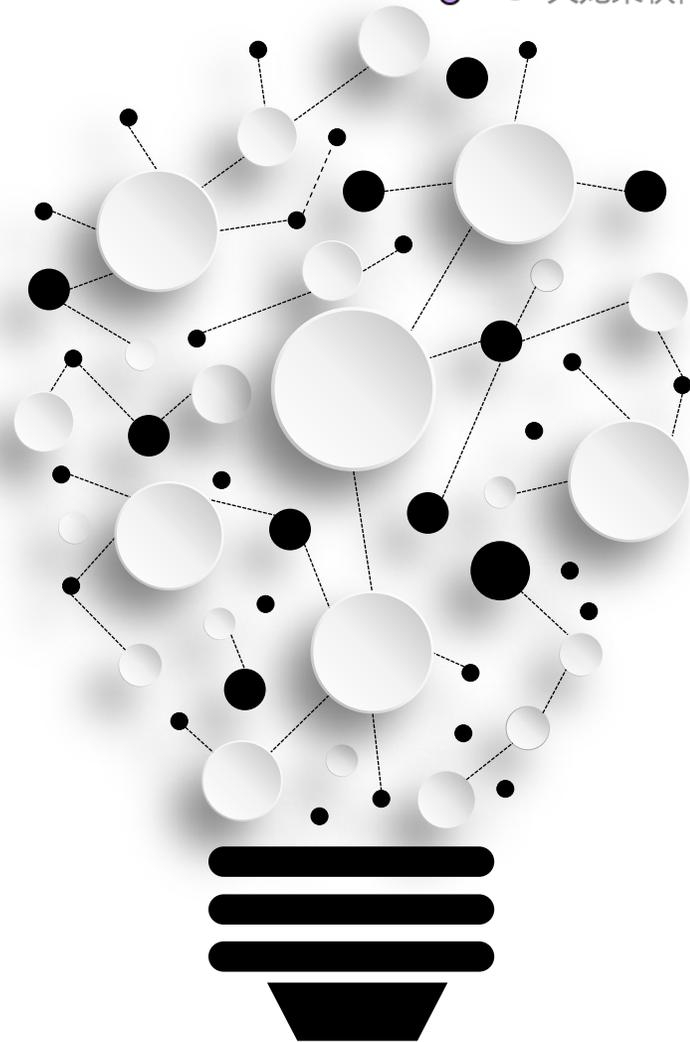
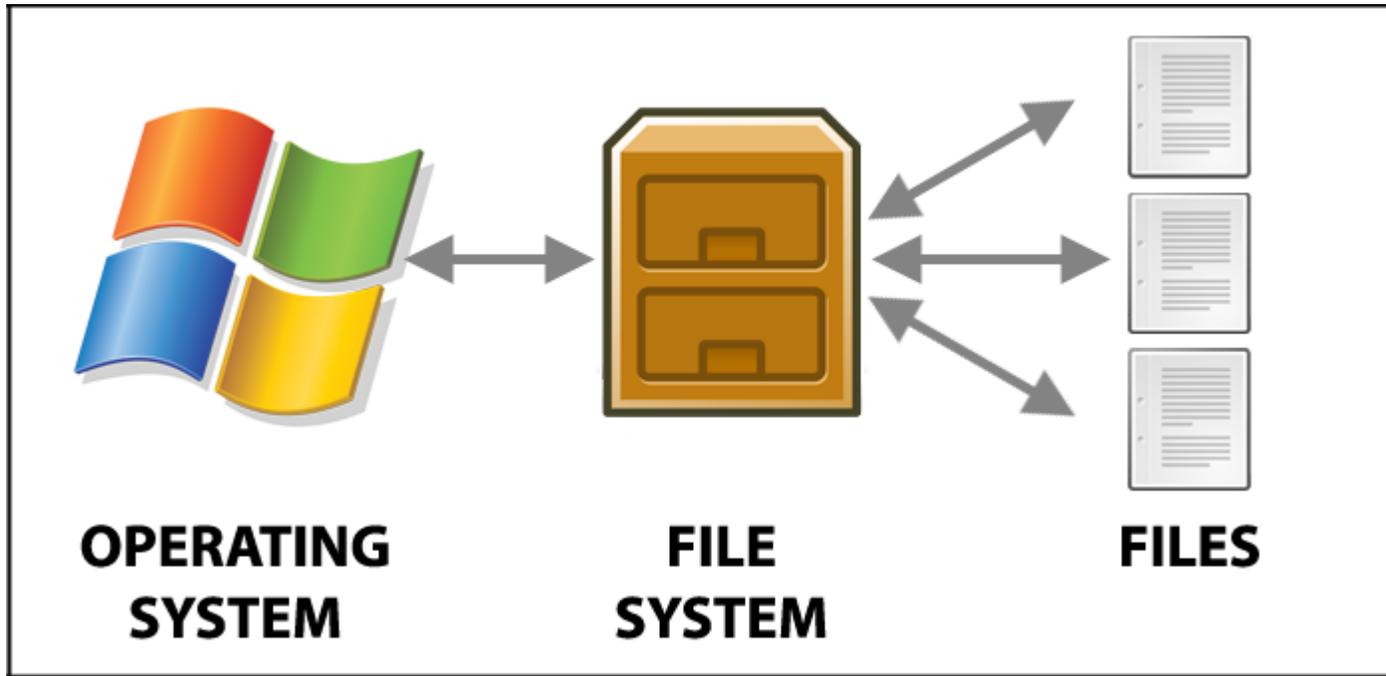


Python3 程序设计

# 文件

动手学 Python, 实践出真知!

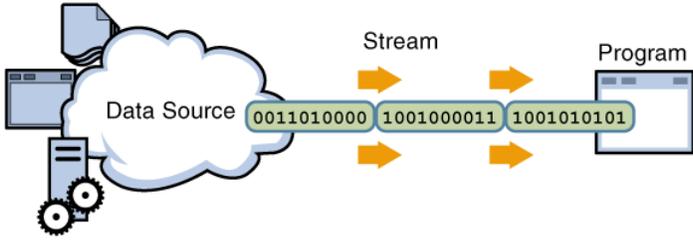




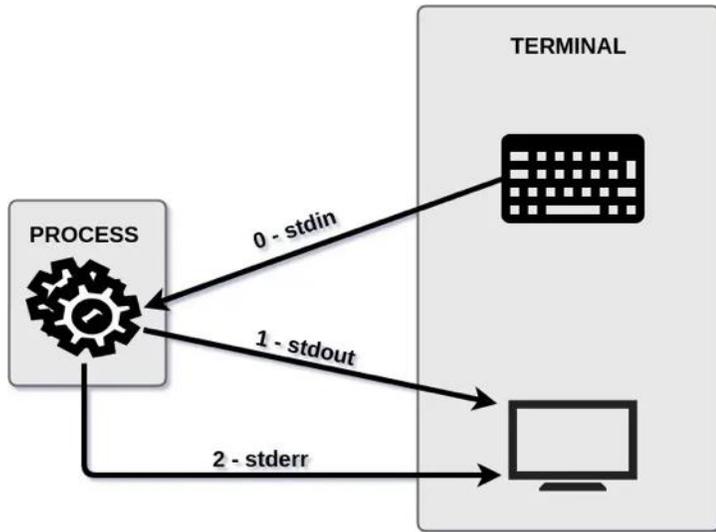
文件

File

# 一切都是文件



Linux/UNIX 将**外围设备抽象成文件**进行统一处理。键盘、显示器、硬盘、以太网卡、目录都是文件。文件本质上就是一串数据流。通过内容区分文件类型是留给应用程序的任务。



执行一个shell命令行时通常会自动打开三个标准文件：标准输入文件（stdin），通常对应终端的键盘；标准输出文件（stdout）和标准错误输出文件（stderr），这两个文件都对应终端的屏幕。

进程将从标准输入文件中得到输入数据，将正常输出数据输出到标准输出文件，而将错误信息送到标准错误文件中。

# 什么是文本文件?

文本文件一般指只有字符原生编码构成的二进制计算机文件，与富文本相比，其**不包含**字样样式的**控制元素**，能够被最简单的文本编辑器直接读取。

由于结构简单，文本文件被广泛用于记录信息。它能够避免其它文件格式遇到的一些问题。此外，当文本文件中的部分信息出现错误时，往往能够比较容易的从错误中恢复出来，并继续处理其余的内容。

文本文件的缺点是它的熵往往较低，也就是说，本可以用更小的存储空间记录这些信息。



# 文本文件：单一编码

**ASCII** (American Standard Code for Information Interchange, 美国信息交换标准代码) 是基于拉丁字母的一套电脑编码系统。它主要用于显示现代英语，而其扩展版本EASCII则可以部分支持其他西欧语言，并等同于国际标准ISO/IEC 646。由于万维网使得ASCII广为通用，直到2007年12月，逐渐被Unicode取代。

**UTF-8** (8-bit Unicode Transformation Format) 是一种针对Unicode的可变长度字节编码，也是一种前缀码。它可以用来表示Unicode标准中的任何字节，且其编码中的第一个字节仍与ASCII兼容，这使得原来处理ASCII字节的软件无须或只须做少部分修改，即可继续使用。因此，它逐渐成为电子邮件、网页及其他存储或发送文字的应用中，优先采用的编码。

**GBK编码**，是在GB2312-80标准基础上的内码扩展规范，使用了双字节编码方案，其编码范围从8140至FEFE (剔除xx7F)，共23940个码位，共收录了21003个汉字，完全兼容GB2312-80标准，支持国际标准ISO/IEC10646-1和国家标准GB13000-1中的全部中日韩汉字，并包含了BIG5编码中的所有汉字。GBK编码方案于1995年10月制定，1995年12月正式发布，目前中文版的WIN95、WIN98、WINDOWS NT以及WINDOWS 2000、WINDOWS XP、WIN 7等都支持GBK编码方案。

# Python的文本文件读取方法

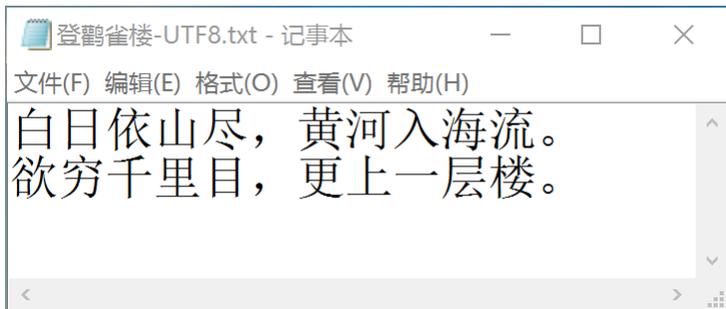
方 法	含 义
<code>&lt;file&gt;.readall()</code>	读入整个文件内容，返回一个字符串或字节流
<code>&lt;file&gt;.read(size=-1)</code>	从文件中读入整个文件内容，如果给出参数，读入前size长度的字符串或字节流
<code>&lt;file&gt;.readline(size=-1)</code>	从文件中读入一行内容，如果给出参数，读入该行前size长度的字符串或字节流
<code>&lt;file&gt;.readlines(hint=-1)</code>	从文件中读入所有行，以每行为元素形成一个列表，如果给出参数，读入hint行

# 读取文件：一次性读取



```
f = open('../data/登鹳雀楼-UTF8.txt')
text = f.read()
print(text)
f.close()
```

```
text = open('../data/登鹳雀楼-UTF8.txt').read()
```



```
In [16]: text = open('登鹳雀楼-UTF8.txt').read()
          print(text)
```

白日依山尽，黄河入海流。

欲穷千里目，更上一层楼。

空行

不同操作系统下的回车和换行有所不同

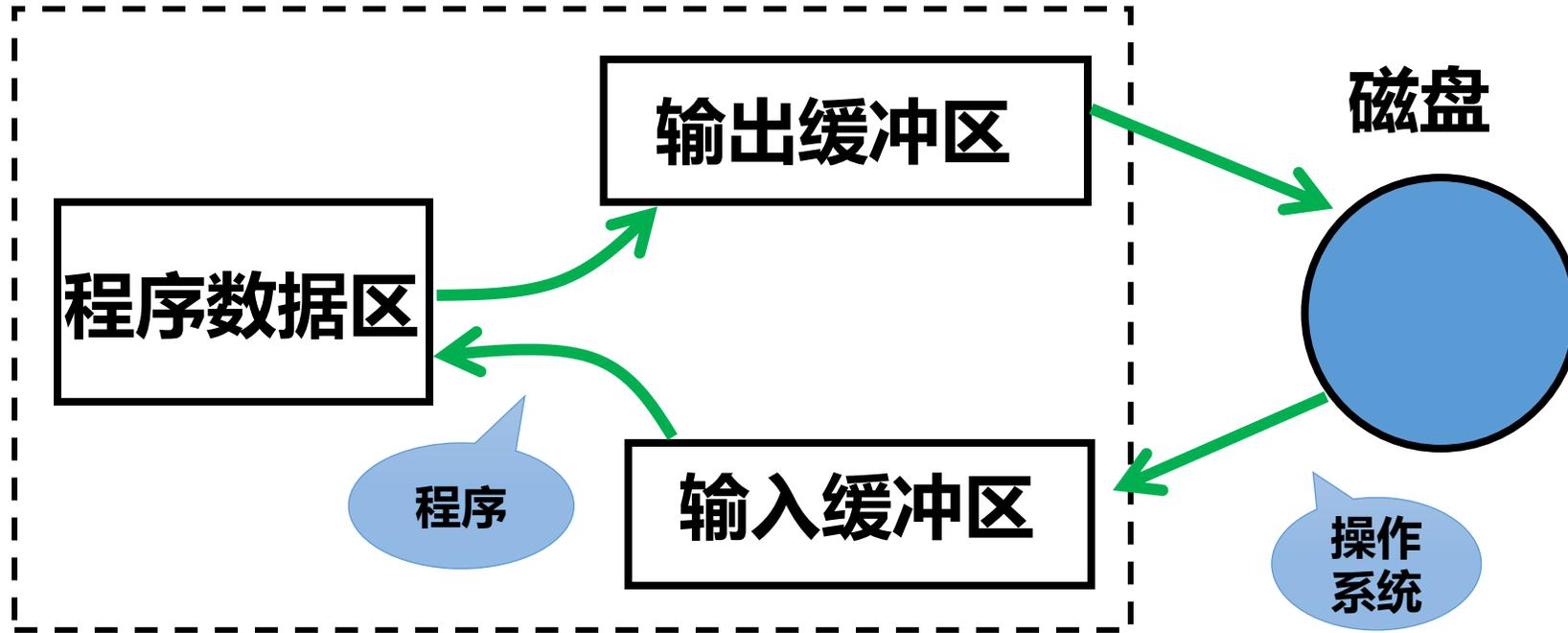
# 读取文件：按行读取

```
f = open('../data/登鹤雀楼-UTF8.txt')
for line in f:
    if (len(line.strip())==0): continue      # 跳过空行
    print(line.strip())                    # strip方法用于去除两端的空白
f.close()
```

# 转换文件格式

```
f1 = open('../data/登鹤雀楼-UTF8.txt')
f2 = open('../data/登鹤雀楼-GBK.txt', mode='w',
encoding='GBK', errors="ignore")
text = f1.read()
f2.write(text)
f1.close()
f2.close()
```

# 文件缓冲区



- 由于CPU 与 I/O 设备间速度不匹配。为了缓和 CPU 与 I/O 设备之间速度不匹配矛盾。
- 文件缓冲区是用以暂时存放读写期间的文件数据而在内存区预留的一定空间。使用文件缓冲区可减少读取硬盘的次数。



- 良好的编程习惯要求在重新赋另一个文件对象前关闭这个文件
- 如果你不显式地关闭文件,那么你可能丢失输出缓冲区的数据

# 使用with-as语句

```
with open('../data/登鹤雀楼-UTF8.txt') as f:  
    text = f.read()  
    print(text)
```

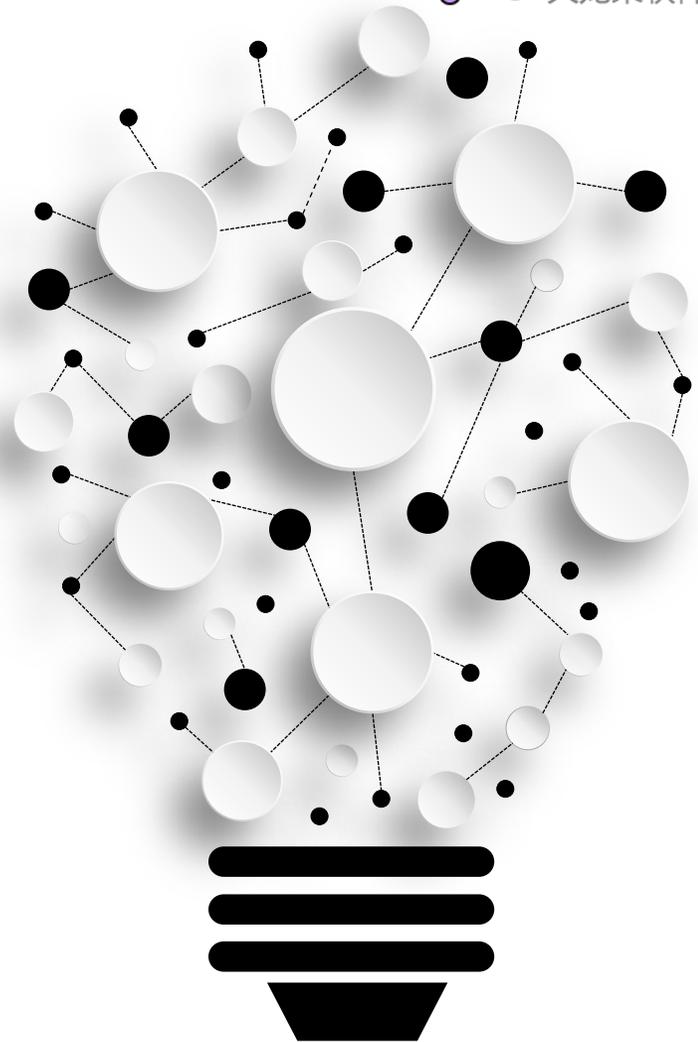
- 写文件时，操作系统往往不会立刻把数据写入磁盘，而是放到内存缓存起来，空闲的时候再慢慢写入
- 只有调用close()方法时，操作系统才保证把没有写入的数据全部写入磁盘
- 忘记调用close()的后果是数据可能只写了一部分到磁盘，剩下的丢失了
- 使用with-as语句：用来处理需要事先设置、事后做清理的工作

# Python3 程序设计

## 3种方法

# 处理CSV文件

动手学 Python, 实践出真知!



# 表格数据和CSV

## 表格数据

- ① 使用二维数据，也称表格数据，由关联关系数据构成，采用表格方式组织
- ② 存储二维数据的常见文件格式有Excel的xlsx和文本形式的CSV字符串

## CSV

- ① Comma-Separated Values的简写，以逗号分隔值
- ② 以纯文本的形式存储表格数据（数字和文本），可以用Excel查看

## 多种分隔符

- ① 每条记录被分隔符分隔为字段（典型分隔符有逗号、分号或制表符）
- ② 每条记录都有同样的字段序列

# 计算招商银行收盘价的平均价

日期	股票代码	名称	收盘价	最高价	最低价	开盘价	成交金额
2018-11-05	'600036	招商银行	30.00	30.23	29.71	29.85	1283209565
2018-11-02	'600036	招商银行	30.33	30.45	29.41	29.90	3665161770
2018-11-01	'600036	招商银行	29.05	29.65	28.66	29.45	1983324170

# 三种方法处理CSV文件



Pandas



readline



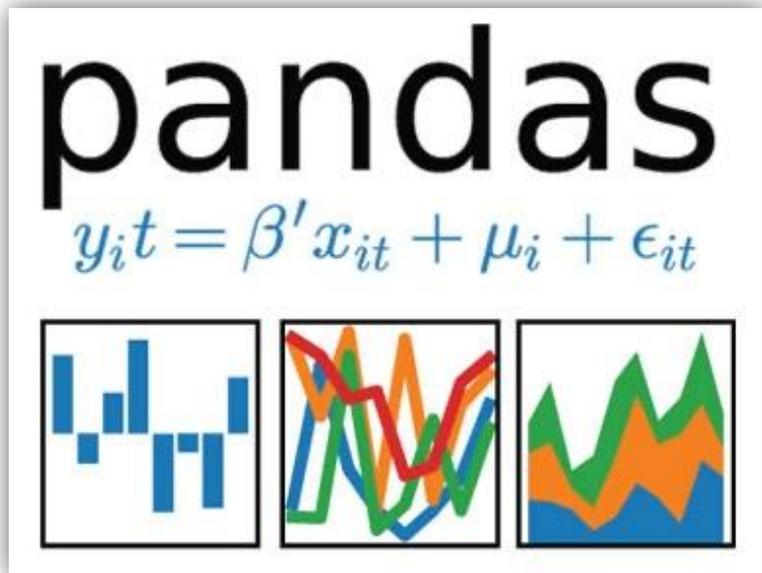
CSV库

# 方法1：使用Pandas库

```
import pandas as pd

df = pd.read_csv('../data/600036_UTF8.csv')
print(df['收盘价'].mean())          # 29.15
```

- 选择适当的工具，会事半功倍
- Python第三方库pandas最初就是为了处理金融数据而生的，其核心数据结构DataFrame用于处理二维数据
- 方法2和方法3只是作为备用



# 方法2：使用文件对象的readline方法

```
f = open('../data/600036_UTF8.csv')
f.readline()      # 跳过表头
tot = 0           # 保存“最高价”这一列的总和
n = 0             # 统计行数
for line in f:
    t = line.strip().split(',') # 切分字符串为列表
    tot += float(t[3])          # 字符串转换为浮点数后才能累加
    n += 1
print('%.2f' %(tot/n))        # 29.15
```

# 方法3：使用内置库csv

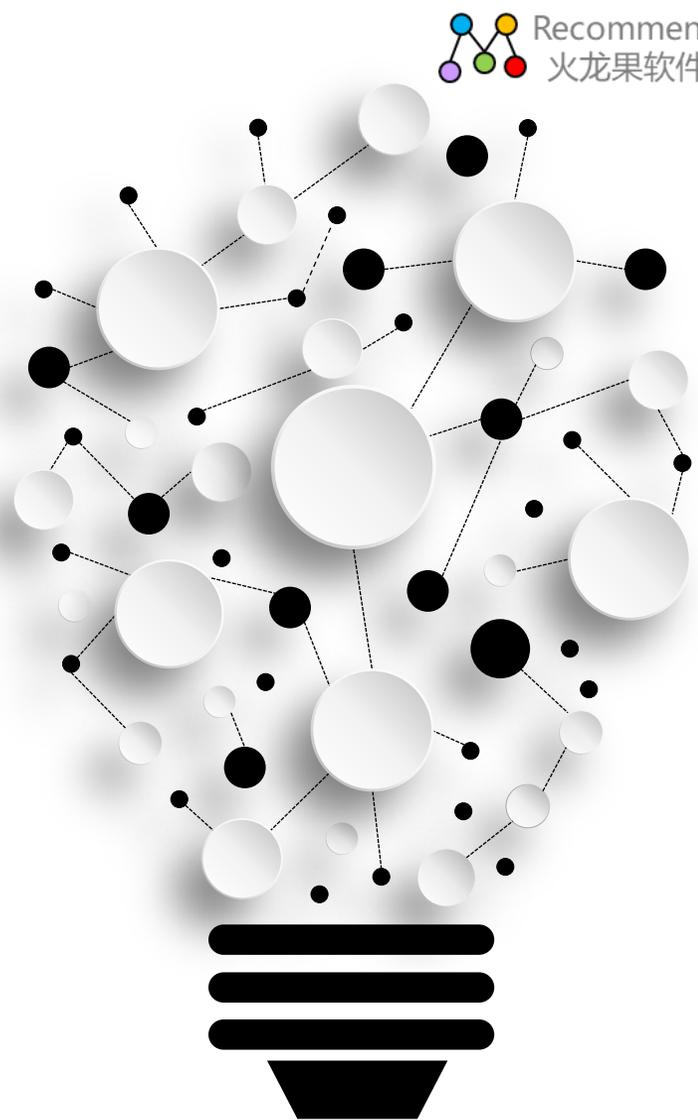
```
import csv

tot = 0 # 保存“最高价”这一列的总和
n = 0 # 统计行数
f = open('../data/600036_UTF8.csv')
reader = csv.DictReader(f)
for row in reader:
    tot += float(row['收盘价']) # 字符串转换为浮点数后才能累加
    n += 1
print('%.2f' %(tot/n)) # 29.15
```

实战

# 批量压缩图片

动手学 Python3, 实践出真知!





# 找到所有符合要求的图片

```
from glob import glob
from PIL import Image
import os

source_dir = 'image'
filenames = glob('{}/*'.format(source_dir))
```

# 输出图片信息

```
for filename in filenames:
    with Image.open(filename) as im:
        width, height = im.size
        print(filename, width, height, os.path.getsize(filename))
```

# 创建目录

```
target_dir = 'output'  
if not os.path.exists(target_dir):  
    os.makedirs(target_dir)
```

# 批量压缩图片

```
for filename in filenames:
    filesize = os.path.getsize(filename)
    if filesize >= threshold:
        print(filename)
        with Image.open(filename) as im:
            width, height = im.size
            new_width = 1024
            new_height = int(new_width * height * 1.0 / width)
            resized_im = im.resize((new_width, new_height))
            output_filename = filename.replace(source_dir, target_dir)
            resized_im.save(output_filename)
```