一 匿名函数和递归函数

```
lambda  参数列表(形参):运算表达式(return后面的返回的内容)
def 函数名():
    缩进
```

```
def fn(x):
    return x*x
print(fn(5))


f = lambda x:x*x  # f 就是函数名
print(f(5))
```

```
总结:
    1.lambda并不会带来程序运行效率的提高，只会使代码更加的简洁
    2.如果使用lambda，lambda内不要有循环，因为可读性不好，有的话请使用标准函数来完成，目的是为了代码有可重用性和可读性
    3.lambda只是为了减少单行函数的定义而存在，如果一个函数只有一个返回值，只有一句代码，就可以使用lambda
```

1.2高阶函数

高阶函数其实就是把函数作为参数传入，就是把一个函数作为另一个函数的参数传入

abs() ==> 完成对数字求绝对值计算

```
print(abs(-5))
```

sum()==> 求和

```
sum(可迭代对象，指定一个相加的参数如果没有默认为0)
print(sum([1,2,3,4,5]))
print(sum((1,2,3,4,5),1)) # 元祖计算总和后 在加第二个参数的1
```

round()==> 四舍五入，精度问题，会有坑

https://www.cnblogs.com/bigc008/p/9682105.html

```
print(round(4.5))  # 4 有坑
print(round(4.6))
```

任意两个数字，对两个数字求绝对值后进行求和

```
def ab_sum(a,b):
    return abs(a) + abs(b)

res = ab_sum(-4,-6)

print(res)
```

```
def ab_sum(a,b):
    return round(a) + round(b)

res = ab_sum(-4.3,-6.6)

print(res)
```

```
def ab_sum(a,b,f):
    return f(a) + f(b)

res = ab_sum(-4.3,-6.6,abs) # 把abs作为参数传入到函数里面，传的是函数本身不是函数调用

print(res)
```

求两个数 相反数的和

```
def xfs(x):  # xfs函数的功能就是 相反数
    return -x  # -5  -5 --5  5

def ab_sum(a,b,f):   # f 是一个形参  他得值 由实参决定
    return f(a) + f(b)  # 4 + -6

res = ab_sum(-4,6,xfs) #  求得是相反数的和

print(res)
```

1.2.1 map()

map返回结果是一个迭代器，想看最终结果要转为list

```
map(func,seq)  第一个参数是给一个函数，第二个是给一个序列类型
```

将列表序列中各个元素加1

```
# list1 = [1,2,3,4,5]
# list2 = []
# for i in list1:
#     list2.append(i+1)
# print(list2)

list1 = [-1,2,-3,4,-5]
print(list(map(abs,list1)))
```

```
list1 = [1,2,3,4,5]
def add1(x):
    return x+1

print(list(map(add1,list1)))
```

```
list1 = [1,2,3,4,5]
def add1(x):
    return '左手'  # 返回 左手

print(list(map(add1,list1)))


list1 = [1,2,3,4,5]
print(list(map(lambda x:x+1,list1)))
```

总结:
1.map内置函数的作用是操作序列中所有元素，并返回一个迭代器，迭代器要转列表才能得到最终的值
2.lambda表达式可以专门配合我们的高阶函数来做简单实现

1.2.2 filter() ==》 过滤不需要的数据

```
filter(func,seq) 结果可以通过list转换
```

保留一个序列中所有偶数

```
list1 = [1,2,3,4,5,6,7,8,9,10]
for i in list1:
    if i % 2 != 0:
        list1.remove(i)
print(list1)   # %  是取余

print(list(filter(lambda x:x%2==0,list1)))
```

1.2.3 sorted()

```
# list1 = [2,4,1,3,5,6,9,7]
#
# # print(list1.sort())
# print(sorted(list1))  # 升序
# print(sorted(list1,reverse=True))  # reverse=True  反转的意思 也就是降序


list1 = ['天天:69','杰杰:78','涛涛:89','广广:100','左手:95']
# list2 = ['天天,69','杰杰,78','涛涛,89','广广,100','左手,95']
def f(x):
    arr = x.split(':')  # arr = '天天','69'
    return int(arr[1])
print(sorted(list1,key=f))  # map(函数，序列类型)  sorted(序列类型，key=函数)

list1 = ['天天:69','杰杰:78','涛涛:89','广广:100','左手:95']
```

```
print(sorted(list1,key=lambda x:int(x.split(':')[1])))
```

1.3 递归函数

如果一个函数的内部调用了自己，就叫递归

如果要定义递归函数，不想让他报错，必须要有出口

不断地向出口接近

1-500

```
def fn():
    print('你好，我是左手')
    fn()
fn()
```

```
x = 1
def func():
    global x
    if x == 500:
        print(x)
    else:
        print(x)
        x += 1
        return func()
func()
```

```
def func(x):
    if x == 501:
        return
    print(x)
    return func(x+1)  # func(1+1)
func(1)
```

作业

```
1. 对于序列 [-1,3,-5,-4] 的每个元素求绝对值
2. 用map来处理字符串列表,把列表中所有的元素后面都加上_xc,比方lulu_xc
3. 将一串数字‘1 3 5 7 8’转化为整型，并以列表形式输出
```