{}] {:.2f}s".format(a,b,c,dur), end="")
22     time.sleep(0.5)
23
24 print('\r'+ '绘制结束'.center(max_line_width))
```