

第一题：

def f1(L):

return list(set(L))

def f2(L):

lst=[]

for x in L:

if x not lst:

lst.append(x)

return lst

def f3(L):

dic={}

for x in L:

dic[x]=0

return list(dic.keys())

L=[1,25,53,1,25,35]

L=[1,1,25,25,35,53]

L=[1,25,35,53]

def f4(L):

pass

L=[1]

def f5(L):

lst=L.copy()

for x in lst:

if L.count(x) >1:

L.remove(x)

return L

第二题：

1.浅拷贝 ：

使用copy.copy,它可以进行对象的浅拷贝(shallow copy),它复制了对象,但对于对象中的元素,依然使用引用（换句话说修改拷贝对象元素，则被拷贝对象元素也被修改）

2.深拷贝 ：

使用copy.deepcopy，它可以进行深拷贝，不仅拷贝了对象，同时也拷贝了对象中的元素，获得了全新的对象，与被拷贝对象完全独立，但这需要牺牲一定的时间和空间。

3.特殊拷贝：

如要复制列表L,使用list(L),要复制一个字典d,使用dict(d),要复制一个集合s,使用set(s)。

总结一下的话：如果你要复制某个对象object, 它属于python内建的类型type,那么你可以使用type(object)来 获得一个拷贝。

4 举例：

import copy

list = [1, 2, 3, 4, ['a', 'b']] #原始对象

b = list #赋值，传对象的引用，依然指向list

c = copy.copy(list) #对象拷贝，浅拷贝（元素依然是共享的引用）

d = copy.deepcopy(list) #对象拷贝，深拷贝

list.append(5) #修改对象list

list[4].append('c') #修改对象list中的['a', 'b']数组对象

print 'list = ', list

print 'b = ', b

print 'c = ', c

print 'd = ', d

输出结果：

list = [1, 2, 3, 4, ['a', 'b', 'c'], 5]

b = [1, 2, 3, 4, ['a', 'b', 'c'], 5]

c = [1, 2, 3, 4, ['a', 'b', 'c']]

d = [1, 2, 3, 4, ['a', 'b']]

第三题：

Python的except用来捕获所有异常， 因为Python里面的每次错误都会抛出 一个异常，所以每个程序的错误都被当作一个运行时错误。

一、下面是使用except的一个例子：

try:

foo = opne(̶file”) #open被错写为opne

except:

sys.exit(̶could not open file!”)

因为这个错误是由于open被拼写成opne而造成的，然后被except捕获，所以debug程序的时候很容易不知道出了什么问题

二、下面这个例子更好点：

try:

foo = opne(̶file”) # 这时候except只捕获IOError

except IOError:

sys.exit(̶could not open file”)

第四题：

pass

1、空语句，什么也不做

2、在特别的时候用来保证格式或是语义的完整性

第五题：

语法：

range(stop)

range(start, stop[, step])

range函数是一个用来创建算数级数序列的通用函数，返回一个结构的整数序列；range函数具有一些特性：

1.如果step参数缺省，默认1；如果start参数缺省，默认0。

2.如果step是正整数，则最后一个元素（start + i \* step）小于stop。

3.如果step是负整数，则最后一个元素（start + i \* step）大于stop。

4.step参数必须是非零整数，否则抛出VauleError异常。

注意：range函数返回一个左闭右开（[left,right)）的序列数

第六题：

使用global修饰变量名