### 元组

#### 特点

##### 元组是位置有序的对象集合

##### **元组不支持任何方法调用**

##### 按偏移（键）访问，支持索引、分片

##### 属于不可变序列

##### 支持嵌套（字典、列表和其它元组等）

#### 基本操作

# 单个元组，注意逗号 (0) 不是元组  
T = (0,)  
# 四个元素元组  
T = 1, 'O', 1.2, 3  
# 可迭代对象的项的元组  
T = tuple('spam')  
print(T) # ('s', 'p', 'a', 'm')  
# 搜索、计数  
T.index('s')  
T.count('p')  
  
# 支持类似字符串操作 结果返回新元组  
print((1, 2) \* 3) # (1, 2, 1, 2, 1, 2)  
  
T = ('a', 'b', 'c', 'd')  
# 元组排序  
list(T).sort()  
print(T)  
sorted(T)  
print(T)  
  
# 元组列表解析  
L = [x \* 2 for x in T]  
print(L)  
  
# 元组不可修改，但元组内的列表可以修改  
T = (1,'a', ['b','d'])  
T[2][1] = 'c'  
print(T)  
  
# 元组 通常用于指关系数据库表的一行  
# 用于字典键

### 文件

#### 常见文件操作

# 创建输出文件（写入）  
output = open(r'C:\spam','w')  
# 创建输入值 （r 默认）  
input = open('data')# 'r')  
# 将整个文件读入单一字符串  
aString = input.read()  
aString = input.readlines()  
# 读取N个字节进字符串  
aString = input.read(N = 5)  
# 读取下一行（包括行尾标识符）到一个字符串  
aString = input.readline()  
# 将字节字符串写到文件  
output.write(aString)  
  
L = ['hello','world']  
# 把列表中所有字符串写到文件  
output.writelines(L)  
  
# 关闭  
output.close()  
  
# 把输出缓冲区刷到硬盘中，但是不关闭文件  
output.flush()  
  
# 修改文件位置到偏移量N处 用于下一次操作  
output.seek(N=3)  
  
# 一行行读取  
for line in input: print(line)  
  
# 指定编码  
open(r'c:\raw.txt','w',encoding='utf-8')  
  
# 二进制文件  
open('f.bin','rb')

a 追加

w 写

r 读

b 二进制处理

+同时读写

？？ pickle 模块 struct模块

* 记得 手动关闭文件 f.close()
* 文件是缓冲的并且是可查找的

# 例一  
f = open('b.txt', 'w')  
f.write("hello world\n")  
f.write("done\n")  
f.writelines(['oh', 'here', 'yours'])  
f.close()  
  
f = open('b.txt', 'r')  
str = f.readline()  
print(str)  
str = f.readline()  
print(str)  
str = f.readline()  
print(str)  
  
f.close()  
  
# 例2  
print(open('b.txt', 'r').read())  
  
for line in open('b.txt', 'r'):  
 print(line, end='\t')

#### 在文件中存储Python对象

X, Y, Z = 43, 44, 45  
S = 'spam'  
L = ['a',47,'b',48]  
T = ('I am tuple',['are','you',20])  
D = dict.fromkeys(['a','b','c'],0)  
  
f = open('c.txt','w')  
f.write(S + '\n')  
f.write("%s,%s,%s\n" % (X,Y,Z))  
f.write(str(L) + '&' + str(T) + '\n')  
f.write(str(D))  
f.close()  
  
#   
f = open('c.txt', 'r')  
line = f.readline()  
print(line)  
  
line = f.readline()  
parts = line.split(',')  
numbers = [int(x) for x in parts]  
print(numbers)  
  
# 用eval()转换原生对象  
line = f.readline()  
parts = line.split('&')  
objects = [eval(x) for x in parts]  
print(objects)  
f.close()  
  
# 用pickle存储python原生对象  
D = {'a': 1, 'b': 2}  
f = open('data.pk1', 'wb')  
import pickle  
  
pickle.dump(D, f)  
f.close()  
  
# 取回字典内容  
f = open('data.pk1','rb')  
E = pickle.load(f)  
print(E)  
  
f.close()

#### 二进制存储与解析

# 生成一个打包的二进制数据文件  
f = open('data.bin', 'wb')  
import struct  
  
data = struct.pack('>i4sh', 7, 'spam', 8)  
print(data)  
f.write(data)  
f.close()

#### 文件上下文管理器

with open('data.bin', 'wb') as f:  
 f.write(b'hello')  
with open('data.bin', 'rb') as f:  
 for line in f.readline():  
 print(line)  
  
# 上面等价于  
f = open('data.bin','rb')  
try:  
 for line in f:  
 pass  
finally:  
 f.close()

### 其它要点总结

* 赋值操作存储对象的引用，而不是对象的拷贝！
* 共享引用问题

L = ['a','b','c']  
X = [L,'d','e']  
D = {'L':L,'X':X,'D':'good'}  
  
print(D)  
# {'L': ['a', 'b', 'c'], 'X': [['a', 'b', 'c'], 'd', 'e'], 'D': 'good'}  
  
L[1] = 'g'  
print(D)  
# {'L': ['a', 'g', 'c'], 'X': [['a', 'g', 'c'], 'd', 'e'], 'D': 'good'}

* 拷贝的四种方式
  + 分片： X = L[:]
  + copy : X.copy()
  + 内置对象: list(L)
  + copy标准库模块
* 比较、相等性和真值
* L1 = [1, ('a', 3)]  
  L2 = [1, ('a', 3)]  
  result = L1 == L2, L1 is L2  
    
  print(result)  
  # (True, False)  
    
  # ==测试相等性  
  # is测试一致性  
    
  # 递归比较大小  
  L1 = [1, ('a', 3)]  
  L2 = [1, ('a', 2)]  
  result = L1 > L2, L1 < L2  
    
  print(result)  
  # (True, False)
* 字典比较大小
* D = {'a': 1, 'd': 2, 'c': 3, 'e': 4}  
  D1 = {'a': 1, 'd': 3, 'c': 1, 'e': 5}  
  print(list(D.items()))  
  sorted(list(D.items()))  
  print(sorted(list(D.items())) < sorted(list(D1.items())))  
  # True 只会比较第一个key
* bool : 非0、非空即为真
* None 相当于 NULL指针
* bool()： 测试一个对象的bool值
* type(): 判断类型 isinstance(elem, type)
* 深复制 与 浅赋值

浅复制 : aList[ : ] adict.copy()

深复制 : copy.deepcoy(x)

* 切记 就算没有圆括号 1,2,3 都是一个元组