7.4函数递归实例解析（斐波那契数列、汉诺塔）

字符串反转

将字符串s反转后输出　　`>>> s[::-1]`

- **函数 + 分支结构**
- **递归链条**
- **递归基例**

```
def rvs(s):
    if s == "" :
        return s
    else :
        return rvs(s[1:])+s[0]
```

斐波那契数列

一个经典数列

$$F(n) = \begin{cases} 1 & n = 1 \\ 1 & n = 2 \\ F(n-1) + F(n-2) & otherwise \end{cases}$$

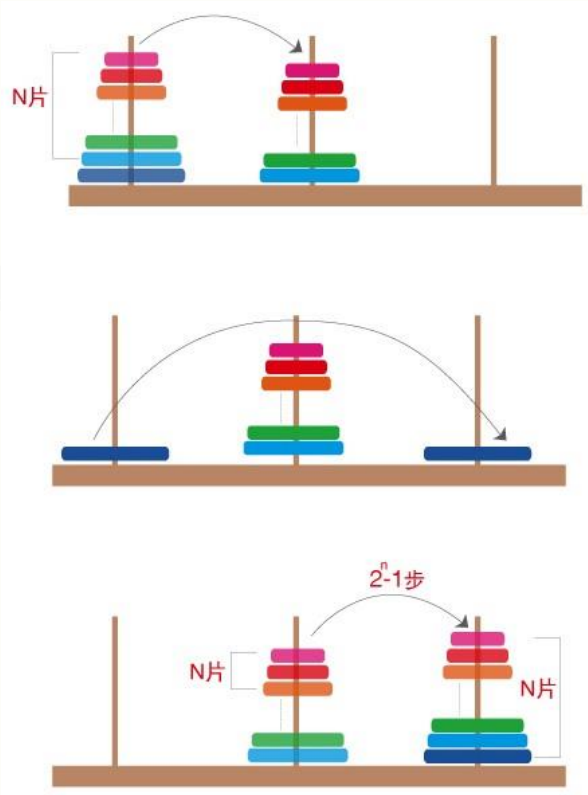
斐波那契数列

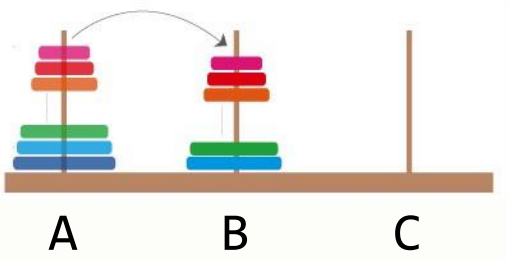
$$F(n) = F(n-1) + F(n-2)$$

- 函数 + 分支结构
- 递归链条
- 递归基例

```
def f(n):
    if n == 1 or n == 2 :
        return 1
    else :
        return f(n-1) + f(n-2)
```

汉诺塔

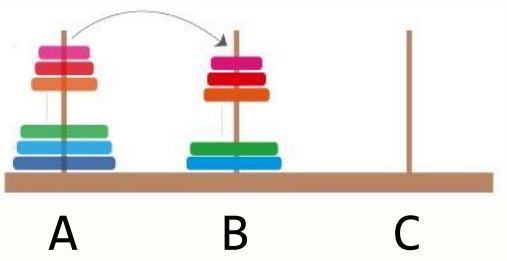




汉诺塔

- 函数 + 分支结构
- 递归链条
- 递归基例

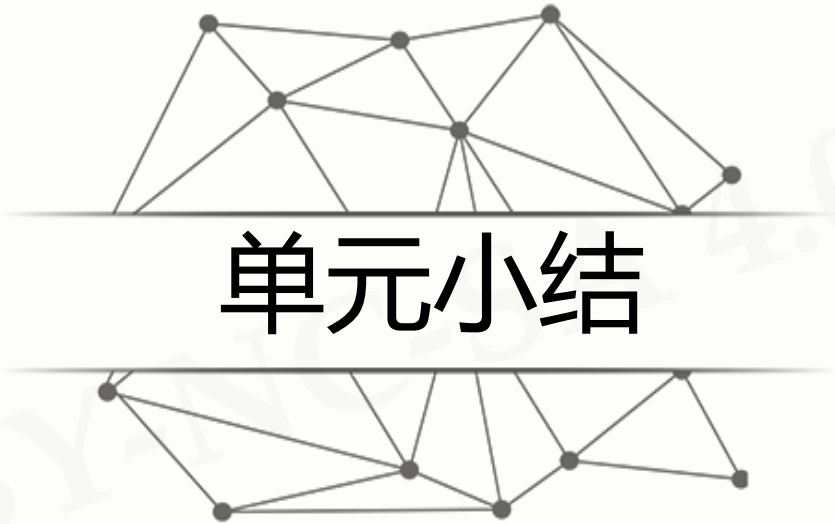
```
count = 0
def hanoi(n, src, dst, mid):
    global count
    if n == 1 :
        print("{}:{}->{}".format(1,src,dst))
        count += 1
    else :
        hanoi(n-1, src, mid, dst)
        print("{}:{}->{}".format(n,src,dst))
        count += 1
        hanoi(n-1, mid, dst, src)
```



汉诺塔

```
count = 0
def hanoi(n, src, dst, mid):
    … (略)
hanoi(3, "A", "C", "B")
print(count)
```

```
>>>
1:A->C
2:A->B
1:C->B
3:A->C
1:B->A
2:B->C
1:A->C
7
```



单元小结

代码复用与函数递归

- **模块化设计：松耦合、紧耦合**
- **函数递归的2个特征：基例和链条**
- **函数递归的实现：函数 + 分支结构**