








Python

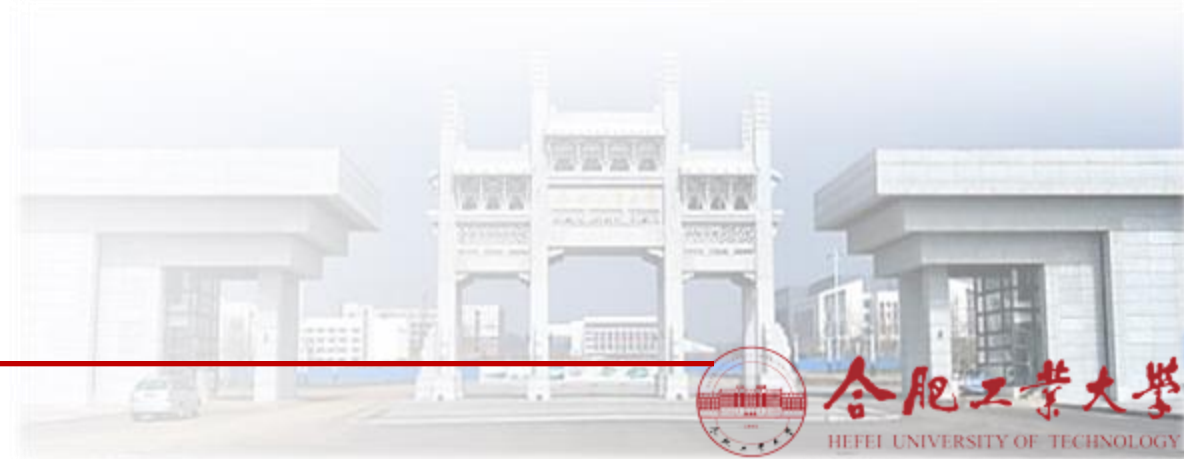
数据分析及可视化-3

程序控制结构



本章学习目标

-  理解条件表达式的值与True或False的等价关系
-  熟练掌握选择结构
-  熟练掌握循环结构
-  理解带else的循环结构执行过程
-  综合案例





Python 基础语法

Python 概念层级

- ◆ 表达式 → 创建、处理对象
- ◆ 语句 → 包含表达式
- ◆ 逻辑单元（语句块） → 函数或者类，由语句组成
- ◆ 模块 → .py 代码文件组成模块
- ◆ 包 → 定义一组有关系的文件，或者模块（包是文件夹，模块是其中的文件，且文件夹中包含一个 `__init__.py` 文件）
- ◆ 程序 → 若干个包 + 若干个文件

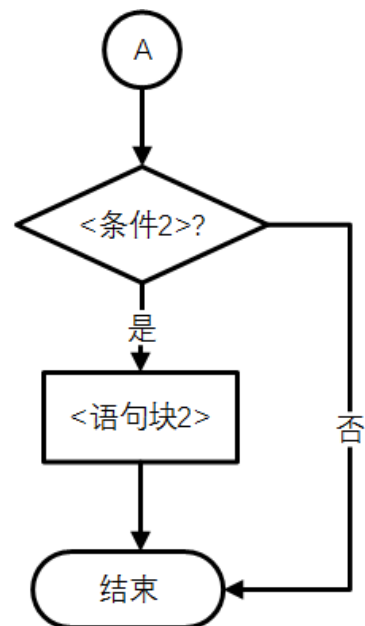
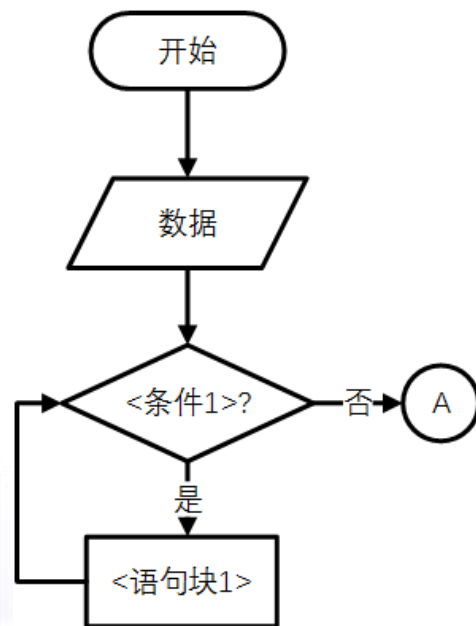




程序基本控制结构

程序基本结构


- ◆ 顺序结构是程序的基础，但单一的顺序结构不可能解决所有问题。
- ◆ 程序由三种基本结构组成：
 - 顺序结构
 - 分支结构
 - 循环结构
- ◆ 这些基本结构都有一个入口和一个出口。任何程序都由这三种基本结构组合而成





程序基本控制结构

 在Python中，几乎所有合法表达式都可以作为条件表达式。

 条件表达式的值等价于True时表示条件成立，等价于False时表示条件不成立。条件表达式的值只要不是False、0（或0.0、0j等）、空值None、空列表、空元组、空集合、空字典、空字符串、空range对象或其他空迭代对象，Python解释器均认为与True等价（注意，等价和相等是有区别的）。





程序基本控制结构

单分支结构: if语句

Python中if-else语句用来形成二分支结构，语法格式如下：

- ```
if <条件>:
 <语句块1>
else:
 <语句块2>
```
- <语句块1>是在if条件满足后执行的一个或多个语句序列
  - <语句块2>是if条件不满足后执行的语句序列
  - 二分支语句用于区分<条件>的两种可能True或者False，分别形成执行路径。

二分支结构还有一种更简洁的表达方式，适合通过判断返回特定值，语法格式如下：

```
<表达式1> if <条件> else <表达式2>
```

```
>>> PM = eval(input("请输入PM2.5数值: "))
>>> print("空气{}污染!".format("存在" if PM >= 75 else "没有"))
```





# 程序基本控制结构

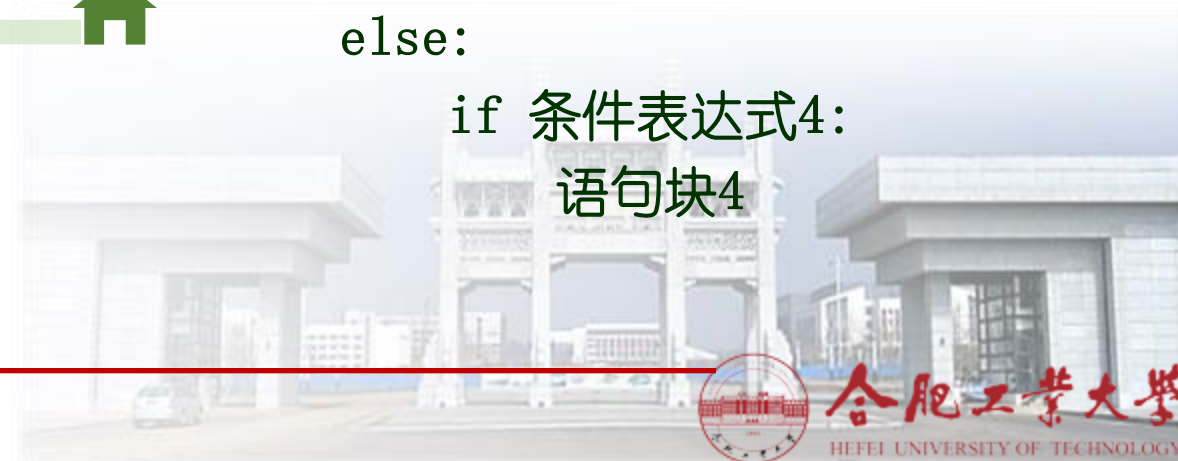
## 嵌套的分支结构



```
if 条件表达式1:
 语句块1
elif 条件表达式2:
 语句块2
elif 条件表达式3:
 语句块3
.....
else:
 语句块n
```



```
if 条件表达式1:
 语句块1
 if 条件表达式2:
 语句块2
 else:
 语句块3
else:
 if 条件表达式4:
 语句块4
```





# 程序基本控制结构

## 循环结构--遍历循环: for语句

for循环非常适合用来遍历容器类对象（列表、元组、字典、集合、字符串以及map、zip等类似对象）中的元素，语法形式为：

**for 循环遍历 in 容器类对象:**  
    **循环体**  
**[else:**  
    **else子句代码块]**

循环N次

for i in range(N):

<语句块>

遍历文件file的每一行

for line in file:

<语句块>

遍历字符串s

for c in s:

<语句块>

遍历列表ls

for item in ls:

<语句块>







# 程序基本控制结构

## 循环结构--遍历循环: for语句

```
1 for s in "BIT":
2 print("循环进行中: " + s)
3 else:
4 s = "循环正常结束"
5 print(s)
```

```
>>>
循环进行中: B
循环进行中: I
循环进行中: T
循环正常结束
```

当for循环正常执行之后，程序会继续执行else语句中内容。else语句只在循环正常执行之后才执行并结束，因此，可以在<语句块2>中放置判断循环执行情况的语句。





# 程序基本控制结构

## 循环结构--无限循环: while语句

无限循环一直保持循环操作直到特定循环条件不被满足才结束, 不需要提前知道确定循环次数, 语法格式如下:

```
1 s, idx = "BIT", 0
2 while idx < len(s):
3 print("循环进行中: " + s[idx])
4 idx += 1
5 else:
6 s = "循环正常结束"
7 print(s)
```

**while <条件>:**  
    **<语句块1>**  
**[else:**  
    **<语句块2>]**

>>>

循环进行中: B

循环进行中: I

循环进行中: T

循环正常结束





# 程序基本控制结构

## 循环结构--break和continue

break用来跳出最内层for或while循环，脱离该循环后程序从循环后代码继续执行。

continue用来结束当前当次循环，即跳出循环体中下面尚未执行的语句，但不跳出当前循环。对比continue和break语句。

```
1 for s in "BIT":
2 for i in range(10):
3 print(s, end="")
4 if s=="I":
5 break
```

```
>>>
BBBBBBBBBBBITTTTTTTTTT
```

```
1 for s in "PYTHON":
2 if s=="T":
3 continue
4 print(s, end="")
```

```
>>>
PYHON
```

```
1 for s in "PYTHON":
2 if s=="T":
3 break
4 print(s,
end="")
```

```
>>>
PY
```





# 综合应用与例题解析

 **例2.1** 编写程序，输入一个正整数，然后输出各位数字之和。例如，输入字符串1234，输出10。

```
num = input('请输入一个自然数：') # input()函数返回字符串
print(sum(map(int, num))) # 把字符串中的每个字符转换为数字，然后求和
```

 **例2.2** 编写程序，输入一个包含若干整数的列表，分别输出列表中的最大值及最大值所在的位置下标。例如，输入[1, 2, 3, 4, 5, 888]，输出888。

```
lst = eval(input('请输入一个包含若干整数的列表：'))
print(max(lst))
Print(lst.index(max(lst)))
```

 **例2.3** 编写程序，输入一个包含若干任意数据的列表，输出该列表中等价于True的元素组成的列表。例如，输入[1, 2, 0, None, False, 'a']，输出[1, 2, 'a']。


```
lst = eval(input('请输入一个包含若干任意元素的列表：'))
print(list(filter(None, lst)))
```







# 综合应用与例题解析

 **例2.4** 编写程序，输入一个字符串，输出翻转（首尾交换）后的字符串，要求使用内置函数实现。例如，输入字符串12345，输出54321。

```
from operator import add
from functools import reduce
text = input('请输入一个字符串：')
print(reduce(add, reversed(text)))
```

 **例2.5** 编写程序，输入一个包含若干整数的列表，输出一个新列表，新列表中奇数在前偶数在后，并且奇数之间的相对顺序不变，偶数之间的相对顺序也不变。

```
lst = eval(input('请输入一个包含若干整数的列表：'))
newLst = sorted(lst, key=lambda num: num%2==0)
print(newLst)
```






# 集合应用案例

## 集合案例

 假设已有若干用户名字及其喜欢的电影清单，现有某用户，已看过并喜欢一些电影，现在想找个新电影看看，又不知道看什么好。

 **思路：**根据已有数据，查找与该用户爱好最相似的用户，也就是看过并喜欢的电影与该用户最接近，然后从那个用户喜欢的电影中选取一个当前用户还没看过的电影，进行推荐。

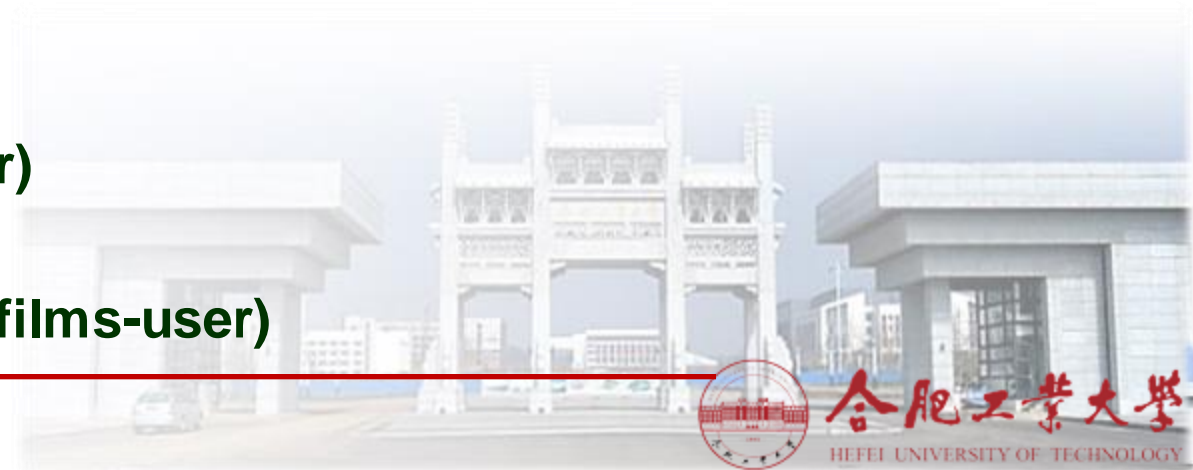




# 集合应用案例

## 集合案例

```
>>> from random import randrange
>>> data = {'user'+str(i):{'film'+str(randrange(1, 10)) for j in range(randrange(15))}\
 for i in range(10)} # 其他用户喜欢看的电影清单
>>> user = {'film1', 'film2', 'film3'} # 待测用户曾经看过并感觉不错的电影
 # 查找与待测用户最相似的用户和Ta喜欢看的电影
>>> similarUser, films = max(data.items(), key=lambda item: len(item[1]&user))
>>> print('历史数据: ')
>>> for u, f in data.items():
 print(u, f, sep=':')
>>> print('和您最相似的用户是: ', similarUser)
>>> print('Ta最喜欢看的电影是: ', films)
>>> print('Ta看过的电影中您还没看过的有: ', films-user)
```





## 集合案例

某次运行结果:

历史数据: user0: {'film5'}

user1: {'film5'}

user2: {'film1', 'film6', 'film2', 'film4', 'film3', 'film7'}

user3: {'film1', 'film9', 'film6', 'film5', 'film8', 'film3', 'film7'}

user4: {'film1', 'film9', 'film6', 'film4', 'film5', 'film3', 'film7'}

user5: {'film1', 'film9', 'film6', 'film2', 'film4'}

user6: {'film1', 'film6', 'film2', 'film8', 'film5', 'film9', 'film7'}

user7: {'film2', 'film6', 'film5', 'film7'}

user8: {'film9', 'film2', 'film4', 'film3', 'film7'}

user9: set()

和您最相似的用户是: user2

Ta最喜欢看的电影是: {'film1', 'film6', 'film2', 'film4', 'film3', 'film7'}

Ta看过的电影中您还没看过的有: {'film7', 'film4', 'film6'}







# 集合应用案例

## 集合案例--过滤无效书评

很多人喜欢爬取书评，然后选择自己喜欢的书或者其他读者评价较高的书，这是一个非常好的思路，也是非常明智的做法。

然而，并不是每个消费者都会认真留言评论，也有部分消费者可能会复制了几个简单的句子或词作为评论。在爬取到原始书评之后可能需要进行简单的处理和过滤，这时就需要制定一个过滤的标准进行预处理，这也是数据处理与分析的关键内容之一。

在下面的代码中，采用了一个最简单的规则：正常书评中，重复的字应该不会超过一定的比例。





# 集合应用案例

## 集合案例--过滤无效书评

```
comments = ['这是一本非常好的书，作者用心了'，
 '作者大大辛苦了'，
 '好书，感谢作者提供了这么多的好案例'，
 '书在运输的路上破损了，我好悲伤。。。'，
 '为啥我买的书上有菜汤。。。。'，
 '啊啊啊啊啊啊，我怎么才发现这么好的书啊，相见恨晚'，
 '书的质量有问题啊，怎么会开胶呢??????'，
 '好好好好好好好好好好'，
 '好难啊看不懂好难啊看不懂好难啊看不懂'，
 '书的内容很充实'，
 '你的书上好多代码啊，不过想想也是，编程的书嘛，肯定代码多一些'，
 '书很不错!!一级棒!!买书就上当当，正版，价格又实惠，让人放心!!!'，
 '无意中来到你小铺就淘到心意的宝贝，心情不错!'，
 '送给朋友的、很不错'，
 '这是一本好书，讲解内容深入浅出又清晰明了，推荐给所有喜欢阅读的朋友同好们。']
```





# 集合应用案例

## 集合案例--过滤无效书评

原始书评:

这是一本非常好的书，作者用心了  
作者大大辛苦了  
好书，感谢作者提供了这么多的好案例  
书在运输的路上破损了，我好悲伤。。。.  
为啥我买的书上有菜汤。。。.  
啊啊啊啊啊啊，我怎么才发现这么好的书啊，相见恨晚  
书的质量有问题啊，怎么会开胶呢?????  
好好好好好好好好好好好好  
好难啊看不懂好难啊看不懂好难啊看不懂  
书的内容很充实  
你的书上好多代码啊，不过想想也是，编程的书嘛，肯定代码多一些  
书很不错!!一级棒!!买书就上当当，正版，价格又实惠，让人放心!!!  
无意中来到你小铺就淘到心意的宝贝，心情不错!  
送给朋友的、很不错  
这是一本好书，讲解内容深入浅出又清晰明了，推荐给所有喜欢阅读的朋友同好们。

过滤后的书评:

这是一本非常好的书，作者用心了  
作者大大辛苦了  
好书，感谢作者提供了这么多的好案例  
书在运输的路上破损了，我好悲伤。。。.  
为啥我买的书上有菜汤。。。.  
啊啊啊啊啊啊，我怎么才发现这么好的书啊，相见恨晚  
书的质量有问题啊，怎么会开胶呢?????  
书的内容很充实  
你的书上好多代码啊，不过想想也是，编程的书嘛，肯定代码多一些  
书很不错!!一级棒!!买书就上当当，正版，价格又实惠，让人放心!!!  
无意中来到你小铺就淘到心意的宝贝，心情不错!  
送给朋友的、很不错  
这是一本好书，讲解内容深入浅出又清晰明了，推荐给所有喜欢阅读的朋友同好们。


```
>>> rule = lambda s:len(set(s))/len(s)>0.5
>>> result = filter(rule, comments)
>>> print('原始书评:')
>>> for comment in comments:
 print(comment)
>>> print('='*30)
>>> print('过滤后的书评:')
>>> for comment in result:
 print(comment)
```



合肥工业大学  
HEFEI UNIVERSITY OF TECHNOLOGY




# 综合应用与例题解析

 **案例3.1** 阿凡提与国王比赛下棋，国王说要是自己输了的话阿凡提想要什么他都可以拿出来。阿凡提说那就要点米吧，棋盘一共64个小格子，在第一个格子里放1粒米，第二个格子里放2粒米，第三个格子里放4粒米，第四个格子里放8粒米，以此类推，后面每个格子里的米都是前一个格子里的2倍，一直把64个格子都放满。编写程序，生成一个列表，其中元素为每个棋盘格子里米的粒数，并输出这些数字的和，也就是一共需要多少粒米。要求使用列表推导式。

```
>>> data = [2**i for i in range(64)]
```

```
>>> print(sum(data))
```

```
>>> Print(int('1'*64,2))
```

 **案例3.2** 编写程序，输入一个包含若干整数的列表，输出由其中的奇数组成的新列表。例如，输入[1, 2, 3, 4, 5, 6, 7, 8]，输出[1, 3, 5, 7]。要求使用列表推导式。

```
>>> data = eval(input('请输入一个包含若干整数的列表：'))
```

```
>>> print([num for num in data if num%2==1])
```







# 综合应用与例题解析



## 案例3.3

编写程序，输入两个包含若干整数的等长列表表示两个向量，输出这两个向量的内积。例如，输入[1, 2, 3]和[4, 5, 6]，内积计算方法为 $1*4 + 2*5 + 3*6 = 32$ ，输出32。要求使用列表推导式。

```
>>> vector1 = eval(input('请输入一个包含若干整数的向量: '))
>>> vector2 = eval(input('请再输入一个包含若干整数的等长向量: '))
>>> print(sum([num1*num2 for num1, num2 in zip(vector1, vector2)]))
```



## 案例3.4

编写程序，输入一个包含若干整数的列表，输出其中的最大值，以及所有最大值的下标组成的列表。例如，输入[1, 2, 3, 1, 2, 3, 3]，输出3和[2, 5, 6]。要求使用列表推导式。

```
>>> data = eval(input('请输入一个包含若干整数的列表: '))
>>> m = max(data)
>>> print([index for index, value in enumerate(data) if value==m])
```





# 综合应用与例题解析



## 案例3.5

编写程序，首先生成包含1000个随机数字的列表，然后统计每个数字的出现次数。

```
>>> from string import digits
>>> from random import choice
>>> lst = [choice(digits) for i in range(1000)]
>>> result = {}
>>> for i in lst:
 result[i] = result.get(i,0) + 1
>>> for digit, fre in sorted(result.items()):
 print(digit, fre, sep=':')
```

#对字典默认按“键”排序

```
>>> for digit, fre in sorted(result.items(),key=lambda x:x[1],reverse=False):
 print(digit, fre, sep=':')
```





# 综合应用与例题解析

## 案例3.6 编写程序，测试列表中的若干整数之间是否有重复。

```
>>> import random
>>> data1 = [random.randint(1, 10)] * 5
>>> data2 = random.choices(range(10), k=5)
>>> data3 = random.sample(range(10), k=5)
>>> for data in (data1, data2, data3):
 print('=' * 20)
 print(data)
 k1 = len(set(data))
 k2 = len(data)
 if k1 == k2:
 print('无重复')
 elif k1 == 1:
 print('完全重复')
 else:
 print('部分重复')
```

```
生成一个包含5个相同随机数的列表
生成一个包含5个允许重复的随机数列表
生成一个包含5个不允许重复的随机数列表
```





# 祝学习进步!

计算机基础教育研究所