人工智能实践教程
从Python入门到机器学习

- 所有代码及ppt均可以由以下链接下载
- https://github.com/shao1chuan/pythonbook
- https://gitee.com/shao1chuan/pythonbook

目录

CONTENTS

01 序列

1.成员有序排列的,且可以通过下标偏移量访问到它的一个或者几个成员,这类类型统称为**序列**。

2.序列数据类型包括:字符串,列表,和元组类型。

3.特点: 成员关系操作符(in , not), 连接操作符(+), 重复操作符(*), 索引与切片操作符

02 列表： 打了激素的数组

列表的创建与删除

数组: 存储同一种数据类型的集和。scores=[12,95.5]

列表(打了激素的数组): 可以存储任意数据类型的集和。

- 创建一个空列表

  list = []

- 创建一个包含元素的列表，元素可以是任意类型，包括数值类型，列表，字符串等均可, 也可以嵌套列表。

  list = ["fentiao", 4, 'gender']

  list = [['粉条', 100], ["粉丝", 90], ["粉带", 98]]

01 序列

以下代码将输出什么?

```
list = ['a', 'b', 'c', 'd', 'e']

print(list[10:])
```

01 序列

以下代码将输出什么?

```
list = ['a', 'b', 'c', 'd', 'e']

print(list[10:])
```

注： 以上代码将输出 [] ,并且不会导致一个 IndexError 。

一个讨厌的小问题是它会导致出现 bug ,并且这个问题是难以追踪的,因为它在运行时不会引发错误。

列表特性练习1

根据用于指定月份，打印该月份所属的季节。

提示: 3,4,5 春季 6,7,8 夏季 9,10,11 秋季 12, 1, 2 冬季

考察点: 列表的成员操作符, if判断语句

'3' in ['3', '4', '5']

列表特性练习2

假定有下面这样的列表:

names = ['fentiao', 'fendai', 'fensi', 'apple']

输出结果为:'I have fentiao, fendai, fensi and apple.'

考察点:

切片:

字符串的join方法:

02 列表

列表的添加

列表可通过append追加一个元素到列表中;

```
In [15]: iplist = ['172.25.254.1','172.25.254.19']

In [16]: iplist.append('172.25.254.17')

In [17]: print iplist
['172.25.254.1', '172.25.254.19', '172.25.254.17']

In [18]: iplist.append(('172.25.254.89','172.25.254.99'))

In [19]: print iplist
['172.25.254.1', '172.25.254.19', '172.25.254.17', ('172.25.254.89',
 '172.25.254.99')]
```

思考:如果我想添加多个元素呢?append方法可以实现么?

02 列表

列表的添加

列表可通过extend方法拉伸, 实现追加多个元素到列表中

```
In [21]: iplist.extend(['172.25.254.189','172.25.254.199'])

In [22]: print iplist
['172.25.254.1', '172.25.254.19', '172.25.254.17', ('172.25.254.89',
 '172.25.254.99'), '172.25.254.189', '172.25.254.199']

In [23]: iplist.append(['172.25.254.189','172.25.254.199'])

In [24]: print iplist
['172.25.254.1', '172.25.254.19', '172.25.254.17', ('172.25.254.89',
 '172.25.254.99'), '172.25.254.189', '172.25.254.199', ['172.25.254.
189', '172.25.254.199']]
```

列表的添加

在指定位置添加元素使用insert方法；

L.insert(index, object)

```
In [33]: list1 = ['fentiao',5,'male']

In [34]: list1.insert(1,"fendai")

In [35]: print list1
['fentiao', 'fendai', 5, 'male']
```

列表的修改

修改列表的元素:直接重新赋值；

```
In [25]: list1 = ['fentiao',5,'male']

In [26]: list1[0] = 'fendai'

In [27]: print list1
['fendai', 5, 'male']
```

02 列表

列表的查看

查看某个列表元素的下表用index方法；

查看某个列表元素出现的次数用count方法;

```
In [43]: list1
Out[43]: ['fentiao', 'fendai', 5, 'male']

In [44]: list1.index('fentiao')
Out[44]: 0

In [45]: list1.append('fentiao')

In [46]: list1.index('fentiao')
Out[46]: 0

In [47]: list1.index('fentiao',1,5)
Out[47]: 4

In [48]: list1.count('fentiao')
Out[48]: 2
```

列表的删除

list.remove()删除列表中的指定元素。

list.pop()根据元素的索引值进行删除指定元素。

list.clear: 清空列表里面的所有元素。

```
In [5]: list1 = ['fentiao', 5, 'male']

In [6]: list1.remove('fentiao')

In [7]: print list1
[5, 'male']

In [8]: list1.remove(list1[1])

In [9]: print list1
[5]
```

列表的删除

del(list[])

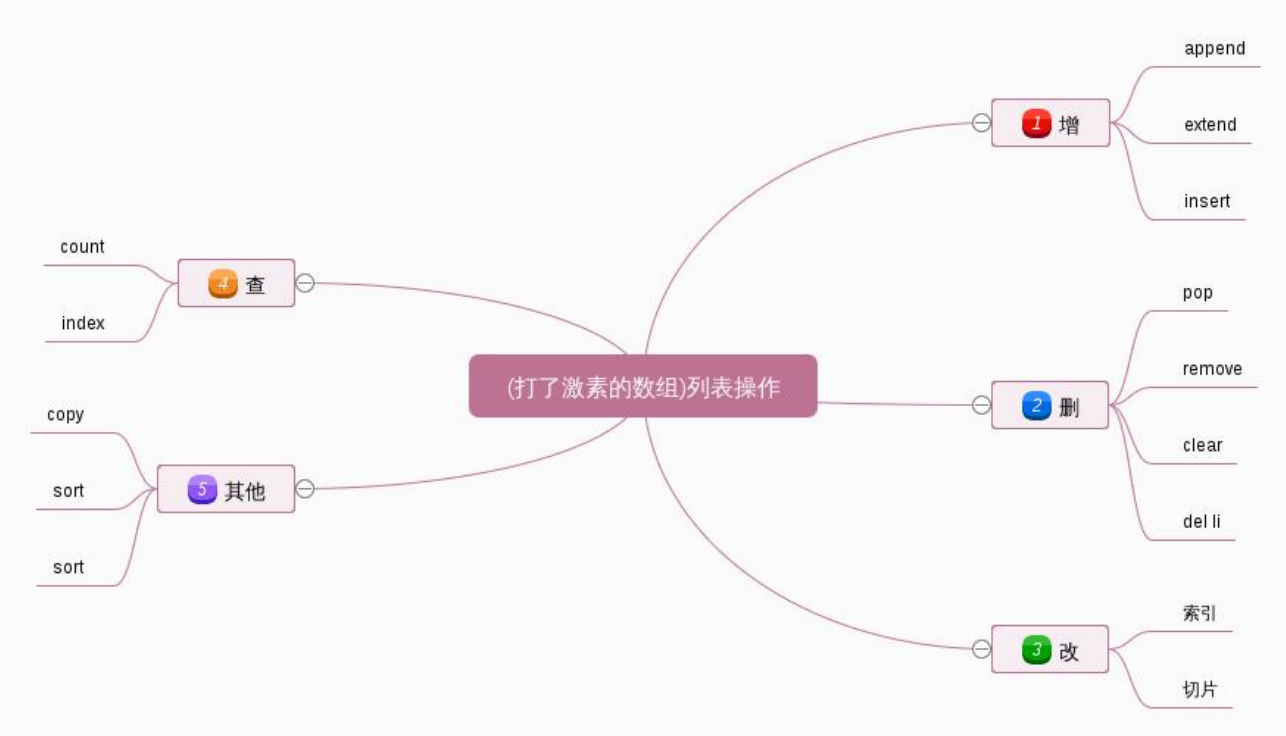
```
In [10]: list1 = ['fentiao', 5, 'male']

In [11]: del(list1)

In [12]: print list1
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-12-b0af4949e4b3> in <module>()
----> 1 print list1

NameError: name 'list1' is not defined
```

总结


(打了激素的数组)列表操作
1 增
append
extend
insert
2 删
pop
remove
clear
del li
3 改
索引
切片
4 查
count
index
5 其他
copy
sort
sort

02

列表

会员信息管理系统

需求分析：

1). 注册、登录、

2). 查询所有会员、修改、删除

补充: is和==的区别

问题: is和==两种运算符在应用上的本质区别是什么?

1). Python中对象的三个基本要素，分别是：id(身份标识)、type(数据类型)和value(值)。

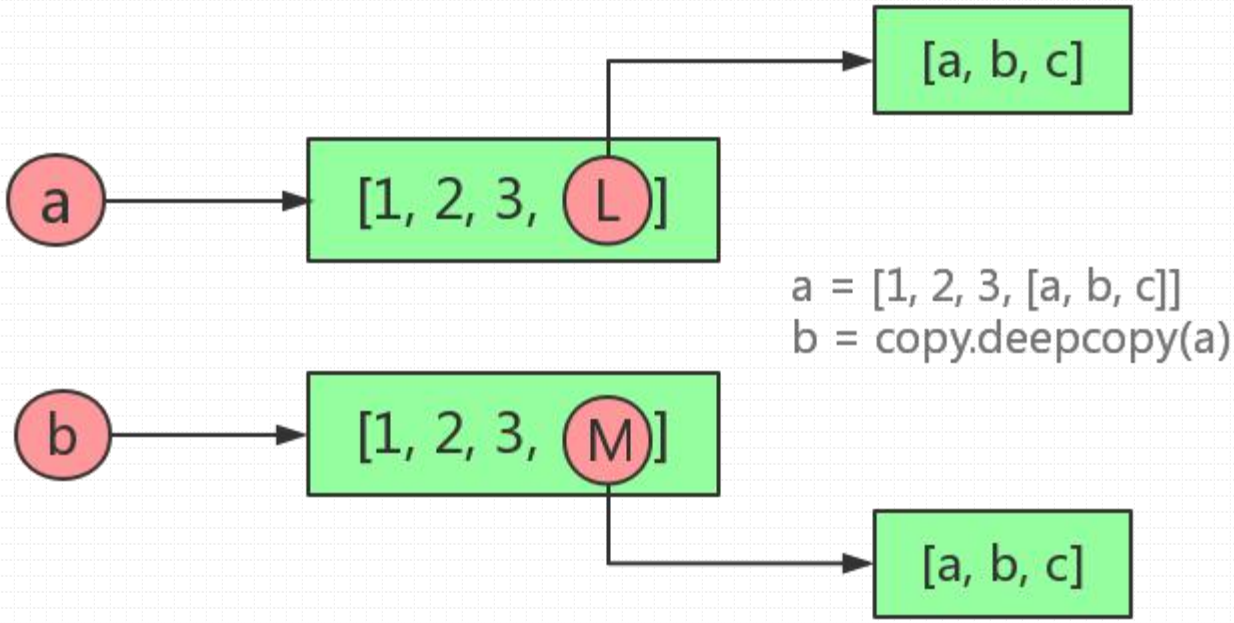
2). is和==都是对对象进行比较判断作用的，但对对象比较判断的内容并不相同。

3). ==用来比较判断两个对象的value(值)是否相等；

is也被叫做同一性运算符， 会判断id是否相同;

补充: 深拷贝和浅拷贝

问题: 深拷贝和浅拷贝的区别?



补充: 深拷贝和浅拷贝

问题: 深拷贝和浅拷贝的区别?/python中如何拷贝一个对象?

赋值： 创建了对象的一个新的引用，修改其中任意一个变量都会影响到另一个。(=)

浅拷贝： 对另外一个变量的内存地址的拷贝，这两个变量指向同一个内存地址的变量值。(li.copy(), copy.copy())

- 公用一个值；
- 这两个变量的内存地址一样；
- 对其中一个变量的值改变，另外一个变量的值也会改变；

深拷贝： 一个变量对另外一个变量的值拷贝。(copy.deepcopy())

- 两个变量的内存地址不同；
- 两个变量各有自己的值，且互不影响；
- 对其任意一个变量的值的改变不会影响另外一个；

03

元组:带了紧箍咒的列表

元组的创建，删除及特性

- 定义空元组

```
tuple = ()
```

- 定义单个值的元组

```
tuple = (fentiao,)
```

- 一般的元组

```
tuple = (fentiao, 8, male)
```

特性:

连接操作符，重复操作符，成员操作符

索引， 切片

注意: 不能对元组的值任意更改；

```
In [30]: t1 = ('fentiao',4,'male')

In [31]: t1[0] = 'fendai'
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-31-8870739c2bca> in <module>()
----> 1 t1[0] = 'fendai'

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

03 元组:带了紧箍咒的列表

元组的创建，删除及特性

特性:

对元组分别赋值,引申对多个变量也可通过元组方式分别赋值

```
In [33]: t1 = ('fentiao',4,'male')

In [34]: name,age,gender = t1

In [35]: print name,age,gender
fentiao 4 male
```

03 元组:带了紧箍咒的列表

多元赋值机制

x, y, z = 1, 2, 'westos' 等价于 (x, y, z) = (1, 2, 'westos') , 等号两边的对象都是元组并且元组的小括号是可选的。

```
# 元组的赋值：有多少个元素，  就用多少个变量接收
t = ('westos', 10, 100)
name, age,score  = t
print(name, age, score)


#
scores = (100, 89, 45, 78, 65)
# 先对元组进行排序
# scoresLi = list(scores)
# scoresLi.sort()
# print(scoresLi)

scores = sorted(scores)
# python3中
minScore, *middleScore, maxScore = scores
print(minScore, middleScore, maxScore)
print("最终成绩为：%.2f" %(sum(middleScore)/len(middleScore)))
```

注：python3中不能用*middleScore

03 元组:带了紧箍咒的列表

环境变量交换

```
x=2; y=10;x, y = y, 2; print(x,y)
```

这种交换方式无需中间变量即可交换两个变量的值。那么具体实现机制是怎样的呢？

03

元组:带了紧箍咒的列表

元组的方法

t.count(value)-->int

返回value在元组中出现的次数；

t.index(value)

返回value在元组中的偏移量(即索引值)

03 元组:带了紧箍咒的列表

深入理解环境变量交换

实现机制：

1). 构造一个元组(y, x);

2). 构造另一个元组(x, y)；

3). 元组(y, x)赋值给(x, y)，元组赋值过程从左到右，依次进行

04 命名元组

用法简介

Tuple还有一个兄弟，叫**namedtuple**。虽然都是tuple，但是功能更为强大。

collections.namedtuple(typename, field_names)

typename：类名称

field_names: 元组中元素的名称

04 命名元组

定义命名元组类与应用

命名元组是一个类，有两种方式来定义命名元组：

```
from collections import namedtuple

User = namedtuple('User', ['name', 'age', 'id'])
User = namedtuple('User', 'name age id')
```

实例化命名元组，获得类的一个实例：

```
user = User('tester', '22', '464643123')
```

访问命名元组: 通过逗号运算符和属性名来访问元组字段的值

user.name; user.age, user.id

04 命名元组

命名元组的属性

类属性 _fields：包含这个类所有字段名的元组

```
>>> user._fields
('name', 'age', 'id')
```

类方法 _make(iterable)：接受一个可迭代对象来生产这个类的实例

```
>>> User._make(['westos', 12, 132001])
User(name='westos', age=12, id=132001)
```

实例方法_replace()：用于修改实例的属性

感谢聆听！

THANK YOU!