The following code is the solution code for 2D-ARRAY under DS problem.

QUES:

1 1 1 0 0 0

0 1 0 0 0 0

1 1 1 0 0 0

0 0 0 0 0 0

0 0 0 0 0 0

0 0 0 0 0 0

We define an hourglass in  to be a subset of values with indices falling in this pattern in 's graphical representation:

a b c

d

e f g

There are  hourglasses in , and an *hourglass sum* is the sum of an hourglass' values. Calculate the hourglass sum for every hourglass in , then print the *maximum* hourglass sum.

For example, given the 2D array:

-9 -9 -9 1 1 1

0 -9 0 4 3 2

-9 -9 -9 1 2 3

0 0 8 6 6 0

0 0 0 -2 0 0

0 0 1 2 4 0

We calculate the following  hourglass values:

-63, -34, -9, 12,

-10, 0, 28, 23,

-27, -11, -2, 10,

9, 17, 25, 18

Our highest hourglass value is  from the hourglass:

0 4 3

1

8 6 6

**Note:** If you have already solved the Java domain's *Java 2D Array* challenge, you may wish to skip this challenge.

**Function Description**

Complete the function *hourglassSum* in the editor below. It should return an integer, the maximum hourglass sum in the array.

hourglassSum has the following parameter(s):

* *arr*: an array of integers

**Input Format**

Each of the  lines of inputs  contains  space-separated integers .

**Output Format**

Print the largest (maximum) hourglass sum found in .

**Sample Input**

1 1 1 0 0 0

0 1 0 0 0 0

1 1 1 0 0 0

0 0 2 4 4 0

0 0 0 2 0 0

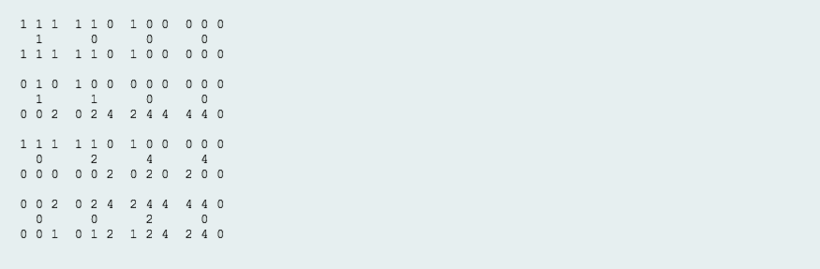
0 0 1 2 4 0

**Sample Output**

19

**Explanation**

 contains the following hourglasses:



The hourglass with the maximum sum () is:

2 4 4

2

1 2 4

CODE:

def hourglassSum(arr):

total =[]

for i in range(len(arr)-2):

for j in range(len(arr)-2):

total.append(sum(arr[i][j:j+3])+arr[i+1][j+1]+sum(arr[i+2][j:j+3]))

return max(total)

#top =sum(arr[i][j:j+3])

#mid=arr[i+1][j+1]

#bottom=sum(arr[i+2][j:j+3])

#hourglass=top+mid+bottom

#if hourglass>total:

# total=hourglass

#return total'''