

python

The Python logo, consisting of two interlocking snakes, one blue and one yellow, is positioned below the word "python".

```
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")

for i in range(4):
    turtle.circle(40, 80)
    turtle.circle(-40, 80)
    turtle.fd(40)
    turtle.circle(16, 180)
    turtle.fd(40 * 2/3)
```

Python语言程序设计

字符串类型及操作



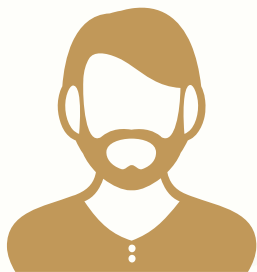
嵩 天
北京理工大学





单元开篇

字符串类型及操作



- 字符串类型的表示
- 字符串操作符
- 字符串处理函数
- 字符串处理方法
- 字符串类型的格式化





字符串类型的表示

字符串

由0个或多个字符组成的有序字符序列

- 字符串由一对单引号或一对双引号表示

"请输入带有符号的温度值:" 或者 'c'

- 字符串是字符的有序序列，可以对其中的字符进行索引

"请" 是 "请输入带有符号的温度值:" 的第0个字符

字符串

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，仅表示单行字符串

"请输入带有符号的温度值：" 或者 'C'

- 由一对三单引号或三双引号表示，可表示多行字符串

''' Python

语言 '''

Q: 老师老师，三引号不是多行注释吗？

Python语言为何提供 2类共4种 字符串表示方式?

字符串

字符串有 2类共4种 表示方法

- 如果希望在字符串中包含双引号或单引号呢？

'这里有个双引号("')' 或者 "这里有个单引号(')'"

- 如果希望在字符串中既包括单引号又包括双引号呢？

''' 这里既有单引号(')又有双引号 (") '''

字符串的序号

正向递增序号 和 反向递减序号



字符串的使用

使用[]获取字符串中一个或多个字符

- **索引：返回字符串中单个字符** **<字符串>[M]**

"请输入带有符号的温度值: "[0] 或者 TempStr[-1]

- **切片：返回字符串中一段字符串** **<字符串>[M: N]**

"请输入带有符号的温度值: "[1:3] 或者 TempStr[0:-1]

字符串切片高级用法

使用[M: N: K]根据步长对字符串切片

- <字符串>[M: N], M缺失表示**至开头**, N缺失表示**至结尾**

"〇一二三四五六七八九十"[:3] 结果是 "〇一二"

- <字符串>[M: N: K], 根据步长K对字符串切片

"〇一二三四五六七八九十"[1:8:2] 结果是 "一三五七"

"〇一二三四五六七八九十"[::-1] 结果是 "十九八七六五四三二一〇"

字符串的特殊字符

转义符 \

- 转义符表达特定字符的本意

"这里有个双引号(\"")" 结果为 这里有个双引号(")

- 转义符形成一些组合，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)



字符串操作符

字符串操作符

由0个或多个字符组成的有序字符序列

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True，否则返回False

字符串操作符

获取星期字符串

- 输入：1-7的整数，表示星期几
- 输出：输入整数对应的星期字符串
- 例如：输入3，输出 星期三

字符串操作符

获取星期字符串

```
#WeekNamePrintV1.py
```

```
weekStr = "星期一星期二星期三星期四星期五星期六星期日"
```

```
weekId = eval(input("请输入星期数字(1-7): "))
```

```
pos = (weekId - 1) * 3
```

```
print(weekStr[pos: pos+3])
```

字符串操作符

获取星期字符串



PY01B13 节用

```
#WeekNamePrintV2.py
```

```
weekStr = "一二三四五六日"
```

```
weekId = eval(input("请输入星期数字(1-7): "))
```

```
print("星期" + weekStr[weekId-1])
```



字符串处理函数

字符串处理函数

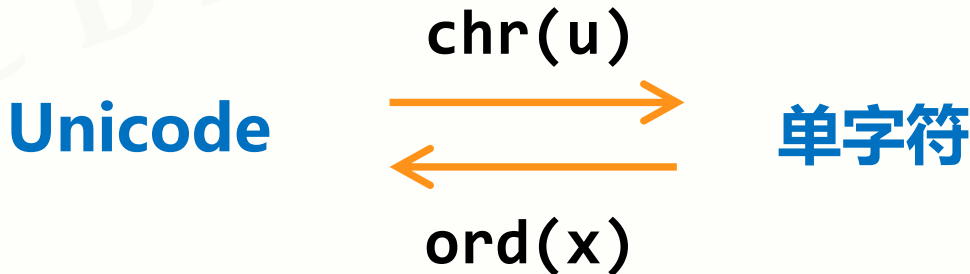
一些以函数形式提供的字符串处理功能

函数及使用	描述
len(x)	长度，返回字符串x的长度 len("一二三456") 结果为 6
str(x)	任意类型x所对应的字符串形式 str(1.23)结果为"1.23" str([1,2])结果为"[1,2]"
hex(x) 或 oct(x)	整数x的十六进制或八进制小写形式字符串 hex(425)结果为"0x1a9" oct(425)结果为"0o651"

字符串处理函数

一些以函数形式提供的字符串处理功能

函数及使用	描述
<code>chr(u)</code>	x为Unicode编码，返回其对应的字符
<code>ord(x)</code>	x为字符，返回其对应的Unicode编码



Unicode编码

Python字符串的编码方式

- 统一字符编码，即覆盖几乎所有字符的编码方式
- 从0到1114111 (0x10FFFF)空间，每个编码对应一个字符
- Python字符串中每个字符都是Unicode编码字符

Unicode编码

一些有趣的例子

```
>>> "1 + 1 = 2 " + chr(10004)
```

```
'1 + 1 = 2 ✓'
```

```
>>> "这个字符𐄀的Unicode值是: " + str(ord("𐄀"))
```

```
'这个字符𐄀的Unicode值是: 9801'
```

```
>>> for i in range(12):
```

```
    print(chr(9800 + i), end="")
```

```
𐄀𐄁𐄂𐄃𐄄𐄅𐄆𐄇𐄈𐄉𐄊𐄋
```



字符串处理方法

字符串处理方法

"方法"在编程中是一个专有名词

- **"方法"特指<a>.()风格中的函数()**
- **方法本身也是函数，但与<a>有关，<a>.()风格使用**
- **字符串或字符串变量是<a>，存在一些可用方法**

字符串处理方法

一些以方法形式提供的字符串处理功能

方法及使用 1/3	描述
<code>str.lower()</code> 或 <code>str.upper()</code>	返回字符串的副本，全部字符小写/大写 <code>"AbCdEfGh".lower()</code> 结果为 <code>"abcdefgh"</code>
<code>str.split(sep=None)</code>	返回一个列表，由str根据sep被分隔的部分组成 <code>"A,B,C".split(",")</code> 结果为 <code>['A', 'B', 'C']</code>
<code>str.count(sub)</code>	返回子串sub在str中出现的次数 <code>"an apple a day".count("a")</code> 结果为 4

字符串处理方法

一些以方法形式提供的字符串处理功能

方法及使用 2/3	描述
<code>str.replace(old, new)</code>	返回字符串str副本，所有old子串被替换为new <code>"python".replace("n","n123.io")</code> 结果为 <code>"python123.io"</code>
<code>str.center(width[,fillchar])</code>	字符串str根据宽度width居中，fillchar可选 <code>"python".center(20,"=")</code> 结果为 <code>'=====python====='</code>

字符串处理方法

一些以方法形式提供的字符串处理功能

方法及使用 3/3	描述
<code>str.strip(chars)</code>	从str中去掉在其左侧和右侧chars中列出的字符 " <code>= python=</code> ". <code>strip(" =np")</code> 结果为 <code>"ytho"</code>
<code>str.join(iter)</code>	在iter变量除最后元素外每个元素后增加一个str " <code>,</code> ". <code>join("12345")</code> 结果为 <code>"1,2,3,4,5"</code> #主要用于字符串分隔等



字符串类型的格式化

字符串类型的格式化

格式化是对字符串进行格式表达的方式

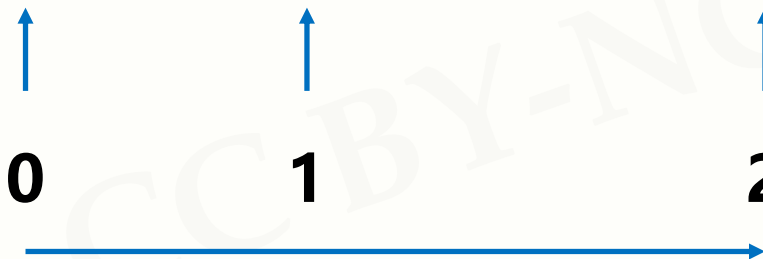
- 字符串格式化使用.format()方法，用法如下：

<模板字符串>.format(<逗号分隔的参数>)

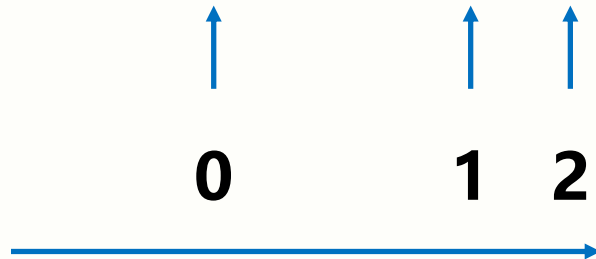
字符串类型的格式化

槽

"{ } : 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

字符串类型的格式化

槽

```
graph TD; A["{1}:计算机{0}的CPU占用率为{2}%"] --> B["2018-10-10"]; A --> C["C"]; A --> D["10"];
```

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

format()方法的格式控制

槽内部对格式化的配置方式

{ <参数序号> : <格式控制标记> }

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

format()方法的格式控制

:	<填充>	<对齐>	<宽度>	<, >	<.精度>	<类型>
引导 符号	用于填充的 单个字符	< 左对齐 > 右对齐 ^ 居中对齐	槽设定的输 出宽度	<pre>>>> "{0:=^20}".format("PYTHON") '=====PYTHON=====' >>> "{0:*>20}".format("BIT") '*****BIT' >>> "{:10}".format("BIT") 'BIT'</pre>		

format()方法的格式控制

:	<填充>	<对齐>	<宽度>	<,>	<.精度>	<类型>
<pre>>>> "{0:,.2f}".format(12345.6789)</pre>				数字的千位 分隔符	浮点数小数 精度或字 符串最大输 出长度	整数类型 b, c, d, o, x, X 浮点数类型 e, E, f, %
<pre>>>> "{0:b},{0:c},{0:d},{0:o},{0:x},{0:X}".format(425)</pre>						
<pre>>>> "{0:e},{0:E},{0:f},{0:%}".format(3.14)</pre>						
<pre>>>> "{0:,.2f},{0:,.2E},{0:,.2f},{0:,.2%}".format(3.14)</pre>						



单元小结

字符串类型及操作

- 正向递增序号、反向递减序号、<字符串>[M:N:K]
- +、*、len()、str()、hex()、oct()、ord()、chr()
- .lower()、.upper()、.split()、.count()、.replace()
- .center()、.strip()、.join() 、.format()格式化



