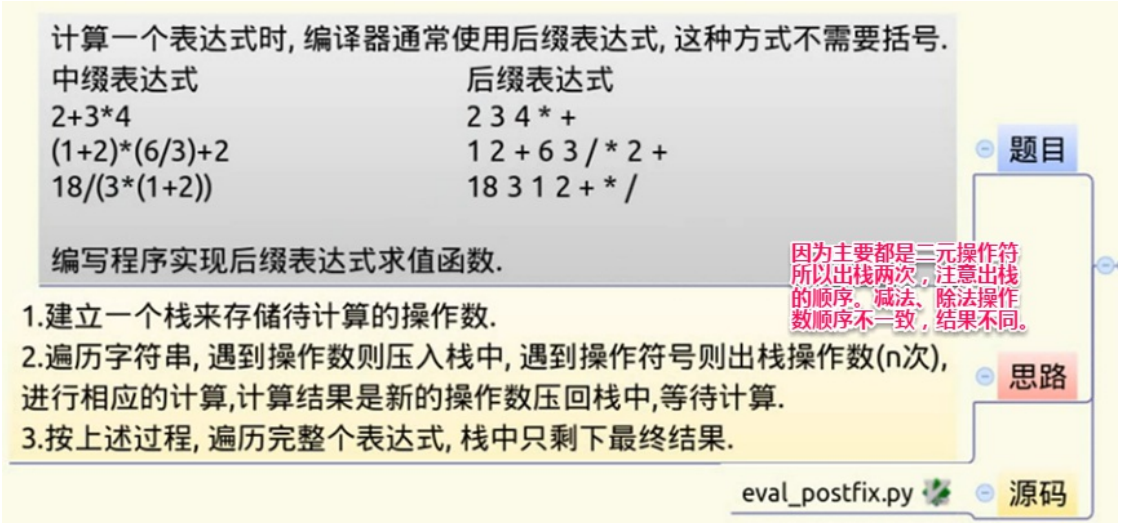
栈-2

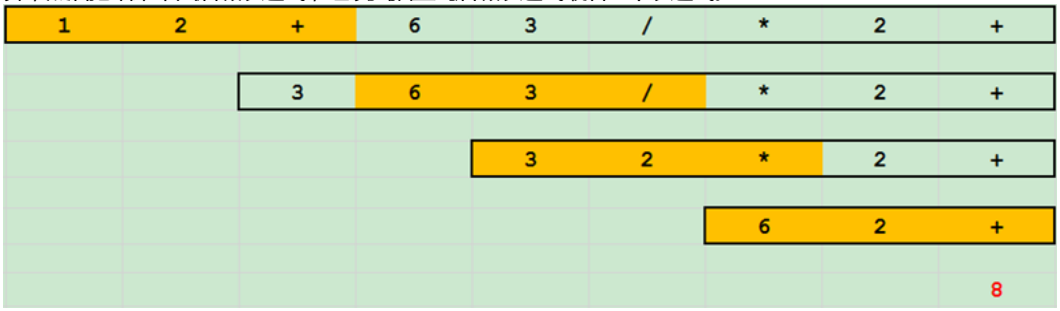
calcSuffix.py

knapSack.py

- **案例**
  - **案例-3：后缀表达式求值**
    - **概述**



    - **思路**
      1. **为什么使用栈？**因为**遇到运算符就将运算符前面两个操作数取出来**，符合栈先进后出的原则。
      2. **后缀表达式如何计算？**从左到右扫描后缀表达式，遇到运算符就把表达式中该运算符前面两个操作数取出并运算，然后把结果带回后缀表达式；继续扫描直到后缀表达式最后一个表达式。



      3. 字典的迭代？默认情况下， dict 迭代的是 key 。如果要迭代 value，可以用 for value in d.itervalues()，如果要同时迭代 key 和 value，可以用 for k, v in d.iteritems() 。
    - **代码**

```python
def calcSuffix(str):

    # 定义操作符
    operator = {
        '+':lambda op1, op2: op1+op2,
        '-':lambda op1, op2: op1-op2,
        '*':lambda op1, op2: op1*op2,
        '/':lambda op1, op2: op1/op2
    }

    # 栈用来存储操作数
    stack = []

    # 后缀表达式是用字符串的形式给出的 , 操作数、操作符通过空格分隔
    # 通过 split() 方法将数字、操作符分隔成一列表方便后续迭代操作
    parts = str.split()

    for part in parts:
        # 如果是数字（字符形式） , 压栈
        # int() 函数 , 强制转换成数字
        if part.isdigit():
            stack.append(int(part))
        # 如果是操作符 , 弹出栈顶两个元素（注意顺序） , 用于计算
        # 计算结果继续压入栈中
        elif part in operator:
            operation = operator[part]
            op2 = stack.pop()
            op1 = stack.pop()
            stack.append(operation(op1, op2))

    # 最后栈中只有一个元素 , 就是后缀表达式计算结果
    return stack[-1]
```

- **案例-4：背包问题**
  - **概述**

  - **思路**
    1. **回溯法是什么？**回溯法（探索与回溯法）是一种选优搜索法，又称为试探法，按选优条件向前搜索，以达到目标。但当探索到某一步时，发现原先选择并不优或达不到目标，就退回一步重新选择，这种走不通就退回再走的技术为回溯法，而满足回溯条件的某个状态的点称为 " 回溯点 " 。
    2. **解决思路？不明白，但是栈经常和回溯法结合到一起。此时空栈代表什么？**
  - **代码**

- **And So On**