



# Python 程序设计课程实验报告

实验 2 设计一个程序遍历至少 2000 个百度百科人物知识

姓 名：李佩石

学 号：22920212204129

院 系：信息学院 人工智能系

专 业：人工智能

年 级：2021 级

指导教师：曹冬林

2023 年 4 月 21 日

## 一.实验任务与基本要求

任务:

要求,爬取至少 2000 个百度百科人物信息,并将信息保存在文件中

基本要求:

考虑到之后便于查询与读取,建议保存在 csv 文件或者数据库中

## 二.主要知识点与难点

主要知识点:

1. 网络爬虫的基本知识
2. 利用 beautifulsoup 库,lxml 库进行网页解析获得文本
3. 利用正则表达式对文本进行格式处理与适当排版
4. Csv 文件的写入与读取

难点问题:

1. 大批量不间断爬取数据,导致网站采取反爬措施,如封禁 ip,重定向网址等等,如何反反爬
2. Csv 单元格容量有限,如何设计储存策略与数据结构
3. 如何优化整体速度

## 三.整体程序设计

整体思路:

通过输入一个人物姓名,来爬取该人物的关系,将关系处理后储存在一个队列中,在下一个循环中出队并爬取出队元素的关系,再入队,对当

前人物处理完后,将其特征保存在另一个队列,以此类推,遍历人物关系,相当于以一个人物为中心生成关系网

如果另一个队列总数没有达到指定数目,如 2000,程序暂停输出提示,人物不足,要求用户再次输入一个姓名,并且尽量与原来的人物关系较远,以不能轻易的重复,如果输入的姓名已被爬取,则再次暂停,直到输入一个新人,并且爬到指定数目为止

基于任务,设计数据结构,划分功能模块

1. 设计 **Person** 类,储存几项关键信息,便于后续依据此来构造爬取

```
# 储存结构体
class Person:
    id = ''
    name = ''
    count = 0

    def __init__(self, s1='', s2='', n=0):
        self.id = s1
        self.name = s2
        self.count = n
```

因为分析目标网址可以发现,不同人物网址的不同之处在于 **id** 与姓名不同,所以记录这两项关键信息便于后续构造网址,

**Count** 记录同名人物个数,便于后续区分和管理与该名相同的条目

考虑到字典占用内存大且不好管理增删类成员,采用类

## 2. 关于工作队列与储存队列的定义

```
# 全局变量设置
# 用列表行使队列操作
workQ = [] # 进行广度遍历的工作队列 储存 id 与姓名
saveQ = [] # 存储已经完成爬取的人物 id 与姓名与同名人物次序
```

作用见注释,同时,saveQ 起到了 url 管理器的作用,这是一般爬虫的必要结构,便于过滤重复(非重名)的人物条目,元素均为 Person 类

## 3. 预计设计以下函数方法

```
37 # 网页 用urllib或requests爬取
38 > def getHtml(url): ...
50
51 # 关系 通过request抓包
52 > def getRelation(id, name, url): ...
85
86
87 # 基本信息
88 > def getBasic(html): ...
106
107
108 # 介绍
109 > def getIntro(html): ...
123
124
125 # 经历
126 > def getExperence(intro, html): ...
141
142
143 # 获得初始人物id,注意 整个程序中仅执行一次,仅根据初始人物数量决定
144 > def spiderGetFirstId(content): ...
157
158
159 # 根据id,姓名构造网址
160 > def urlConvert(id, name): ...
```

```

164
165 # 对输入姓名进行归一化处理,以满足自定义数据结构
166 > def initializeInputName(content): ...
170
171
172 # 执行四项信息爬取
173 > def spiderBaidubaike(id, name): ...
186
187
188 # 写入
189 > def csvWriteInfo(id, name, count, relation, basic, intro, experience)
210
211
212 # 中心人物的关系提取处理后装入工作队
213 > def updateWorkQ(relation): ...
230
231
232 # 执行爬取,返回多条信息,写入,更新工作队
233 > def spiderSearchWrite(ele): ...

```

#### 4. 确定创建 csv 文件并写入 csv 列标题

```

246 # 四条信息 关系 基本信息 简介 经历 , '基本信息', '简介', '经历'
247
248 # csv header
249 fieldnames = ['id', 'name', '同名次序', '关系', '基本信息', '简介']
250 filename = 'result_data.csv'
251 # 写入列标题
252 > with open(filename, 'w+', encoding='utf-8-sig', newline='') as f: ...
255

```

### 四.中间细节问题及解决

#### 1. 关系内容的储存格式

由于初始提取的人物关系,包含关系名,姓名,id,三个部分,则考虑到姓名的连接符与关系名的冒号后,将单词条拼接为格式如好友:

李白@1043

并在末尾加上'\n'换行符,既可以排版又可以分隔

## 2. Csv 单个单元格不能储存过多数据

则考虑,将履历信息,单独储存在 txt 文件中,每一个人物,就一个 txt,注意在代码文件夹下建立新文件夹,所有履历信息均存储于此,另外,甚至可以考虑,将除了关系以外的信息都在 txt 中,因为考虑到实验 3 实际上只需要关系,最多在需要输出特殊信息时,查找特定文件然后输出即可,总之一些信息是不必存在 csv 中的

## 3. 优化时间

这里主要是对 workQ 的限制,实验中发现,workQ 往往是 saveQ 的三到五倍,而 workQ 实际上最后远远大于指定的数目,则,在有特定数目的要求下,可以对 workQ 进行限制,

即,如果 workQ 达到了指定数目的 2-3 倍,则停止入队,仅仅出队并对出队元素进行处理,不再入队,直到出队导致 workQ 数量下降不满足这一条件

如此可以减少无用条目的录入,限制 workQ 的长度,如此可以限制内存与时间

## 4. 关于反反爬

这里有多项措施,此次采用的有:

(1) 增加 time.sleep() 函数在每次爬取之前与之后,随机生成延迟时间,以减小请求频率

(2)再请求函数里,增加禁止重定向,避免网站自动将请求转到无用网址中

通过查询资料,了解的可以尝试应用的方法,之后尝试:

- (1)充分利用请求函数,将其默认参数均主动设置有效参数,而不是默认值
- (2)使用动态生成 ip,爬取一定数目即更换
- (3)在请求头中添加 **cookie** 等更多信息,模拟浏览器行为更充分,关于 **cookie** 的获取,一可以直接查看复制粘贴,一可以爬取 **cookie** 并保存为 **session**,添加到请求头中,实现自动更新,这是目前看来最稳妥,生存周期最长的方案
- (4)更换爬取方式,采用 **selenium** 或 **pypeteer** 库进行爬取,这两种爬取方式均会打开浏览器,模仿电脑操作,模拟效果更好,而且更利于直接获得全部动态元素代码,不必重新爬取数据包

## 五.实验总结

1. 本次实验,建立在实验 1 之上,但是对实验 1 进行了大幅度优化与结构设计,使得功能模块化,清晰可读,便于实验 2 目的的实现
2. 在整体程序的设计与实现过程中,深刻感受到面向对象程序设计的优势与简便,模块化思维贯穿始终
3. 对各种异常的判断,问题的处理,任务的达成与错误的预防和提示

有充分的考虑,加强了对程序现实交互过程的思考,对不确定性的预防与分类

4. 体会到合理的数据结构设计与储存方案设计会极大简化程序的开发编写
5. 再次深入接触网页与网络,对 html 更为熟悉,理解动态网页生成机制,知道了并查询资料了解数据包的传输
6. 学习掌握初步的几种反反爬措施,并且明白,反爬措施是及时更新的,这就导致反反爬措施不可能一劳永逸,爬虫程序要及时的维护与更新,才能有更长的使用寿命

## 六.附录：完整程序代码与运行结果

```
import re
import csv
import copy
import urllib
import requests
from lxml import etree
from urllib import parse
from urllib import request
from bs4 import BeautifulSoup

import time
import random

# time 功能
# time.sleep(random.random()*2)
n = 0

# 储存结构体
```



```

class Person:
    id = ''
    name = ''
    count = 0

    def __init__(self, s1='', s2='', n=0):
        self.id = s1
        self.name = s2
        self.count = n

# 全局变量设置
# 用列表行使队列操作
workQ = [] # 进行广度遍历的工作队列 储存 id 与姓名
saveQ = [] # 存储已经完成爬取的人物 id 与姓名与同名人物次序

# 网页 用 urllib 或 requests 爬取
def getHtml(url):
    # 延时
    time.sleep(random.random() * n)

    # 用 request 库 爬取网页
    # 重构请求头
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:6.0) Gecko/20100101 Firefox/6.0'}

    req = requests.get(url)
    req.encoding = "utf-8"
    html = req.text
    return html

# 关系 通过 request 抓包
def getRelation(id, name, url):
    # 延时
    time.sleep(random.random() * n)
    # 爬取关系 b 抓包

    headers = {
        'User-Agent ': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) '
        'AppleWebKit/537.36 (KHTML, like Gecko) '
        'Chrome/111.0.0.0 Safari/537.36 Edg',
        # 必须加 cookie, 虽然不知道咋加
        'Cookie': '',
        'Accep': '',
    }

```

```

        "Referer": url
    }
    # 将 API 链接的查询字符串传给 Params 参数
    params = {
        'lemmaId': id,
        'lemmaTitle': name
    }

    # 暂时避免不了 request 爬取抓包
    res =
requests.get('https://baike.baidu.com/starmap/api/gethumanrelationcard'
, headers=headers, params=params,
              allow_redirects=False)
    # 返回结果力 JSQN 格式,调用 json.()方法解析
    items = res.json()
    b = []
    # 拼接成如'好友:李白@1043\n好友:苏涣@10652\n' 标准形式
    for x in items['list']:
        s = x['relationName'] + ':' + x['lemmaTitle'] + '@' +
str(x['lemmaId']) + '\n'
        b.append(s)
    b = "".join(b)

    return b

# 基本信息
def getBasic(html):
    # 解析基本信息 a
    soup = BeautifulSoup(html, features="html.parser")
    company_items = soup.find_all("div", class_="basic-info J-basic-info
cmn-clearfix")

    a = []
    for i in company_items:
        x = i.text.strip()
        a.append(x)

    a = "".join(a)
    a = "".join(a.split('\xa0'))
    a = "\n".join(a.split('\n\n'))
    a = re.sub(u"\\[.*?]", "", a)
    a = "\n".join(a.split('\n\n'))
    a = "\n".join(a.split('\n\n'))

```

```

        return a

# 介绍
def getIntro(html):
    # 解析介绍 c
    # 构造 _Element 对象
    html1 = etree.HTML(html)
    # 使用 xpath 匹配数据, 得到匹配字符串列表
    sen_list = html1.xpath(
        '//*[@class="lemma-summary" or
contains(@class,"lemmaWgt-lemmaSummary")]//text()')
    # 过滤数据, 去掉空白
    sen_list_after_filter = [item.strip('\n') for item in sen_list]
    # 将字符串列表连成字符串并返回
    c = ''.join(sen_list_after_filter)
    c = re.sub(u"\\[.*?]", "", c)

    return c

# 经历
def getExpearence(intro, html):
    # 解析生平 d

    # 构造 _Element 对象
    html1 = etree.HTML(html)
    sen_list = html1.xpath('//*[@class="para"]//text()')
    # 过滤数据, 去掉空白
    sen_list_after_filter = [item.strip('\n') for item in sen_list]
    # 将字符串列表连成字符串并返回
    d = ''.join(sen_list_after_filter)
    d = re.sub(u"\\[.*?]", "", d)
    # 由于履历中有介绍重复一部分,将这一部分正则除去
    d = d.replace(intro, '', 1)

    return d

# 获得初始人物 id,注意 整个程序中仅执行一次,仅根据初始人物数量决定
def spiderGetFirstId(content):
    url = 'https://baike.baidu.com/item/' + parse.quote(content)

    # 获取目标网址所有信息
    demo = getHtml(url) # 定义所有信息的文本

```

```

soup = BeautifulSoup(demo, 'html.parser') # BeautifulSoup 中的方法

a = soup.find('link', rel="alternate", hreflang="x-default")
a = a.get('href')
a = a.split('/')[1] # 获得 lemmaid
# time.sleep(random.random() * n)
# input("暂停")print(demo)
return a

# 根据 id,姓名构造网址
def urlConvert(id, name):
    url = 'https://baike.baidu.com/item/' + parse.quote(name) + '/' + id
    return url

# 对输入姓名进行归一化处理,以满足自定义数据结构
def intializeInputName(content):
    id = spiderGetFirstId(content)
    name = content
    return Person(id, name, 0)

# 执行四项信息爬取
def spiderBaidubaike(id, name):
    url = urlConvert(id, name)
    html = getHtml(url)
    relation = getRelation(id, name, url)
    basic = getBasic(html)
    intro = getIntro(html)
    experience = getExperience(intro, html)

    # print(relation)
    # print(basic)
    # print(intro)
    # print(experience)
    return relation, basic, intro, experience

# 写入
def csvWriteInfo(id, name, count, relation, basic, intro, experience):
    order = str(total).zfill(6) # 序号
    row = [{
        'id': id,
        'name': name,

```

```

        '同名次序': count,
        '关系': relation,
        '基本信息': basic,
        '简介': intro,
        # '经历': experience

    }]

    # 按照列标题写入
    with open(filename, 'a+', encoding='utf-8-sig', newline='') as f:
        writer = csv.DictWriter(f, fieldnames=fieldnames)
        writer.writerow(row)

    # 将经历单独写入新文件 txt,按次序对应命名

    with open('experience\\' + order + name + '@' + id + '.txt', 'w+',
encoding='utf-8-sig', newline='') as f:
        f.write(experience)

# 中心人物的关系提取处理后装入工作队
def updateWorkQ(relation):
    # 以下处理将关系字符串分为关系名,名字,id 三个一组的列表
    b = re.split('@|:|\n', relation)
    i = 0
    name = []
    id = []
    # 首位为 1 计数下标,模 3 为 2 即为名字,整除即为 id, 分别加入相应列表
    for t in b:
        i = i + 1
        if i % 3 == 2:
            name.append(t)
        elif i % 3 == 0:
            id.append(t)
            n = i // 3 - 1
            # 每处理 3 个,取两个列表的同下标元素为一组,初始化个人信息,入工作队
            a = Person(id[n], name[n])
            workQ.append(a)

# 执行爬取,返回多条信息,写入,更新工作队
def spiderSearchWrite(ele):
    # 获得信息 ,basic,intro,experience ,basic, intro,
experience='', '', ''
    relation, basic, intro, experience = spiderBaidubaike(ele.id,
ele.name)

```

```

    # 写入文件
    csvWriteInfo(ele.id, ele.name, ele.count, relation, basic, intro,
experience)
    # 更新工作队,数目大于 N 个,不再入队,减少损耗,但如果要遍历所有人物,还是要放
开跑
    if len(workQ) <= 2 * N:
        updateWorkQ(relation)
    else:
        print("总数达到,不再入队")

# 四条信息 关系 基本信息 简介 经历 , '基本信息', '简介', '经历'

# csv header
fieldnames = ['id', 'name', '同名次序', '关系', '基本信息', '简介']
filename = 'result_data.csv'
# 写入列标题
with open(filename, 'w+', encoding='utf-8-sig', newline='') as f:
    writer = csv.DictWriter(f, fieldnames=fieldnames)
    writer.writeheader()

total = 0
N = 3000
flag1 = 0
flag2 = 0
k = 0
# 可能不断输入姓名
print("请输入人物姓名:", end='')
start_time = time.time()
while 1:
    content = input() # '成龙'

    ele = initializeInputName(content)
    workQ.append(ele)

    while 1:
        # 跳出条件
        # 完成爬取数目,总程序结束
        if total == N:
            flag1 = 1
            break
        # 工作队空了,终止本层循环,跳出,并提示再次输入,并且尽可能与第一个人物关
系远一些
        elif workQ == []:

```

```

        flag1 = 2
        print("目前条目数不足,请再次输入人物,尽可能与之前输入人物关系较
远:", end='')
        break

# 从工作队取出一个元素,查重,计数并储存
ele = copy.deepcopy(workQ.pop(0))

# 以下为对出队元素进行检查,满足要求则进行检索爬取
# 确定该名为第几个该名人物(为 1 则无重名,其为第一个)
count = 1
flag2 = 0
for a in saveQ:
    if a.name == ele.name:
        if a.id != ele.id:
            count = count + 1
        else: # 当前人信息已经在储存队中
            flag2 = 1
# 当前人在储存队中,且之前工作队为空,即输入的为工作队仅有的元素,原输入
不符合要求,此时要求重新输入
if flag1 == 2 and flag2 == 1:
    print("该条目已爬取,请另外输入,要求同上:", end='')
    break
# 在工作队非空的情况下,条目重复,直接跳过该轮循环
elif flag2 == 1:
    continue
# 重新输入后,新条目非重复,则置 flag1 为 0,避免影响不同分支
elif flag1 == 2 and flag2 == 0:
    flag1 = 0
# 若该条信息首次出现,更新重复数,存入储存队,总数加 1
ele.count = count
saveQ.append(ele)
total = total + 1
# 输出当前处理对象,以及加上他的关系后工作队个数
print(total, ':', ele.name, ele.id, ' ', end='')
print("此时工作队元素个数(检测是否爆栈):", len(workQ))
# 执行爬取,写入,
try:
    spiderSearchWrite(ele)
except:
    print("故障中止结束")
    flag1 = 1
    break

```

```

    if flag1 == 1:
        print("总爬取写入结束")
        break
    elif flag1 == 2:
        pass

print("爬取总数 total:", total)
end_time = time.time()
print("耗时:", (end_time - start_time))

```

运行结果:

控制台输出示例:

```

D:\Library\ProgramData\Anaconda3\python.exe
C:/Users/leeyu/Desktop/code_example/CourseExperiment/01.tryRunning/001.py
请输入人物姓名:成龙
1 : 成龙 71648      此时工作队元素个数(检测是否爆栈): 0
2 : 林凤娇 2824170  此时工作队元素个数(检测是否爆栈): 34
3 : 房祖名 530870   此时工作队元素个数(检测是否爆栈): 37
4 : 吴卓林 4342611  此时工作队元素个数(检测是否爆栈): 50
5 : 房道龙 962305   此时工作队元素个数(检测是否爆栈): 51
6 : 房仕胜 12509588 此时工作队元素个数(检测是否爆栈): 55
7 : 房仕德 2196680  此时工作队元素个数(检测是否爆栈): 58
8 : 邓丽君 27007    此时工作队元素个数(检测是否爆栈): 62
9 : 吴绮莉 7985285  此时工作队元素个数(检测是否爆栈): 71
10 : 董又霖 19322280 此时工作队元素个数(检测是否爆栈): 74
11 : 罗维 79212     此时工作队元素个数(检测是否爆栈): 76
12 : 于占元 4392373 此时工作队元素个数(检测是否爆栈): 76
13 : 周华健 6700    此时工作队元素个数(检测是否爆栈): 89
14 : 梅艳芳 230174  此时工作队元素个数(检测是否爆栈): 103
15 : 陈木胜 4321147 此时工作队元素个数(检测是否爆栈): 114
16 : 陈泽民 5235    此时工作队元素个数(检测是否爆栈): 116
17 : 刘浩存 23284525 此时工作队元素个数(检测是否爆栈): 119
18 : 郭麒麟 310636  此时工作队元素个数(检测是否爆栈): 128

2986 : 杨采钰 6041646 此时工作队元素个数(检测是否爆栈): 5946
2987 : 李浩菲 17848783 此时工作队元素个数(检测是否爆栈): 5946
2988 : 任重 9484322  此时工作队元素个数(检测是否爆栈): 5944
2989 : 曲栅栅 10173091 此时工作队元素个数(检测是否爆栈): 5951
2990 : 刘江 8780372  此时工作队元素个数(检测是否爆栈): 5950
2991 : 潘泰名 5205163 此时工作队元素个数(检测是否爆栈): 5951
2992 : 姚芊羽 1260876 此时工作队元素个数(检测是否爆栈): 5953
2993 : 岳跃 2647     此时工作队元素个数(检测是否爆栈): 5958
2994 : 王策 12709887 此时工作队元素个数(检测是否爆栈): 5958
2995 : 李雪健 1260954 此时工作队元素个数(检测是否爆栈): 5958
2996 : 温峥嵘 6000079 此时工作队元素个数(检测是否爆栈): 5963
2997 : 李解 3630362  此时工作队元素个数(检测是否爆栈): 5963
2998 : 宋祖儿 2474501 此时工作队元素个数(检测是否爆栈): 5965
2999 : 倪妮 24849    此时工作队元素个数(检测是否爆栈): 5968
3000 : 毛晓彤 6469760 此时工作队元素个数(检测是否爆栈): 5973
总爬取写入结束
爬取总数total: 3000
耗时: 1456.907041311264

进程已结束,退出代码0

```



result\_data.csv      experience

人物姓名,id,基本信息,简介

B2	A	B	C	D	E	F	G	H	I	J	K	L	M
1	id	name	同名次序	关系	基本信息	简介							
2	71648	戚龙	1	妻子: 林凤娇#2824170	中文名	戚龙 (Jackie Chan)							
3	2824170	林凤娇	1	丈夫: 戚龙#71648	中文名	林凤娇							
4	4538070	房祖名	1	父亲: 戚龙#71648	中文名	房祖名							
5	4342611	吴卓林	1	母亲: 吴倩莉#7985285	中文名	吴卓林							
6	962305	房祖光	1	儿子: 房祖胜#1250658	中文名	房祖光 (1914年12月22日—2008年8月26日)							
7	1250658	房祖胜	1	父亲: 房祖光#962305	中文名	房祖胜							
8	2196680	房祖伟	1	父亲: 房祖光#962305	中文名	房祖伟							
9	27007	邓丽君	1	父亲: 邓和#8098483	中文名	邓丽君 (1953年1月29日—1995年5月8日)							
10	7985285	吴倩莉	1	前男友: 叶崇仁#8343204	中文名	吴倩莉							
11	1932280	董又霖	1	姨姐: 林凤娇#2824170	中文名	董又霖 (Jeffrey)							
12	79212	罗维	1	父: 戚龙#71648	中文名	罗维 (1918年—1996年)							
13	4392373	于元元	1	女儿: 于素秋#5790674	中文名	于元元 (1905年—1997年)							
14	6700	周华健	1	妻子: 康粹兰#10170000	中文名	周华健 (Emil Wakin Chau)							
15	250174	陈婉珍	1	母亲: 覃爱金#2804996	中文名	陈婉珍 (1963年10月10日—2003年12月30日)							
16	4321447	陈冠希	1	好友: 戚龙#71648	中文名	陈冠希 (Benson Chan, 1981年—2020年8月23日)							
17	5235	陈浩民	1	儿子: 陈冠希#147585	中文名	陈冠希							
18	23284525	刘浩存	1	搭档: 郭麒麟#4310636	中文名	刘浩存							
19	310636	郭麒麟	1	父亲: 郭德纲#175780	中文名	郭麒麟							
20	406416	杨紫琼	1	父亲: 杨捷德#16024518	中文名	杨紫琼 (Michelle Yeoh)							
21	6961934	麦长青	1	好友: 陈洁民#219832	中文名	麦长青							
22	915273	谭晶	1	丈夫: 邓中翰#573885	中文名	谭晶							
23	3416080	林威	1	搭档: 张学友#411598	中文名	林威, Wai Lam							
24	6719	刘耀威	1	搭档: 周华健#66700	中文名	刘耀威							
25	69991	范大宇	1	丈夫: 陈冠希#138056	中文名	范大宇							
26	333755	曹格	1	丈夫: 乐基儿#8167354	中文名	曹格 (Gogo)							
27	436893	戴安娜	1	搭档: 戚龙#71648	中文名	戴安娜 (Winnie)							
28	948086	蓝心金	1	搭档: 廖凡#835741	中文名	蓝心金 (Zoe Zhang)							
29	1238650	威尔·史密斯	1	儿子: 贾登·史密斯#175427	中文名	威尔·史密斯 (Will Smith)							
30	511610	于荣光	1	父亲: 于鸣魁#382195	中文名	于荣光							

人物生平:

名称	修改日期
000001成龙@71648.txt	2023/4/19 18:41
000002林凤娇@2824170.txt	2023/4/19 18:41
000003房祖名@530870.txt	2023/4/19 18:41
000004吴卓林@4342611.txt	2023/4/19 18:41
000005房道龙@962305.txt	2023/4/19 18:41
000006房仕胜@12509588.txt	2023/4/19 18:41
000007房仕德@2196680.txt	2023/4/19 18:41
000008邓丽君@27007.txt	2023/4/19 18:41
000009吴绮莉@7985285.txt	2023/4/19 18:41
000010董又霖@19322280.txt	2023/4/19 18:41
000011罗维@79212.txt	2023/4/19 18:41

000003房祖名@530870.txt - 记事本			
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)			
<p>房祖名早年经历编辑播报1982年房祖名出生于美国洛杉矶。1997年在香港学习电吉他。2001年—2003年跟随李宗盛学习唱歌和录音。2003年房祖名得到成龙、杨受成的帮助进入娱乐圈。房祖名演艺经历编辑播报2003年，他在电影《千机变2花都大战》中担任男主角，并演唱电影主题曲《最动听》。2004年，发行首张专辑《房祖名》；8月28日，担任中国电影华表奖演出嘉宾，并演唱歌曲《边走边唱》。房祖名(2张)2005年1月，在央视春节联欢晚会上演唱创作单曲《要强》；4月凭借电影《千机变2花都大战》获得第24届香港电影金像奖最佳新演员提名；同年出演电影《早熟》。2006年，客串电影《春田花花同学会》。2007年与谢霆锋、余文乐、吴京合作主演电影《男儿本色》，饰演尽忠职守的警察卫景灏；同年出演姜文执导的电影《太阳照常升起》。2008年，与梁家辉、李心洁、郑希怡合作出演爱情电影《战鼓》，剧中他饰演的Sid是一个黑道老大的儿子，因为与另外一个黑道老大的情妇相爱而被人追杀；同年凭借电影《太阳照常升起》获得第64届威尼斯电影节最佳男主角提名。同年，演唱歌曲《北京欢迎你》。2009年，主演电影《追影》，饰演推动电影情节发展的重要线索人物许三官；同年与赵薇、陈坤、胡军合作出演古装战争电影《花木兰》，饰演费小虎一角；12月，参加澳门回归十周年晚会。房祖名(7张)2010年6月16日，主演的爱情电影《分手说爱你》在香港上映，截止6月27日止，该片票房累计为780万港币；同年发行专辑《乱》。2011年，主演由郭帆、李阳执导的爱情电影《李献计历险记》，剧中他饰演的李献计是一个为了爱情不惜卖肾、换铁手的另类青年；之后出演喜剧电影《东成西就2011》，饰演摩呼罗迦一角；同年参演电影《辛亥革命》；8月7日参加东方卫视《舞林大会》。2012年，主演动作电影《宝岛双雄》，由于影片的剧情、风格和成龙的动作喜剧比较接近，因此该片也被外界视为房祖名的转型之作；之后与范文芳共同主演爱情电影《爱谁谁》，他所饰演的李世民是一位有计谋的爱情专家；同年与安以轩、李欣汝、曾志伟共同主演爱情喜剧电影《给野兽献花》</p>			
第 1 行, 第 1 列		100%	Windows (CRLF) 带有 BOM 的 UTF