Python基础语法精讲

嵩 天

一二维数据格式化

嵩 天

一二维数据格式化

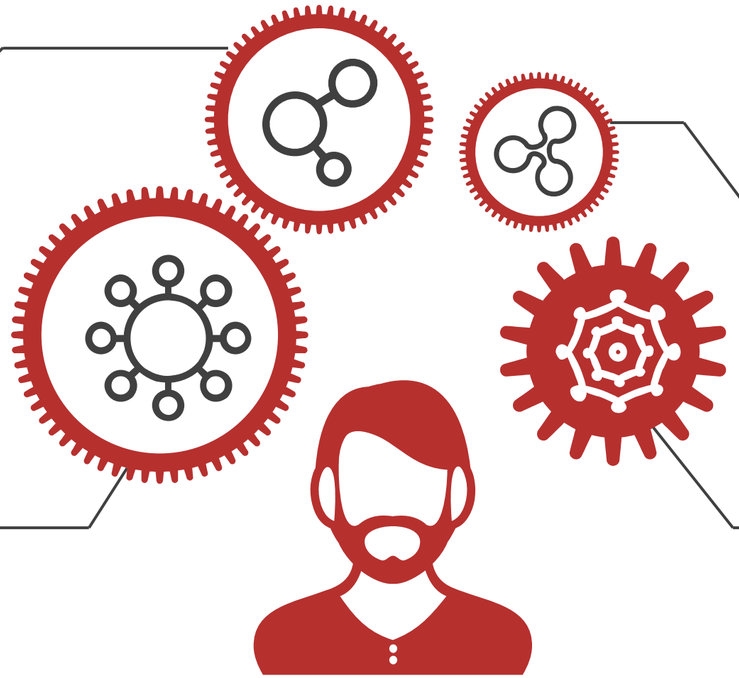
单元开篇

Python123

单元开篇

(2) 一维数据的表示与处理

(3) 二维数据的表示与处理



(1) 数据组织的维度

(4) CSV格式与一二维数据存储

一二维数据格式化

—二维数据格式化

数据组织的维度

数据组织的维度

维度：一组数据的组织形式

3.14

3.1413　3.1404

3.1398　3.1401

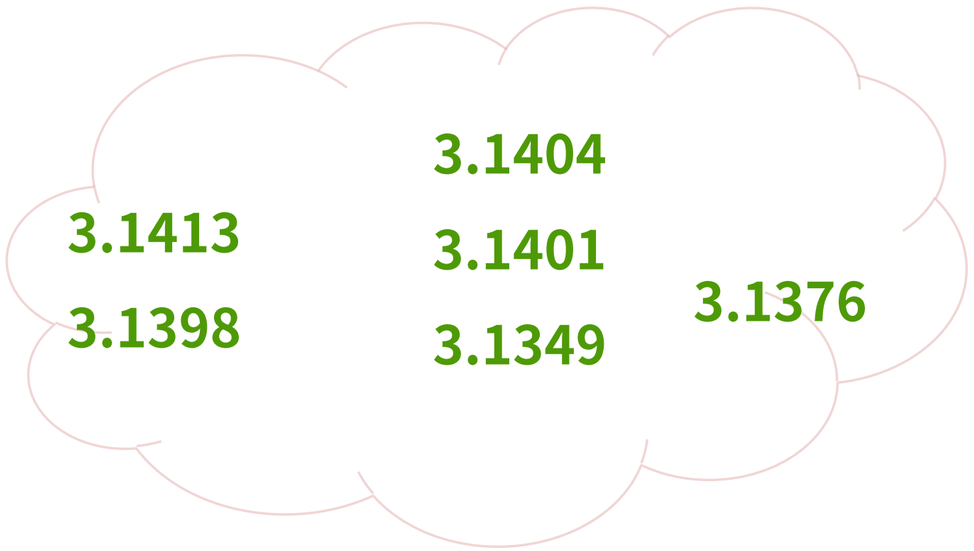
3.1349

一个数据表达一个含义

一组数据表达一个或多个含义

数据组织的维度

从一组数据到一组有组织的数据



3.1413,3.1398,3.1404,3.1401,3.1349,3.1376

或

3.1398, 3.1349, 3.1376

3.1413, 3.1404, 3.1401

数据组织的维度

数据的维度

- 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
- 二维数据：由多个一维数据构成，是一维数据的组合形式，最常用
- 多维数据：由一维或二维数据在新维度上扩展形成
- 高维数据：利用“键值对”展示数据间关系的数据组织方式，最常用

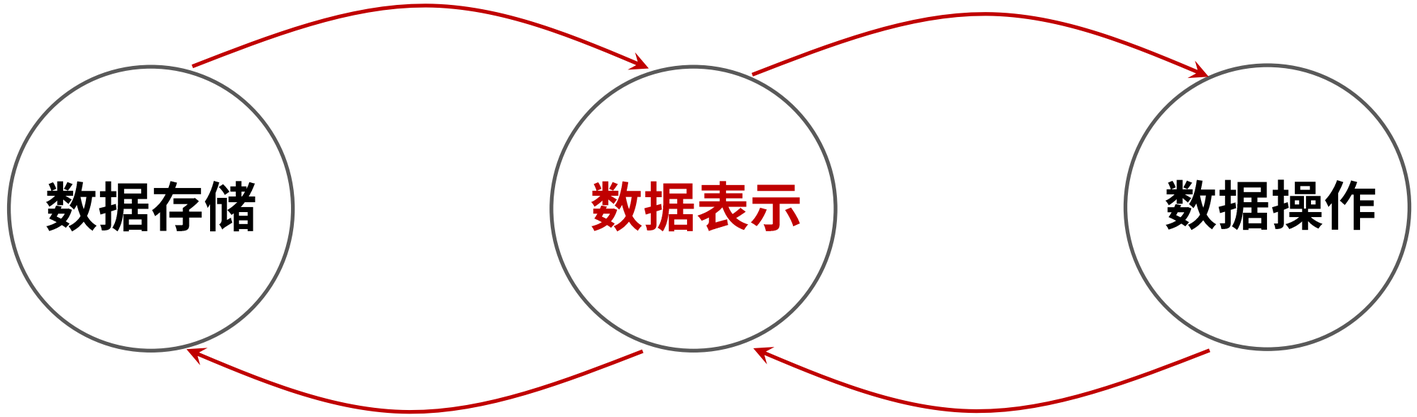
数据组织的维度

数据的维度

- 一维数据：对应列表（有序）和集合（无序）等类型
- 二维数据：对应列表（二维列表）类型
- 多维数据：对应列表（多维列表）类型
- 高维数据：对应字典类型

数据的操作周期

存储 <-> 表示 <-> 操作



存储格式　　数据类型　　操作方式

—二维数据格式化

一维数据的表示与处理

Python123

一维数据的表示

两种表示方式

- 如果数据间有序：使用列表类型

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

- 如果数据间无序：使用集合类型

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

一维数据的存储

- 空格分隔

  中国 美国 日本 德国 法国 英国 意大利

- 逗号分隔

  中国,美国,日本,德国,法国,英国,意大利

- 其他特殊符号分隔

  中国$美国$日本$德国$法国$英国$意大利

哪种好?

一维数据的存储

- 空格分隔

  中国 美国 日本 德国 法国 英国 意大利

- 逗号分隔

  中国,美国,日本,德国,法国,英国,意大利

- 其他特殊符号分隔

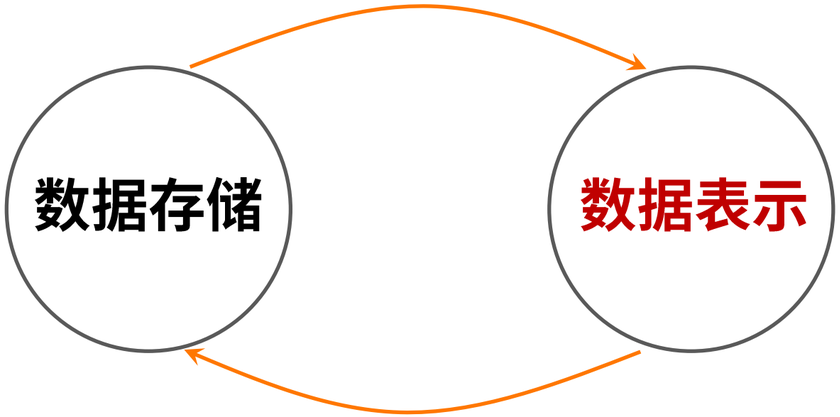
  中国$美国$日本$德国$法国$英国$意大利

哪种好?

从标准化角度，逗号分隔的CSV方式好

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

存储格式　　**数据类型**

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']
```

一维数据的读入处理

从逗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

采用逗号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

采用特殊符号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()
```

一维数据的表示与处理

- 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()

—二维数据格式化

二维数据的表示与处理

Python 123

二维数据的表示

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

列表表示方式

- 二维数据本质上是表格数据
- 二维列表表达数据的层次关系

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

二维数据据的存储

- 元数据（自定义数据）
- Excel格式
- 数据库格式
- CSV格式

哪种好？

从应用广泛性角度，CSV格式好

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()
```

注意：每行形成的列表被当作一个元素

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()
```

注意：每个列表元素本身写入到CSV的一行中

二维数据的逐一处理

采用二层循环

```
ls = [[], [], []] #二维列表
for row in ls:
    for column in row:
        print(ls[row][column])
        #其他处理代码
```

—二维数据格式化

CSV格式与一二维数据存储

Python 123

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及所有数据库软件可读入并导出，一般编辑软件可编辑

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格式的特殊说明

- 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

二维数据的格式化

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

二维数据的格式化

精化小结

- 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()

一二维数据格式化

单元小结

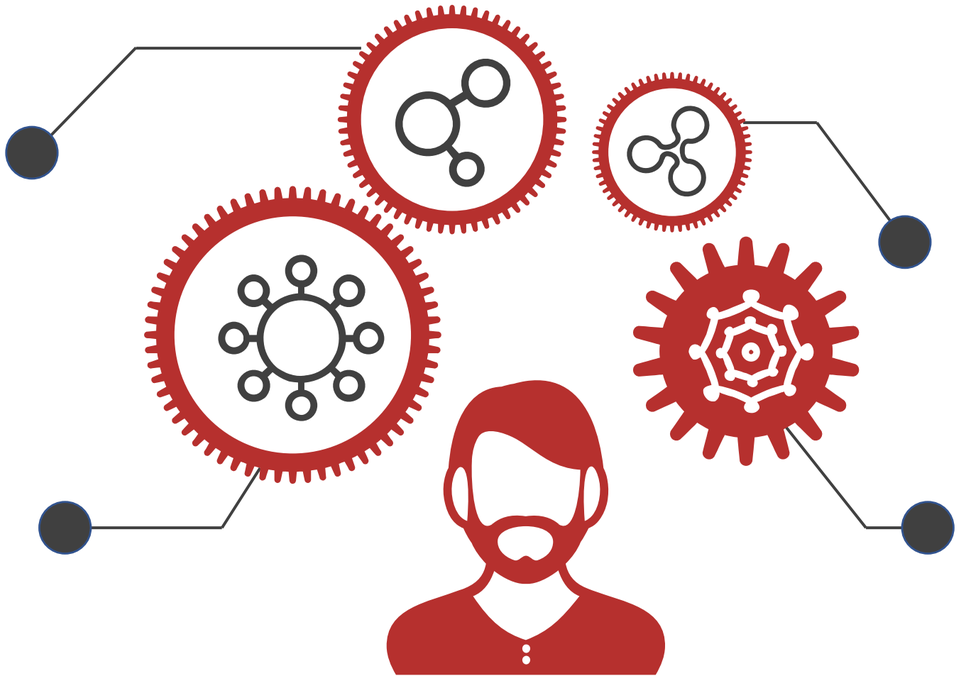
单元开篇

(2) 一维数据的表示与处理

列表/集合表示、CSV等格式存储

(1) 数据组织的维度

一维、二维、多维、高维



(3) 二维数据的表示与处理

二维列表表示、.split()、.join()

(4) CSV格式与一二维数据存储

行列存储、 特殊问题、存取方法

一二维数据格式化


Python 123
Thank you