语法：
如果语句很长，使用反斜杠 **** 来实现多行语句

- 按字面意义级联字符串，如 **"this " "is " "string"** 会被自动转换为 **this is string**。
- 字符串可以用 **+** 运算符连接在一起，用 ***** 运算符重复。

Python 中的字符串不能改变。翻转：str[::-1]或 reversed（str）

- 字符串切片 **str[start:end:step]**，其中 start（包含）是切片开始的索引，end（不包含）是切片结束的索引。步长参数 step

输入输出：import sys？
sys.stdin.write/ sys.stdout.write(x + '\n')
末尾不输出多余空格：elements = ["apple", "cherry"] output = ' '.join(elements) print(output)
或 print(*elements)
保留小数点后几位：number = 123.456789
formatted_number = f"{number:.2f}" # 保留两位小数 f"{要控制的数:.位数 f}"
print(formatted_number) # 输出: 123.46
.format() 方法支持多种类型的参数传递方式：

- **位置参数**：按照顺序填入 {} 占位符。
- **关键字参数**：通过名称指定填入 {name} 占位符。
- **组合使用**：可以混合使用位置和关键字参数。

导入：将整个模块(somemodule)导入，格式为： **import somemodule**
从某个模块中导入某个函数,格式为： **from somemodule import somefunction**
赋值：可以为多个对象指定多个变量。例如：a, b, c = 1, 2, "runoob"
数值的除法包含两个运算符：/ 返回一个浮点数，// 返回一个整数（向小取整**/但**得到的并不一定是整数类型的数，它与分母分子的数据类型有关系，>>> 7.0//2 >>3.0）。
>>> True==1>>True >>> 1 is True>>False
布尔值在控制流中的应用 if+条件（若等价于或为 True 则执行下面语句）
所有非零的数字和非空的字符串、列表、元组等数据类型都被视为 True，只有 **0、空字符串、空列表、空元组**等被视为 False
is 与 == 区别：is 用于判断两个变量引用对象是否为同一个， ==引用变量的值是否相等。
列表截取可以接收第三个参数，参数作用是截取的步长，以下实例在索引 1 到索引 4 的位置并设置为步长为 2 lst[1:4:2]>>下标为 1 和 3 的元素
删除：del obj， lst.pop()删除末尾，.remove(obj)移除某个值第一个匹配项
列表函数 or 方法 浅拷贝深拷贝
list.sort(key= , reverse=False) list.insert(index, obj) 将对象插入列表 min(list) 返回最小
list.index(obj) 从列表中找出某个值第一个匹配项的索引位置 enumerate?
list.pop([index=-1]) 移除列表中的一个元素（默认最后一个元素），并且返回该元素的值
列表推导式[表达式 for 变量 in 列表 if 条件]
names = ['Bob','Tom','alice','Jerry','Wendy','Smith']
new_names = [name.upper()**for** name **in** names **if** len(name)>3]
>>['ALICE', 'JERRY', 'WENDY', 'SMITH']
vec = [2, 4, 6]
[[x, x**2] for x in vec]
>>[[2, 4], [4, 16], [6, 36]]（用于生成矩阵？）
元组 tuple 的元素不可改变，但它可以包含可变的对象，比如 list 列表。
tup1 = () # 空元组

tup2 = (20,) # 一个元素，需要在元素后添加逗号。不添加逗将被解释为一个普通的值
集合无序、可变，用于存储唯一的元素。元素不会重复，可以进行交集、并集、差集等
(a-b) 差　　(a|b) **并**　　(a&b) 交
(a ^ b)　# a 和 b 中不同时存在的元素
字典　无序的 **键(key) : 值(value)** 的集合，键必须是不可变类型且唯一的
dict[key]>>value。创建时如果同一个键被赋值两次，后一个值会被记住。
dict.update(dict2) dict2 的 **key/value(键/值)** 对更新到字典 dict 里。
dict.get(key[, value]) value 可选，如果指定键的值不存在时，返回该默认值。如果不指定默认值，则返回 None。
dict.setdefault(key, default=None) 如果键不存在，添加键并将值设为默认值。
字典推导式：{x: x**2 for x in (2, 4, 6)}>>{2: 4, 4: 16, 6: 36}
map： n, m = map(int, input().split())
list(map(function or type, iterable-可迭代对象))　>>一个列表
数据类型转换：type() 是否可行主要取决于数据本身是否包含足够的信息来表示目标类型
算术运算：*可以用于重复某个字符串或元素　**幂运算　%取模 返回余数
数学函数 math　abs(x)>>绝对值　ceil(x)>>上入 math.ceil(4.2)>>5　floor(x)下舍　sqrt 平方根
Math 里面可以求对数 math.loga(b)
海象运算符（:=）在表达式内部同时进行赋值操作并返回赋值的值　if (n := len(a)) > 10:
循环：for 和 while
while 和 for 可以搭配 else　else 控制的是循环结束后的代码

```
count = 0
while count < 5:
  print (count, " 小于 5")    >>0>>1>>2>>3>>4
  count = count + 1
else:
  print (count, " 大于或等于 5")  >>5
```

range(起点，开终点，步长)函数：生成数列、遍历索引等
continue 跳过当前循环块中的剩余语句，进行下一轮循环。**break 跳出当前循环体**
函数　必需参数须以正确的顺序传入函数。调用时的数量必须和声明时的一样，否则报错
调用函数时，如果没有传递参数，则会使用默认参数 printinfo(name, age = 35)
lambda 函数的语法只包含一个语句，如下：lambda [arg1 [,arg2,.....argn]]: 表达式
return 用于退出函数，选择性地向调用方返回一个表达式。不带参数值的 return 返回 None
数据结构
Stack 栈　LIFO 后进先出　列表提供了一些方法适合用于栈操作，特别是 append() 和 pop()

```
stack = []
stack.append(1)
stack.append(2)
stack.append(3)
print(stack)  # 输出: [1, 2, 3]
top_element = stack.pop()
print(top_element)  # 输出: 3
print(stack)      # 输出: [1, 2]
```

Queue 队列 FIFO 先进先出　直接用列表不方便实现（O(n)）
collections.deque 适合用于实现队列　（用它自己的函数）（O(1)）

```
from collections import deque
queue = deque()  # 创建一个空队列
```

```
queue.append('a')    # 向队尾添加元素
queue.append('b')
queue.append('c')
print("队列状态:", queue)  # 输出: 队列状态: deque(['a', 'b', 'c'])

first_element = queue.popleft()    # 从队首移除元素
print("移除的元素:", first_element)  # 输出: 移除的元素: a
print("队列状态:", queue)            # 输出: 队列状态: deque(['b', 'c'])
```

嵌套列表解析 矩阵 matrix row column

```
>>> matrix = [
...     [1, 2, 3, 4],
...     [5, 6, 7, 8],
...     [9, 10, 11, 12],
... ]
```

将 3X4 的矩阵列表转换为 4X3 列表： [[row[i] for row in matrix] for i in range(4)]

```
>>[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

或

```
>>> transposed = []
   >>> for i in range(4):
            transposed.append([row[i] for row in matrix])
```

遍历技巧

字典，关键字和对应的值可以使用 items() 方法同时解读出来

```
knights = {'gallahad': 'the pure', 'robin': 'the brave'}
for k, v in knights.items():
    print(k, v)
>>gallahad the pure
    robin the brave
```

序列，索引位置和对应值可以使用 enumerate() 函数同时得到：

```
for i, v in enumerate(['tic', 'tac', 'toe']):
   print(i, v)  >>0 tic   1 tac   2 toe
```

错误和异常处理

```
while True:
   try:
      x = int(input("请输入一个数字: "))
       break
   except ValueError:
      print("您输入的不是数字，请再次尝试输入！")
```

- 首先，执行 try 子句
- 如果没有异常发生，忽略 except 子句，try 子句执行后结束。
- 如果在执行 try 子句的过程中发生了异常， try 子句余下的部分将被忽略。如果异常的类型和 except 之后的名称相符，那么对应的 except 子句将被执行。
- 如果一个异常没有与任何的 except 匹配，那么这个异常将会传递给上层的 try 中。

一个 except 子句可以同时处理多个异常，放在一个括号里成为一个元组

可选的 **else** 子句必须放在所有 except 子句之后，在 try 子句没有发生任何异常的时候执行。

冒泡排序

```
def bubbleSort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1] :
                arr[j], arr[j+1] = arr[j+1], arr[j]
```

筛选素数 对于每个素数 i，将所有 i 的倍数（从 2*i 开始）标记为 False，即它们不是素数。

```
n = int(input())
isPrime = [True] * (n + 1)
for i in range(2, n + 1):
  if isPrime[i]:
    for j in range(2 * i, n + 1, i):
      isPrime[j] = False
```

隐式 dp：机智的股民老张-最大利润

```
a = list(map(int,input().split()))
income = 0
stack = []
min_value = float("inf")
for i in range (len(a)):
   min_value = min(a[i],min_value)
   income = max(income,a[i]-min_value)
print(income)
```

旋转矩阵 洋葱

```
def dfs(n, s, x, y):
  if n == 1:        # 递归的终止条件
    return s[x][y] # 如果只剩一个元素，直接返回该元素值
  if n == 0:
    return 0
  curr = 0    # 记录当前边缘的累加和
  directions = [(0, 1), (1, 0), (0, -1), (-1, 0)]
  for i in range(4 * (n - 1)):        # 遍历边缘的元素，总共需要遍历 4 * (n - 1) 个元素
    dx, dy = directions[(i // (n - 1)) % 4]  # 根据当前索引计算方向
    x += dx
    y += dy
    curr += s[x][y]
  return max(curr, dfs(n - 2, s, x + 1, y + 1)) # 下一层，起始坐标向内移动一格，缩小一圈

n = int(input())
s = [list(map(int, input().split())) for _ in range(n)]
result = dfs(n, s, 0, 0)   # 初始调用 dfs
print(result)
```