

# python

The Python logo, consisting of two interlocking snakes, one blue and one yellow, is positioned below the word "python".

```
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")

for i in range(4):
    turtle.circle(40, 80)
    turtle.circle(-40, 80)
    turtle.fd(40)
    turtle.circle(16, 180)
    turtle.fd(40 * 2/3)
```

Python语言程序设计

## 第6章 辅学内容

---



嵩 天  
北京理工大学





# 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*



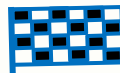
# 程序的循环结构

- *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





# 本课概要

# 第6章 组合数据类型

3.14

从一个数据到一组数据

一个数据  
表达一个含义



3.1413 3.1404  
3.1398 3.1401 3.1376  
3.1349

一组数据

表达一个或多个含义



# 第6章 组合数据类型

- 6.1 集合类型及操作

- 6.2 序列类型及操作

元组类型

列表类型

- 6.3 实例9: 基本统计值计算

- 6.4 字典类型及操作

- 6.5 模块5: jieba库的使用

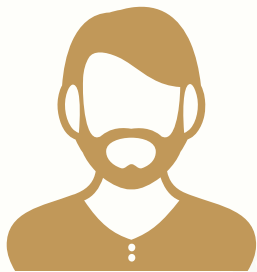
- 6.6 实例10: 文本词频统计



# 第6章 组合数据类型

## 方法论

- Python三种主流组合数据类型的使用方法



## 实践能力

- 学会编写处理一组数据的程序





# 练习与作业

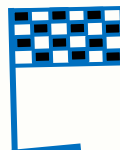
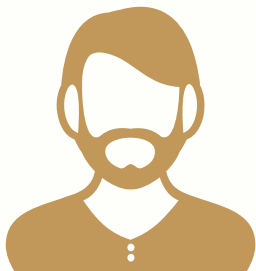
# 第6章 组合数据类型

## 练习 (可选)

- 5道编程 @Python123

## 测验

- 10道单选+2道编程 @Python123









# python

The Python logo, consisting of two interlocking snakes, one blue and one yellow, is positioned below the word "python".

```
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")

for i in range(4):
    turtle.circle(40, 80)
    turtle.circle(-40, 80)
    turtle.circle(40, 80/2)
    turtle.fd(40)
    turtle.circle(16, 180)
    turtle.fd(40 * 2/3)
```

Python语言程序设计

# 集合类型及操作

---



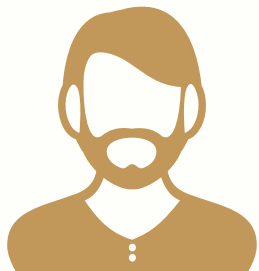
嵩 天  
北京理工大学





# 单元开篇

# 集合类型及操作



- 集合类型定义
- 集合操作符
- 集合处理方法
- 集合类型应用场景





# 集合类型定义

# 集合类型的定义

**集合是多个元素的无序组合**

- **集合类型与数学中的集合概念一致**
- **集合元素之间无序，每个元素唯一，不存在相同元素**
- **集合元素不可更改，不能是可变数据类型    为什么？**

# 集合类型的定义

**集合是多个元素的无序组合**

- **集合用大括号 {} 表示，元素间用逗号分隔**
- **建立集合类型用 {} 或 set()**
- **建立空集合类型，必须使用set()**

# 集合类型的定义

```
>>> A = {"python", 123, ("python",123)} #使用{}建立集合
{123, 'python', ('python', 123)}

>>> B = set("pypy123") #使用set()建立集合
{'1', 'p', '2', '3', 'y'}

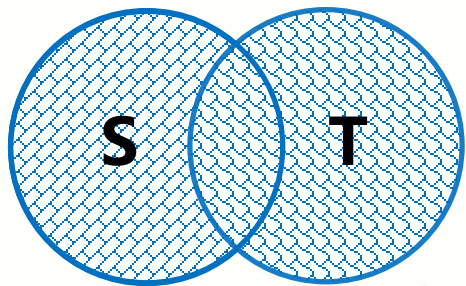
>>> C = {"python", 123, "python",123}
{'python', 123}
```



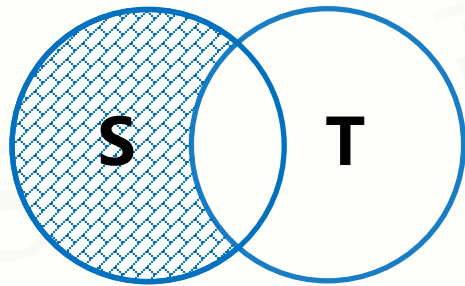


# 集合操作符

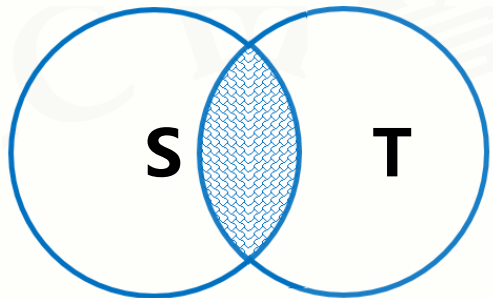
# 集合间操作



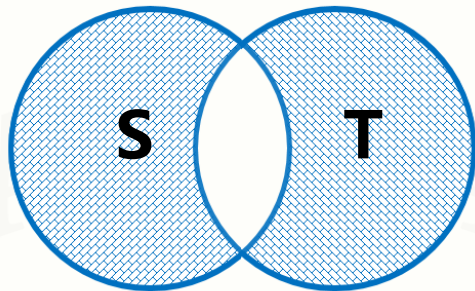
$S | T$   
并



$S - T$   
差



$S \& T$   
交



$S \wedge T$   
补

# 集合操作符

## 6个操作符

操作符及应用	描述
$S \mid T$	并，返回一个新集合，包括在集合S和T中的所有元素
$S - T$	差，返回一个新集合，包括在集合S但不在T中的元素
$S \& T$	交，返回一个新集合，包括同时在集合S和T中的元素
$S \wedge T$	补，返回一个新集合，包括集合S和T中的非相同元素
$S \leq T$ 或 $S < T$	返回True/False，判断S和T的子集关系
$S \geq T$ 或 $S > T$	返回True/False，判断S和T的包含关系

# 集合操作符

## 4个增强操作符

操作符及应用	描述
$S \mid= T$	并，更新集合S，包括在集合S和T中的所有元素
$S -= T$	差，更新集合S，包括在集合S但不在T中的元素
$S \&= T$	交，更新集合S，包括同时在集合S和T中的元素
$S \wedge= T$	补，更新集合S，包括集合S和T中的非相同元素

# 集合类型的定义

```
>>> A = {"p", "y", 123}
```

```
>>> B = set("pypy123")
```

```
>>> A-B
```

```
{123}
```

```
>>> B-A
```

```
{'3', '1', '2'}
```

```
>>> A&B
```

```
{'p', 'y'}
```

```
>>> A|B
```

```
{'1', 'p', '2', 'y', '3', 123}
```

```
>>> A^B
```

```
{'2', 123, '3', '1'}
```



# 集合处理方法

# 集合处理方法

操作函数或方法	描述
S.add(x)	如果x不在集合S中，将x增加到S
S.discard(x)	移除S中元素x，如果x不在集合S中，不报错
S.remove(x)	移除S中元素x，如果x不在集合S中，产生KeyError异常
S.clear()	移除S中所有元素
S.pop()	随机返回S的一个元素，更新S，若S为空产生KeyError异常

# 集合处理方法

操作函数或方法	描述
S.copy()	返回集合S的一个副本
len(S)	返回集合S的元素个数
x in S	判断S中元素x, x在集合S中, 返回True, 否则返回False
x not in S	判断S中元素x, x不在集合S中, 返回True, 否则返回False
set(x)	将其他类型变量x转变为集合类型



# 集合处理方法

```
>>> try:
>>> A = {"p", "y", 123}
>>> for item in A:
    print(item, end="")
p123y
>>> A
{'p', 123, 'y'}
```

```
>>> while True:
    print(A.pop(), end="")
except:
    pass
p123y
>>> A
set()
```



# 集合类型应用场景

# 集合类型应用场景

## 包含关系比较

```
>>> "p" in {"p", "y", 123}
```

```
True
```

```
>>> {"p", "y"} >= {"p", "y", 123}
```

```
False
```

# 集合类型应用场景

**数据去重：集合类型所有元素无重复**

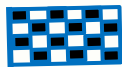
```
>>> ls = ["p", "p", "y", "y", 123]
>>> s = set(ls)      # 利用了集合无重复元素的特点
{'p', 'y', 123}
>>> lt = list(s)     # 还可以将集合转换为列表
['p', 'y', 123]
```



# 单元小结

# 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





# 小花絮

# 为什么嵩老师不建议低龄儿童学编程？

背景：近年来，在英语/奥数之外，少儿编程又成为了新的商业热点，学生家长的焦虑感陡增。

观点：嵩老师建议青少年学编程，但不建议**低龄**儿童（小学三年级及以下）学习编程。

- 儿童学习需要**符合**认知规律成长及发展心理学
- 编程思维逻辑性很强，将**限制**低龄儿童创造性思维培养
- 进阶编程需要懂得**大量**计算机知识，不适合低龄儿童掌握







# python

The Python logo, consisting of two interlocking snakes, one blue and one yellow, is positioned below the word "python".

```
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")

for i in range(4):
    turtle.circle(40, 80)
    turtle.circle(-40, 80)
    turtle.circle(40, 80/2)
    turtle.fd(40)
    turtle.circle(16, 180)
    turtle.fd(40 * 2/3)
```

Python语言程序设计

# 序列类型及操作

---



嵩 天  
北京理工大学





# 单元开篇

# 序列类型及操作



- 序列类型定义
- 序列处理函数及方法
- 元组类型及操作
- 列表类型及操作
- 序列类型应用场景





# 序列类型定义

# 序列类型定义

**序列是具有先后关系的一组元素**

- **序列是一维元素向量，元素类型可以不同**
- **类似数学元素序列：**  $s_0, s_1, \dots, s_{n-1}$
- **元素间由序号引导，通过下标访问序列的特定元素**

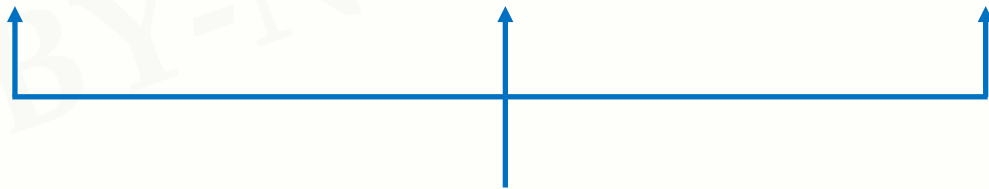
# 序列类型定义

序列是一个基类类型

字符串类型

元组类型

列表类型

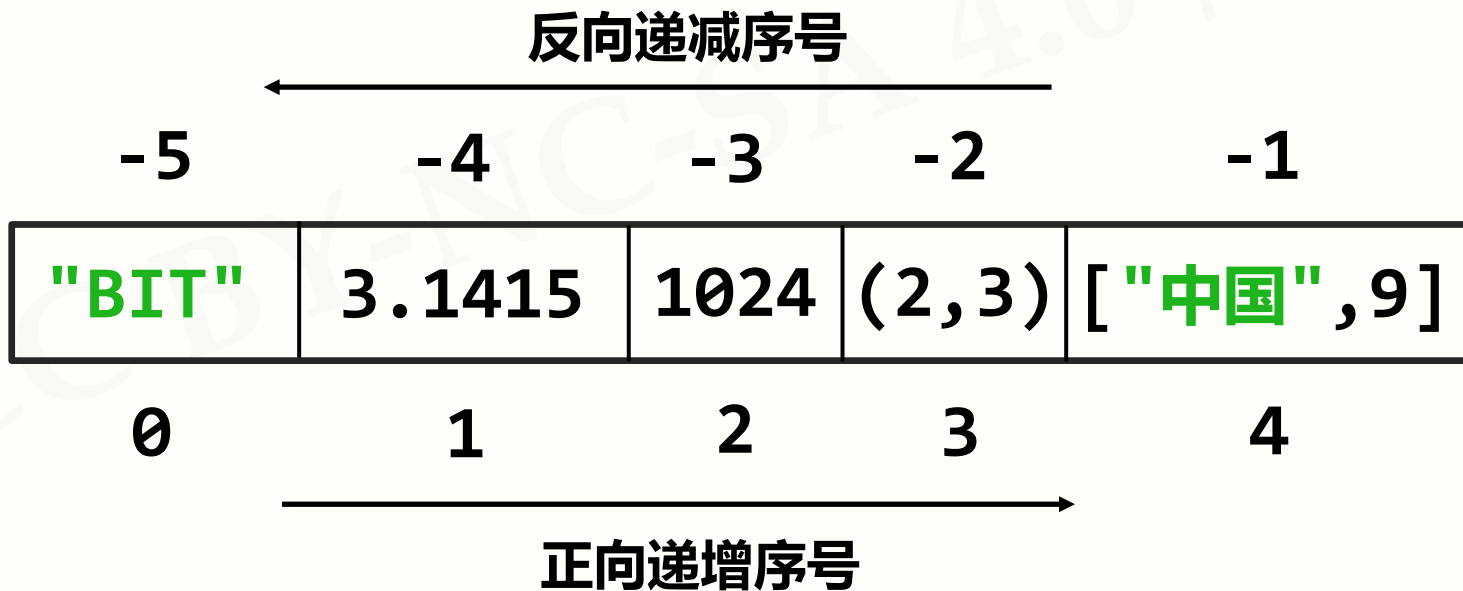


序列类型



# 序列类型定义

## 序号的定义





# 序列处理函数及方法

# 序列类型通用操作符

## 6个操作符

操作符及应用	描述
<code>x in s</code>	如果x是序列s的元素，返回True，否则返回False
<code>x not in s</code>	如果x是序列s的元素，返回False，否则返回True
<code>s + t</code>	连接两个序列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s中的第i个元素，i是序列的序号
<code>s[i: j]</code> 或 <code>s[i: j: k]</code>	切片，返回序列s中第i到j以k为步长的元素子序列

# 序列类型操作实例

```
>>> ls = ["python", 123, ".io"]
```

```
>>> ls[::-1]
```

```
['.io', 123, 'python']
```

```
>>> s = "python123.io"
```

```
>>> s[::-1]
```

```
'oi.321nohtyp'
```

# 序列类型通用函数和方法

## 5个函数和方法

函数和方法	描述
<code>len(s)</code>	返回序列s的长度，即元素个数
<code>min(s)</code>	返回序列s的最小元素，s中元素需要可比较
<code>max(s)</code>	返回序列s的最大元素，s中元素需要可比较
<code>s.index(x)</code> 或 <code>s.index(x, i, j)</code>	返回序列s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>	返回序列s中出现x的总次数

# 序列类型操作实例

```
>>> ls = ["python", 123, ".io"]
```

```
>>> len(ls)
```

```
3
```

```
>>> s = "python123.io"
```

```
>>> max(s)
```

```
'y'
```



# 元组类型及操作

# 元组类型定义

元组是序列类型的一种扩展

- 元组是一种序列类型，一旦创建就不能被修改
- 使用小括号 () 或 tuple() 创建，元素间用逗号，分隔
- 可以使用或不使用小括号

```
def func():  
    return 1,2
```



# 元组类型定义

```
>>> creature = "cat", "dog", "tiger", "human"
```

```
>>> creature
```

```
('cat', 'dog', 'tiger', 'human')
```

```
>>> color = (0x001100, "blue", creature)
```

```
>>> color
```

```
(4352, 'blue', ('cat', 'dog', 'tiger', 'human'))
```

# 元组类型操作

## 元组继承序列类型的全部通用操作

- 元组继承了序列类型的全部通用操作
- 元组因为创建后不能修改，因此没有特殊操作
- 使用或不使用小括号

# 元组类型操作

```
>>> creature = "cat", "dog", "tiger", "human"
```

```
>>> creature[::-1]
```

```
('human', 'tiger', 'dog', 'cat')
```

```
>>> color = (0x001100, "blue", creature)
```

```
>>> color[-1][2]
```

```
'tiger'
```



# 列表类型及操作

# 列表类型定义

**列表是序列类型的一种扩展，十分常用**

- **列表是一种序列类型，创建后可以随意被修改**
- **使用方括号 [] 或list() 创建，元素间用逗号，分隔**
- **列表中各元素类型可以不同，无长度限制**

# 列表类型定义

```
>>> ls = ["cat", "dog", "tiger", 1024]
```

```
>>> ls
```

```
['cat', 'dog', 'tiger', 1024]
```

```
>>> lt = ls
```

```
>>> lt
```

```
['cat', 'dog', 'tiger', 1024]
```

ls

lt



```
['cat', 'dog', 'tiger', 1024]
```

方括号 [] 真正创建一个列表，赋值仅传递引用

# 列表类型操作函数和方法

函数或方法	描述
<code>ls[i] = x</code>	替换列表ls第i元素为x
<code>ls[i: j: k] = lt</code>	用列表lt替换ls切片后所对应元素子列表
<code>del ls[i]</code>	删除列表ls中第i元素
<code>del ls[i: j: k]</code>	删除列表ls中第i到第j以k为步长的元素
<code>ls += lt</code>	更新列表ls，将列表lt元素增加到列表ls中
<code>ls *= n</code>	更新列表ls，其元素重复n次

# 列表类型操作

```
>>> ls = ["cat", "dog", "tiger", 1024]
```

```
>>> ls[1:2] = [1, 2, 3, 4]
```

```
['cat', 1, 2, 3, 4, 'tiger', 1024]
```

```
>>> del ls[::3]
```

```
[1, 2, 4, 'tiger']
```

```
>>> ls*2
```

```
[1, 2, 4, 'tiger', 1, 2, 4, 'tiger']
```



# 列表类型操作函数和方法

函数或方法	描述
ls.append(x)	在列表ls最后增加一个元素x
ls.clear()	删除列表ls中所有元素
ls.copy()	生成一个新列表，赋值ls中所有元素
ls.insert(i,x)	在列表ls的第i位置增加元素x
ls.pop(i)	将列表ls中第i位置元素取出并删除该元素
ls.remove(x)	将列表ls中出现的第一个元素x删除
ls.reverse()	将列表ls中的元素反转

# 列表类型操作

```
>>> ls = ["cat", "dog", "tiger", 1024]
```

```
>>> ls.append(1234)
```

```
['cat', 'dog', 'tiger', 1024, 1234]
```

```
>>> ls.insert(3, "human")
```

```
['cat', 'dog', 'tiger', 'human', 1024, 1234]
```

```
>>> ls.reverse()
```

```
[1234, 1024, 'human', 'tiger', 'dog', 'cat']
```

# 列表功能默写

- 定义空列表lt
- 向lt新增5个元素
- 修改lt中第2个元素
- 向lt中第2个位置增加一个元素
- 从lt中第1个位置删除一个元素
- 删除lt中第1-3位置元素
- 判断lt中是否包含数字0
- 向lt新增数字0
- 返回数字0所在lt中的索引
- lt的长度
- lt中最大元素
- 清空lt

# 列表功能默写

- 定义空列表lt 

```
>>> lt = []
```
- 向lt新增5个元素 

```
>>> lt += [1,2,3,4,5]
```
- 修改lt中第2个元素 

```
>>> lt[2] = 6
```
- 向lt中第2个位置增加一个元素 

```
>>> lt.insert(2, 7)
```
- 从lt中第1个位置删除一个元素 

```
>>> del lt[1]
```
- 删除lt中第1-3位置元素 

```
>>> del lt[1:4]
```

# 列表功能默写

```
>>> 0 in lt
```

■ 判断lt中是否包含数字0

```
>>> lt.append(0)
```

■ 向lt新增数字0

```
>>> lt.index(0)
```

■ 返回数字0所在lt中的索引

```
>>> len(lt)
```

■ lt的长度

```
>>> max(lt)
```

■ lt中最大元素

```
>>> lt.clear()
```

■ 清空lt



# 序列类型应用场景

# 序列类型应用场景

## 数据表示：元组 和 列表

- 元组用于元素不改变的应用场景，更多用于固定搭配场景
- 列表更加灵活，它是最常用的序列类型
- 最主要作用：表示一组有序数据，进而操作它们

# 序列类型应用场景

## 元素遍历

*for* item *in* ls :

<语句块>

*for* item *in* tp :

<语句块>



# 序列类型应用场景

## 数据保护


- 如果不希望数据被程序所改变，转换成元组类型

```
>>> ls = ["cat", "dog", "tiger", 1024]
```

```
>>> lt = tuple(ls)
```

```
>>> lt
```

```
('cat', 'dog', 'tiger', 1024)
```



# 单元小结

# 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





# 小花絮

各种二维码，都扫过了吗？

一不小心，弄出这么多二维码，随缘吧...



这里面有：MOOC、公众号、教材、微信群、嵩老师







# python

The Python logo, consisting of two interlocking snakes, one blue and one yellow, is positioned below the word "python".

```
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")

for i in range(4):
    turtle.circle(40, 80)
    turtle.circle(-40, 80)
    turtle.fd(40)
    turtle.circle(16, 180)
    turtle.fd(40 * 2/3)
```

# 实例9: 基本统计值计算

---



嵩 天  
北京理工大学







# "基本统计值计算"问题分析

# 问题分析

## 基本统计值

- 需求：给出一组数，对它们有个概要理解
- 该怎么做呢？

总个数、求和、平均值、方差、中位数...

# 问题分析

## 基本统计值

- 总个数: `len()`

- 求和: `for ... in`

- 平均值: 求和/总个数

- 方差:

各数据与平均数差的平方和的平均数

- 中位数: 排序, 然后...

奇数找中间1个, 偶数找中间2个取平均



# "基本统计值计算"实例讲解

# 基本统计值计算

#CalStatisticsV1.py

```
def getNum():          #获取用户不定长度的输入
    nums = []
    iNumStr = input("请输入数字(回车退出): ")
    while iNumStr != "":
        nums.append(eval(iNumStr))
        iNumStr = input("请输入数字(回车退出): ")
    return nums
```

```
def mean(numbers):    #计算平均值
    s = 0.0
    for num in numbers:
        s = s + num
    return s / len(numbers)
```

- 获取多数据输入

- 通过函数分隔功能

# 基本统计值计算

```
def dev(numbers, mean): #计算方差
    sdev = 0.0
    for num in numbers:
        sdev = sdev + (num - mean)**2
    return pow(sdev / (len(numbers)-1), 0.5)
```

```
def median(numbers):    #计算中位数
    sorted(numbers)
    size = len(numbers)
    if size % 2 == 0:
        med = (numbers[size//2-1] + numbers[size//2])/2
    else:
        med = numbers[size//2]
    return med
```

```
n = getNum()
m = mean(n)
print("平均值:{},方差:{:.2},中位数:{}.".format(m, dev(n,m),median(n)))
```

- 获取多数据输入

- 通过函数分隔功能

**准备好电脑，与老师一起编码吧！**



# "基本统计值计算"举一反三



```
def dev(numbers, mean): #计算方差
```

```
    sdev = 0.0
```

```
    for num in numbers:
```

```
        sdev = sdev + (num - mean)**2
```

```
    return pow(sdev / (len(numbers)-1), 0.5)
```

```
def median(numbers):    #计算中位数
```

```
    sorted(numbers)
```

```
    size = len(numbers)
```

```
    if size % 2 == 0:
```

```
        med = (numbers[size//2-1] + numbers[size//2])/2
```

```
    else:
```

```
        med = numbers[size//2]
```

```
    return med
```

```
n = getNum()
```

```
m = mean(n)
```

```
print("平均值:{},方差:{:.2},中位数:{}.".format(m, dev(n,m),median(n)))
```

```
#CalStatisticsV1.py
```

```
def getNum():    #获取用户不定长度的输入
```

```
    nums = []
```

```
    iNumStr = input("请输入数字(回车退出): ")
```

```
    while iNumStr != "":
```

```
        nums.append(eval(iNumStr))
```

```
        iNumStr = input("请输入数字(回车退出): ")
```

```
    return nums
```

```
def mean(numbers):    #计算平均值
```

```
    s = 0.0
```

```
    for num in numbers:
```

```
        s = s + num
```

```
    return s / len(numbers)
```



# 举一反三

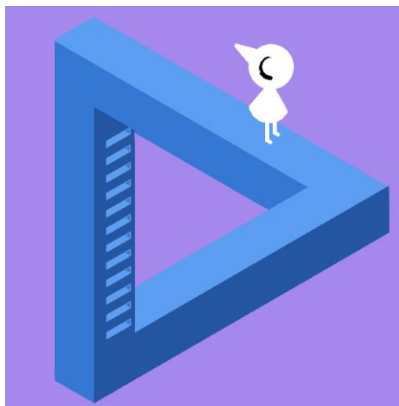
## 技术能力扩展

- **获取多个数据：从控制台获取多个不确定数据的方法**
- **分隔多个函数：模块化设计方法**
- **充分利用函数：充分利用Python提供的内置函数**



# 小花絮

# Turtle绘图作品



<https://python123.io>

- 每个小小成就汇成人生发展的历史坐标
- 回想小时候的第一幅画、第一次考试、第一个满分
- 请在Python123上留下你自己的turtle作品
- 此生只做一次！创意不限、精彩不限、回忆无限！







# python

The Python logo, consisting of two interlocking snakes, one blue and one yellow, is positioned below the word "python".

```
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")

for i in range(4):
    turtle.circle(40, 80)
    turtle.circle(-40, 80)
    turtle.circle(40, 80/2)
    turtle.fd(40)
    turtle.circle(16, 180)
    turtle.fd(40 * 2/3)
```

Python语言程序设计

# 字典类型及操作

---



嵩 天  
北京理工大学





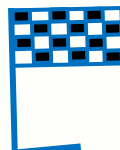
# 单元开篇



# 字典类型及操作



- 字典类型定义
- 字典处理函数及方法
- 字典类型应用场景





# 字典类型定义

# 字典类型定义

## 理解“映射”

- 映射是一种键(索引)和值(数据)的对应



# 字典类型定义

## 理解“映射”

- 映射是一种键(索引)和值(数据)的对应

内部颜色：蓝色

外部颜色：红色



"streetAddr" : "中关村南大街5号"  
"city" : "北京市"  
"zipcode" : "100081"

# 字典类型定义

## 理解“映射”

- 映射是一种键(索引)和值(数据)的对应

["python", 123, ".io"]



0



1



2



内部颜色：蓝色

外部颜色：红色

序列类型由0..N整数作为数据的默认索引

映射类型则由用户为数据定义索引

# 字典类型定义

字典类型是“映射”的体现

- 键值对：键是数据索引的扩展
- 字典是键值对的集合，键值对之间无序
- 采用大括号{}和dict()创建，键值对用冒号: 表示

{<键1>:<值1>, <键2>:<值2>, ... , <键n>:<值n>}

# 字典类型的用法

在字典变量中，通过键获得值

$\langle \text{字典变量} \rangle = \{ \langle \text{键1} \rangle : \langle \text{值1} \rangle, \dots, \langle \text{键n} \rangle : \langle \text{值n} \rangle \}$

$\langle \text{值} \rangle = \langle \text{字典变量} \rangle [\langle \text{键} \rangle] \quad \langle \text{字典变量} \rangle [\langle \text{键} \rangle] = \langle \text{值} \rangle$

**[ ] 用来向字典变量中索引或增加元素**

# 字典类型定义和使用

```
>>> d = {"中国": "北京", "美国": "华盛顿", "法国": "巴黎"}
```

```
>>> d
```

```
{'中国': '北京', '美国': '华盛顿', '法国': '巴黎'}
```

```
>>> d["中国"]
```

```
'北京'
```

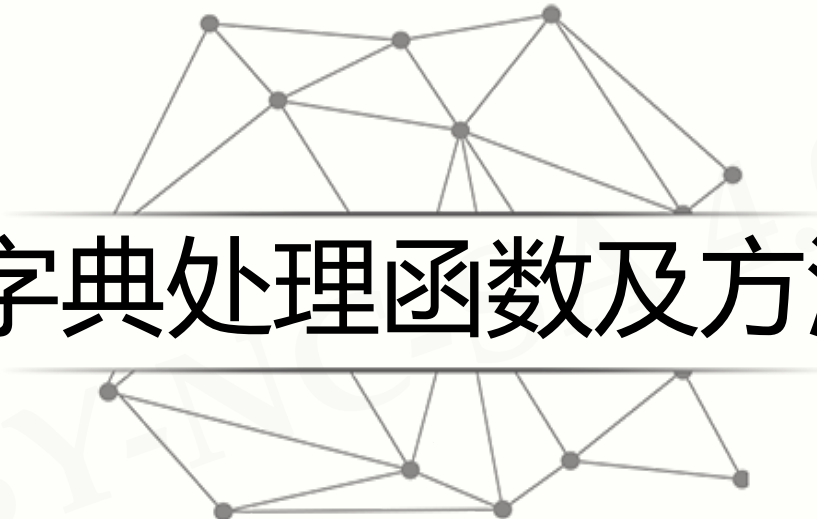
```
>>> de = {} ; type(de)
```

```
<class 'dict'>
```

**type(x)**

**返回变量x的类型**





# 字典处理函数及方法

# 字典类型操作函数和方法

函数或方法	描述
<code>del d[k]</code>	删除字典d中键k对应的数据值
<code>k in d</code>	判断键k是否在字典d中，如果在返回True，否则False
<code>d.keys()</code>	返回字典d中所有的键信息
<code>d.values()</code>	返回字典d中所有的值信息
<code>d.items()</code>	返回字典d中所有的键值对信息

# 字典类型操作

```
>>> d = {"中国":"北京", "美国":"华盛顿", "法国":"巴黎"}
```

```
>>> "中国" in d
```

```
True
```

```
>>> d.keys()
```

```
dict_keys(['中国', '美国', '法国'])
```

```
>>> d.values()
```

```
dict_values(['北京', '华盛顿', '巴黎'])
```

# 字典类型操作函数和方法

函数或方法	描述
d.get(k, <default>)	键k存在，则返回相应值，不在则返回<default>值
d.pop(k, <default>)	键k存在，则取出相应值，不在则返回<default>值
d.popitem()	随机从字典d中取出一个键值对，以元组形式返回
d.clear()	删除所有的键值对
len(d)	返回字典d中元素的个数

# 字典类型操作

```
>>> d = {"中国":"北京", "美国":"华盛顿", "法国":"巴黎"}
```

```
>>> d.get("中国","伊斯兰堡")
```

```
'北京'
```

```
>>> d.get("巴基斯坦","伊斯兰堡")
```

```
'伊斯兰堡'
```

```
>>> d.popitem()
```

```
('美国', '华盛顿')
```

# 字典功能默写

■ 定义空字典d

```
>>> d = {}
```

■ 向d新增2个键值对元素

```
>>> d["a"] = 1; d["b"] = 2
```

■ 修改第2个元素

```
>>> d["b"] = 3
```

■ 判断字符"c"是否是d的键

```
>>> "c" in d
```

■ 计算d的长度

```
>>> len(d)
```

■ 清空d

```
>>> d.clear()
```



# 字典类型应用场景

# 字典类型应用场景

## 映射的表达

- 映射无处不在，键值对无处不在
- 例如：统计数据出现的次数，数据是键，次数是值
- 最主要作用：表达键值对数据，进而操作它们




# 字典类型应用场景

## 元素遍历

```
for k in d :
```

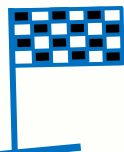
＜语句块＞



# 单元小结

# 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()





# 小花絮



**冈仁波齐：每个人心中都有一个神圣的目标，你的是什么？**







# python

The Python logo, consisting of two interlocking snakes, one blue and one yellow, is positioned below the word "python".

```
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")

for i in range(4):
    turtle.circle(40, 80)
    turtle.circle(-40, 80)
    turtle.fd(40)
    turtle.circle(16, 180)
    turtle.fd(40 * 2/3)
```

Python语言程序设计

# 模块5: jieba库的使用

---



嵩 天  
北京理工大学







# jieba库基本介绍

# jieba库概述

**jieba是优秀的中文分词第三方库**

- **中文文本需要通过分词获得单个的词语**
- **jieba是优秀的中文分词第三方库，需要额外安装**
- **jieba库提供三种分词模式，最简单只需掌握一个函数**

## (cmd命令行) pip install jieba

```
命令提示符
98%
99%
99%
99%
99%
99%
99%
100%
7.3MB 61kB/s
Installing collected packages: jieba
Running setup.py install for jieba ... done
Successfully installed jieba-0.39
C:\Users\Tian Song>
```

# jieba分词的原理

## jieba分词依靠中文词库

- 利用一个中文词库，确定中文字符之间的关联概率
- 中文字符间概率大的组成词组，形成分词结果
- 除了分词，用户还可以添加自定义的词组



# jieba库使用说明

# jieba分词的三种模式

## 精确模式、全模式、搜索引擎模式

- **精确模式：**把文本精确的切分开，不存在冗余单词
- **全模式：**把文本中所有可能的词语都扫描出来，有冗余
- **搜索引擎模式：**在精确模式基础上，对长词再次切分

# jieba库常用函数

函数	描述
<code>jieba.lcut(s)</code>	<p>精确模式，返回一个列表类型的分词结果</p> <pre>&gt;&gt;&gt;jieba.lcut("中国是一个伟大的国家")</pre> <pre>['中国', '是', '一个', '伟大', '的', '国家']</pre>
<code>jieba.lcut(s, cut_all=True)</code>	<p>全模式，返回一个列表类型的分词结果，存在冗余</p> <pre>&gt;&gt;&gt;jieba.lcut("中国是一个伟大的国家",cut_all=True)</pre> <pre>['中国', '国是', '一个', '伟大', '的', '国家']</pre>

# jieba库常用函数

函数	描述
<code>jieba.lcut_for_search(s)</code>	搜索引擎模式，返回一个列表类型的分词结果，存在冗余 <code>&gt;&gt;&gt;jieba.lcut_for_search("中华人民共和国是伟大的")</code> <code>['中华', '华人', '人民', '共和', '共和国', '中华人民共和国', '是', '伟大', '的']</code>
<code>jieba.add_word(w)</code>	向分词词典增加新词w <code>&gt;&gt;&gt;jieba.add_word("蟒蛇语言")</code>



# jieba分词要点

**jieba.lcut(s)**



# 小花絮

# 课程的实践平台为何叫Python123?

**123 代表了一种符合认知的发展过程**

**Python123 表达了一种提供最好学习价值的愿望**

**赶快注册个账号试试吧!**





# python

The Python logo, consisting of two interlocking snakes, one blue and one yellow, is positioned below the word "python".

```
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")
for i in range(4):
    turtle.circle(40, 80)
    turtle.circle(-40, 80)
    turtle.circle(40, 80/2)
    turtle.fd(40)
    turtle.circle(16, 180)
    turtle.fd(40 * 2/3)
```

# 实例10: 文本词频统计

---



嵩 天  
北京理工大学





# "文本词频统计"问题分析

# 问题分析

## 文本词频统计

- 需求：一篇文章，出现了哪些词？哪些词出现得最多？
- 该怎么做呢？

英文文本



中文文本



# 问题分析

## 文本词频统计

- 英文文本: *Hamlet*      分析词频

<https://python123.io/resources/pye/hamlet.txt>

- 中文文本: 《三国演义》 分析人物

<https://python123.io/resources/pye/threekingdoms.txt>

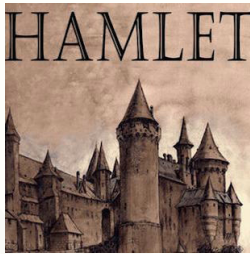


# "Hamlet英文词频统计"实例讲解

```
#CalHamletV1.py
```

```
def getText():  
    txt = open("hamlet.txt", "r").read()  
    txt = txt.lower()  
    for ch in '!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~':  
        txt = txt.replace(ch, " ")  
    return txt
```

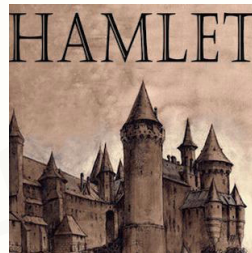
```
hamletTxt = getText()  
words = hamletTxt.split()  
counts = {}  
for word in words:  
    counts[word] = counts.get(word,0) + 1  
items = list(counts.items())  
items.sort(key=Lambda x:x[1], reverse=True)  
for i in range(10):  
    word, count = items[i]  
    print("{0:<10}{1:>5}".format(word, count))
```



- 文本去噪及归一化
- 使用字典表达词频

>>>

the	1138
and	965
to	754
of	669
you	550
i	542
a	542
my	514
hamlet	462
in	436



- 运行结果由大到小排序
- 观察单词出现次数

**准备好电脑，与老师一起编码吧！**



# "《三国演义》人物出场统计"实例讲解(上)

```
#CalThreeKingdomsV1.py
```

```
import jieba
```

```
txt = open("threekingdoms.txt", "r", encoding="utf-8").read()
```

```
words = jieba.lcut(txt)
```

```
counts = {}
```

```
for word in words:
```

```
    if len(word) == 1:
```

```
        continue
```

```
    else:
```

```
        counts[word] = counts.get(word,0) + 1
```

```
items = list(counts.items())
```

```
items.sort(key=lambda x:x[1], reverse=True)
```

```
for i in range(15):
```

```
    word, count = items[i]
```

```
    print("{0:<10}{1:>5}".format(word, count))
```



- 中文文本分词
- 使用字典表达词频

>>>

曹操 953

孔明 836

将军 772

却说 656

玄德 585

关公 510

丞相 491

二人 469

不可 440

荆州 425

玄德曰 390

孔明曰 390

不能 384

如此 378

张飞 358



- 中文文本分词
- 使用字典表达词频



**准备好电脑，与老师一起编码吧！**



# "《三国演义》人物出场统计"实例讲解(下)

# 《三国演义》人物出场统计

将词频与人物相关联，面向问题

词频统计



人物统计

```
#CalThreeKingdomsV2.py
import jieba
txt = open("threekingdoms.txt", "r", encoding="utf-8").read()
excludes = {"将军", "却说", "荆州", "二人", "不可", "不能", "如此"}
words = jieba.lcut(txt)
counts = {}
for word in words:
    if len(word) == 1:
        continue
    elif word == "诸葛亮" or word == "孔明":
        rword = "孔明"
    elif word == "关公" or word == "云长":
        rword = "关羽"
    elif word == "玄德" or word == "玄德曰":
        rword = "刘备"
    elif word == "孟德" or word == "丞相":
        rword = "曹操"
    else:
        rword = word
    counts[rword] = counts.get(rword, 0) + 1
for word in excludes:
    del counts[word]
items = list(counts.items())
items.sort(key=lambda x: x[1], reverse=True)
for i in range(10):
    word, count = items[i]
    print("{0:<10}{1:>5}".format(word, count))
```



- 中文文本分词
- 使用字典表达词频
- 扩展程序解决问题

>>>

曹操	1451
孔明	1383
刘备	1252
关羽	784
张飞	358
商议	344
如何	338
主公	331
军士	317
吕布	300



## - 根据结果进一步优化

隆重发布《三国演义》人物出场顺序前20:

曹操、孔明、刘备、关羽、张飞、吕布、赵云、孙权、  
司马懿、周瑜、袁绍、马超、魏延、黄忠、姜维、马岱、  
庞德、孟获、刘表、夏侯惇

**准备好电脑，与老师一起编码吧！**



# "文本词频统计"举一反三

```
#CalThreeKingdomsV2.py
import jieba
txt = open("threekingdoms.txt", "r", encoding="utf-8").read()
excludes = {"将军", "却说", "荆州", "二人", "不可", "不能", "如此"}
words = jieba.lcut(txt)
counts = {}
for word in words:
    if len(word) == 1:
        continue
    elif word == "诸葛亮" or word == "孔明":
        rword = "孔明"
    elif word == "关公" or word == "云长":
        rword = "关羽"
    elif word == "玄德" or word == "玄德曰":
        rword = "刘备"
    elif word == "孟德" or word == "丞相":
        rword = "曹操"
    else:
        rword = word
    counts[rword] = counts.get(rword, 0) + 1
for word in excludes:
    del counts[word]
items = list(counts.items())
items.sort(key=lambda x: x[1], reverse=True)
for i in range(10):
    word, count = items[i]
    print("{0:<10}{1:>5}".format(word, count))
```



- 中文文本分词
- 使用字典表达词频
- 扩展程序解决问题



# 举一反三

## 应用问题的扩展

- 《红楼梦》、《西游记》、《水浒传》 ...
- 政府工作报告、科研论文、新闻报道 ...
- 进一步呢？ 未来还有词云...



# 小花絮



# 全国计算机等级考试二级 Python科目

<http://ncre.neea.edu.cn>

全国计算机等级考试（简称NCRE）是教育部批准，由教育部考试中心主办，面向社会，用于考查应试人员计算机应用知识与技能的全国性计算机水平考试体系。

二级Python语言科目在 **2018年9月** 首考，异常火爆，快去报名试试吧！

