Python递归函数与匿名函数

一. 函数的设计概念：高内聚 低耦合

1) 实现低耦合

1.力求让函数独立于它外部的其他代码

2.只有在真正必要的情况下才使用全局变量

3.避免直接改变在另一个模块文件中的变量

2) 实现高内聚

1.每一个函数都应该有一个单一的目标，在设计完美的情况下，一个函数应该只做一件事，并且这件事可以用一个简单的说明语句来总结.

2.一个函数中的代码行数应该控制在一个合理范围，如果一个函数的代码需要翻几页才能看完，此时应考虑是否可以对它进行再细分

二. Python支持递归函数

什么是递归函数?

在一个函数内部可以调用其他函数，如果一个函数在自身内部调用其自身，这种直接或间接的调用自身以循环的函数，我们称之为递归函数. 它允许程序遍历拥有任意的，不可预知的形状的结构.

In [1]:
```
#示例代码1.递归求和

def calc(x):
    if not x:
        return 0
    else:
        return x[0] + calc(x[1:])

l = [1, 2, 3, 4]
print(calc(l))
```

```
10
```

在每一层这个函数都递归的调用自己来计算列表剩余值的和, 这个和随后的加到前面的一项中, 当列表变为空的时候, 递归循环结束并返回0, 在每个层级上 要加和的列表变得越来越小, 直到它变为空, 变为空时循环结束.

In [2]:
```
#示例代码2.处理任意结构，计算一个嵌套的子列表结构中所有数字的总和

x = [1, [2, 3, 4], [5], [1, 2, 3, 4]]

def sumtree(x):
    res = 0
    for j in x:
        if isinstance(j, int):
            res += j
        if isinstance(j, list):
            res += sumtree(j)
    return res
print(sumtree(x))
```

```
25
```

二.Python中函数灵活性的体现

In [4]:
```
#1.可以把函数对象赋值给其他名称并且引用
def echo(message):
    print(message)
x = echo
x('hello')
```

```
hello
```

In [5]:
```
#2.可以把函数作为参数传递给另一个函数
def f1(func, args):
    func(args)
f1(echo, 'hello world')
```

```
hello world
```

In [6]:
```
#3.可以把函数和参数放入元组中进行循环
x = [(echo, 'hello'), (echo, 'world')]
for (func, args) in x:
    func(args)
```

```
hello
world
```

三.匿名函数

lambda函数是一种快速定义单行的最小函数

为什么使用匿名函数?

1) lambda可以省去函数的定义过程让代码更精简

2) 对于一些抽象的不会在别的地方重复使用的函数, 有时候给函数起个名字都是个难题, 此时使用lambda不需要考虑的命名的问题

3) 使用lambda在某些时候可以让代码更容易理解

4) lambda是一个表达式而不是一个语句

In [7]:
```
#示例代码1.

def f(x, y):
    return x * y
print(f(2, 3))

g = lambda x, y: x*y
print(g(3, 4))

#在lambda中冒号前是参数，可以有多个用逗号隔开，冒号右边是返回值，构造的是一个函数的对象
```

```
6
12
```

In [8]:
```
#示例代码2. 匿名函数结合函数式编程可以让代码更加简洁

#传统写法
l = [1, 2, 3, 4, 5]
res = []
for j in l:
    res.append(j + 10)
else:
    print(res)
```

```
[11, 12, 13, 14, 15]
```

In [10]:
```
#列表解析写法
res = [j + 10 for j in l]
print(res)
```

```
[11, 12, 13, 14, 15]
```

In [12]:
```
#函数式编程结合匿名函数写法
print(list((map((lambda x: x + 10), l))))
```

```
[11, 12, 13, 14, 15]
```