

Python程序语言入门与应用



python

Life is short, use Python
人生苦短，我用Python



课程回顾



❖ 第5章 函数和模块

- ❖ 函数的基本使用
- ❖ 函数的参数传递
- ❖ datetime库的使用
- ❖ 实例7：七段数码管绘制
- ❖ 代码复用和模块化设计
- ❖ 函数的递归
- ❖ 实例8：科赫曲线绘制
- ❖ Python内置函数



上周练习题



5.1 实现isOdd()函数，参数为整数。参数如果是奇数，返回真（True），否则返回假（False）。

```
def isOdd(number):  
    if number % 2 != 0:  
        return True  
    else:  
        return False
```

其下的代码只有在.py文件作为脚本直接执行时才会被执行，而import到其他脚本中是不会被执行的，从而使当前的.py文件既可以直接运行，也可以作为模块被其他.py文件导入。

```
if __name__ == '__main__':  
    number = int(input("请输入一个整数: "))  
    if isOdd(number):  
        print("{}是一个奇数.".format(number))  
    else:  
        print("{}不是一个奇数.".format(number))
```

```
> python 5-1isOdd.py  
请输入一个整数: 13  
13是一个奇数。
```

```
> python 5-1isOdd.py  
请输入一个整数: 20  
20不是一个奇数。
```

运行示例



作业实例



作业1

```
def isOdd(num):  
    if num%2==0:  
        return False  
    else:  
        return True  
n=eval(input("请输入一个整数: "))  
a = isOdd(n)  
def isOddPrint(a):  
    if a==True: ←  
        → print("这是一个奇数。")  
    elif a==False: ←  
        → print("这是一个偶数。")  
print(isOddPrint(a))
```

一般将所有自定义函数放在代码头部

作业2

```
def isodd(number):  
    if number%2==1:  
        True  
    else: ← 缺少return  
        False  
c='奇数' if isodd(int(input('请  
输入一个整数')))) else "偶数"  
print(c)
```

作业3

```
def isOdd(x):  
    if x % 2 == 1:  
        return True  
else:    return False
```



上周练习题



5.2 实现multi()函数，参数个数不限，返回所有参数的乘积。

```
def multi(*num):  
    product = 1  
    for n in num:  
        product *= n  
    return product  
  
if __name__ == '__main__':  
    multiplication = multi(1, 3, 5, 7, 9)  
    print("所有参数的乘积是: ", multiplication)
```

```
> python 5-2multi.py  
所有参数的乘积是: 945
```

运行示例



上周练习题



5.3 使用datetime库，获取系统时间并按照自己喜欢个一种格式输出。

```
from datetime import datetime

today = datetime.now()
print(today)
print(today.strftime("%Y-%m-%d %H:%M:%S"))
```

```
> python 5-3mysysdatetime.py
2019-10-18 08:36:48.736266
2019-10-18 08:36:48
```

运行示例



作业实例



作业1

```
import datetime as dt
a = dt.datetime.now()
print("现在是{}年{}月{}日{}时{}分{}秒"
      ".format(a.year, a.month, a.day, a.hour,
a.minute, a.second))
```

作业2

```
from datetime import datetime
ty=datetime.now()
print(ty)
print("今天是{0:%Y}年{0:%b}.{0:%d},{0:%H}:{0:%M}".format(ty))
ny=datetime(2020,1,1,0,0,0)
print(type(ny))
print(type(ty))
d=(ny-ty).days
print("2019年还剩{:d}天".format(d))
```



新乡医学院

Python程序语言入门与应用

深入Python

第六章 Python组合数据类型



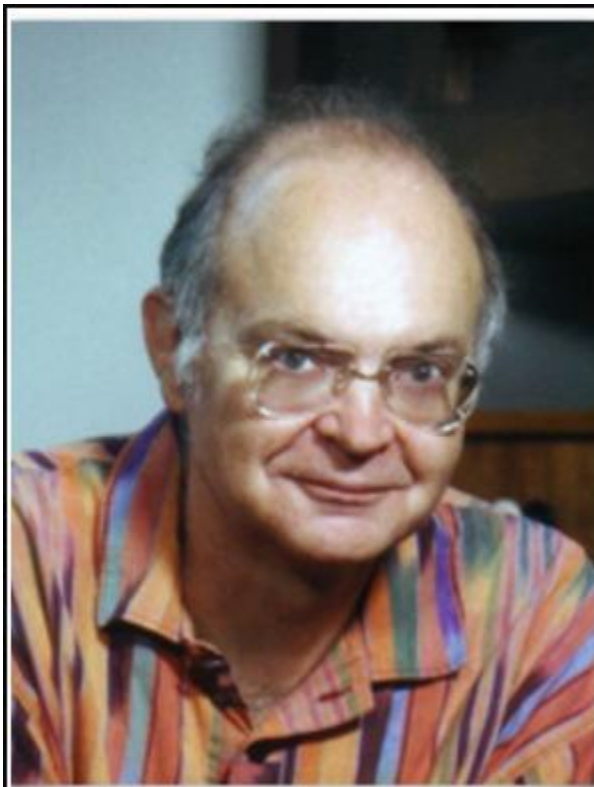
郭长江

changjiangguo@xxmu.edu.cn

生命科学技术学院

新乡医学院





Computers are good at following
instructions, but not at reading your
mind.

— Donald Knuth —

AZ QUOTES

计算机根据指令执行，而不是人的意志。

——唐纳德·克努特

排版软件Tex发明人《计算机程序设计的艺术》作者



学习目标




基本要求

掌握

-  列表、字典的表示与使用

理解

-  3种基本组合数据类型

-  元组、集合等类型的表示和使用

了解

-  组合数据类型进行文本词频统计



本课概要



第6章 Python组合数据类型

- 6.1 组合数据类型概述
- 6.2 列表类型和操作
- 6.3 实例：基本统计值计算
- 6.4 字典类型和操作
- 6.5 jieba库的使用
- 6.6 实例：文本词频统计



6.1 组合数据类型概述



6.1 组合数据类型概述



❧ 数据处理

❧ 单个变量表示的数据

❧ 数字、字符串（第3章 基本数据类型）

❧ 多个变量组成的一组数据

❧ 一组单词{python, data, class}，计算每个单词长度

❧ 一个年级学生信息，统计性别比例

❧ 组合数据类型

❧ 将多个同类型或不同类型的数据组织起来，提供单一表示



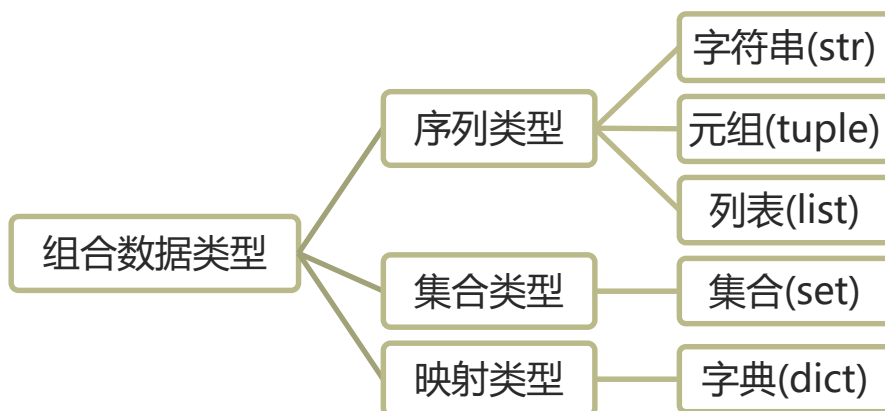
6.1 组合数据类型概述



组合数据类型

分为3类

- 序列类型：元素向量，元素之间存在先后关系，可通过序号访问（索引），元素之间不排他，可重复。
- 集合类型：元素集合，元素之间无序，相同元素在集合中唯一存在。
- 映射类型："键-值"数据项的组合，每个元素是一个键-值对，键名唯一，通过键名访问值。





6.1 组合数据类型概述



序列类型

- 序列类型是一维元素向量，来源于数学中的数列概念

$$S = s_0, s_1, s_2, \dots, s_{n-1}$$

- 元素之间存在顺序关系，序列中可存在数值相同但位置不同的元素。 `[1, 1, 2, 3, 5, 8, 13]`

- 支持成员关系操作符(in)、长度计算函数(len)、切片([N:M])。 `if 'A' in 'ABC', len('python'), 'python'[2:4]`

- 元素本身也可以是序列类型。 `['abc', (1,2,3), 123]`



图 6.2 序列类型的索引体系



6.1 组合数据类型概述



序列类型

字符串 "abcdefg"

- 单一字符的有序组合

- 一般看做基本数据类型

元组 (10, 'abc', 1024)

- 包含0个或多个数据项的**不可变**序列类型

- 生成之后是**固定的**，任何数据项不能替换或删除

列表 [10, 'abc', 1024]

- 包含0个或多个数据项的**可变**序列类型

- 生成之后**可修改**，任何数据项可替换或删除



6.1 组合数据类型概述



序列类型

元组(tuple)

- 比较特殊的序列类型，生成后是固定的，各数据项不能修改
- 常用于表达固定数据项、函数多返回值、多变量同步赋值、循环遍历等
- 表示方法：逗号和圆括号() (可选) tuple(s)函数可以生成元组

```
>>> animal = "cat", "dog", "tiger"
>>> animal
('cat', 'dog', 'tiger')
>>> color = ("red", 0x001100, "blue", animal)
>>> color
('red', 4352, 'blue', ('cat', 'dog', 'tiger'))
>>> color[2]
'blue'
>>> color[-1][1]
'dog'
```

元组嵌套

tuple() 可将字符串，列表，字典，集合转化为元组



6.1 组合数据类型概述



序列类型

12个通用操作符和函数

操作符	描述
<code>x in s</code>	如果x是s的元素，返回True，否则返回False
<code>x not in s</code>	如果x不是s的元素，返回True，否则返回False
<code>s + t</code>	连接s和t
<code>s * n</code> 或 <code>n * s</code>	将序列s复制n次
<code>s[i]</code>	索引，返回序列s的第i个元素
<code>s[i:j]</code>	切片，返回序列s第i到j个元素（不包含）的子序列
<code>s[i:j:k]</code>	步骤切片，返回序列s第i到j个元素以k为步数的子序列
<code>len(s)</code>	序列s的元素个数（长度）
<code>min(s)</code>	序列s中的最小元素
<code>max(s)</code>	序列s中的最大元素
<code>s.index(x[, i[, j]])</code>	序列s中从i开始到j位置中第一次出现元素x的位置
<code>s.count(x)</code>	序列s中出现x的总次数



6.1 组合数据类型概述



☞ 集合类型(set)

- ☞ 与数学中集合概念一致
- ☞ 包含0个或多个数据项的**无序**组合，元素不可重复
- ☞ 元素类型只能是**固定数据类型**，如数字、字符串和元组，而列表、字典和集合都是可变数据类型，不能作为集合元素出现
- ☞ 能进行**哈希运算**(`hash()`)的类型可作为集合元素

```
>>> hash(1.23)
530343892119149569
>>> hash('python')
-4478659149325180121
>>> hash((1,2,'a'))
5161284069083137971
```

```
>>> hash([1,'a'])
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    hash([1,'a'])
TypeError: unhashable type: 'list'
```



6.1 组合数据类型概述



☞ 集合类型(set)

- ☞ 无序组合，没有索引和位置概念，也不能切片
- ☞ 集合中的元素可以动态增加和删除
- ☞ 表示方法：逗号和花括号{}

```
>>> S = {425, "BIT", (10, "CS"), 42.5}
>>> S
{425, 42.5, 'BIT', (10, 'CS')}
>>> T = {425, "BIT", (10, "CS"), 425, "BIT", 42.5}
>>> T
{425, 42.5, 'BIT', (10, 'CS')}
```

- ☞ 使用集合类型可过滤重复元素



6.1 组合数据类型概述



集合类型

set(x)函数可以生成集合，x可以是任何组合数据类型，返回一个无重复且无序的元素集合

```
>>> set("apple")
{'l', 'e', 'p', 'a'}
>>> set((1,1,2,3,5))
{1, 2, 3, 5}
>>> set(['a', 'b', 'c', 'a'])
{'c', 'b', 'a'}
```



6.1 组合数据类型概述



集合类型

10个操作符

操作符	描述
$S - T$ 或 <code>S.difference(T)</code>	返回一个新集合，包含在集合S但不在集合T中的元素（差集）
$S -= T$ 或 <code>S.difference_update(T)</code>	更新集合S，包含在集合S但不在集合T中的元素
$S \& T$ 或 <code>S.intersection(T)</code>	返回一个新集合，包含同时在集合S和集合T中的元素（交集）
$S \&= T$ 或 <code>S.intersection_update(T)</code>	更新集合S，包含同时在集合S和集合T中的元素
$S \wedge T$ 或 <code>S.symmetric_difference(T)</code>	返回新集合，包含在集合S和T中但不同时在其中的元素（补集）
$S \wedge= T$ 或 <code>S.symmetric_difference_update(T)</code>	更新集合S，包含在集合S和T中但不同时在其中的元素
$S T$ 或 <code>S.union(T)</code>	返回一个新集合，包含在集合S和T中的所有元素（并集）
$S = T$ 或 <code>S.update(T)</code>	更新集合S，包含在集合S和T中的所有元素
$S \leq T$ 或 <code>S.issubset(T)</code>	如果S与T相同或者S是T的子集，返回True，否则返回False
$S \geq T$ 或 <code>S.issuperset(T)</code>	如果S与T相同或者S是T的超集，返回True，否则返回False

$S < T$: S是否是T的真子集

$S > T$: S是否是T的真超集



6.1 组合数据类型概述



集合类型

10个操作函数或方法

操作符	描述
S.add(x)	如果数据项x不在集合S中，将x增加到s
S.clear()	移除S中的所有数据项（清空）
S.copy()	返回集合S的一个副本
S.pop()	随机返回集合S的一个元素（S不能为空）
S.discard(x)	如果x在S中，移除该元素
S.remove(x)	如果x在S中，移除该元素，如果不在则产生异常
S.isdisjoint(T)	如果S和T中没有相同元素，返回True
len(S)	返回集合S的元素个数
x in S	如果x是S的元素，返回True，否则返回False
x not in S	如果x不是S的元素，返回True，否则返回False



6.1 组合数据类型概述



集合类型

使用场景

成员关系测试、元素去重、删除数据项

```
>>> "BIT" in {"python", "BIT", 123, "good"}      # 成员关系测试
True
>>> tup = ("python", "BIT", 123, "BIT", 123)      # 元素去重
>>> set(tup)
{123, 'BIT', 'python'}
>>> newtup = tuple(set(tup)-{'python'})           # 去重并删除数据项
>>> newtup
(123, 'BIT')
```

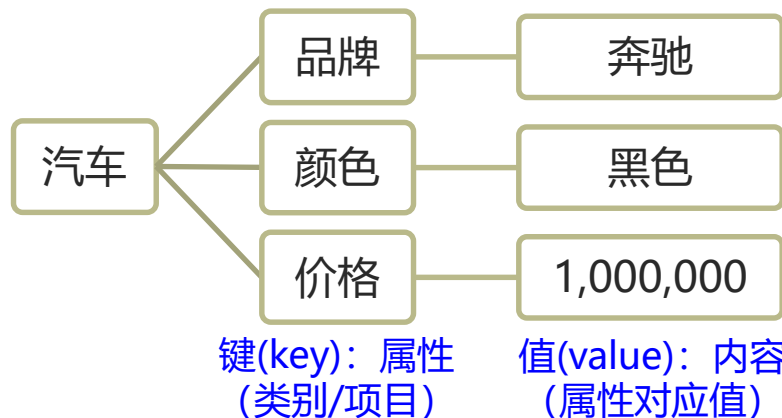



6.1 组合数据类型概述



映射类型(map)

- "键-值"数据项的组合，每个元素都是一个键值对。
- 元素之间是无序的，键值对是二元关系（映射）



- 映射类型实现：字典 (dict)



6.2 列表类型和操作



6.2 列表类型和操作



❖ 列表(list)

- ❖ 包含0个或多个对象引用的有序序列
- ❖ 无长度限制，可自由增删元素，使用灵活
- ❖ 表示方法
 - ❖ 逗号(,)和方括号([])
 - ❖ list()函数可以将元组或字符串转化为列表

```
>>> ls = [123, 'python', 123, [1, 2, 3]]
>>> ls
[123, 'python', 123, [1, 2, 3]]
>>> ls[3][-1]
3
>>> list("python")
['p', 'y', 't', 'h', 'o', 'n']
>>> list((1, 2, 3, 4))
[1, 2, 3, 4]
```

[], list()表示空列表



6.2 列表类型和操作



🔗 列表引用

🔗 列表必须通过显式的数据赋值才能生成，简单将一个列表赋值给另外一个列表不会生成新的列表对象，而是对原列表对象的一个**引用**，真实数据值只存储一份。

```
>>> ls = [1, 2, 3]
>>> lt = ls
#lt是对ls的引用, 并不包含真实数据
>>> lt
[1, 2, 3]
>>> ls[0] = 0
>>> ls
[0, 2, 3]
>>> lt
[0, 2, 3]
>>> lt[1] = 0
>>> ls
[0, 0, 3]
>>> lt
[0, 0, 3]
```



6.2 列表类型和操作



列表类型的操作

除序列类型操作外，列表还有14个常用函数或方法。

操作符	描述
<code>ls[i] = x</code>	替换列表ls第i数据项为x
<code>ls[i:j] = lt</code>	用列表lt替换列表ls第i到第j项（不含）数据
<code>ls[i:j:k] = lt</code>	用列表lt替换列表ls第i到第j项（不含）以k为步数数据
<code>del ls[i:j]</code>	删除列表ls第i到第j（不含）项数据，同 <code>ls[i:j] = []</code>
<code>del ls[i:j:k]</code>	删除列表ls第i到第j（不含）项以k为步数的数据
<code>ls += lt</code> 或 <code>ls.extend(lt)</code>	将列表lt元素增加到ls中
<code>ls *= n</code>	更新列表ls，其元素重复n次
<code>ls.append(x)</code>	在列表ls最后增加一个元素x
<code>ls.clear()</code>	删除列表ls中的所有元素
<code>ls.copy()</code>	生成一个新列表，复制ls中的所有元素
<code>ls.insert(i,x)</code>	在列表ls的第i位置增加元素x
<code>ls.pop(i)</code>	在列表ls的第i项元素取出并删除该元素，i默认为-1
<code>ls.remove(x)</code>	将列表ls中出现的第一个元素x删除
<code>ls.reverse()</code>	列表ls中的元素顺序反转



6.2 列表类型和操作



列表操作

```
>>> vlist = list(range(5))
>>> vlist
[0, 1, 2, 3, 4]
>>> len(vlist[2:])
3
>>> 2 in vlist
True
>>> vlist[3] = '3'
>>> vlist
[0, 1, 2, '3', 4]
>>> vlist[1:3] = ['1', '2']
>>> vlist
[0, '1', '2', '3', 4]
```

```
>>> vlist = [0,1,2,3,4]
>>> vlist[1:3] = ['1']
>>> vlist
[0, '1', 3, 4]
>>> vlist[1:2] = [1,2]
>>> vlist
[0, 1, 2, 3, 4]
```

```
>>> vlist.extend([5,6])
>>> vlist
[0, '1', '2', '3', 4, 5, 6]
>>> vlist.append(7)
>>> vlist
[0, '1', '2', '3', 4, 5, 6, 7]
>>> vlist.append([8,9])
>>> vlist
[0, '1', '2', '3', 4, 5, 6, 7, [8, 9]]
>>> vlist.pop()
[8, 9]
>>> vlist
[0, '1', '2', '3', 4, 5, 6, 7]
>>> vlist.reverse()
>>> vlist
[7, 6, 5, 4, '3', '2', '1', 0]
```

当时用一个列表改变另一个列表值时，不要
求2个列表的长度一样，但遵循“多增少减”
的原则。



6.2 列表类型和操作



列表遍历

通过for-in语句对其元素进行遍历

```
for <变量名> in <列表名>:  
    <语句块>
```

```
>>> ls = ['a', 'b', 'c']  
>>> for e in ls:  
    print(e, end=' ')  
abc  
>>> for n, e in enumerate(ls):  
    print(n, e)  
0 a  
1 b  
2 c
```

enumerate()函数用于将一个可遍历的数据对象(如列表、元组或字符串)组合为一个索引序列,同时列出数据和数据索引,一般用在for循环中。



6.3 实例：基本统计值计算



6.3 实例：基本统计值计算



基本统计值

统计学中一组数据的基本统计值

平均值、标准差、最大值、最小值、中位数、及上下四分位数

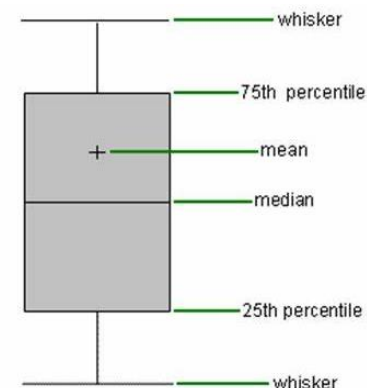
算术平均数

$$m = (\sum_{i=0}^{n-1} s_i) / n$$

标准差 $d = \sqrt{(\sum_{i=0}^{n-1} (s_i - m)^2) / (n - 1)}$

中位数：数据数量为奇数时为最中间位置的数据，数量为偶数时为最中间两个数据的平均值。

四分位数（Quartile）：把所有数据由小到大排序后处于25%和75%位置上的值。



箱线图





6.3 实例：基本统计值计算



❧ 基本统计值计算

❧ IPO描述

- ❧ 输入：从用户输入、文件、网络等途径获取一组数据
- ❧ 处理：适当的数据结构和算法
- ❧ 输出：平均值、标准差、最大值、最小值和中位数

❧ 函数方式编程

- ❧ `getNum()`函数获取数据
- ❧ `mean()`函数计算平均值
- ❧ `dev()`函数计算标准差
- ❧ `max_median_min()`函数计算最大值、中位数、最小值



6.3 实例：基本统计值计算



基本统计值计算

6.1 CalStatistics

```
def getNum(): # 获取用户输入
    nums = []
    iNumStr = input("请输入一行数字 (逗号隔开) :")
    nums = [eval(i) for i in iNumStr.split(",")]
    return nums
```

```
def mean(numbers): # 计算平均值
    s = 0
    for num in numbers:
        s = s + num
    return s / len(numbers)
```

```
def dev(numbers, mean): # 计算标准差
    sdev = 0
    for num in numbers:
        sdev = sdev + (num - mean) ** 2
    return (sdev / (len(numbers) - 1)) ** 0.5
```

`ls = [<表达式> for <变量> in lt]`
对lt每个元素进行某种操作后所产生的所有元素重新组合为新的列表ls

```
def max_median_min(numbers):
    new = sorted(numbers)
    size = len(numbers)
    mini = new[0]
    maxi = new[-1]
    med = 0
    if size % 2 == 0:
        med = (new[size//2-1] + new[size//2]) / 2
    else:
        med = new[size//2]
    return (maxi, med, mini)
```

```
n = getNum()
m = mean(n)
dev = dev(n, m)
mmm = max_median_min(n)
print("输入的数字为: ", n)
print("平均值:{0:.3f}, 标准差:{1:.3f}".format(m, dev))
print("最大值:{0[0]}, 中位数:{0[1]}, 最小值:{0[2]}".format(mmm))
```

列表排序之**sorted()函数**：不改变列表本身，返回新列表
`sorted(ls)`:从小到大正向排序；`sorted(ls, reverse=True)`:从大到小反向排序
列表排序之**sort()方法**：改变列表本身，没有返回值
`ls.sort()`:从小到大正向排序；`ls.sort(reverse=True)`:从大到小反向排序



6.3 实例：基本统计值计算



基本统计值计算

运行结果1:

请输入一行数字（逗号隔开）:1,2,3,4,5,6,7,8,9

输入的数字为: [1, 2, 3, 4, 5, 6, 7, 8, 9]

平均值:5.000, 标准差:2.739

最大值:9, 中位数:5, 最小值:1

运行结果2:

请输入一行数字（逗号隔开）:12,19,5,23,45,56,32,68,26,40

输入的数字为: [12, 19, 5, 23, 45, 56, 32, 68, 26, 40]

平均值:32.600, 标准差:19.766

最大值:68, 中位数:29.0, 最小值:5



6.4 字典类型和操作

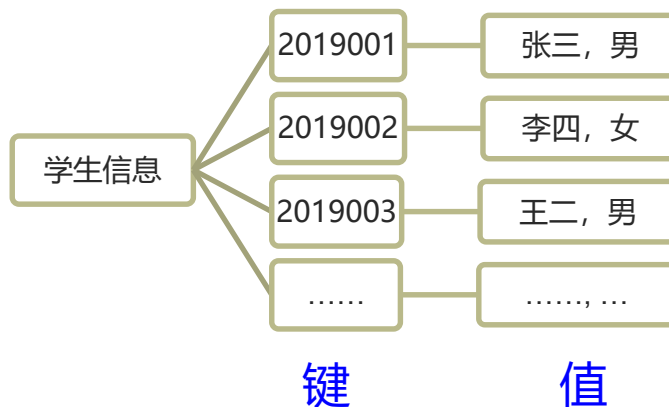


6.4 字典类型和操作



数据存储与检索

- 列表按照整数索引来查找数据，查找方式不灵活
- 如根据学号检索学生姓名、性别信息，列表无能为力
- 根据一个信息查找另外一个信息，构成"键值对"
 - 姓名：电话、用户名：密码、国家名：首都名等等



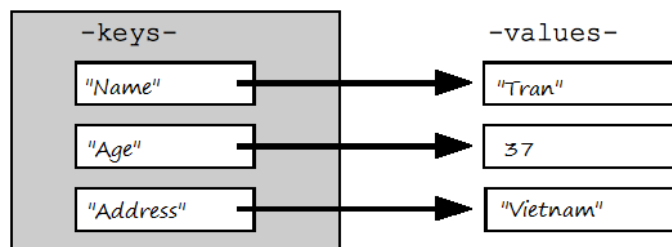
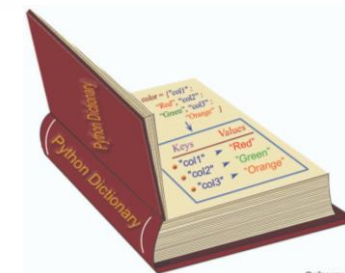


6.4 字典类型和操作



字典(dict)

- 包含0个或多个键值对的集合
- 没有长度限制，可以根据键索引值的内容



表示方法:

- 逗号(,)、冒号(:)和花括号({})
 - 键值以冒号连接，键值对以逗号隔开
 - 键名通常为字符串，也可以是数字，且不能重复
- {<键1>:<值1>, <键2>:<值2>, ..., <键n>:<值n>}

一个键只能
对应一个值

{ }为空字典



6.4 字典类型和操作



字典

- 是集合类型的延续，键值对元素没有顺序之分
- 可通过键索引访问值 $\langle \text{值} \rangle = \langle \text{字典变量} \rangle [\langle \text{键} \rangle]$
- 键值对赋值可以修改和增加元素 $\langle \text{字典变量} \rangle [\langle \text{键} \rangle] = \langle \text{值} \rangle$

访问

修改

增加

```
>>> Dcountry={"中国":"北京", "美国":"华盛顿", "法国":"巴黎"}
>>> print(Dcountry)
{'中国': '北京', '美国': '华盛顿', '法国': '巴黎'}
>>> Dcountry["中国"]
'北京'
>>> Dcountry["中国"]="大北京"
>>> print(Dcountry)
{'中国': '大北京', '美国': '华盛顿', '法国': '巴黎'}
>>> Dcountry["英国"]="伦敦"
>>> print(Dcountry)
{'中国': '大北京', '美国': '华盛顿', '法国': '巴黎', '英国': '伦敦'}
```




6.4 字典类型和操作



字典类型的操作

字典类型的操作函数和方法

操作符	描述
d.keys()	返回字典d的所有的键信息
d.values()	返回字典d的所有的值信息
d.items()	返回字典d所有的键值对
d.get(key, <default>)	字典d中存在键key返回相应值，否则返回默认值default
d.pop(key, <default>)	字典d中存在键key返回相应值，并删除键值对，否则返回默认值default
d.popitem()	随机从字典d中取出一个键值对，以元组(key, value)形式返回
d.clear()	删除字典d所有键值对（清空）
del d[key]	删除字典d中键值对key:value
key in d	如果字典d中存在键key，返回True，否则返回False



6.4 字典类型和操作



字典操作

```
>>> Dcountry={"中国":"北京", "美国":"华盛顿"}
>>> Dcountry.keys()
dict_keys(['中国', '美国'])
>>> list(Dcountry.keys())
['中国', '美国']
>>> Dcountry.values()
dict_values(['北京', '华盛顿'])
>>> Dcountry.items()
dict_items([('中国', '北京'), ('美国', '华盛顿')])
>>> '中国' in Dcountry
True
>>> Dcountry.get("美国", "东京")
'华盛顿'
>>> Dcountry.get("日本", "东京")
'东京'
```

遍历字典:

```
for key in d:
    for key in d.keys():
    for value in d.values():
    for key, value in d.items():
```

```
>>> for key in Dcountry:
        print(key)
'中国'
'美国'
>>> for value in Dcountry.values():
        print(value)
'北京'
'华盛顿'
>>> for key, value in
Dcountry.items():
        print(key,value)
'中国 北京'
'美国 华盛顿'
```



6.5 jieba库的使用



6.5 jieba库的使用



分词

英文文本提取单词

"China is a great country"

字符串的split()方法即可实现

中文缺少分隔符而分词困难

"**中国**是一个**伟大**的国家"

jieba (结巴)库

第三方中文分词函数库

pip指令安装

`pip install jieba` →

```
Collecting jieba
  Downloading
https://files.pythonhosted.org/packages/71/46/c6f9179f73b818d58
27202ad1c4a94e371a29473b7f043b736b4dab6b8cd/jieba-0.39.zip
(7.3MB)
| 7.3MB 656kB/s
Building wheels for collected packages: jieba
  Building wheel for jieba (setup.py) ... done
  Created wheel for jieba: filename=jieba-0.39-cp37-none-any.whl
  size=7282597
  sha256=d7a80becf96bacc05456c105f72178cce2cf7799fec73b35faa
  89c1b6d96b103
  Stored in directory:
  C:\Users\iSynBio\AppData\Local\pip\Cache\wheels\c9\c7\63\a9ec0
  322ccc7c365fd51e475942a82395807186e94f0522243
  Successfully built jieba
Installing collected packages: jieba
Successfully installed jieba-0.39
```



6.5 jieba库的使用






jieba库

分词原理

利用中文词库，将待分词的内容与分词词库进行比对，通过图结构和动态规划方法找到最大概率的词组。

三种分词模式

-  精确模式：对句子精确分词，适合文本分析
-  全模式：获取所有词语，速度快，不能消除歧义
-  搜索引擎模式：对长词再次切分，适合搜索引擎分词



6.5 jieba库的使用



jieba库主要函数

函 数	描 述
<code>jieba.cut(s)</code>	精确模式，返回一个可迭代的数据类型
<code>jieba.cut(s, cut_all=True)</code>	全模式，输出文本s中所有可能的单词
<code>jieba.cut_for_search(s)</code>	搜索引擎模式，适合搜索引擎建立索引的分词结果
<code>jieba.lcut(s)</code>	精确模式，返回一个列表类型，建议使用
<code>jieba.lcut(s, cut_all=True)</code>	全模式，返回一个列表类型，建议使用
<code>jieba.lcut_for_search(s)</code>	搜索引擎模式，返回一个列表类型，建议使用
<code>jieba.add_word(w)</code>	向分词词典中增加新词w

```
>>> import jieba
>>> jieba.lcut("中华人民共和国是一个伟大的国家")
['中华人民共和国', '是', '一个', '伟大', '的', '国家']
>>> jieba.lcut("中华人民共和国是一个伟大的国家", cut_all=True)
['中华', '中华人民', '中华人民共和国', '华人', '人民', '人民共和国',
'共和', '共和国', '国是', '一个', '伟大', '的', '国家']
>>> jieba.lcut_for_search("中华人民共和国是一个伟大的国家")
['中华', '华人', '人民', '共和', '共和国', '中华人民共和国', '是',
'一个', '伟大', '的', '国家']
```



6.6 实例：文本词频统计



- 对一段文本（文章、报道、小说、论文等），统计其中频繁出现的词语，进而简要分析文章内容。
- 相当于词语计数器：以词为键、以次数为值
- 常见方式：词云（word cloud）





6.6 实例：文本词频统计



🔗 英文词频统计

🔗 文本来源：莎士比亚《哈姆雷特》 (hamlet.txt)

🔗 步骤

- 🔗 将特殊标点符号替换为空格，分解并提取英文单词，单词统一为小写形式；
- 🔗 以字典类型存储单词，并对单词进行计数；
- 🔗 对单词的统计值从高到低排序，排除语法性词汇 (the, you, a, in等) ，输出前10高频词语



6.6 实例：文本词频统计



《哈姆雷特》词频统计

```
# 6.2CalHamlet.py
def getText():
    txt = open("hamlet.txt", "r").read()    # 读取小说内容
    txt = txt.lower()
    for ch in '!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~':
        txt = txt.replace(ch, " ")    # 将文本中特殊字符替换为空格
    return txt

excludes = {"the", "and", "of", "you", "a", "i", "my", "in", "to", "well", "this", "it", "that", "is", "not", "his",
            "but", "with", "was", "for", "your", "me", "be", "as", "he", "what", "him", "at", "them", "so", "have", "will", "do",
            "on", "all", "our", "by", "like", "or", "no", "we", "are", "if", "good", "come", "shall", "sir", "o", "thou", "they",
            "now", "more", "let", "from", "her", "how"} # 排除语法性词汇

hamletTxt = getText()
words = hamletTxt.split()
counts = {}
for word in words:
    if word not in excludes:
        counts[word] = counts.get(word, 0) + 1    # word存在时取其次数, 否则创建并且次数设为0
top_counts = sorted(list(counts.values()), reverse=True)[:10] # 按值倒序排序, 取前10
for count in top_counts:
    for key in counts:
        if counts[key] == count:
            print("{0:<10}{1:>5}".format(key, count))
```

(运行结果)

```
>python 6.2CalHamlet.py
hamlet      462
lord        309
king        194
horatio     157
claudius    120
queen       117
polonius    116
laertes     103
gertrude    95
thy         87
```



6.6 实例：文本词频统计



🔗 中文词频统计

- 🔗 文本来源：罗贯中《三国演义》(threekingdoms.txt)
- 🔗 数百个历史人物，到底谁是主角：
 - 🔗 刘备？诸葛亮？曹操？孙权？关羽？张飞？ ...
- 🔗 Python程序统计各人物的出场次数
 - 🔗 jieba库进行中文精确分词
 - 🔗 排除无关词汇，如**将军**、**二人**、**不可**、**荆州**
 - 🔗 合并人名称谓，如玄德=刘备，孔明=诸葛亮，孟德=丞相=曹操



6.6 实例：文本词频统计



```
# 6.3CalThreeKingdoms.py
import jieba
excludes = {"将军", "却说", "荆州", "二人", "不可", "不能", "如此"}
txt = open("threekingdoms.txt", "r", encoding='utf-8').read()
words = jieba.lcut(txt)
counts = {}
for word in words:
    if len(word) == 1: # 过滤单字词汇
        continue
    elif word in ("孔明", "孔明曰"):
        rword = "诸葛亮"
    elif word in ("关公", "云长", "云长曰"):
        rword = "关羽"
    elif word in ("翼德", "翼德曰"):
        rword = "张飞"
    elif word in ("玄德", "玄德曰"):
        rword = "刘备"
    elif word in ("孟德", "丞相", "孟德曰"):
        rword = "曹操"
    else:
        rword = word
    if rword not in excludes:
        counts[rword] = counts.get(rword, 0) + 1
top_counts = sorted(list(counts.values()), reverse=True)[:5]
for count in top_counts:
    for key in counts:
        if counts[key] == count:
            print("{0:<10}{1:>5}".format(key, count))
```

```
> python .\6.3CalThreeKingdoms.py
Building prefix dict from the
default dictionary ...
Loading model from cache
D:\Temp\jieba.cache
Loading model cost 0.615 seconds.
Prefix dict has been built
succesfully.
```

曹操	1451
诸葛亮	1383
刘备	1252
关羽	784
张飞	376

《三国演义》词频
统计结果



能力扩展



Python制作词云

第三方库: wordcloud

三国演义词云分析
[ThreeKingdomsWordcloud.py](#)

代码

https://github.com/amueller/word_cloud

说明

https://amueller.github.io/word_cloud/



Single Word



Minimal Example



Masked wordcloud




本章要点



Python组合数据类型

 序列、集合、映射

 元组

 列表

 字典

 jieba库分词



程序练习题



6.1 重复元素判定

代码文件名: 6.1haveRepeat.py

编写一个函数haveRepeat(), 接受列表作为参数, 如果一个元素在列表中出现了不止一次, 则返回True, 但不改变原列表值。

6.2 文本字符分析

代码文件名: 6.2chrSort.py

编程程序接受字符串, 按字符出现频率的降序打印字母。

6.3 《红楼梦》人物统计

代码文件名: 6.3calHongLou.py

编程程序统计《红楼梦》中前20位出场最多的人物。

.py代码文件打包(6.学号+姓名)发送到
python_xxmu@163.com



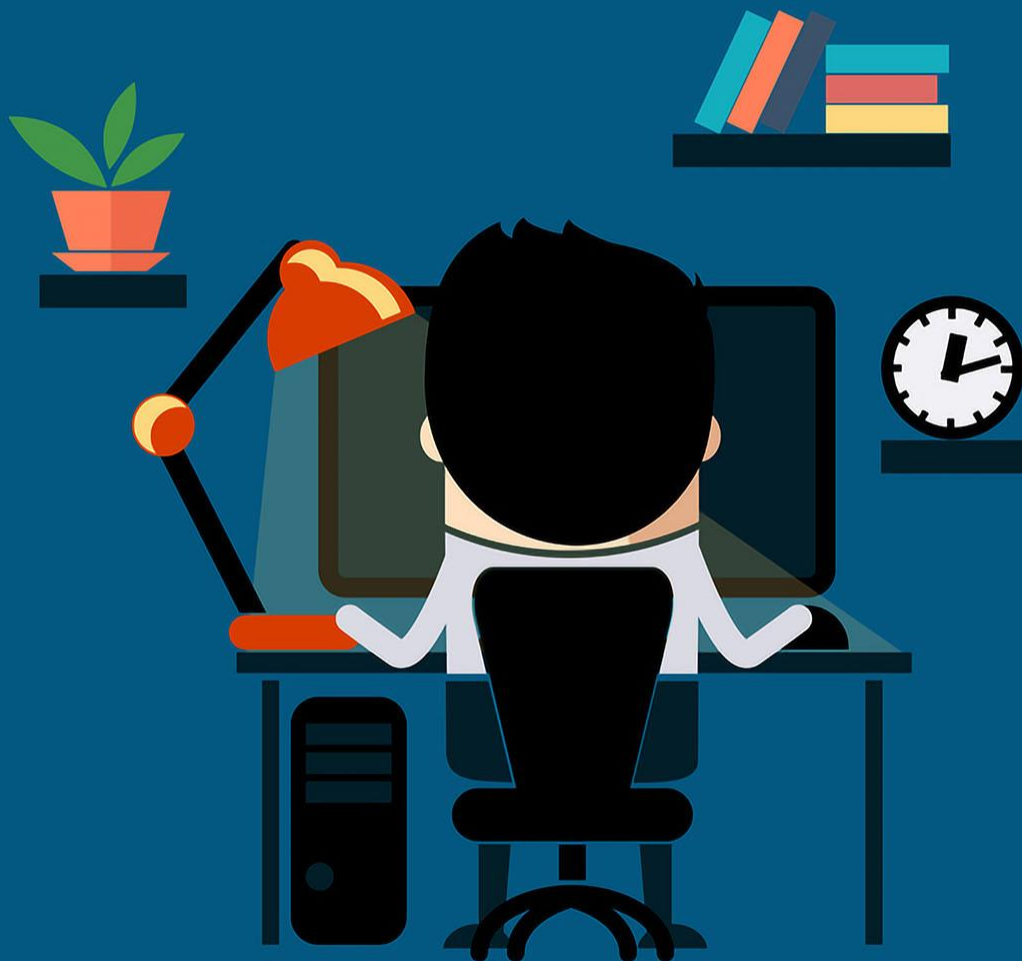
下周课程



第七章 Python文件操作

- 文件的读写方法
- 一二维数据的存储格式和读写方法
- PIL库和json库的使用

编程辣么好，还等什么？开始学习吧！



Programing is an Art