Project Euler 31: Coin Sums

https://www.metaquant.org

英格兰的货币单位由英磅和便士构成，并且日常流通的共有八种硬币：

$$1p, 2p, 5p, 10p, 20p, 50p, £1(100p), £2(200p)$$

两英磅可以由以下方式构成：

$$1 \times £1 + 1 \times 50p + 2 \times 20p + 1 \times 5p + 1 \times 2p + 3 \times 1p$$

求使用以上八种硬币，组成两英磅的方式共有多少种？

分析：这是一首典型的动态规划入门问题，可以用动态规划的方式来解决。首先我们要把所求问题分解成一个规模更小的问题。我们从一便士开始，我们只能用一便士来组成一便士，所以组成一便士的方法只有一种。再来看二便士，我们可以用两个一便士或者一个二便士来组成二便士，所以组成二便士的方法有两种。对于三便士，可以选用的只能是一便士和二便士两种硬币，我们可以用三个一便士或者一个一便士一个两便士两种方法组成三便士。对于以上的观察，我们可以画一个表格：

	only 1p	<=2p	<=5p	<=10p	<=20p	<=50p	<=100p	<=200p
1p	1	1	1	1	1	1	1	1
2p	1	2	2	2	2	2	2	2
3p	1	2	2	2	2	2	2	2
4p	1	3	3	3	3	3	3	3
5p	1	3	4	4	4	4	4	4
6p	1	4	5	5	5	5	5	5
7p	1	4	6	6	6	6	6	6
8p	1	5	7	7	7	7	7	7
9p	1	5	8	8	8	8	8	8
10p	1	6	10	11	11	11	11	11

图 1: 使用表格解决动态规划问题

表格中行表示我们想要的构成的货币量，列则为我们能够使用的硬币，表格中的数字即为方法数，比如第三行第二列表示只能使用一便士和二便士的情况下，要组成三便士共有两种方法。通过观察这个表格我们可以发现，第一行和第一列都是一，第一行都是一是因为构成一便士只有一种方法，第一列是一是因为如果只允许使用一便士，那么任何货币量都只能用一种方法构成。这个表格也是我们在分析动态规划问题中经常会使用的表格，用行表示我们的目标，用列表示不断缩小的问题规模，而行和列对应的数字即为我们想要优化的数值，通过观察表格中数值之间的相互关系，我们就可以分析出动态规划问题中的问题分解方式或者说状态转移方程。

比如当我们要求构成四便士共有多少种方式，我们可以用四便士减去两便士等于两便士，然后我们查看表格中构成两便士的方法有几种，答案是两种，分别是两个一便士和一个两便士，在它们基础上各加上一个两便士，就可以得到构成四便士的两种方法，两个一便士加上一个两便士，和两个二便士。此外我们知道总可以用四个一便士来组成四便士，所以很容易得到组成四便士的方法有三种。如果只允许使用一便士和二便士，我们可以用类似的方法推出组成五便士的方法有三种，如果允许使用五便士，则组成五便士的方法有三种，因为可以用一个五便士。至此我们可以总结如下：设表格中的第 i 行第 j 列对应的方法数量为 $w(i,j)$，那么我们有：$w(0,j)=1, w(i,0)=1, coins=[1,2,5,10,20,50,100,200]$，则：

- 如果 $i<coins[j]$，则 $w(i,j)=w(i,j-1)$；
- 如果 $i=coins[j]$，则 $w(i,j)=w(i,j-1)+1$；
- 如果 $i>coins[j], w(i,j)=w(i,j-1)+w(i-coins[j],j)$。

上述关系可以进一步简化为：

$$w(i,j)=\begin{cases} w(i,j-1)+w(i-coins[j],j) & i\geq coins[j] \\ w(i,j-1) & i<coins[j] \end{cases}$$

根据以上的递推关系式，我们可以使用 numpy 数组构建结果矩阵，逐项递推出最终结果，则有 $w(200,200)$ 即为所求。

```
def main(target=200):
    import numpy as np
    coins = [1, 2, 5, 10, 20, 50, 100, 200]
    x = np.ones((target,len(coins)))
    for i in range(1,target):
        for j in range(1,len(coins)):
            left = x[i][j-1]
            up = x[i-coins[j]][j] if (i+1-coins[j])>=0 else 0
            x[i][j] = left + up
    return int(x[-1,-1])
```