Python程序设计教案

<table>
<tr><td>本次授课内容</td><td colspan="2">3.1 Python序列概述
3.2 列表
3.3 元组与生成器表达式</td></tr>
<tr><td>本次课的
教学目的</td><td colspan="2">掌握列表、元组的类型特点与方法
掌握运算符和内置函数对列表、元组的操作
掌握切片操作
掌握列表推导式、生成器表达式的工作原理</td></tr>
<tr><td>本次课教学
重点与难点</td><td colspan="2">列表推导式、生成器表达式
切片</td></tr>
<tr><td>教学方法
教学手段</td><td colspan="2">PPT、边讲边练</td></tr>
<tr><td rowspan="3">课堂教学
时间分配</td><td>教学内容</td><td>时间分配（分）</td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td>课堂教学设计</td><td colspan="2">重点介绍列表对象的特点和应用，以及列表推导式和切片的用法，然后介绍元组的特点及其与列表的区别。</td></tr>
<tr><td>实　验</td><td colspan="2">列表方法以及列表推导式、生成器表达式的工作原理</td></tr>
<tr><td>思考题及作业题</td><td colspan="2">课后习题1-5，8-16</td></tr>
<tr><td>备　注</td><td colspan="2"></td></tr>
<tr><td>教学后记</td><td colspan="2"></td></tr>
</table>

课堂重点内容详解

第 一 节 课

列表是包含若干元素的有序连续内存空间。在形式上，列表的所有元素放在一对方括号中，相邻元素之间使用逗号分隔。在 Python 中，同一个列表中元素的数据类型可以各不相同，可以同时包含整数、实数、字符串等基本类型的元素，也可以包含列表、元组、字典、集合、函数以及其他任意对象。如果只有一对方括号而没有任何元素则表示空列表。下面几个都是合法的列表对象：

```
[10, 20, 30, 40]
['crunchy frog', 'ram bladder', 'lark vomit']
['spam', 2.0, 5, [10, 20]]
[['file1', 200,7], ['file2', 260,9]]
[{3}, {5:6}, (1, 2, 3)]
```

使用“=”直接将一个列表常量赋值给变量即可创建列表对象。

```
>>> a_list = ['a', 'b', 'c', 'd', 'e']
>>> a_list = []                          #创建空列表
```

也可以使用 list()函数把元组、range 对象、字符串、字典、集合或其他可迭代对象转换为列表。

```
>>> list((3, 5, 7, 9, 11))    #将元组转换为列表
[3, 5, 7, 9, 11]
>>> list(range(1, 10, 2))     #将 range 对象转换为列表
[1, 3, 5, 7, 9]
>>> list('hello world')        #将字符串转换为列表
['h', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd']
>>> list({3, 7, 5})            #将集合转换为列表，集合中的元素是无序的
[3, 5, 7]
>>> list({'a':3, 'b':9, 'c':78}) #将字典的“键”转换为列表
['a', 'b', 'c']
>>> list({'a':3, 'b':9, 'c':78}.items()) #将字典的元素转换为列表
[('a', 3), ('b', 9), ('c', 78)]
>>> x = list()                   #创建空列表
>>> x = list('Python')           #把字符串转换为列表
>>> x
['P', 'y', 't', 'h', 'o', 'n']
>>> x[1]                            #下标为 0 的元素，第一个元素
'y'
>>> x[-3]                           #下标为-1 的元素，最后一个元素
'h'
```

表 3-1 列表对象常用方法

方法	说明
append(x)	将 x 追加至列表尾部
extend(L)	将列表 L 中所有元素追加至列表尾部
insert(index, x)	在列表 index 位置处插入 x
remove(x)	在列表中删除第一个值为 x 的元素，如果列表中不存在 x

	则抛出异常
pop([index])	删除并返回列表中下标为 index 的元素，index 则默认为-1
index(x)	返回列表中第一个值为 x 的元素的索引，若不存在值为 x 的元素则抛出异常
count(x)	返回 x 在列表中的出现次数
reverse()	对列表所有元素进行原地逆序，首尾交换
sort(key=None, reverse=False)	对列表中的元素进行原地排序，key 用来指定排序规则，reverse 为 False 表示升序，True 表示降序

列表推导式可以使用非常简洁的方式对列表或其他可迭代对象的元素进行遍历、过滤或再次计算，快速生成满足特定需求的新列表。列表推导式的语法形式为：

```
[expression  for expr1 in sequence1 if condition1
             for expr2 in sequence2 if condition2
             for expr3 in sequence3 if condition3
             ...
             for exprN in sequenceN if conditionN]
```

在形式上，切片使用 2 个冒号分隔的 3 个数字来完成。

```
[start:end:step]
```

其中第一个数字 start 表示切片开始位置，默认为 0；第二个数字 end 表示切片截止（但不包含）位置（默认为列表长度）；第三个数字 step 表示切片的步长（默认为 1）。当 start 为 0 时可以省略，当 end 为列表长度时可以省略，当 step 为 1 时可以省略，省略步长时还可以同时省略最后一个冒号。另外，当 step 为负整数时，表示反向切片，这时 start 应该在 end 的右侧才行。

第 二 节 课

可以把元组看作是轻量级列表或者简化版列表，支持和列表类似的操作，但功能不如列表强大。在形式上，元组的所有元素放在一对圆括号中，元素之间使用逗号分隔，如果元组中只有一个元素则必须在最后增加一个逗号。

```
>>> x = (1, 2, 3)     #直接把元组赋值给一个变量
>>> type(x)           #使用 type()函数查看变量类型
<class 'tuple'>
>>> x[0]              #元组支持使用下标访问特定位置的元素
1
>>> x[-1]             #最后一个元素，元组也支持双向索引
3
>>> x[1] = 4          #元组是不可变的
TypeError: 'tuple' object does not support item assignment
>>> x = (3,)          #如果元组中只有一个元素，必须在后面多写一个逗号
>>> x
(3,)
>>> x = ()            #空元组
>>> x = tuple()       #空元组
>>> tuple(range(5))   #将其他迭代对象转换为元组
```

```
(0, 1, 2, 3, 4)
```

使用生成器对象的元素时，可以根据需要将其转化为列表或元组，也可以使用生成器对象的__next__()方法或者内置函数 next()进行遍历，或者直接使用 for 循环来遍历其中的元素。但是不管用哪种方法访问其元素，只能从前往后正向访问每个元素，没有任何方法可以再次访问已访问过的元素，也不支持使用下标访问其中的元素。当所有元素访问结束以后，如果需要重新访问其中的元素，必须重新创建该生成器对象，enumerate、filter、map、zip 等其他迭代器对象也具有同样的特点。

```
>>> g = ((i+2)**2 for i in range(10)) #创建生成器对象
>>> g
<generator object <genexpr> at 0x0000000003095200>
>>> tuple(g)                                #将生成器对象转换为元组
(4, 9, 16, 25, 36, 49, 64, 81, 100, 121)
>>> list(g)                                 #生成器对象已遍历结束
[]
>>> g = ((i+2)**2 for i in range(10)) #重新创建生成器对象
>>> g.__next__()           #使用生成器对象的__next__()方法获取元素
4
>>> g.__next__()                            #获取下一个元素
9
>>> next(g)                #使用函数 next()获取生成器对象中的元素
16
>>> g = map(str, range(20))        #map 对象也具有同样的特点
>>> '2' in g
True
>>> '2' in g                            #这次判断会把所有元素都给“看”没了
False
>>> '8' in g
False
```

Python 程序设计教案

<table>
<tr><td>本次授课内容</td><td colspan="2">3.4 字典
3.5 集合
3.6 序列解包</td></tr>
<tr><td>本次课的
教学目的</td><td colspan="2">掌握字典对象的应用
掌握集合对象的应用
掌握序列解包的应用</td></tr>
<tr><td>本次课教学
重点与难点</td><td colspan="2">字典对象的 get()方法，访问字典对象时可以使用“键”作为下标。
集合元素的无序性。
序列解包赋值的同时性。</td></tr>
<tr><td>教学方法
教学手段</td><td colspan="2">PPT、边讲边练</td></tr>
<tr><td rowspan="3">课堂教学
时间分配</td><td>教学内容</td><td>时间分配（分）</td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td>课堂教学设计</td><td colspan="2">字典对象的创建与删除，字典元素的读取，字典元素的添加与修改。
集合对象的创建与常用方法。
序列解包的语法与应用。</td></tr>
<tr><td>实　验</td><td colspan="2">字典元素访问与修改，集合对象应用。</td></tr>
<tr><td>思考题及作业题</td><td colspan="2">课后习题 6、7</td></tr>
<tr><td>备　注</td><td colspan="2"></td></tr>
<tr><td>教学后记</td><td colspan="2"></td></tr>
</table>

课堂重点内容详解

第 一 节 课

字典中的每个元素表示一种映射关系或对应关系，根据提供的“键”作为下标就可以访问对应的“值”，如果字典中不存在这个“键”会抛出异常。

```
>>> aDict = {'age': 39, 'score': [98, 97], 'name': 'Dong', 'sex': 'male'}
>>> aDict['age']                    #指定的“键”存在，返回对应的“值”
39
>>> aDict['address']                #指定的“键”不存在，抛出异常
KeyError: 'address'
```

字典对象提供了一个 get()方法用来返回指定“键”对应的“值”，并且允许指定该键不存在时返回特定的“值”。例如：

```
>>> aDict.get('age')               #如果字典中存在该“键”则返回对应的“值”
39
>>> aDict.get('address', 'Not Exists.')
                                   #指定的“键”不存在时返回指定的默认值
'Not Exists.'
```

当以指定“键”为下标为字典元素赋值时，有两种含义：

1）若该“键”存在，则表示修改该“键”对应的值；

2）若不存在，则表示添加一个新的“键:值”对，也就是添加一个新元素。

```
>>> aDict = {'age': 35, 'name': 'Dong', 'sex': 'male'}
>>> aDict['age'] = 39                  #修改元素值
>>> aDict
{'age': 39, 'name': 'Dong', 'sex': 'male'}
>>> aDict['address'] = 'Yantai'        #添加新元素
>>> aDict
{'age': 39, 'address': 'Yantai', 'name': 'Dong', 'sex': 'male'}
```

例 3-4 首先生成包含 1000 个随机字符的字符串，然后统计每个字符的出现次数，注意字典方法 get()方法运用。

```
>>> import string
>>> import random
>>> x = string.ascii_letters + string.digits
>>> z = ''.join((random.choice(x) for i in range(1000)))
                                   #choice()用于从多个元素中随机选择一个
>>> d = dict()
>>> for ch in z:                          #遍历字符串，统计频次
    d[ch] = d.get(ch, 0) + 1              #已出现次数加 1

>>> for k, v in sorted(d.items()):     #查看统计结果
    print(k, ':', v)
```

第 二 节 课

直接将集合赋值给变量即可创建一个集合对象。

```
>>> a = {3, 5}                                  #创建集合对象
```

也可以使用 set()函数将列表、元组、字符串、range 对象等其他可迭代对象转换为集合，如果原来的数据中存在重复元素，则在转换为集合的时候只保留一个；如果原序列或迭代对象中有不可哈希的值，无法转换成为集合，抛出异常。

```
>>> a_set = set(range(8, 14))           #把 range 对象转换为集合
>>> a_set
{8, 9, 10, 11, 12, 13}
>>> b_set = set([0, 1, 2, 3, 0, 1, 2, 3, 7, 8])
                                        #转换时自动去掉重复元素
>>> b_set
{0, 1, 2, 3, 7, 8}
>>> x = set()                                #空集合
```

当不再使用某个集合时，可以使用 del 命令删除整个集合。

（1）集合元素增加与删除

```
>>> s = {1, 2, 3}
>>> s.add(3)                              #添加元素，重复元素自动忽略
>>> s
{1, 2, 3}
>>> s.update({3,4})                      #更新当前字典，自动忽略重复的元素
>>> s
{1, 2, 3, 4}
>>> s.discard(5)                          #删除元素，不存在则忽略该操作
>>> s
{1, 2, 3, 4}
>>> s.remove(5)                           #删除元素，不存在就抛出异常
KeyError: 5
>>> s.pop()                               #删除并返回一个元素
1
```

（2）集合运算

```
>>> a_set = set([8, 9, 10, 11, 12, 13])
>>> b_set = {0, 1, 2, 3, 7, 8}
>>> a_set | b_set                          #并集
{0, 1, 2, 3, 7, 8, 9, 10, 11, 12, 13}
>>> a_set & b_set                         #交集
{8}
>>> a_set - b_set                         #差集
{9, 10, 11, 12, 13}
>>> a_set ^ b_set                         #对称差集
{0, 1, 2, 3, 7, 9, 10, 11, 12, 13}
```

例 3-8 电影评分与推荐。

```
from random import randrange
```

```
data = {'user'+str(i):{'film'+str(randrange(1, 15)):randrange(1, 6)
                       for j in range(randrange(3, 10))}
       for i in range(10)}

#模拟当前用户打分数据，为5部随机电影打分
user = {'film'+str(randrange(1, 15)):randrange(1,6) for i in range(5)}
f = lambda item:(-len(item[1].keys()&user),
                 sum(((item[1].get(film)-user.get(film))**2
                      for film in user.keys()&item[1].keys())))
similarUser, films = min(data.items(), key=f)

print('known data'.center(50, '='))
for item in data.items():
    print(len(item[1].keys()&user.keys()),
          sum(((item[1].get(film)-user.get(film))**2
               for film in user.keys()&item[1].keys())),
          item,
          sep=':')
print('current user'.center(50, '='))
print(user)
print('most similar user and his films'.center(50, '='))
print(similarUser, films, sep=':')
print('recommended film'.center(50, '='))
print(max(films.keys()-user.keys(), key=lambda film: films[film]))
```