第3章 Python复杂数据类型

AI程序设计课程组

目录页

列表

列表概述

- 列表（list）是一个有序的序列结构，序列中的元素可以是不同的数据类型
- 操作: 索引、切片、加、乘和检查成员等

A = [1,'xiaoWang','a', [2, 'b']]

列表创建

- 将列表中的各元素用逗号分隔开，并用中括号将所有元素包裹起来

```
: classesinfo=['AI Programming',40,'Machine Learning2',64]
  print(classesinfo)
  print(type(classesinfo))
```

```
['AI Programming', 40, 'Machine Learning2', 64]
<class 'list'>
```

列表索引

- Python语言中所有的索引都是从 0 开始计数的，如果列表中有 n 个元素，那么最后一个元素的索引是 n-1
- ['AI Programming',40,'Machine Learning2',64]

Index(索引位置)	0	1	2	3
Value(值)	'AI Programming	40	Machine Learning2	64

```
In [23]: print(classesinfo[2])
         classesinfo[0]

         Machine Learning2

Out[23]: 'AI Programming'
```

列表中元素的增加

在列表中增加元素的方式有多种，具体如下：

- 通过append可以向列表添加元素
- 通过运算符“+”可以向列表添加元素
- 通过extend可以将另一个列表的元素添加到列表中
- 通过insert在指定位置index前插入数据元素

列表元素的增加

在列表中增加元素的方式有多种，具体如下：

- append()方法
- 运算符“+”

In [9]:
```
classesinfo.append('Intelligent Hardware')
print(classesinfo)
classesinfo+=[48]
print(classesinfo)
```

```
['AI Programming', 40, 'Machine Learning2', 64, 'Intelligent Hardware']
['AI Programming', 40, 'Machine Learning2', 64, 'Intelligent Hardware', 48]
```

列表元素的增加

列表元素的增加：

- extend()方法在列表后扩充，效果与“+=”一致

[63]:
```
print(classesinfo)
classesinfo.extend(['Intelligent Hardware',40 ])
print(classesinfo)
```

```
['AI Programming', 40, 'MachineLearning', 64]
['AI Programming', 40, 'MachineLearning', 64, 'Intelligent Hardware', 40]
```

列表元素的增加

列表元素的增加：

- Insert()方法在指定的索引位置添加数据元素

```
print(classesinfo)
classesinfo.insert(2,'c1')
print(classesinfo)
```

```
['AI Programming', 40, 'MachineLearning', 64, 'Intelligent Hardware', 40]
['AI Programming', 40, 'c1', 'MachineLearning', 64, 'Intelligent Hardware', 40]
```

列表元素的删除

列表中删除元素的方式有多种，具体如下

- remove：根据元素的值进行删除
- del：根据下标进行删除
- pop：删除指定位置元素

使用remove()方法删除元素

```
['AI Programming', 40, 'Machine Learning2', 64, 'Intelligent Hardware']
```

```
#利用列表remove函数删除特定的元素
classesinfo.remove('Intelligent Hardware')
print(classesinfo)
```

```
['AI Programming', 40, 'Machine Learning2', 64]
```

列表元素的删除

remove()方法适用于知道要删除的值的情况，当我们不知道具体元素值，但是知道元素的索引位置时，我们可以使用 del 函数配合列表索引，删除索引位置的元素或者使用 pop()方法。

```
['AI Programming', 40, 'Machine Learning2', 64, 'Intelligent Hardware', 48]
```

```
#利用del函数对特定位置元素删除
del(classesinfo[5])
print(classesinfo)
```

```
['AI Programming', 40, 'Machine Learning2', 64, 'Intelligent Hardware']
```

列表元素的删除

使用 pop()函数删除特定位置元素

```
['AI Programming', 40, 'Machine Learning2', 64]
```

```
classesinfo.pop(2)
print(classesinfo)
```

```
['AI Programming', 40, 64]
```

查找列表中的元素

在在列表中查找元素的方法包括：

- in（存在），如果对象的值出现在列表中则结果为true，否则为false。
- not in（不存在），如果对象的值不出现在列表中则结果为true，否则false。

```
classesinfo = ['AI Programming', 40, 'Machine Learning', 64, 'Intelligent Hardware']
print(40 in classesinfo)
print('English' not in classesinfo)
print(30 in classesinfo)
```

```
True
True
False
```

修改列表中的元素

通过下表修改列表中的元素

```
classesinfo = ['AI Programming', 40, 'Machine Learning', 64, 'Intelligent Hardware']
classesinfo[1] = 50
print(classesinfo)
```

```
['AI Programming', 50, 'Machine Learning', 64, 'Intelligent Hardware']
```

列表长度

- Python内置的用于判断列表长度的函数为 len()

```
print(len(classesinfo))
classesinfo
```

```
7

['AI Programming', 40, 'c1', 'MachineLearning', 64, 'Intelligent Hardware', 40]
```

列表切片

- 切片操作需要提供起始索引位置和最后索引位置，然后用冒号：将两者分开

  列表名称[起始索引位置:最后索引位置:步长]

- 如果未输入步长，则默认步长为 1
- 切片操作返回一系列从起始索引位置开始到最后索引位置结束的数据元素
- 注意：起始索引位置的值包含在返回结果中，而最后索引位置的值不包含在返回结果中

列表切片

- 注意：起始索引位置的值包含在返回结果中，而最后索引位置的值不包含在返回结果中

[103]:
```
print(classesinfo)
print(classesinfo[3:-1])
print(classesinfo[5:2:-1])
print(classesinfo[3:len(classesinfo)])
```

```
['AI Programming', 40, 'c1', 'MachineLearning', 64, 'Intelligent Hardware', 40]
['MachineLearning', 64, 'Intelligent Hardware']
['Intelligent Hardware', 64, 'MachineLearning']
['MachineLearning', 64, 'Intelligent Hardware', 40]
```

列表其他操作

- 列表排序和比较Sort和sorted
- 列表的最大值和最小值
- 判断元素是否在列表中

列表排序和比较

列表排序sort()和sorted()方法，只能对同种类型数据操作：List.sort(key=none,reverse=true)方法对列表元素进行排序（默认按照从小到大进行排序），排序后改变原列表元素的顺序。sorted()方法由python提供对列表型数据进行排序，排序后不改变原列表数据的顺序。

```
#利用sort函数对列表进行排序
numbers=[18,45, 67, 12, 1, 3]
print(numbers)
numbers.sort()
print(numbers)
```

```
[18, 45, 67, 12, 1, 3]
[1, 3, 12, 18, 45, 67]
```

```
#利用sorted函数对列表进行排序，
numbers=[18,45, 67, 12, 1, 3]
tempData=sorted(numbers)
print(tempData)
print(numbers)
```

```
[1, 3, 12, 18, 45, 67]
[18, 45, 67, 12, 1, 3]
```

列表最大值和最小值

- max()方法和min()方法

In [128]:
```
#列表元素最大和最小值
print(max(numbers))
print(min(numbers))
```

```
67
1
```

嵌套列表

- 列表中的元素也可以是列表，这样可以将列表看成更高维的数组

In [137]:
```
#嵌套列表，可以表示高维数组
classesinfo=[['AI Programming', 40],[ 'MachineLearning', 64]\
             ,[ 'Intelligent Hardware', 40]]
print(classesinfo[1][0])
print(classesinfo[1][1])
```

```
MachineLearning
64
```

元组

元组

- 元组（tuple）数据结构与列表类似，其中元素可以有不同的类型
- 但是元组中的**元素是不可变**的，即一旦初始化之后，就不能够再做修改（报错：元组对象不支持赋值）

```
tup1 = ('physics', 'chemistry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5 )
tup3 = "a", "b", "c", "d"          # 不要括号也可以
```

- 元组中只包含一个元素时，需要在元素后面添加逗号“，”，否则括号会被当做运算符

```
In [39]: tuple1=(50)#不加逗号，为整型数据
         type(tuple1)

Out[39]: int

In [40]: tuple2=(50,)#加逗号，为元组数据
         type(tuple2)

Out[40]: tuple
```

元组元素的增加


```
:  numbers1=(18,45, 67, 12, 1, 3)
   numbers1[45]=34
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-156-8e9cd6a02f1d> in <module>
      1 numbers1=(18,45, 67, 12, 1, 3)
----> 2 numbers1[45]=34

TypeError: 'tuple' object does not support item assignment
```

不可修改

元素的增加

```
[141]: #但是元组中的元素是不可变的，即一旦初
       numbers1=(18,45, 67, 12, 1, 3)
       numbers1+=(23,2)
       print(numbers1)
```

```
t[141]: (18, 45, 67, 12, 1, 3, 23, 2)
```

删除元组

- 元组中的元素值是不允许删除的，但可以使用del语句来删除整个元组

```
classesinfo = ('AI Programming', 40, 'Machine Learning', 64, 'Intelligent Hardware')
print(classesinfo)

del classesinfo
print(classesinfo)
```

```
('AI Programming', 40, 'Machine Learning', 64, 'Intelligent Hardware')
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-14-39a112560d78> in <module>
      3
      4 del classesinfo
----> 5 print(classesinfo)

NameError: name 'classesinfo' is not defined
```

元组操作

- 索引
- 切片
- 重复（*）

```
aa = (1, 2, 3, 4)
print(aa * 4)
```

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

- 成员操作符(in/not in)
- 排序（sorted）

字典

字典概述

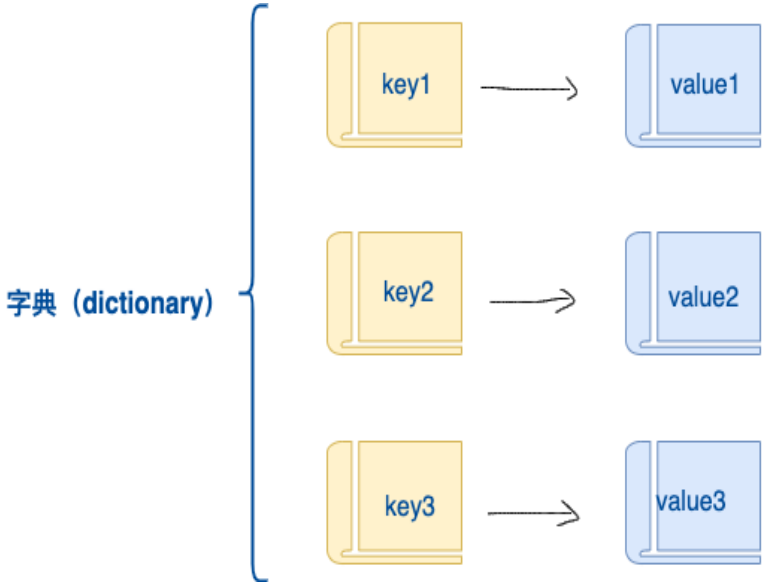
- 字典（dict）在其他语言中被称作哈希映射（hash map）或者相关数组（associative arrays）
- 字典是一种大小可变的键值对集，其中的键（key）和值（value）都是Python对象

```
d = {key1 : value1, key2 : value2, key3 : value3 }
```

- 字典用在需要高速查找的地方

字典概述

- 注意：dict 作为 Python 的关键字和内置函数，变量名不建议命名为 dict
- 键必须是唯一的，但值则不必
- 值可以取任何数据类型，但键必须是不可变的，如字符串，数字



字典创建

- 创建字典

```
tinydict1={'ai':'hello'}
tinydict2={'ai':'hello',1:'world'}
print(tinydict1)
print(tinydict2)
```

```
{'ai': 'hello'}
{'ai': 'hello', 1: 'world'}
```

- 创建空字典：使用大括号 {} 来创建空字典

```
emptyDict={}
print(emptyDict)
```

```
{}
```

- 创建空字典：使用内建函数 dict() 创建字典

```
emptyDict = dict()
print(emptyDict)
```

```
{}
```

字典创建与索引

In [168]:
```
#字典创建于索引
addressinfo = {'zs': '2341', 'ls': '9102', 'ww': '3258',1:8888}
print(addressinfo)
#此处的‘1’不是索引，是键值#
print(addressinfo[1])
print(addressinfo['zs'])
#若访问不存在的键，则会报错
print(addressinfo[3])
```

```
{'zs': '2341', 'ls': '9102', 'ww': '3258', 1: 8888}
8888
2341

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent c
```

字典索引（访问/查找）

- 字典中某值的索引还可以通过 get 方法，如果字典不包含某个键，可以返回 None，或者自己指定的值
- 我们可以通过 in 判断是否存在某个键，其语法跟在列表和元组中判断是否存在某个值是相同的
  如果不太确定字典中有哪些键或者值，我们可以使用 keys() 方法或者values()方法

```
#dict主键查询
addressinfo = {'zs': '2341', 'ls': '9102', 'ww': '3258',1:8888}
print(addressinfo.get('zs'))
print('ls' in addressinfo)
```

字典修改与增加

- 直接对字典中键对应的值进行赋值。Key是唯一的，value的值可以重复。

[175]:
```
print(addressinfo)
addressinfo[1]=6666
print(addressinfo)
addressinfo['tj']='8686'
print(addressinfo)
```

```
{'zs': '2341', 'ls': '9102', 'ww': '3258', 1: 8888}
{'zs': '2341', 'ls': '9102', 'ww': '3258', 1: 6666}
{'zs': '2341', 'ls': '9102', 'ww': '3258', 1: 6666, 'tj': '8686'}
```

字典修改与增加

- update方法实现字典元素的修改与增加，可一次增加一个或多个元素

```
#update方法实现字典元素的修改与增加
print(addressinfo)
addressinfo.update({1:8888})
print(addressinfo)
addressinfo.update({'cs':'8644'})
print(addressinfo)
```

```
{'zs': '2341', 'ls': '9102', 'ww': '3258', 1: 6666, 'tj': '8686'}
{'zs': '2341', 'ls': '9102', 'ww': '3258', 1: 8888, 'tj': '8686'}
{'zs': '2341', 'ls': '9102', 'ww': '3258', 1: 8888, 'tj': '8686', 'cs': '8644'
```

字典元素的删除

- del() 方法或字典中的pop()方法删除字典元素

```
#字典元素删除操作
print(addressinfo)
del(addressinfo['cs'])
print(addressinfo)
addressinfo.pop('tj')
print(addressinfo)
```

```
{'zs': '2341', 'ls': '9102', 'ww': '3258', 1: 8888, 'tj': '8686', 'cs': '8644'}
{'zs': '2341', 'ls': '9102', 'ww': '3258', 1: 8888, 'tj': '8686'}
{'zs': '2341', 'ls': '9102', 'ww': '3258', 1: 8888}
```

字典元素的清空

- clear()方法

In [201]:
```
#清空字典中所有元素
print(addressinfo)
addressinfo.clear()
print(addressinfo)
```

```
{'zs': '2341', 'ls': '9102', 'ww': '3258', 1: 8888}
{}
```

字典其他操作

函数及描述	实例
len(dict) 计算字典元素个数，即键的总数。	>>> dict = {'Name': 'Runoob', 'Age': 7, 'Class': 'First'} >>> len(dict) 3
str(dict) 输出字典，以可打印的字符串表示。	>>> dict = {'Name': 'Runoob', 'Age': 7, 'Class': 'First'} >>> str(dict) "{'Name': 'Runoob', 'Class': 'First', 'Age': 7}"
type(variable) 返回输入的变量类型，如果变量是字典就返回字典类型。	>>> dict = {'Name': 'Runoob', 'Age': 7, 'Class': 'First'} >>> type(dict) <class 'dict'>

字典其他操作

在有些情况下，我们需要取出字典中的键值对用于下一步的分析，此时可以使用 items() 方法，该方法将返回所有键值对，并将其保存在一个元组列表（列表中的元素为元组）中

```
myclass = {'name': 'AI Programming', 'credit': 2, 'adress': '综合楼258'}
print("Value: %s" % myclass.items())
```

```
Value: dict_items([('name', 'AI Programming'), ('credit', 2), ('adress', '综合楼258')])
```

字典其他操作

dict.keys()方法返回在字典中的所有可用的键的列表

```
myclass = {'name': 'AI Programming', 'credit': 2, 'adress': '综合楼258'}
print(myclass.keys())
```

```
dict_keys(['name', 'credit', 'adress'])
```

字典其他操作

dict.values()方法返回在字典中的所有可用的值的列表

```
myclass = {'name': 'AI Programming', 'credit': 2, 'adress': '综合楼258'}
print(myclass.values())
```

```
dict_values(['AI Programming', 2, '综合楼258'])
```

集合

集合

- 集合（set）是一种无序集，它是一组键的集合，不存储值
- 在集合中，重复的键是不被允许的。集合可以用于去除重复值
- 集合也可以进行数学集合运算，如并、交、差以及对称差等。

集合创建

- 集合的创建有两种方式：使用 set() 函数或者使用大括号{}
- 需要注意的是，创建空集合，必须使用 set()，而不是{}，因为{}表示创建一个空的字典

```
#集合定义
basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
#自动去重
print(basket)
subbasket={'orange','pear','watermelon'}
# 集合a中包含而集合b中不包含的元素
print(basket-subbasket)
# 集合a和b中都包含了的元素
print(basket&subbasket)
# 集合a或b中包含的所有元素
print(basket|subbasket)
# 不同时包含于a和b的元素
print(basket^subbasket)
```

集合的添加

- s.add()将元素 x 添加到集合 s 中，如果元素已存在，则不进行任何操作。

In [45]:
```
#元素添加到集合中，如果元素已存在，则不进行任何操作。
print(basket)
basket.add('banana')
print(basket)
```

```
{'apple', 'pear', 'orange'}
{'apple', 'banana', 'pear', 'orange'}
```

集合的添加

- s.update(x)也可以添加元素，且参数可以是列表，元组，字典等
- x也可以有多个，用逗号分开

```
basket = {'apple', 'pear', 'orange'}
basket.update({'pear', 'grape'})
print(basket)
```

```
{'pear', 'orange', 'apple', 'grape'}
```

```
basket.update([1, 3], [2, 4])
print(basket)
```

```
{1, 2, 'pear', 3, 'orange', 4, 'apple', 'grape'}
```

集合的删除

- s.remove(x)将元素 x 从集合 s 中移除，如果元素不存在，则会发生错误
- s.discard(x)将元素 x 从集合 s 中移除，如果元素不存在，不会发生错误
- s.pop() 随机删除集合中的一个元素

```
basket = {'apple', 'pear', 'orange'}
basket.remove('apple')
print(basket)
```

```
{'pear', 'orange'}
```

```
basket = {'apple', 'pear', 'orange'}
basket.pop()
print(basket)
```

```
{'pear', 'orange'}
```

集合运算

- 集合支持数学集合运算，如并、交、差以及对称差等

表2 Python的集合运算

函数	其他表示法	说明
a.add(x)	N/A	把元素x添加到集合a中
a.remove(x)	N/A	把元素x从集合a中删除
a.union(b)	a | b	a和b中全部的唯一元素
a.intersection(b)	a & b	a和b都有的元素
a.difference(b)	a - b	a中不属于b的元素
a.symmetric_difference(b)	a ^ b	a或b中不同时属于a和b的元素
a.issubset(b)	N/A	如果a的全部元素都包含于b，则为True
a.issuperset(b)	N/A	如果b的全部元素都包含于a，则为True
a.isdisjoint(b)	N/A	如果a和b没有公共元素，则为True

总结

- 列表、元组、字典、集合
- 创建、索引、增减、切片操作