lambda是一个表达式，而不是语句，可以使用在def函数不能定义的地方（例如一个序列中）

lambda是处理一些简单的逻辑，def用来处理更复杂的逻辑。

例：程序function\_detail/function\_detail9.py

f = lambda x, y, z: x + y + z  
print(f(1, 2, 3)) # 6  
print()  
  
# lambda使用在序列中  
L = [lambda x: x \*\* 2, lambda x: x \*\* 3, lambda x: x \*\* 4]  
for f in L:  
 print(f(2)) # 4 8 16  
print()  
  
# 不带参数的lambda  
D = {'already': (lambda: 2 + 2), 'got': (lambda: 2 \* 4)}  
print(D['got']()) # 8  
print(D['already']()) # 4  
  
lower = (lambda x, y: x if x < y else y)  
print(lower('aa', 'bb')) # aa  
  
# 嵌套lambda，类似嵌套函数，避免使用  
action = (lambda x: (lambda y: x + y))  
act = action(99)  
print(act(3)) # 102  
print(act(5)) # 104

map：序列中映射函数（lambda）

例：程序function\_detail/function\_detail10.py

counters = [1, 2, 3, 4]  
  
  
def inc(x): return x + 10  
  
  
# map将counters每一个元素分别调用inc()  
print(list(map(inc, counters))) # [11, 12, 13, 14]  
print(list(map((lambda x: x + 10), counters))) # [11, 12, 13, 14]  
  
# map可以将N个参数的函数用于N个序列  
# 如果序列的长度不一致，则按最短序列匹配  
# 1 \*\* 2, 2 \*\* 3, 3 \*\* 4  
print(list(map(pow, [1, 2, 3], [2, 3, 4, 5]))) # [1, 8, 81]  
print(list(map(pow, [1, 2, 3, 4], [2, 3, 4]))) # [1, 8, 81]  
  
# filter过滤  
print(list(filter((lambda x: x > 0), range(-5, 5)))) # [1, 2, 3, 4]