华东师范大学计算机科学技术系作业

	华东师范大学计算机科学技术系作业	
课程名称：编程导论 Python	年级：2018级	作业成绩：
指导教师：杨燕	姓名：吴子靖	提交作业日期：2018年11月29日
专业：计算机系	学号：10185102141	作业编号： 7

一、异常处理练习。假设输入一组任意长度列表，我们要对该列表中第10个元素进行加1操作，请利用try-except模型自己实现一个异常处理，可以捕获IndexError异常。
（15分）

In [1]:

```
L=list(map(int,input().split()))
try:
    L[9]+=1
    print(L)
except IndexError:
    print("列表中元素不足十个")
```

```
1 2 3 4 5 6 7 8 9 10
[1, 2, 3, 4, 5, 6, 7, 8, 9, 11]
```

In [2]:

```
L=list(map(int,input().split()))
try:
    L[9]+=1
    print(L)
except IndexError:
    print("列表中元素不足十个")
```

```
1 2 3 4 5
列表中元素不足十个
```

二、 用二分法的递归方式求n个元素列表的最大值和最小值，改写本章的<程序：求数列最大最小值——二分法>，传递参数时不用分片，而是用它在原来列表的索引及长度。然后分析程序的开销，开销的增长趋势是什么?
（15分）

In [17]:

```
def find_min_max(L,n):
    if n==2:
        if L[0]<L[1]:
            return L[0],L[1]
        else:
            return L[1],L[0]
    if n==1:
        return L[0],L[0]
    if n==0:
        return [],[]
    else:
        m,p=find_min_max(L,n-1)
        if m>L[n-1]:
            m=L[n-1]
        if p<L[n-1]:
            p=L[n-1]
        return m,p
L=[12,56,24,98,56]
a,b=find_min_max(L,len(L))
print("最小值为:",a)
print("最小值为:",b)
```

```
最小值为: 12
最小值为: 98
```

分析：二分法通过不断分片，最终得到若干个长度为1或2的列表，再通过逐一比较，再得到若干个长度为2或1的新列表，直到最后只剩两个元素，所用的时间会随着n的增长而增长，且n加1，比较次数就加若干，是非线性函数，而我所用的方法是将n个元素的范围缩短到n-1,n-2,n-3……直到2个元素，然后逐个进行大小比较，n每加1，就要多比较两次，是一个斜率为2的一次函数，随着n不断增大，由于基数的增加，二分法的增幅会不断增大，导致所用的时间开销会逐渐大于后者，直到最后会有极大差距，远慢于我所用的办法

三、用二分法的递归方式实现求给定数列L中所有元素的平均数。例如给定数列L=[12,32,45,78,22]，则该数列平均数为(12+32+45+78+22)/5=37.8。
（15分）

In [13]:

```
def average(L,m=len(L)):
    if len(L)==1:
        return (L[0]/m)
    else:
        return (average(L[:len(L)//2])+average(L[len(L)//2:]))
average([12,32,45,78,22])
```

Out[13]:

37.8

四、用递归方法实现求给定正整数n的阶乘n!。例如n=3，则n的阶乘为$1*2*3=6$。
（15分）

In [6]:

```
def jiecheng(x):
    if x==1:
        return 1
    else:
        return (x*jiecheng(x-1))
jiecheng(int(input()))
```

3

Out[6]:

6

五、用递归方法实现求给定列表L中所有元素的最小值。如L=[11,15,9,14,8,5]，则最小数为5。
（15分）

In [8]:

```
def min_L(L):
    if len(L)==1:
        return L[0]
    if len(L)==0:
        return []
    min1=min_L(L[0:len(L)//2])
    min2=min_L(L[len(L)//2:])
    if min1>min2:
        min1=min2
    return min1
min_L([11,15,9,14,8,5])
```

Out[8]:

5

六、调用import time库，编写一个程序能测试<程序：非递归实现my_remove>和它的较好程序的时间差异。建议所要删除的元素是一个长列表的最后一个。
（15分）

In [27]:

```
import time
def my_remove(L,x):
    if not x in L:
        return L
    A=[]
    for i in range (len(L)):
        if L[i]==x:
            A=A+L[i+1:]
            break
        A=A+[L[i]]
    return A
def my_remove_better(L,x):
    A=L[:]
    for i in range (len(L)):
        if L[i]==x:
            A=A[:i]+A[i+1:]
            break
        return A
L=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
time1=time.clock()
my_remove(L,20)
time2=time.clock()-time1
time3=time.clock()
my_remove_better(L,20)
time4=time.clock()-time3
time5=time2-time4
print("原始程序所用时间为:",time2)
print("改进程序所用时间为:",time4,end=",")
print("比原始程序快了:",time5)
```

```
原始程序所用时间为: 9.930385351708537e-05
改进程序所用时间为: 7.664898203074699e-05,比原始程序快了: 2.2654871486338
38e-05
```

七、解释<程序：递归实现my_remove2使用二分法>的终止条件，为何要考虑长度为1的情形？（10分）

当列表的长度为1时，如果x在L内，那么返回L[1:]得到空列表，没有问题。但是如果x不在L之内，A1=L[0:(1//2)]=L[0:0],即A1=L，相当于再进行一次my_remove2(L,x),而同样的又一次运算A1=L[0:0]，反反复复没有止境的算下去，造成了死循环。所以为了防止列表长度为1，而x又不在L内的情形，需要在终止条件额外编写 return L;防止死循环的产生。