第五课: Python 字符串格式化方法

魏守科 博士, 教授

目录页 CONTENTS PAGE

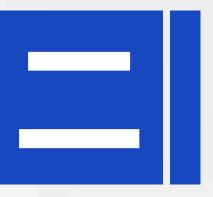
- **简单字符串格式化法**
- %字符串格式化方法
- format()字符串格式化法
- 四 f-string字符串格式化法

简单字符串格式化

• 应用举例

简单的方法

```
名字 = '老李'
性别 = '男'
年龄 = 60
print('我的名字' + 名字 + '.')
print('我的名字',名字,'.')
print('我的名字' + 名字 + 年龄 + '岁.') # error
print('我的名字',名字,年龄,'岁.')
```



%格式化字符串

- %格式化语法
- 应用举例

%格式化语法

- %格式化字符串: Python最早的方法, 它会把字符串中的格式化符%按顺序用后面参数替换,
- %s, %d, %f格式化符:
 - %s: 把后面的值格式化为字符类型
 - %d: 把后面的值格式化为整数类型
 - %f: 把后面的值格式化为浮点类型

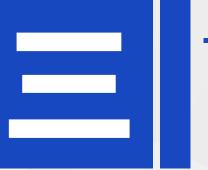
名字 = '老李'

性别 = '男'

年龄 = 60

print("姓名为%s,性别是%s,年龄为%d岁。"%(名字,性别,年龄))

```
x = 1
y = 3
z = x/y
print('%d除以%d的结果为%f.'%(x,y,z))
print('%d除以%d的结果 '\
 '保留小数点后一位数为%.1f.'%(x,y,z))
print('%d除以%d的结果 '\
 '保留小数点后两位数为%.2f.'%(x,y,z))
print('%d除以%d的结果 '\
 '保留小数点后三位数为%.3f.'%(x,y,z))
```



format()格式化字符串

- format()格式化
- 应用举例

format()方法

- %虽然强大,但用起来难免有些麻烦,代码也不是特别美观,
- 在python 2.5 之后,提供了更加优雅的str.format()方法。
 - ✔ 用{}做占位符,
 - ✔ 不分插入的类型

名字 = '老李'

性别 = '男'

年龄 = 60

print("姓名为{ }, 性别是{ }, 年龄为{ }岁。" .format(名字, 性别, 年龄))

```
x = 1
y = 3
z = x/y
print('{ }除以{ }的结果为{ }。'.format(x,y,z))
print('{ }除以{ }的结果 '\
 '保留小数点后一位数为{:.1f }。'.format(x,y,z))
print('{ }除以{ }的结果 '\
 '保留小数点后一位数为{:.2f }。'.format(x,y,z))
print('{ }除以{ }的结果 '\
 '保留小数点后一位数为{:.3f }。'.format(x,y,z))
```



f-string格式化字符串

- f-string法
- 应用举例

f-string法

f-string是2015年python 3.6 新添加的一种字符串格式化方法f-string是一个文字字符串,前缀为'f',其中包含大括号内的表达式。表达式会将大括号中的内容替换为其值。

名字 = '老李'

性别 = '男'

年龄 = 60

print(f "姓名为{名字},性别是{性别},年龄为{年龄}岁。")

```
x = 1
y = 3
z = x/y
print(f'{ x}除以{ x}的结果为{z}。')
print(f'{ x}除以{ x}的结果 '\
 '保留小数点后一位数为{z:.1f}。')
print(f'print('{ x}除以{ y}的结果 '\
 '保留小数点后一位数为{z:.2f}。')
print(f'{ x}除以{ y}的结果 '\
 '保留小数点后一位数为{z:.3f}。')
```

总结



%字符串格式化方法,%s,%d,%f等, 繁琐,不太美观

