人工智能程序设计

计算机学院

李晶晶

華南師範大學
South China Normal University

目录

- 数值数据类型
- 数据结构类型
  - 列表
  - 元组
  - 字典
  - 集合

元组（tuple）

- 元组（tuple）是序列类型中比较特殊的类型，因为它一旦创建就不能被修改
- 创建元组
  - 使用圆括号（）
    - ✓可以通过把若干元素放在一对圆括号中创建元组，如果只有一个元素的话则需要多加一个逗号，例如(3,)。
    - ✓圆括号可以省略
    - ✓()为空元组，(0,)为单元素元组（注意逗号,），(0)为括号表达式
  - 使用tuple( )函数
    - ✓将其他数据结构对象转化成元组类型

元组（tuple）

- 元组基本操作
  - ➢元组元素提取
    - ✓索引访问

```
>>>tp = (1, 2, 3, ["god", "hello", 1], "world", (1, 2))
>>>tp[3]
>>>tp[-1]
>>>tp[-1][0]
>>>tp[7]
```

元组（tuple）

- 元组基本操作
  - ➢元组元素提取
    - ✓索引访问

```
>>>tp = (1, 2, 3, ["god", "hello", 1], "world", (1, 2))
>>>tp[3]
['god', 'hello', 1]
>>>tp[-1]
(1, 2)
>>>tp[-1][0]
1
 >>>tp[7]
indexError: tuple index out of range
```

元组（tuple）

- 元组基本操作
  - ➢元组元素提取
    - ✓切片访问

```
>>>tp = (1, 2, 3, ["god", "hello", 1], "world", (1, 2))
>>>tp[3]
['god', 'hello', 1]
>>>tp[-2::-1]
('world', ['god', 'hello', 1], 3, 2, 1)
>>> tp[1:10]
>>>(2, 3, ['god', 'hello', 1], 'world', (1, 2))
```

元组（tuple）

- 元组基本操作
  - ➢+=（不可变类型）

```
>>>tuple =(10,20,30)
>>> tuple += (40,50)
>>> tuple

>>> tuple1 = (10,20,30)
>>> tuple1 += [1,2,3,4]

>>>numbers = [1,2,3,4]
>>>numbers += (8,9)
>>>numbers
```

元组（tuple）

- 元组基本操作
  - 元组可能包含可变对象

```
>>>tp = (1, 2, 3, ["god", "hello", 1], "world", (1, 2))
>>>tp[3][2] = 77
>>>tp
```

元组（tuple）

- 元组常用方法和函数
  - tuple.count
  - tuple.index
  - sorted
  - len
  - +
  - *

元组与列表的区别

- 元组是不可变的，不能直接修改元组中元素的值，也不能为元组增加或删除元素。
  - 因此，元组没有提供append()、extend()和insert()等方法，也没有remove()和pop()方法。
- 元组的访问速度比列表更快，开销更小。
  - 如果定义了一系列常量值，主要用途只是对它们进行遍历或其他类似操作，那么一般建议使用元组而不用列表。
- 元组可以使得代码更加安全。
  - 例如，调用函数时使用元组传递参数可以防止在函数中修改元组，而使用列表则无法保证这一点。
- 元组可用作字典的键，也可以作为集合的元素，但列表不可以，包含列表的元组也不可以。

上机实践

- 将列表[‘pen’, ‘paper’, 10, False, 2.5]转换为元组类型，并提取当中的布尔值
  - 创建列表
  - 查看类型
  - 转为tuple
  - 查看类型
  - 查询元组元素False的位置
  - 提取元素

问题2

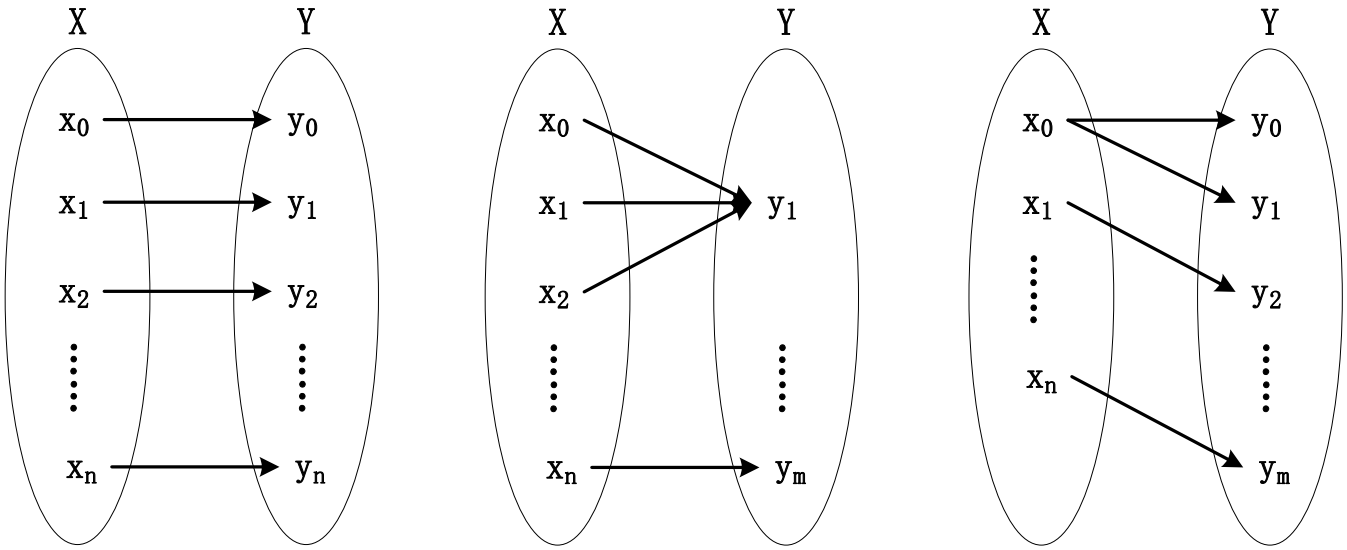
- 问题描述
  - 统计给定字符串mstr=“Hello world, I am using Python to program.”中各个字符出现的次数。

问题2

- 问题分析
  - 输入：字符串mstr=“Hello world, I am using Python to program.”
  - 处理：对字符串每个字符进行遍历，将该字符作为键值插入字典，或更新其出现次数
  - 输出：字典类型，里面记录每个字符以及对应的出现次数

字典（dict）

- 一组键/值对的数据结构



映射（f(x)=y）可以有一对一、多对一，但不能是一对多

字典（dict）

- 每个键对应于一个值
- 在字典中，键不能重复
- 根据键可以查询到值
- 字典长度是可变的，可以通过对键信息赋值实现增加或修改键值对
- 字典的键只能使用不可变的对象，但字典的值可以使用不可变或可变的对象

字典（dict）

- 创建字典
  - 花括号{ }创建
    - ✓{<键1>:<值1>, <键2>:<值2>, … , <键n>:<值n>}

      其中，键和值通过冒号连接，不同键值对通过逗号隔开。
- dict( )函数创建
  - 主要是将包含双值子序列的序列对象转换为字典类型
  - 直接向dict函数传入键和值创建
    - ✓ Dict(key_1 = value_1, key_2 = value_2, …, key_n = valuen)

字典（dict）

- 创建字典
  - 字典中键值对中的值还可以是字典，如：

```
mdict2 = {'H':{'count':1,'freq':0.2},
'e':{'count':1,'freq':0.2}, 'l':{'count':2,'freq':0.4},
'o':{'count':1,'freq':0.2}}
```

字典（dict）

- 提取字典元素
  - dict[key]
  - in语句测试键是否存在
  - 使用字典方法get
    - ✓dict.get(key, default=None)

字典（dict）

- Python字典符合数据库中数据表格的概念，它能表示基于关系模型的数据库
- 可以用字典存放：mdict = {'H':[1,0.2], 'e':[1,0.2], 'l':[2,0.4], 'o':[1,0.2]}，这时，mdict[‘H’][1]即为字母’H’出现的频率

字符	频次	频率
H	1	0.2
e	1	0.2
l	2	0.4
o	1	0.2

字典（dict）

- 字典常用函数和方法
  - 增添字典元素
    - ✓键访问赋值增添

      dict_name[newkey]=new_value
    - ✓update

      dict_name.update(dict)

      将两个字典中的键值对进行合并，如果存在相同键，传入字典中的键所对应的值会替换原有值，实现值更新的效果

字典（dict）

- 字典常用函数和方法
  - 增添字典元素

```
>>>mydict = {"one" : 1, "two" : 2, "three" : 3, "four" : 4}
>>> mydict["two"] = 3
>>> mydict["five"] = 7
>>> mydict
>>> mydict1 = {"six" : 6, "seven" : 7}
>>> mydict.update(mydict1)
>>> mydict
>>> mydict1 = {"six" : 6, "seven" : 81}
>>> mydict.update(mydict1)
>>> mydict
```

字典（dict）

- 字典常用函数和方法
  - ➢删除字典元素
    - ✓del
    - ✓pop( )
    - ✓clear( )

```
>>>mydict = {"one" : 1, "two" : 2, "three" : 3, "four" : 4}
>>> mydict_copy = mydict.copy()
>>> del mydict_copy["four"]
>>> mydict_copy
>>>mydict_copy.pop(“two”)
>>>value = mydict_copy.pop(“one”)
>>>print(value)
>>>mydict_copy = mydict.copy()
>>>mydict_copy.clear()
```

字典（dict）

- 字典常用函数和方法
  - 修改字典元素
    - ✓dict_name[key]=new_value
  - 查询和获取字典元素信息
    - ✓keys: 用于获取字典中的所有键
    - ✓values：用于获取字典中的所有值
    - ✓items：得到字典所有键值对

```
mydict = {"one" : 1, "two" : 2, "three" : 3}
all_value = mydict.values()
print(list(all_value))
all_items = mydict.items()
print(list(all_items))
```

字典（dict）

- 字典常用函数和方法
  - 以给定键值创建值为空的字典
    - ✓dict.fromkeys( )

      dict.fromkeys(seq[, value])
  - 使用dict利用已有数据创建字典

```
>>>adict = dict.fromkeys([ ‘name ‘, ‘age ‘, ‘sex ‘])
>>>adict
{ ‘name': None, ‘age': None, ‘sex': None}
```

```
>>>keys = [ ‘a ‘, ‘b ‘, ‘c ‘, ‘d ‘]
>>>values = [1, 2, 3, 4]
>>>dictionary = dict(zip(keys,values))
>>>dictionary
{ ‘a':1, ‘b':2, ‘c':3, ‘d':4}
```

字典（dict）

- 字典操作
  - 与其他组合类型一样，字典可以通过for...in语句对其元素进行遍历，基本语法结构如下：

for <变量名

语句块

```
mydict = {"one" : 1, "two" : 2, "three" : 3,
"four" : 4}
for all_value in mydict.values():
    print(all_value, end=" ")
for key in mydict:
    print(key)
for key, value in mydict.items():
    print((key,value))
for all_items in mydict:
    print(all_items, end=" ")
for all_items in mydict.items():
```

字典（dict）

```
mydict = {"one" : 1, "two" : 2, "three" : 3, "four" : 4}
for all_value in mydict.values():
    print(all_value, end=" ")
for key in mydict:
    print(key)
for key, value in mydict.items():
    print((key,value))
for all_items in mydict:
    print(all_items, end=" ")
for all_items in mydict.items():
    print(all_items, end=" ")
```

问题2

- 演示代码
  - 输入：字符串mstr=“Hello world, I am using Python to program.”
  - 处理：对字符串每个字符进行遍历，将该字符作为键值插入字典，或更新其出现次数
  - 输出：字典类型，里面记录每个字符以及对应的出现次数

字典与列表

字符	频次	频率
H	1	0.2
e	1	0.2
l	2	0.4
o	1	0.2

- 列表是序列，字典不是序列
- 序列是使用索引方式获取元素的，而字典是使用键来获取元素的。
- 序列元素的插入是和索引相关的，而字典元素的插入是和键相关的。因此列表有append函数、分片功能等，而字典没有。
- 列表是一种通用的数据结构，它里面的元素可以千变万化。列表可以被视为有序列的一群东西，可见其功能的广泛，而字典则不然，它是key-value这种结构的独特组合，所以，字典对于寻找某一个key的记录，会有快速的方式来实现，这种实现方式叫作哈希(Hash)方式。

上机实践

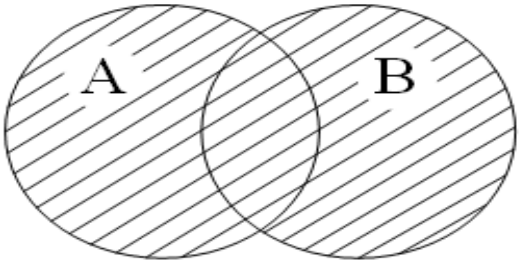
- 统计单词出现次数
  - 输入：字符串mstr = 'I am using Python to write Hello world. I like Python. I am using python to program.'
  - 处理：将每个单词分离（去掉标点符号，不区分大小写），统计单词的个数存入字典
  - 输出：字典类型，里面记录每个单词以及对应的出现次数

集合

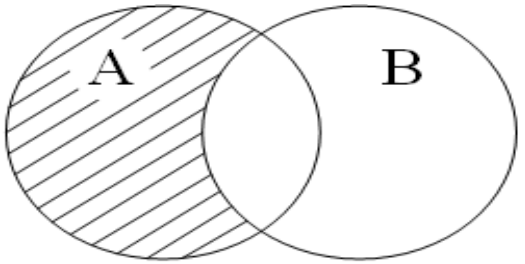
- 可变集合（set）
  - {}
  - set( )将数据结构转换为可变集合类型
- 不可变集合（frozenset）
  - frozenset( )将数据结构转换为不可变集合类型

集合

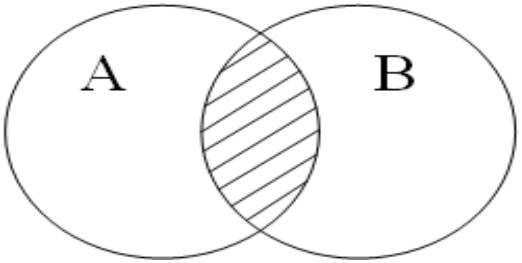
- 集合类型的4种基本操作，交集（&）、并集（|）、差集（-）、补集（^），操作逻辑与数学定义相同



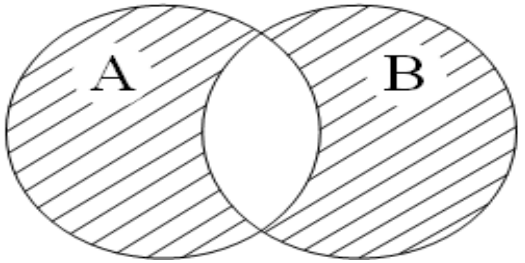
A|B



A-B



A&B



A^B

集合

- 集合类型10个操作符

操作符	描述
S – T 或 S.difference(T)	返回一个新集合，包括在集合S中但不在集合T中的元素
S-=T或S.difference_update(T)	更新集合S，包括在集合S中但不在集合T中的元素
S & T或S.intersection(T)	返回一个新集合，包括同时在集合S和T中的元素
S&=T或S.intersection_update(T)	更新集合S，包括同时在集合S和T中的元素。
S^T或s.symmetric_difference(T)	返回一个新集合，包括集合S和T中元素，但不包括同时在其中的元素
S=^T或 s.symmetric_difference_update(T)	更新集合S，包括集合S和T中元素，但不包括同时在其中的元素
S|T或S.union(T)	返回一个新集合，包括集合S和T中所有元素
S=|T或S.update(T)	更新集合S，包括集合S和T中所有元素
S<=T或S.issubset(T)	如果S与T相同或S是T的子集，返回True，否则返回False，可以用S<T判断S是否是T的真子集
S>=T或S.issuperset(T)	如果S与T相同或S是T的超集，返回True，否则返回False，可以用S>T判断S是否是T的真超集

集合

- 集合类型有10个操作函数或方法

可变集合

函数或方法	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中所有数据项
S.copy()	返回集合S的一个拷贝
S.pop()	随机返回集合S中的一个元素，如果S为空，产生KeyError异常
S.discard(x)	如果x在集合S中，移除该元素；如果x不在，不报错
S.remove(x)	如果x在集合S中，移除该元素；不在产生KeyError异常
S.isdisjoint(T)	如果集合S与T没有相同元素，返回True
len(S)	返回集合S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

序列解包

- 本质：对多个变量同时赋值
- 把一个序列或可迭代的对象中的多个元素的值同时赋值给多个变量，要求等号左侧变量的数量和等号右侧值的数量必须一致。
- 可用于列表、元组、字典、集合、字符串等等。

序列解包

```
x, y, z = 1, 2, 3                    # 多个变量同时赋值
x, y, z = (False, 3.5, 'exp')        # 元组支持序列解包
x, y, z = [1, 2, 3]                  # 列表支持序列解包
x, y = y, x                          # 交换两个变量的值
data = {'a': 97, 'b': 98}
x, y = data.values()                 # 使用字典的“值”进行序列解包
```

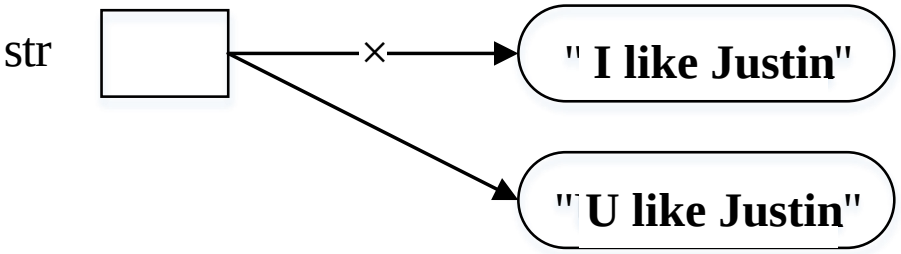
可变与不可变类型

- 不可变类型

str="I like Justin"变成"U like Justin"

str=‘U’+str [1:len(str)]

str=str.replace(“I”，“U”）



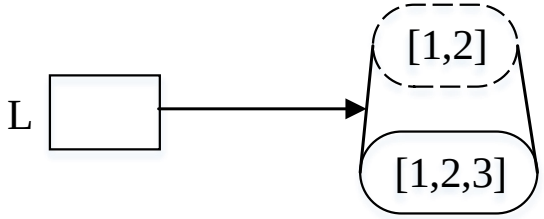
可变与不可变类型

- 总结分片和“+”
  - 不管是可变还是不可变的序列类型：
    - ✓分片必定产生新的序列；
    - ✓“+”号在等号右边，必定产生新的序列。然后将新的序列地址赋予给等号左边的变量

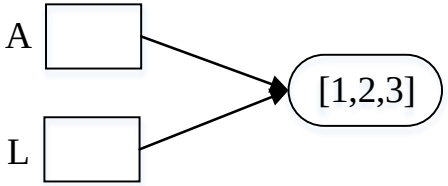
可变与不可变类型

- 可变类型
  - 列表的append操作
  - L=A
  - L=A[:]

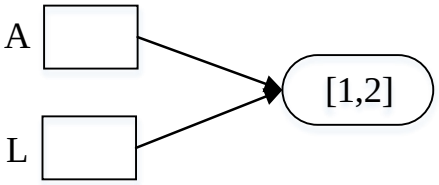
L=[1,2]，执行L.append(3)后



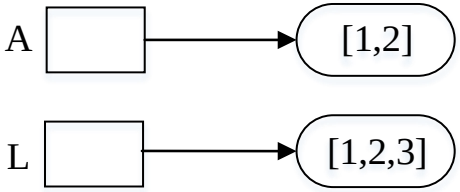
A=[1,2]，则L=A



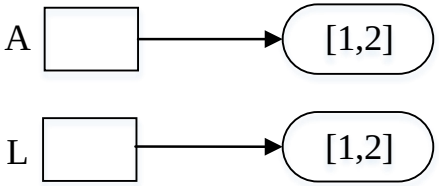
L=A　　L.append(3)



A=[1,2]，则L=A[:]



L=A[:]　L.append(3)



可变与不可变类型

- 向列表中添加新元素L=L+[i]， L.append(i)，L+=[i]，这三种方法的差别：
  - L=L+[i]在每一次执行时都会将原列表复制一次，L指向新列表，并在新列表中加入新元素。
  - L.append(i)只是将新元素直接添加到原列表中，不会产生新列表。
  - L+=[i]的执行效果和L.append(i)类似，也是在原列表中直接添加元素，不会复制原列表。

可变与不可变类型

- 增强赋值语句
  - “A+=B”、“A-=B”、“A*=B”这类语句都是增强赋值语句。
  - 虽然Python的增强赋值语句的独特之处
    - ✓对于不可变变量来说，A+=B其实就等价于A=A+B
    - ✓对于可变变量来说，A+=B是直接在原值的基础上做修改。

小结

- 数据结构类型
  - 列表
  - 元组
  - 字典
  - 集合