Lab03/Hw03 Review

2022 SICP

如何写出递归函数？

1. 我的函数输入是什么？输出是什么？它能完成什么事情？
2. 相信你的函数能完成相应的工作！
3. 基本情况是什么？最后一步需要干什么？
4. 或许可以先写一个循环版本

Lab03p3 Climb Stairs

1. `count_stairs_ways` 输入是什么？输出是什么？

   - 输入：台阶数量n
   - 输出：每次走1步或2步时，总计的上楼梯方案

Lab03p3 Climb Stairs

2．最后一步怎么走？

- 走1步，即从第$n-1$级台阶走到第n级台阶，走到第$n-1$级台阶的方案数为`count_stairs_ways(n-1)`
- 走2步，即从第$n-2$级台阶走到第n级台阶，走到第$n-2$级台阶的方案数为`count_stairs_ways(n-2)`
- 总方案数`count_stairs_ways(n-1)+count_stairs_ways(n-2)`

Lab03p3 Climb Stairs

3．基本情况是？

- 到达第1级台阶有1种方案
- 到达第2级台阶有2种方案

```
if n == 1:
    return 1
elif n == 2:
    return 2
```

Lab03p3 Climb Stairs

```
def count_stair_ways(n):
    if n == 1:
        return 1
    elif n == 2:
        return 2
    return count_stair_ways(n-1) + count_stair_ways(n-2)
```

Lab03p3 Climb Stairs

```
def count_stair_ways(n):
    if n == 1:
        return 1
    elif n == 2:
        return 2
    return count_stair_ways(n-1) + count_stair_ways(n-2)
```

最多走k步怎么办？

Lab03p3 Climb Stairs

2．最后一步怎么走？

- 走$k(k \leq n)$步，即从第$n-k$级台阶走到第n级台阶，走到第$n-k$级台阶的方案数为`count_k(n-k,k)`

- 总方案数为$\sum_{i=1}^{\min(k,n)} count_k(n-i,k)$

Lab03p3 Climb Stairs

```
def count_k(n, k):
    if n == 0:
        return 1
    else:
        total, i = 0, 1
        while i <= min(n, k):
            total += count_k(n-i, k)
            i += 1
        return total
```

$$\sum_{i=1}^{\min(k,n)} count\_k(n-i,k)$$

Lab03p3 Climb Stairs

- 可以利用summation

$$\sum_{i=1}^{\min(k,n)} count\_k(n-i,k)$$

```
def count_k(n, k):
    if n == 0:
        return 1
    return summation(min(n, k), lambda i: count_k(n-i, k))
```

Lab03p5 Maximum Subsequence

1. max_subseq输入是什么？输出是什么？

- 输入：数字n与长度l
- 输出：找出数字n中长度不超过l的子序列，返回最大的子序列

Lab03p5 Maximum Subsequence

2．最后一位数字选什么？

- 包含末尾数字，最大子序列为

$$max\_subseq(n//10, l-1) * 10 + n \% 10$$

- 不包含末尾数字，最大子序列为

$$max\_subseq(n//10, l)$$

- 取两种情况的最大值

```
return max(max_subseq(n//10, l-1) * 10 + n % 10, max_subseq(n//10, l))
```

Lab03p5 Maximum Subsequence

3. 基本情况是？

- 长度$l == 0$，最大子序列为0
- 数字$n == 0$，最大子序列为0

```
if l == 0 or n == 0:
    return 0
```

Lab03p5 Maximum Subsequence

```
def max_subseq(n, l):
    if l == 0 or n == 0:
        return 0
    return max(max_subseq(n//10, l),
max_subseq(n//10, l-1) * 10 + n % 10)
```

HW03p2 Ping-pong

1. pingpong输入是什么？输出是什么？

- 输入：序数n
- 输出：pingpong序列的第n项

HW03p2 Ping-pong

- 一种错误的做法：

```
def pingpong(n):
    def addOrSub(i):
        if i == 1:
            return 1
        elif i % 6 == 0 or number_of_six(i) != 0:
            return -addOrSub(i-1)
        else:
            return addOrSub(i-1)
    if n == 1:
        return 1
    else:
        return pingpong(n-1) + addOrSub(n-1)
```

TLE，你的程序跑得太慢啦

每次递归都要重新确定方向

HW03p2 Ping-pong

```
def pingpong(n):
    def addOrSub(i):
        if i < 6:
            return 1
        elif i % 6 == 0 or number_of_six(i) != 0:
            return -addOrSub(i-1)
        else:
            return addOrSub(i-1)
    def countPingpong(n, direction):
        if n <= 6:
            return n
        elif n % 6 == 0 or number_of_six(n) != 0:
            return countPingpong(n-1, -direction) - direction
        else:
            return countPingpong(n-1, direction) + direction
    return countPingpong(n, addOrSub(n))
```

带着方向进行递归

HW03p2 Ping-pong

- 换个思路，能不能从1开始数？

- 先写一个循环版本：

```
def pingpong(n):
    curr, result, direct = 1, 1, 1
    while curr != n:
        if curr % 6 == 0 or number_of_six(curr) > 0:
            result = result - direct
            direct = -direct
        else:
            result = result + direct
            # direct = direct
        curr = curr + 1
    return result
```

HW03p2 Ping-pong

- 改写成递归
  - 没有赋值语句怎么办？
  - 利用辅助函数的参数来记录

```
def pingpong(n):
    def helper(curr, result, direct):
        if curr == n:
            return result
        if curr % 6 == 0 or number_of_six(curr) != 0:
            return helper(curr + 1, result - direct, - direct)
        return helper(curr + 1, result + direct, direct)
    return helper(1, 1, 1)
```

HW03p4 Count Change

1. count_change输入是什么？输出是什么？

- 输入：需要找零的金额$total$，货币系统$next_money$
  - $next_money$输入是什么？输出是什么？
  - 输入一个货币面额（最小为1），返回下一个比该面额大的货币面额，如果没有则返回$None$
- 输出：找零方案数

HW03p4 Count Change

2．当前用什么面额的货币？

- 再用一张当前面额，找零方案数为$change(total - minMoney, minMoney)$
- 用下一张面额，找零方案数为$change(total, next_money(minMoney))$
- 怎么记录当前面额？
  - 辅助函数！

HW03p4 Count Change

3．基本情况是？

- $total < minMoney$，得到了不合法的找零方案，返回0
- $total == minMoney$，找零方案合法，返回1
- $minMoney\ is\ None$，没有更大的货币面额了，方案不合法，返回0

HW03p6 Multiadder

1. multiadder输入是什么？输出是什么？

- 输入：加法参数个数n
- 输出：可以调用n次的求和函数，即

```
lambda x1: lambda x2: … lambda xn: x1 + x2 + … + xn
```

HW03p6 Multiadder

3．基本情况是什么？

- $n == 1$时，返回$lambda\ x\colon\ x$

HW03p6 Multiadder

```
lambda x1: lambda x2: … lambda xn: x1 + x2 + … + xn
```

HW03p6 Multiadder

```
lambda x1: lambda x2: … lambda xn: x1 + x2 + … + xn
```

HW03p6 Multiadder

```
lambda x1: lambda x2: … lambda xn: x1 + x2 + … + xn
```

```
lambda x12: lambda x3: … lambda xn: x12 + x3 + … + xn
```

$$multiadder(n-1)$$

$$multiadder(n-1)(x1+x2)$$

```
def multiadder(n):
    if n == 1:
        return lambda x: x
    return lambda x: lambda y: multiadder(n - 1)(x + y)
```

HW03p6 Multiadder

- 换个思路：

- 对于$multiadder(5)(1)(2)$
  - 目前已经累加了2个数，还差3个数没有输入
  - 当前的和curSum为3，但multiadder参数不够，无法记录
  - 辅助函数！

HW03p6 Multiadder

```
def multiadder(n):
    def helper(n, curSum):
        def inner(x):
            if n == 1:
                return curSum + x
            return helper(n - 1, curSum + x)
        return inner
    return helper(n, 0)
```