

Python语言程序设计

# 第7章 文件和数据格式化

---





# 本课概要

# 第7章 文件和数据格式化

## 格式化

字符串格式化



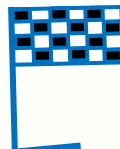
数据格式化



`"{ }{ }{ }".format()`

将字符串按照一定规格和式样  
进行规范

将一组数据按照一定规格和式样进行  
规范：表示、存储、运算等



# 第7章 文件和数据格式化



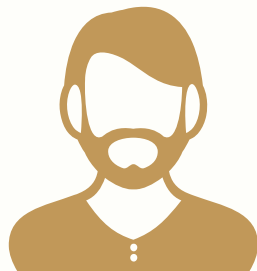
- 7.1 文件的使用
- 7.2 实例11: 自动轨迹绘制
- 7.3 一维数据的格式化和处理
- 7.4 二维数据的格式化和处理
- 7.5 模块6: wordcloud库的使用
- 7.6 实例12: 政府工作报告词云



# 第7章 文件和数据格式化

## 方法论

- 从Python角度理解的文件和数据表示



## 实践能力

- 学会编写带有文件输入输出的程序





# 前课复习

# 数字类型及操作

- 整数类型的无限范围及4种进制表示
- 浮点数类型的近似无限范围、小尾数及科学计数法
- +、-、\*、/、//、%、\*\*、二元增强赋值操作符
- abs()、divmod()、pow()、round()、max()、min()
- int()、float()、complex()



# 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





# 程序的分支结构

- 单分支 *if* 二分支 *if-else* 及紧凑形式
- 多分支 *if-elif-else* 及条件之间关系
- *not and or > >= == <= < !=*
- 异常处理 *try-except-else-finally*



# 程序的循环结构

- *for...in* 遍历循环: 计数、字符串、列表、文件...
- *while* 无限循环
- *continue* 和 *break* 保留字: 退出当前循环层次
- 循环 *else* 的高级用法: 与 *break* 有关



# 函数的定义与使用

- 使用保留字`def`定义函数，`lambda`定义匿名函数
- 可选参数(赋初值)、可变参数(\*b)、名称传递
- 保留字`return`可以返回任意多个结果
- 保留字`global`声明使用全局变量，一些隐式规则



# 代码复用与函数递归

- 模块化设计：松耦合、紧耦合
- 函数递归的2个特征：基例和链条
- 函数递归的实现：函数 + 分支结构



# 集合类型及操作

- 集合使用{}和set()函数创建
- 集合间操作：交(&)、并(|)、差(-)、补(^)、比较(>=<)
- 集合类型方法：.add()、.discard()、.pop()等
- 集合类型主要应用于：包含关系比较、数据去重



# 序列类型及操作

- 序列是基类类型，扩展类型包括：字符串、元组和列表
- 元组用`()`和`tuple()`创建，列表用`[]`和`set()`创建
- 元组操作与序列操作基本相同
- 列表操作在序列操作基础上，增加了更多的灵活性



python

语言程序设计

# 字典类型及操作

- 映射关系采用键值对表达
- 字典类型使用{}和dict()创建，键值对之间用:分隔
- d[key] 方式既可以索引，也可以赋值
- 字典类型有一批操作方法和函数，最重要的是.get()



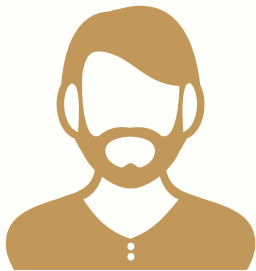


# 练习与作业



# 第7章 文件和数据格式化

## 练习



## 7.1 文件的使用

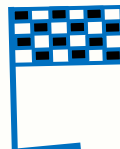
---



# 文件的使用



- 文件的类型
- 文件的打开和关闭
- 文件内容的读取
- 数据的文件写入





# 文件的类型

# 文件的理解

## 文件是数据的抽象和集合

- 文件是存储在辅助存储器上的数据序列
- 文件是数据存储的一种形式
- 文件展现形态：文本文件和二进制文件

# 文件的理解

## 文本文件 vs. 二进制文件

- 文件文件和二进制文件只是文件的展示方式
- 本质上，所有文件都是二进制形式存储
- 形式上，所有文件采用两种方式展示

# 文本文件

文件是数据的抽象和集合

- 由**单一特定编码**组成的文件，如UTF-8编码
- 由于存在编码，也被看成是存储着的长字符串
- 适用于例如：.txt文件、.py文件等

# 二进制文件

文件是数据的抽象和集合

- 直接由比特0和1组成，**没有统一字符编码**
- 一般存在二进制0和1的组织结构，即文件格式
- 适用于例如：.png文件、.avi文件等



# 文本文件 vs. 二进制文件

"中国是个伟大的国家!"

- 文本形式

中国是个伟大的国家!

- 二进制形式

```
b' \xd6\xd0\xb9\xfa\xca\xc7\xb8\xf6\xce\xb0\xb4\xf3\xb5\x  
c4\xb9\xfa\xbc\xd2\xa3\xa1 '
```

# 文本文件 vs. 二进制文件

f.txt文件保存: "中国是个伟大的国家!"

```
#文本形式打开文件
```

```
tf = open("f.txt", "rt")
```

```
print(tf.readline())
```

```
tf.close()
```

```
>>>
```

```
中国是个伟大的国家!
```

# 文本文件 vs. 二进制文件

f.txt文件保存: "中国是个伟大的国家!"

#二进制形式打开文件

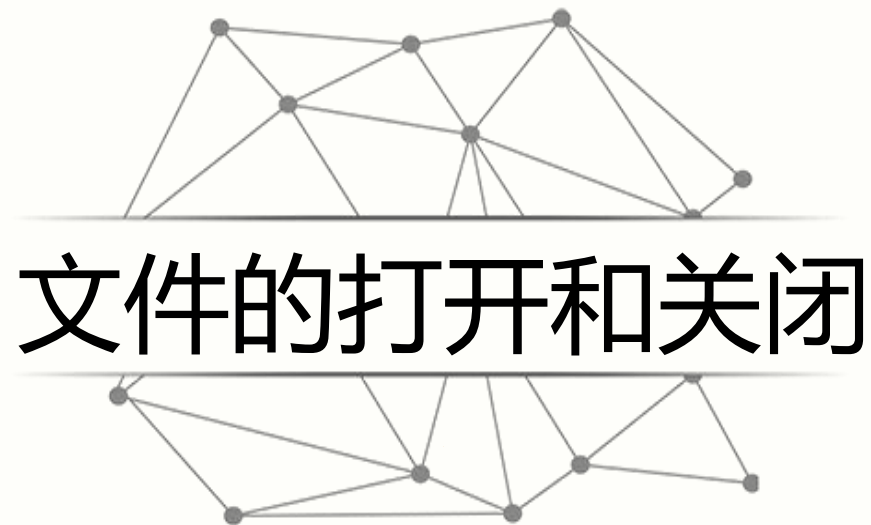
```
bf = open("f.txt", "rb")
```

```
print(bf.readline())
```

```
bf.close()
```

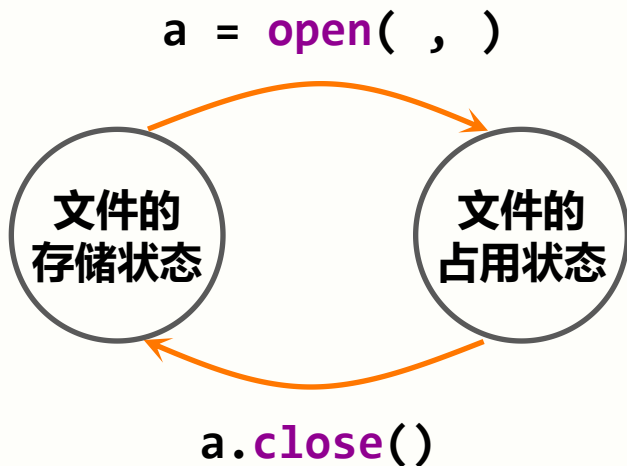
```
>>>
```

```
b'\xd6\xd0\xb9\xfa\xca\xc7\xb8\xf6\xce\xb0  
\xb4\xf3\xb5\xc4\xb9\xfa\xbc\xd2\xa3\xa1'
```



# 文件的打开关闭

## 文件处理的步骤: 打开-操作-关闭

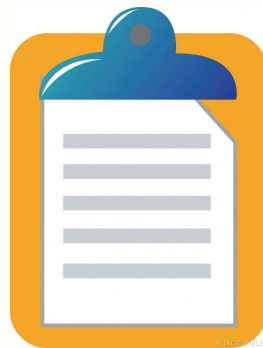


```
a.read(size)
a.readline(size)
a.readlines(hint)
```

读文件

```
-----
a.write(s)
a.writelines(lines)
a.seek(offset)
```

写文件



# 文件的打开

**<变量名> = open(<文件名>, <打开模式>)**

**文件句柄**

**文件路径和名称**

源文件同目录可省路径

**文本 or 二进制**

**读 or 写**

# 文件路径

＜变量名＞ = **open**(＜文件名＞, ＜打开模式＞)

D:\PYE\f.txt



文件路径和名称

"D:/PYE/f.txt"

". /PYE/f.txt"

源文件同目录可省路径

"D:\\PYE\\f.txt"

"f.txt"

# 打开模式

文件的打开模式	描述
'r'	只读模式，默认值，如果文件不存在，返回FileNotFoundError
'w'	覆盖写模式，文件不存在则创建，存在则完全覆盖
'x'	创建写模式，文件不存在则创建，存在则返回FileExistsError
'a'	追加写模式，文件不存在则创建，存在则在文件最后追加内容
'b'	二进制文件模式
't'	文本文件模式，默认值
'+'	与r/w/x/a一同使用，在原功能基础上增加同时读写功能



# 打开模式

```
f = open("f.txt")
```

```
f = open("f.txt", "rt")
```

```
f = open("f.txt", "w")
```

```
f = open("f.txt", "a+")
```

```
f = open("f.txt", "x")
```

```
f = open("f.txt", "b")
```

```
f = open("f.txt", "wb")
```

- 文本形式、只读模式、默认值
- 文本形式、只读模式、同默认值
- 文本形式、覆盖写模式
- 文本形式、追加写模式+ 读文件
- 文本形式、创建写模式
- 二进制形式、只读模式
- 二进制形式、覆盖写模式

# 文件的关闭

**〈变量名〉.close()**



**文件句柄**

# 文件使用

#文本形式打开文件

```
tf = open("f.txt", "rt")  
print(tf.readline())  
tf.close()
```

#二进制形式打开文件

```
bf = open("f.txt", "rb")  
print(bf.readline())  
bf.close()
```



# 文件内容的读取

# 文件内容的读取

操作方法	描述
<code>&lt;f&gt;.read(size=-1)</code>	读入全部内容，如果给出参数，读入前size长度  <code>&gt;&gt;&gt;s = f.read(2)</code>  中国
<code>&lt;f&gt;.readline(size=-1)</code>	读入一行内容，如果给出参数，读入该行前size长度  <code>&gt;&gt;&gt;s = f.readline()</code>  中国是一个伟大的国家！

# 文件内容的读取

操作方法	描述
<code>&lt;f&gt;.readlines(hint=-1)</code>	<p>读入文件所有行，以每行为元素形成列表</p> <p>如果给出参数，读入前hint行</p> <pre>&gt;&gt;&gt;s = f.readlines()</pre> <pre>['中国是一个伟大的国家！']</pre>

# 文件的全文本操作

## 遍历全文本：方法一

```
fname = input("请输入要打开的文件名称:")
```

```
fo = open(fname, "r")
```

```
txt = fo.read()
```

```
#对全文txt进行处理
```

- 一次读入，统一处理

```
fo.close()
```

# 文件的全文本操作

## 遍历全文本：方法二

```
fname = input("请输入要打开的文件名称:")
```

```
fo = open(fname, "r")
```

```
txt = fo.read(2)
```

```
while txt != "":
```

```
    #对txt进行处理
```

```
    txt = fo.read(2)
```

```
fo.close()
```

- 按数量读入，逐步处理



# 文件的逐行操作

## 逐行遍历文件：方法一

```
fname = input("请输入要打开的文件名称:")
```

```
fo = open(fname, "r")
```

```
for line in fo.readlines():
```

```
    print(line)
```

- 一次读入，分行处理

```
fo.close()
```

# 文件的逐行操作

## 逐行遍历文件：方法二

```
fname = input("请输入要打开的文件名称:")
```

```
fo = open(fname, "r")
```

```
for line in fo:
```

```
    print(line)
```

- 分行读入，逐行处理

```
fo.close()
```



# 数据的文件写入

# 数据的文件写入

操作方法	描述
<code>&lt;f&gt;.write(s)</code>	<p>向文件写入一个字符串或字节流</p> <pre>&gt;&gt;&gt;f.write("中国是一个伟大的国家!")</pre>
<code>&lt;f&gt;.writelines(lines)</code>	<p>将一个元素全为字符串的列表写入文件</p> <pre>&gt;&gt;&gt;ls = ["中国", "法国", "美国"] &gt;&gt;&gt;f.writelines(ls)  中国法国美国</pre>

# 数据的文件写入

操作方法	描述
<code>&lt;f&gt;.seek(offset)</code>	<p>改变当前文件操作指针的位置, offset含义如下: 0 – 文件开头; 1 – 当前位置; 2 – 文件结尾</p> <p><b>&gt;&gt;&gt;f.seek(0) #回到文件开头</b></p>

# 数据的文件写入

```
fo = open("output.txt", "w+")
```

```
ls = ["中国", "法国", "美国"]
```

```
fo.writelines(ls)
```

```
for line in fo:
```

```
    print(line)
```

```
fo.close()
```

- 写入一个字符串列表

>>> (没有任何输出)

# 数据的文件写入

```
fo = open("output.txt", "w+")
```

```
ls = ["中国", "法国", "美国"]
```

```
fo.writelines(ls)
```

```
fo.seek(0)
```

```
for line in fo:
```

```
    print(line)
```

```
fo.close()
```

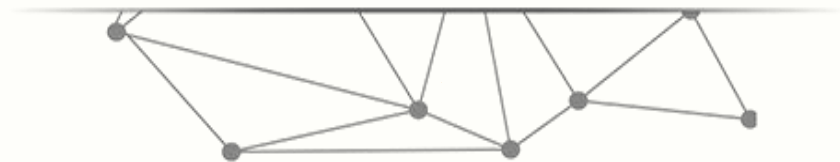
- 写入一个字符串列表

```
>>>
```

中国法国美国



# 单元小结





# 文件的使用

- 文件的使用方式：打开-操作-关闭
- 文本文件&二进制文件，`open( , )`和`.close()`
- 文件内容的读取：`.read()` `.readline()` `.readlines()`
- 数据的文件写入：`.write()` `.writelines()` `.seek()`



## 7.2 实例11：自动轨迹绘制

---





# "自动轨迹绘制"问题分析

# 问题分析

## 自动轨迹绘制

- **需求：根据脚本来绘制图形？**
- **不通过写代码而通过写数据绘制轨迹**
- **数据脚本是自动化最重要的第一步**

# 问题分析

## 自动轨迹绘制

300,0,144,1,0,0

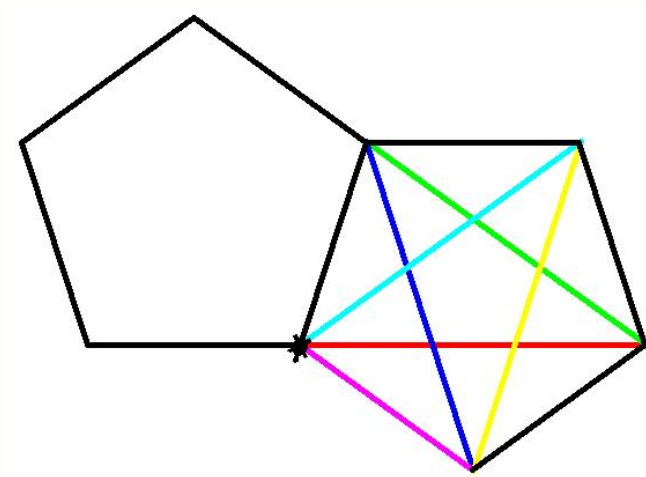
300,0,144,0,1,0

300,0,144,0,0,1

300,0,144,1,1,0

300,0,108,0,1,1

184,0,72,1,0,1





# "自动轨迹绘制"实例讲解

# 自动轨迹绘制

## 基本思路

- **步骤1：定义数据文件格式（接口）**
- **步骤2：编写程序，根据文件接口解析参数绘制图形**
- **步骤3：编制数据文件**

# 数据接口定义

非常具有个性色彩

300,0,144,1,0,0

300,1,144,0,1,0

行进距离

转向判断

0: 左转 1:右转

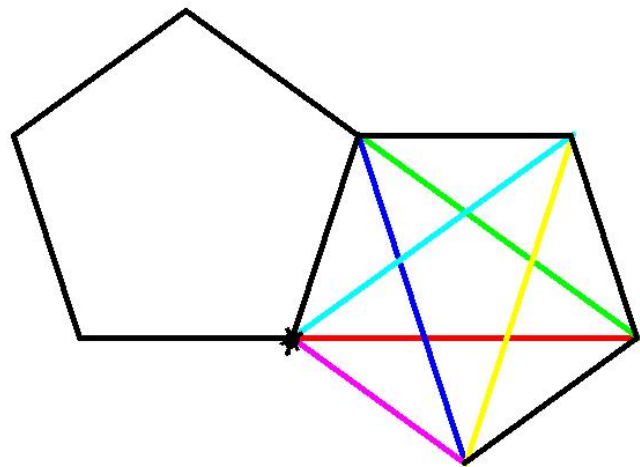
转向角度

RGB三个通道颜色

0-1之间浮点数



```
#AutoTraceDraw.py
import turtle as t
t.title('自动轨迹绘制')
t.setup(800, 600, 0, 0)
t.pencolor("red")
t.pensize(5)
#数据读取
datals = []
f = open("data.txt")
for line in f:
    line = line.replace("\n", "")
    datals.append(list(map(eval, line.split(","))))
f.close()
#自动绘制
for i in range(len(datals)):
    t.pencolor(datals[i][3], datals[i][4], datals[i][5])
    t.fd(datals[i][0])
    if datals[i][1]:
        t.right(datals[i][2])
    else:
        t.left(datals[i][2])
```



# 数据文件

300,0,144,1,0,0

300,0,144,0,1,0

300,0,144,0,0,1

300,0,144,1,1,0

300,0,108,0,1,1

184,0,72,1,0,1

184,0,72,0,0,0

184,0,72,0,0,0

184,0,72,0,0,0

184,1,72,1,0,1

184,1,72,0,0,0

184,1,72,0,0,0

184,1,72,0,0,0

184,1,72,0,0,0

184,1,720,0,0,0

data.txt



**准备好电脑，与老师一起编码吧！**

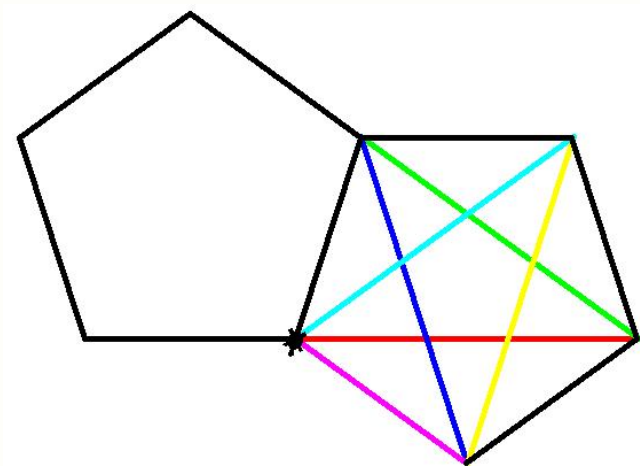


"自动轨迹绘制"举一反三

```

import turtle as t
t.title('自动轨迹绘制')
t.setup(800, 600, 0, 0)
t.pencolor("red")
t.pensize(5)
datals = []
f = open("data.txt")
for line in f:
    line = line.replace("\n", "")
    datals.append(list(map(eval, line.split(","))))
f.close()
for i in range(len(datals)):
    t.pencolor(datals[i][3], datals[i][4], datals[i][5])
    t.fd(datals[i][0])
    if datals[i][1]:
        t.right(datals[i][2])
    else:
        t.left(datals[i][2])

```



# 举一反三

## 理解方法思维

- **自动化思维：数据和功能分离，数据驱动自动运行**
- **接口化设计：格式化设计接口，清晰明了**
- **二维数据应用：应用维度组织数据，二维数据最常用**

# 举一反三

## 应用问题的扩展

- 扩展接口设计，增加更多控制接口
- 扩展功能设计，增加弧形等更多功能
- 扩展应用需求，发展自动轨迹绘制到动画绘制

## 7.3 一维数据的格式化和处理

---

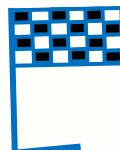




# 一维数据的格式化和处理



- 数据组织的维度
- 一维数据的表示
- 一维数据的存储
- 一维数据的处理





# 数据组织的维度

# 从一个数据到一组数据

3.14



3.1413

3.1398

3.1404

3.1401

3.1349

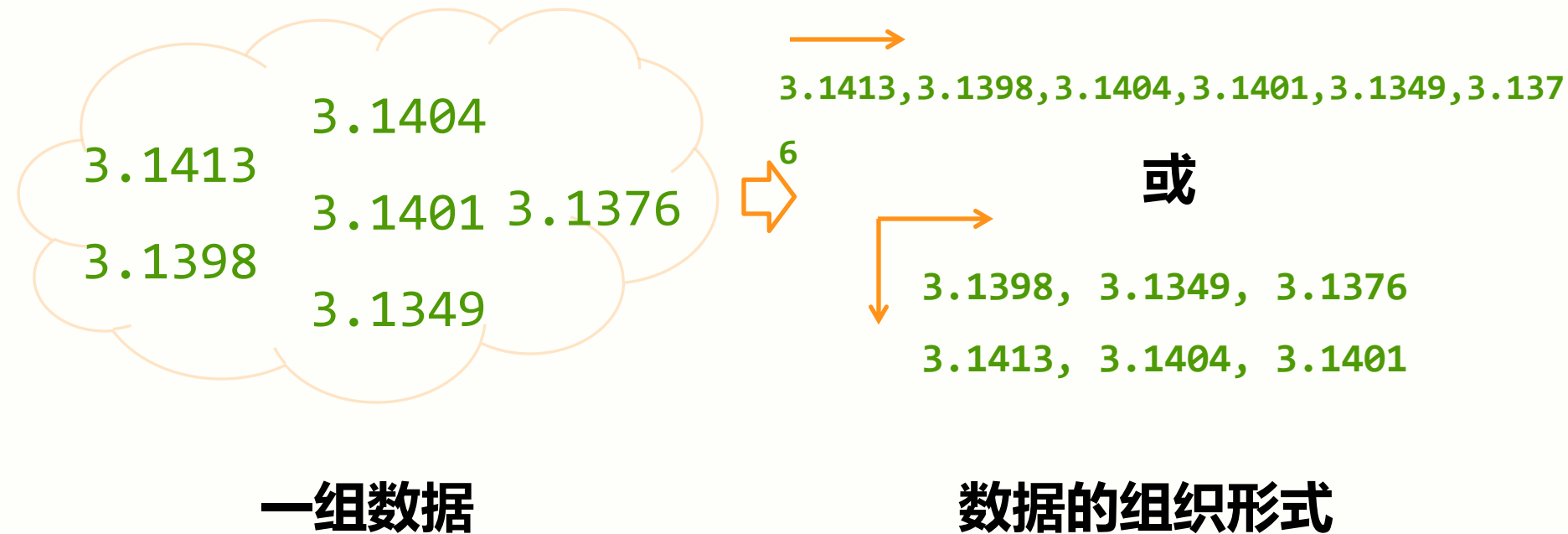
**一个数据**

**表达一个含义**

**一组数据**

**表达一个或多个含义**

# 维度：一组数据的组织形式



# 一维数据

由对等关系的有序或无序数据构成，采用线性方式组织

3.1413, 3.1398, 3.1404, 3.1401, 3.1349, 3.1376

- 对应列表、数组和集合等概念

# 二维数据

由多个一维数据构成，是一维数据的组合形式

排名	学校名称	省市	总分	指标得分
				生源质量（新生高考成绩得分） ▾
1	清华大学	北京	94.0	100.0
2	北京大学	北京	81.2	96.1
3	浙江大学	浙江	77.8	87.2
4	上海交通大学	上海	77.5	89.4
5	复旦大学	上海	71.1	91.8
6	中国科学技术大学	安徽	65.9	91.9
7	南京大学	江苏	65.3	87.1
8	华中科技大学	湖北	63.0	80.6
9	中山大学	广东	62.7	81.1
10	哈尔滨工业大学	黑龙江	61.6	76.4

表格是典型的二维数据

其中，表头是二维数据的一部分

# 多维数据

由一维或二维数据在新维度上扩展形成

排名	学校名称	省市	总分	指标得分
				生源质量（新生高考成绩得分） ▾
1	清华大学	北京市	95.9	100.0
2	北京大学	北京市	82.6	98.9
3	浙江大学	浙江省	80	88.8
4	上海交通大学	上海市	78.7	90.6
5	复旦大学	上海市	70.9	90.4
6	南京大学	江苏省	66.1	90.7
7	中国科学技术大学	安徽省	65.5	90.1
8	哈尔滨工业大学	黑龙江省	63.5	80.9
9	华中科技大学	湖北省	62.9	83.5
10	中山大学	广东省	62.1	81.8

时间维度



2016

排名	学校名称	省市	总分	指标得分
				生源质量（新生高考成绩得分） ▾
1	清华大学	北京	94.0	100.0
2	北京大学	北京	81.2	96.1
3	浙江大学	浙江	77.8	87.2
4	上海交通大学	上海	77.5	89.4
5	复旦大学	上海	71.1	91.8
6	中国科学技术大学	安徽	65.9	91.9
7	南京大学	江苏	65.3	87.1
8	华中科技大学	湖北	63.0	80.6
9	中山大学	广东	62.7	81.1
10	哈尔滨工业大学	黑龙江	61.6	76.4

2017

# 高维数据

仅利用最基本的二元关系展示数据间的复杂结构

```
{
  "firstName" : "Tian" ,
  "lastName"  : "Song" ,
  "address"   : {
    "streetAddr" : "中关村南大街5号" ,
    "city"       : "北京市" ,
    "zipcode"    : "100081"
  } ,
  "professional" : ["Computer Networking" , "Security"]
}
```

键值对



# 数据的操作周期

存储 <-> 表示 <-> 操作





# 一维数据的表示

如果数据间有序：使用列表类型

```
ls = [3.1398, 3.1349, 3.1376]
```

- 列表类型可以表达一维有序数据
- for循环可以遍历数据，进而对每个数据进行处理

# 一维数据的表示

如果数据间无序：使用集合类型

```
st = {3.1398, 3.1349, 3.1376}
```

- 集合类型可以表达一维无序数据
- for循环可以遍历数据，进而对每个数据进行处理



一维数据的存储

# 一维数据的存储

## 存储方式一：空格分隔

中国 美国 日本 德国 法国 英国 意大利

- 使用一个或多个空格分隔进行存储，不换行
- 缺点：数据中不能存在空格

# 一维数据的存储

## 存储方式二：逗号分隔

中国,美国,日本,德国,法国,英国,意大利

- 使用英文半角逗号分隔数据进行存储，不换行
- 缺点：数据中不能有英文逗号

# 一维数据的存储

## 存储方式三：其他方式

中国\$美国\$日本\$德国\$法国\$英国\$意大利

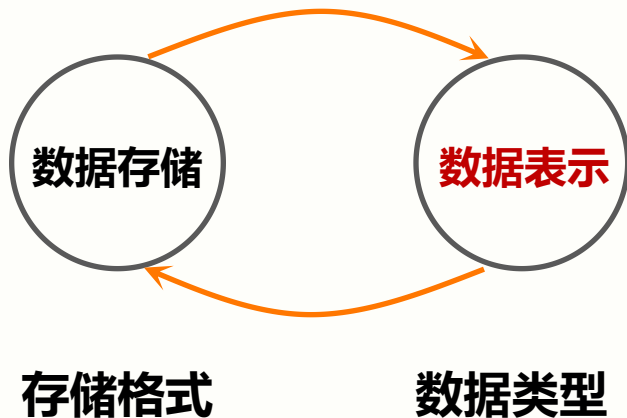
- 使用其他符号或符号组合分隔，建议采用特殊符号
- 缺点：需要根据数据特点定义，通用性较差





# 数据的处理

存储 <-> 表示



- 将存储的数据读入程序
- 将程序表示的数据写入文件

# 一维数据的读入处理

## 从空格分隔的文件中读入数据

中国 美国 日本 德国 法国 英国 意大利

```
txt = open(fname).read()
```

```
ls = txt.split()
```

```
f.close()
```

```
>>> ls
```

```
['中国', '美国', '日本', '德国',  
, '法国', '英国', '意大利']
```

# 一维数据的读入处理

## 从特殊符号分隔的文件中读入数据

中国\$美国\$日本\$德国\$法国\$英国\$意大利

```
txt = open(fname).read()
```

```
ls = txt.split("$")
```

```
f.close()
```

```
>>> ls
```

```
['中国', '美国', '日本', '德国',  
, '法国', '英国', '意大利']
```

# 一维数据的写入处理

采用空格分隔方式将数据写入文件

```
ls = ['中国', '美国', '日本']
```

```
f = open(fname, 'w')
```

```
f.write(' '.join(ls))
```

```
f.close()
```

# 一维数据的写入处理

采用特殊分隔方式将数据写入文件

```
ls = ['中国', '美国', '日本']
```

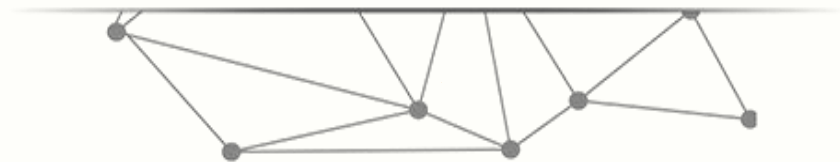
```
f = open(fname, 'w')
```

```
f.write('$'.join(ls))
```

```
f.close()
```

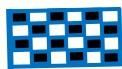


# 单元小结



# 一维数据的格式化和处理

- 数据的维度：一维、二维、多维、高维
- 一维数据的表示：列表类型(有序)和集合类型(无序)
- 一维数据的存储：空格分隔、逗号分隔、特殊符号分隔
- 一维数据的处理：字符串方法 `.split()` 和 `.join()`





## 7.4 二维数据的格式化和处理

---



# 二维数据的格式化和处理



- 二维数据的表示
- CSV数据存储格式
- 二维数据的存储
- 二维数据的处理





## 二维数据的表示

# 二维数据的表示

## 使用列表类型

排名	学校名称	省市	总分	指标得分
				生源质量（新生高考成绩得分） ▾
1	清华大学	北京	94.0	100.0
2	北京大学	北京	81.2	96.1
3	浙江大学	浙江	77.8	87.2
4	上海交通大学	上海	77.5	89.4
5	复旦大学	上海	71.1	91.8
6	中国科学技术大学	安徽	65.9	91.9
7	南京大学	江苏	65.3	87.1
8	华中科技大学	湖北	63.0	80.6
9	中山大学	广东	62.7	81.1
10	哈尔滨工业大学	黑龙江	61.6	76.4

- 列表类型可以表达二维数据

- 使用二维列表

```
[ [3.1398, 3.1349, 3.1376],  
  [3.1413, 3.1404, 3.1401] ]
```

# 二维数据的表示

## 使用列表类型

```
[ [3.1398, 3.1349, 3.1376],  
  [3.1413, 3.1404, 3.1401] ]
```

- 使用两层for循环遍历每个元素
- 外层列表中每个元素可以对应一行，也可以对应一列

# 一二维数据的Python表示

数据维度是数据的组织形式


## - 一维数据：列表和集合类型

`[3.1398, 3.1349, 3.1376]` 数据间有序

`{3.1398, 3.1349, 3.1376}` 数据间无序

## - 二维数据：列表类型

`[ [3.1398, 3.1349, 3.1376],  
[3.1413, 3.1404, 3.1401] ]`



# CSV格式与二维数据存储

# CSV数据存储格式

## CSV: Comma-Separated Values

- 国际通用的一二维数据存储格式，一般.csv扩展名
- 每行一个一维数据，采用逗号分隔，无空行
- Excel和一般编辑软件都可以读入或另存为csv文件



# CSV数据存储格式

城市	环比	同比	定基
北京	101.5	120.7	121.4
上海	101.2	127.3	127.8
广州	101.3	119.4	120.0
深圳	102.0	140.0	145.5
沈阳	100.0	101.4	101.6



城市,环比,同比,定基

北京,101.5,120.7,121.4

上海,101.2,127.3,127.8

广州,101.3,119.4,120.0

深圳,102.0,140.0,145.5

沈阳,100.0,101.4,101.6

# CSV数据存储格式

## CSV: Comma-Separated Values

- 如果某个元素缺失，逗号仍要保留
- 二维数据的表头可以作为数据存储，也可以另行存储
- 逗号为英文半角逗号，逗号与数据之间无额外空格

# 二维数据的存储

**按行存? 按列存?**

- **按行存或者按列存都可以，具体由程序决定**
- **一般索引习惯：ls[row][column]，先行后列**
- **根据一般习惯，外层列表每个元素是一行，按行存**



# 二维数据的处理

# 二维数据的读入处理

## 从CSV格式的文件中读入数据

```
fo = open(fname)
ls = []
for line in fo:
    line = line.replace("\n", "")
    ls.append(line.split(","))
fo.close()
```

# 二维数据的写入处理

## 将数据写入CSV格式的文件

```
ls = [[], [], []] #二维列表
```

```
f = open(fname, 'w')
```

```
for item in ls:
```

```
    f.write(','.join(item) + '\n')
```

```
f.close()
```

# 二维数据的逐一处理

## 采用二层循环

```
ls = [[1,2], [3,4], [5,6]] #二维列
```

表

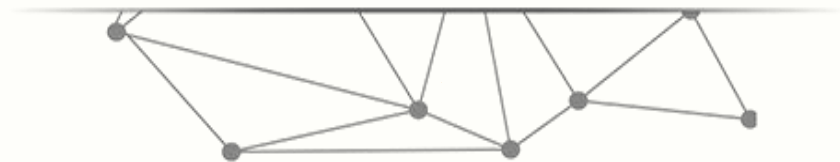
```
for row in ls:
```

```
    for column in row:
```

```
        print(column)
```



# 单元小结





# 二维数据的格式化和处理

- 二维数据的表示：列表类型，其中每个元素也是一个列表
- CSV格式：逗号分隔表示一维，按行分隔表示二维
- 二维数据的处理：for循环+.split()和.join()



## 7.5 模块6: wordcloud库的使用

---





# wordcloud库基本介绍

# wordcloud库概述

## wordcloud是优秀的词云展示第三方库



- **词云以词语为基本单位，更加直观和艺术地展示文本**

# wordcloud库的安装

(cmd命令行) `pip install wordcloud`

```
命令提示符
Microsoft Windows [版本 10.0.16299.371]
(c) 2017 Microsoft Corporation. 保留所有权利。

C:\Users\Tian Song>pip install wordcloud
Collecting wordcloud
  Downloading https://files.pythonhosted.org/packages/bc/e8/cab8479b25297b3347cfb55e85a5014e8c53b80e513eaf1ba58c7b3a6acd/wordcloud-1.4.1.tar.gz (172kB)
    100% |#####| 174kB 33kB/s

Installing collected packages: wordcloud
  Running setup.py install for wordcloud ... done
Successfully installed wordcloud-1.4.1
You are using pip version 9.0.1, however version 10.0.0 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\Tian Song>
```

```
命令提示符
python36-32\lib\site-packages (from wordcloud)
Requirement already satisfied: numpy>=1.6.1 in c:\users\tian song\appdata\local\programs\python36-32\lib\site-packages (from wordcloud)
Requirement already satisfied: pillow in c:\users\tian song\appdata\local\programs\python\python36-32\lib\site-packages (from wordcloud)
Requirement already satisfied: six>=1.10 in c:\users\tian song\appdata\local\programs\python\python36-32\lib\site-packages (from matplotlib->wordcloud)
Requirement already satisfied: pytz in c:\users\tian song\appdata\local\programs\python\python36-32\lib\site-packages (from matplotlib->wordcloud)
Requirement already satisfied: cycler>=0.10 in c:\users\tian song\appdata\local\programs\python\python36-32\lib\site-packages (from matplotlib->wordcloud)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\tian song\appdata\local\programs\python\python36-32\lib\site-packages (from matplotlib->wordcloud)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\users\tian song\appdata\local\programs\python\python36-32\lib\site-packages (from matplotlib->wordcloud)
Requirement already satisfied: olefile in c:\users\tian song\appdata\local\programs\python\python36-32\lib\site-packages (from pillow->wordcloud)
Installing collected packages: wordcloud
  Running setup.py install for wordcloud ... done
Successfully installed wordcloud-1.4.1
You are using pip version 9.0.1, however version 10.0.0 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\Tian Song>
```



# wordcloud库使用说明

# wordcloud库基本使用

**wordcloud库把词云当作一个WordCloud对象**

- **wordcloud.WordCloud()代表一个文本对应的词云**
- **可以根据文本中词语出现的频率等参数绘制词云**
- **词云的绘制形状、尺寸和颜色都可以设定**

# wordcloud库常规方法

**w = wordcloud.WordCloud()**

- 以WordCloud对象为基础
- 配置参数、加载文本、输出文件



# wordcloud库常规方法

**w = wordcloud.WordCloud()**

方法	描述
w.generate(txt)	向WordCloud对象w中加载文本txt <code>&gt;&gt;&gt;w.generate("Python and WordCloud")</code>
w.to_file(filename)	将词云输出为图像文件，.png或.jpg格式 <code>&gt;&gt;&gt;w.to_file("outfile.png")</code>

# wordcloud库常规方法

```
import wordcloud
```

```
c = wordcloud.WordCloud()
```

```
c.generate("wordcloud by Python")
```

```
c.to_file("pywordcloud.png")
```

- 步骤1：配置对象参数

- 步骤2：加载词云文本

- 步骤3：输出词云文件

# wordcloud库常规方法



↑  
200  
↓

← 400 →

# wordcloud库常规方法

"wordcloud by Python"



文本



① 分隔: 以空格分隔单词

② 统计: 单词出现次数并过滤

③ 字体: 根据统计配置字号

④ 布局: 颜色环境尺寸



词云

# 配置对象参数

**w = wordcloud.WordCloud(<参数>)**

参数	描述
width	指定词云对象生成图片的宽度，默认400像素 <b>&gt;&gt;&gt;w=wordcloud.WordCloud(width=600)</b>
height	指定词云对象生成图片的高度，默认200像素 <b>&gt;&gt;&gt;w=wordcloud.WordCloud(height=400)</b>

# 配置对象参数

参数	描述
min_font_size	指定词云中字体的最小字号，默认4号 <code>&gt;&gt;&gt;w=wordcloud.WordCloud(min_font_size=10)</code>
max_font_size	指定词云中字体的最大字号，根据高度自动调节 <code>&gt;&gt;&gt;w=wordcloud.WordCloud(max_font_size=20)</code>
font_step	指定词云中字体字号的步进间隔，默认为1 <code>&gt;&gt;&gt;w=wordcloud.WordCloud(font_step=2)</code>

# 配置对象参数

参数	描述
font_path	指定字体文件的路径，默认None <code>&gt;&gt;&gt;w=wordcloud.WordCloud(font_path="msyh.ttc")</code>
max_words	指定词云显示的最大单词数量，默认200 <code>&gt;&gt;&gt;w=wordcloud.WordCloud(max_words=20)</code>
stop_words	指定词云的排除词列表，即不显示的单词列表 <code>&gt;&gt;&gt;w=wordcloud.WordCloud(stop_words={"Python"})</code>

# 配置对象参数

参数	描述
mask	<p>指定词云形状，默认为长方形，需要引用imread()函数</p> <pre>&gt;&gt;&gt;from scipy.misc import imread &gt;&gt;&gt;mk=imread("pic.png") &gt;&gt;&gt;w=wordcloud.WordCloud(mask=mk)</pre>
background_color	<p>指定词云图片的背景颜色，默认为黑色</p> <pre>&gt;&gt;&gt;w=wordcloud.WordCloud(background_color="white")</pre>



# wordcloud应用实例

```
import wordcloud  
txt = "life is short, you need python"  
w = wordcloud.WordCloud( \  
    background_color = "white")  
w.generate(txt)  
w.to_file("pywcloud.png")
```



以空格分隔单词

```
import jieba
```

```
import wordcloud
```

```
txt = "程序设计语言是计算机能够理解和\
识别用户操作意图的一种交互体系，它按照\
特定规则组织计算机指令，使计算机能够自\
动进行各种运算处理。"
```

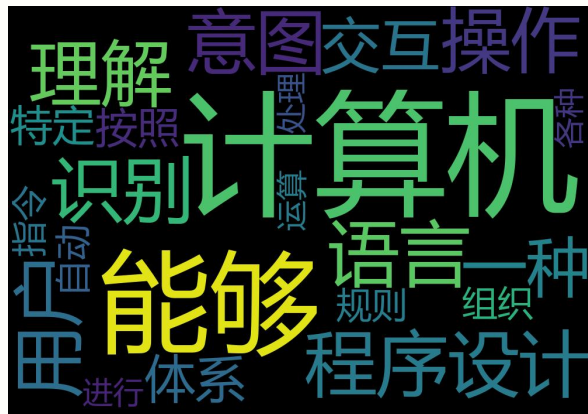
```
w = wordcloud.WordCloud( width=1000,\
```

```
    font_path="msyh.ttc",height=700)
```

```
w.generate(" ".join(jieba.lcut(txt)))
```

```
w.to_file("pywcloud.png")
```

中文需要先分词并组成空格分隔字符串



## 7.6 实例12: 政府工作报告词云

---





# "政府工作报告词云"问题分析

# 问题分析

## 直观理解政策文件

- 需求：对于政府工作报告等政策文件，如何直观理解？
- 体会直观的价值：生成词云 & 优化词云

政府工作报告等文件  有效展示的词云

# 问题分析

**《决胜全面建成小康社会 夺取新时代中国特色社会主义伟大胜利》**

**在中国共产党第十九次全国代表大会上的报告**

**(2017年10月18日)**

**习近平**

**新时代中国特色社会主义.txt**

# 问题分析

**《中共中央 国务院关于实施乡村振兴战略的意见》**

**2018一号文件**

**(2018年01月02日)**

**中共中央 国务院**

**关于实施乡村振兴战略的意见.txt**



# "政府工作报告词云"实例讲解(上)



# 政府工作报告词云

## 基本思路

- 步骤1：读取文件、分词整理
- 步骤2：设置并输出词云
- 步骤3：观察结果，优化迭代



```
#GovRptWordCloudv1.py
```

```
import jieba
```

```
import wordcloud
```

```
f = open("新时代中国特色社会主义思想.txt", "r", encoding="utf-8")
```

```
t = f.read()
```

```
f.close()
```

```
ls = jieba.lcut(t)
```

```
txt = " ".join(ls)
```

```
w = wordcloud.WordCloud(    font_path = "msyh.ttc",\  
    width = 1000, height = 700, background_color = "white", \  
    )
```

```
w.generate(txt)
```

```
w.to_file("grwordcloud.png")
```



```
#GovRptWordCloudv1.py
```

```
import jieba
```

```
import wordcloud
```

```
f = open("关于实施乡村振兴战略的意见.txt", "r", encoding="utf-8")
```

```
t = f.read()
```

```
f.close()
```

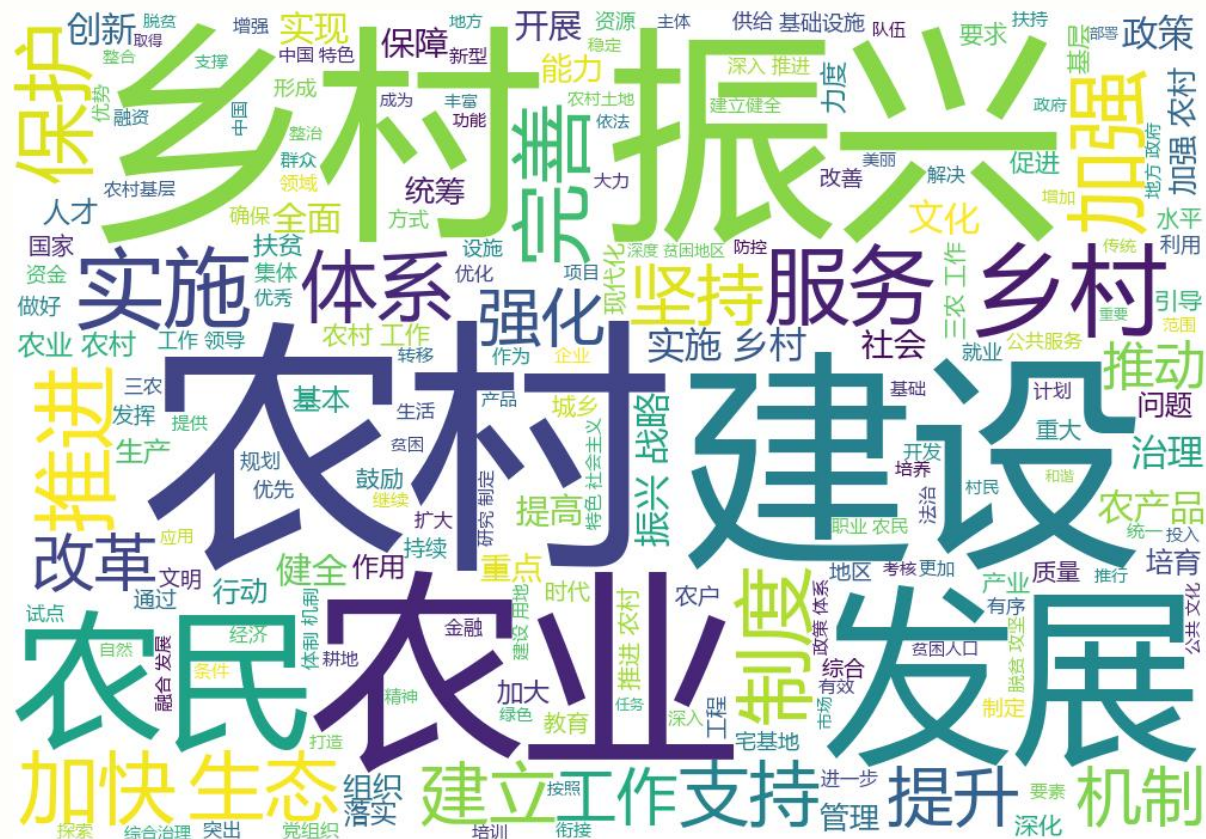
```
ls = jieba.lcut(t)
```

```
txt = " ".join(ls)
```

```
w = wordcloud.WordCloud(    font_path = "msyh.ttc",\  
    width = 1000, height = 700, background_color = "white", \  
    )
```

```
w.generate(txt)
```

```
w.to_file("grwordcloud.png")
```



# 2018年一号文件

```
#GovRptWordCloudv1.py
```

```
import jieba
```

```
import wordcloud
```

```
f = open("新时代中国特色社会主义思想.txt", "r", encoding="utf-8")
```

```
t = f.read()
```

```
f.close()
```

```
ls = jieba.lcut(t)
```

```
txt = " ".join(ls)
```

```
w = wordcloud.WordCloud(    font_path = "msyh.ttc",\  
    width = 1000, height = 700, background_color = "white", \  
    max_words = 15)
```

```
w.generate(txt)
```

```
w.to_file("grwordcloud.png")
```

推进政治  
发展  
中国特色  
加强社会  
建设  
制度  
社会主义  
人民  
中国

新时代中国特色社会主义思想



```
#GovRptWordCloudv1.py
```

```
import jieba
```

```
import wordcloud
```

```
f = open("关于实施乡村振兴战略的意见.txt", "r", encoding="utf-8")
```

```
t = f.read()
```

```
f.close()
```

```
ls = jieba.lcut(t)
```

```
txt = " ".join(ls)
```

```
w = wordcloud.WordCloud(    font_path = "msyh.ttc",\  
    width = 1000, height = 700, background_color = "white", \  
    max_words = 15)
```

```
w.generate(txt)
```

```
w.to_file("grwordcloud.png")
```



服务实施 乡村振兴 加快  
推进 加强 制度  
农村 发展  
完善 保障 乡村

# 2018年一号文件

**准备好电脑，与老师一起编码吧！**



# "政府工作报告词云"实例讲解(下)



# 政府工作报告词云

更有形的词云



fivestar.png



bitlogo.png

```
#GovRptWordCloudv2.py
```

```
import jieba
```

```
import wordcloud
```

```
from imageio import imread
```

```
mask = imread("fivestar.png")
```

```
f = open("新时代中国特色社会主义.txt", "r", encoding="utf-8")
```

```
t = f.read()
```

```
f.close()
```

```
ls = jieba.lcut(t)
```

```
txt = " ".join(ls)
```

```
w = wordcloud.WordCloud(    font_path = "msyh.ttc", mask = mask\
    width = 1000, height = 700, background_color = "white", \
    )
```

```
w.generate(txt)
```

```
w.to_file("grwordcloud.png")
```



# 新时代中国特色社会主义



# 2018年一号文件

```
#GovRptWordCloudv2.py
```

```
import jieba
```

```
import wordcloud
```

```
from imageio import imread
```

```
mask = imread("bitlogo.png")
```

```
f = open("新时代中国特色社会主义思想.txt", "r", encoding="utf-8")
```

```
t = f.read()
```

```
f.close()
```

```
ls = jieba.lcut(t)
```

```
txt = " ".join(ls)
```

```
w = wordcloud.WordCloud(    font_path = "msyh.ttc", mask = mask\
    width = 1000, height = 700, background_color = "white", \
    )
```

```
w.generate(txt)
```

```
w.to_file("grwordcloud.png")
```



# 新时代中国特色社会主义



# 2018年一号文件





"政府工作报告词云"举一反三

# 举一反三

## 扩展能力

- 了解wordcloud更多参数，扩展词云能力
- 特色词云：设计一款属于自己的特色词云风格
- 更多文件：用更多文件练习词云生成

