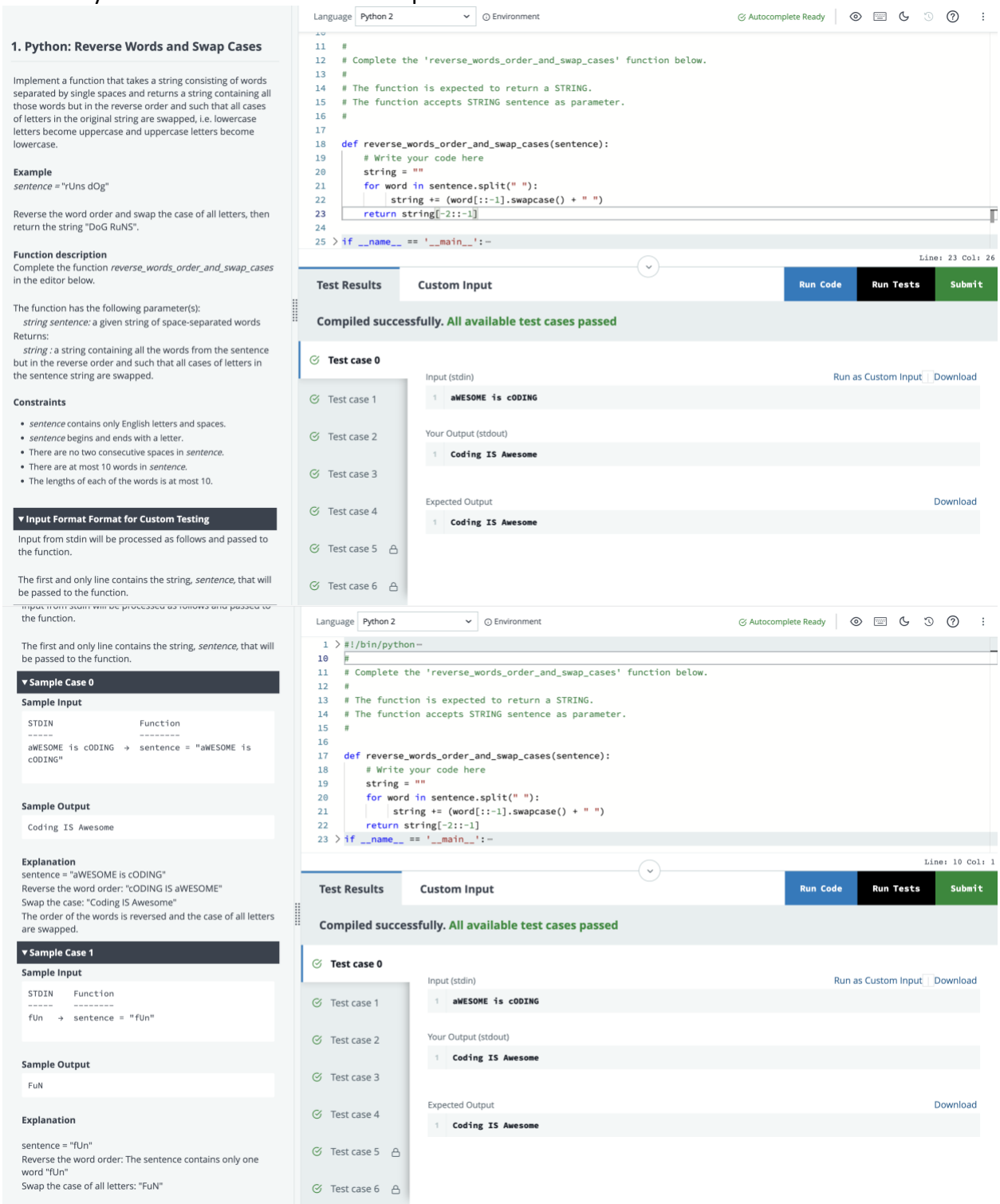
Python Basic

1. Python: Reverse Words and Swap Cases



1. Python: Reverse Words and Swap Cases

Implement a function that takes a string consisting of words separated by single spaces and returns a string containing all those words but in the reverse order and such that all cases of letters in the original string are swapped, i.e. lowercase letters become uppercase and uppercase letters become lowercase.

Example
sentence = "rUns dOg"

Reverse the word order and swap the case of all letters, then return the string "DoG RuNS".

Function description
Complete the function *reverse_words_order_and_swap_cases* in the editor below.

The function has the following parameter(s):
string sentence: a given string of space-separated words
Returns:
string : a string containing all the words from the sentence but in the reverse order and such that all cases of letters in the sentence string are swapped.

Constraints

- *sentence* contains only English letters and spaces.
- *sentence* begins and ends with a letter.
- There are no two consecutive spaces in *sentence*.
- There are at most 10 words in *sentence*.
- The lengths of each of the words is at most 10.

▼ Input Format Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first and only line contains the string, *sentence*, that will be passed to the function.

▼ Sample Case 0

Sample Input

```
STDIN                Function
-----                --------
aWESOME is cODING  →  sentence = "aWESOME is cODING"
```

Sample Output

```
Coding IS Awesome
```

Explanation
sentence = "aWESOME is cODING"
Reverse the word order: "cODING IS aWESOME"
Swap the case: "Coding IS Awesome"
The order of the words is reversed and the case of all letters are swapped.

▼ Sample Case 1

Sample Input

```
STDIN    Function
-----    --------
fUn   →  sentence = "fUn"
```

Sample Output

```
FuN
```

Explanation

sentence = "fUn"
Reverse the word order: The sentence contains only one word "fUn"
Swap the case of all letters: "FuN"

Language: Python 2 | Environment | Autocomplete Ready

```
#
# Complete the 'reverse_words_order_and_swap_cases' function below.
#
# The function is expected to return a STRING.
# The function accepts STRING sentence as parameter.
#

def reverse_words_order_and_swap_cases(sentence):
    # Write your code here
    string = ""
    for word in sentence.split(" "):
        string += (word[::-1].swapcase() + " ")
    return string[-2::-1]

if __name__ == '__main__':
```

Line: 23 Col: 26

Test Results | Custom Input | Run Code | Run Tests | Submit

Compiled successfully. All available test cases passed

Test case 0

Input (stdin) — Run as Custom Input | Download

```
aWESOME is cODING
```

Your Output (stdout)

```
Coding IS Awesome
```

Expected Output — Download

```
Coding IS Awesome
```

Test case 1 | Test case 2 | Test case 3 | Test case 4 | Test case 5 | Test case 6

Language: Python 2 | Environment | Autocomplete Ready

```
#!/bin/python
#
# Complete the 'reverse_words_order_and_swap_cases' function below.
#
# The function is expected to return a STRING.
# The function accepts STRING sentence as parameter.
#

def reverse_words_order_and_swap_cases(sentence):
    # Write your code here
    string = ""
    for word in sentence.split(" "):
        string += (word[::-1].swapcase() + " ")
    return string[-2::-1]
if __name__ == '__main__':
```

Line: 10 Col: 1

Test Results | Custom Input | Run Code | Run Tests | Submit

Compiled successfully. All available test cases passed

Test case 0

Input (stdin) — Run as Custom Input | Download

```
aWESOME is cODING
```

Your Output (stdout)

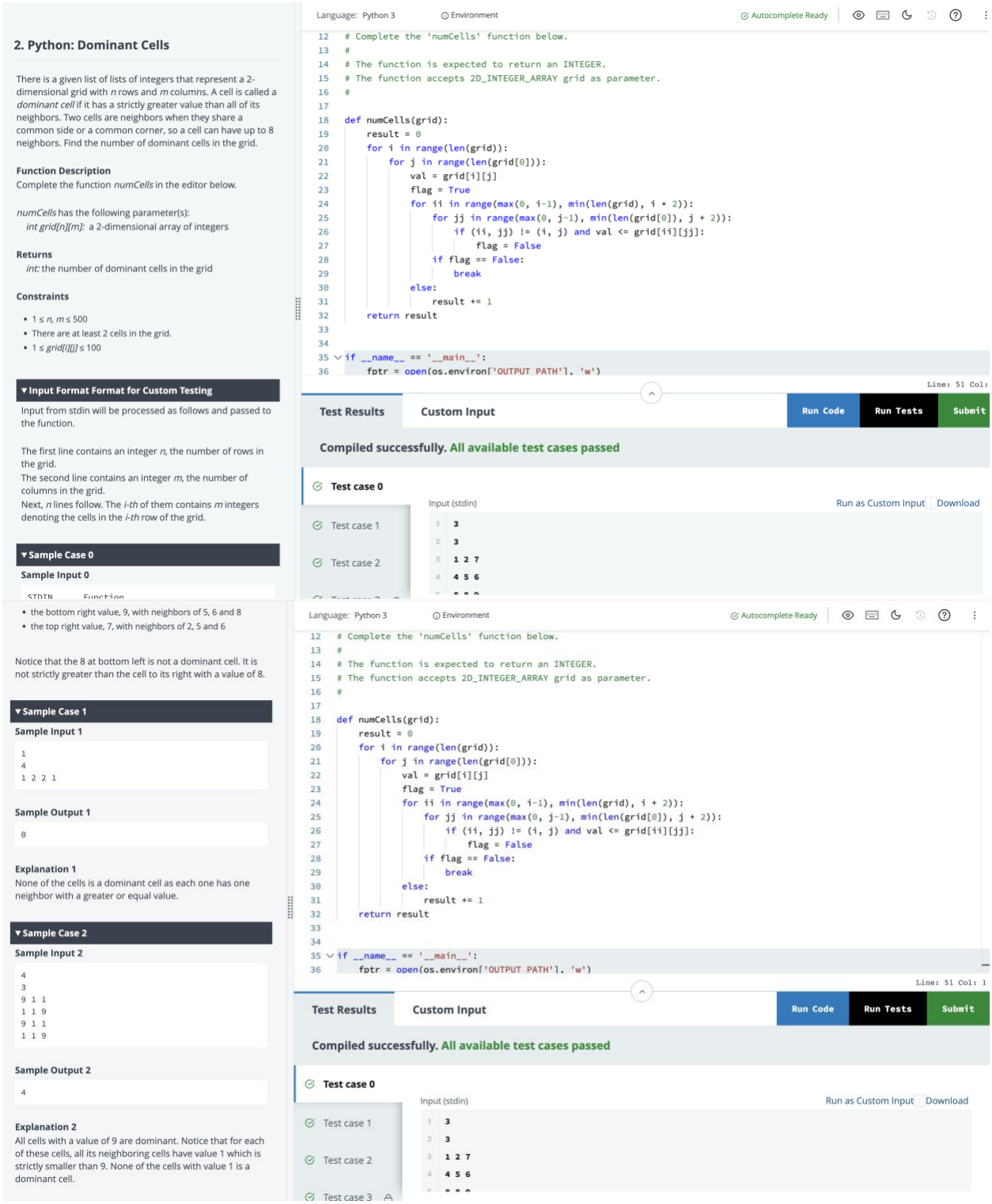
```
Coding IS Awesome
```

Expected Output — Download

```
Coding IS Awesome
```

Test case 1 | Test case 2 | Test case 3 | Test case 4 | Test case 5 | Test case 6

2. Python Dominant Cells



2. Python: Dominant Cells

There is a given list of lists of integers that represent a 2-dimensional grid with *n* rows and *m* columns. A cell is called a *dominant cell* if it has a strictly greater value than all of its neighbors. Two cells are neighbors when they share a common side or a common corner, so a cell can have up to 8 neighbors. Find the number of dominant cells in the grid.

Function Description
Complete the function *numCells* in the editor below.

numCells has the following parameter(s):
int grid[n][m]: a 2-dimensional array of integers

Returns
int: the number of dominant cells in the grid

Constraints

- $1 \le n, m \le 500$
- There are at least 2 cells in the grid.
- $1 \le grid[i][j] \le 100$

▼ Input Format Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer *n*, the number of rows in the grid.
The second line contains an integer *m*, the number of columns in the grid.
Next, *n* lines follow. The *i-th* of them contains *m* integers denoting the cells in the *i-th* row of the grid.

▼ Sample Case 0

Sample Input 0

```
STDIN       Function
```

- the bottom right value, 9, with neighbors of 5, 6 and 8
- the top right value, 7, with neighbors of 2, 5 and 6

Notice that the 8 at bottom left is not a dominant cell. It is not strictly greater than the cell to its right with a value of 8.

▼ Sample Case 1

Sample Input 1

```
1
4
1 2 2 1
```

Sample Output 1

```
0
```

Explanation 1
None of the cells is a dominant cell as each one has one neighbor with a greater or equal value.

▼ Sample Case 2

Sample Input 2

```
4
3
9 1 1
1 1 9
9 1 1
1 1 9
```

Sample Output 2

```
4
```

Explanation 2
All cells with a value of 9 are dominant. Notice that for each of these cells, all its neighboring cells have value 1 which is strictly smaller than 9. None of the cells with value 1 is a dominant cell.

Language: Python 3 | Environment | Autocomplete Ready

```
# Complete the 'numCells' function below.
#
# The function is expected to return an INTEGER.
# The function accepts 2D_INTEGER_ARRAY grid as parameter.
#

def numCells(grid):
    result = 0
    for i in range(len(grid)):
        for j in range(len(grid[0])):
            val = grid[i][j]
            flag = True
            for ii in range(max(0, i-1), min(len(grid), i + 2)):
                for jj in range(max(0, j-1), min(len(grid[0]), j + 2)):
                    if (ii, jj) != (i, j) and val <= grid[ii][jj]:
                        flag = False
                if flag == False:
                    break
            else:
                result += 1
    return result


if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')
```

Line: 51 Col:

Test Results | Custom Input | Run Code | Run Tests | Submit

Compiled successfully. All available test cases passed

Test case 0

Test case 1
Test case 2

Input (stdin) | Run as Custom Input | Download

```
3
3
1 2 7
4 5 6
```

Language: Python 3 | Environment | Autocomplete Ready

```
# Complete the 'numCells' function below.
#
# The function is expected to return an INTEGER.
# The function accepts 2D_INTEGER_ARRAY grid as parameter.
#

def numCells(grid):
    result = 0
    for i in range(len(grid)):
        for j in range(len(grid[0])):
            val = grid[i][j]
            flag = True
            for ii in range(max(0, i-1), min(len(grid), i + 2)):
                for jj in range(max(0, j-1), min(len(grid[0]), j + 2)):
                    if (ii, jj) != (i, j) and val <= grid[ii][jj]:
                        flag = False
                if flag == False:
                    break
            else:
                result += 1
    return result


if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')
```

Line: 51 Col: 1

Test Results | Custom Input | Run Code | Run Tests | Submit

Compiled successfully. All available test cases passed

Test case 0

Test case 1
Test case 2
Test case 3

Input (stdin) | Run as Custom Input | Download

```
3
3
1 2 7
4 5 6
```

the function.

The first and only line contains the string, *sentence*, that will be passed to the function.

▼Sample Case 0

Sample Input

```
STDIN                Function
-----                --------
aWESOME is cODING  →  sentence = "aWESOME is
cODING"
```

Sample Output

```
Coding IS Awesome
```

Explanation

sentence = "aWESOME is cODING"
Reverse the word order: "cODING IS aWESOME"
Swap the case: "Coding IS Awesome"
The order of the words is reversed and the case of all letters are swapped.

▼Sample Case 1

Sample Input

```
STDIN   Function
-----   --------
fUn  →  sentence = "fUn"
```

Sample Output

```
FuN
```

Explanation

sentence = "fUn"
Reverse the word order: The sentence contains only one word "fUn"
Swap the case of all letters: "FuN"

Language Python 2 | Environment | Autocomplete Ready

```
# Complete the 'reverse_words_order_and_swap_cases' function below.
#
# The function is expected to return a STRING.
# The function accepts STRING sentence as parameter.
#

def reverse_words_order_and_swap_cases(sentence):
    # Write your code here
    string = ""
    for word in sentence.split(" "):
        string += word[::-1] + " "
    return string[-2::-1].swapcase()


if __name__ == '__main__':
```

Line: 22 Col: 37

Test Results | Custom Input | Run Code | Run Tests | Submit

Compiled successfully. All available test cases passed

Test case 8 · Test case 9 · Test case 10 · Test case 11 · Test case 12 · Test case 13 · Test case 14

Input (stdin) — Run as Custom Input | Download

```
aWESOME is cODING
```

Your Output (stdout)

```
Coding IS Awesome
```

Expected Output — Download

```
Coding IS Awesome
```

dimensional grid with *n* rows and *m* columns. A cell is called a *dominant cell* if it has a strictly greater value than all of its neighbors. Two cells are neighbors when they share a common side or a common corner, so a cell can have up to 8 neighbors. Find the number of dominant cells in the grid.

Function Description
Complete the function *numCells* in the editor below.

numCells has the following parameter(s):
int grid[n][m]: a 2-dimensional array of integers

Returns
int: the number of dominant cells in the grid

Constraints

- $1 \le n, m \le 500$
- There are at least 2 cells in the grid.
- $1 \le grid[i][j] \le 100$

▼Input Format Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer *n*, the number of rows in the grid.
The second line contains an integer *m*, the number of columns in the grid.
Next, *n* lines follow. The *i-th* of them contains *m* integers denoting the cells in the *i-th* row of the grid.

▼Sample Case 0

Sample Input 0

```
STDIN       Function
-----       --------
3       →   n = 3
3       →   m = 3
1 2 7   →   grid = [[1, 2, 7], [4, 5, 6], [8, 8,
9]]
4 5 6
8 8 9
```

Language: Python 3 | Environment | Autocomplete Ready

```
def numCells(grid):
    result = 0
    for i in range(len(grid)):
        for j in range(len(grid[0])):
            val = grid[i][j]
            flag = True
            for ii in range(max(0, i-1), min(len(grid), i + 2)):
                for jj in range(max(0, j-1), min(len(grid[0]), j + 2)):
                    if (ii, jj) != (i, j) and val <= grid[ii][jj]:
                        flag = False
                if flag == False:
                    break
            else:
                result += 1
    return result
```

Line: 10 Col: 1

Test Results | Custom Input | Run Code | Run Tests | Submit

Compiled successfully. All available test cases passed

Test case 0 · Test case 1 · Test case 2 · Test case 3 · Test case 4 · Test case 5 · Test case 6

Input (stdin) — Run as Custom Input | Download

```
3
3
1 2 7
4 5 6
8 8 9
```

Your Output (stdout)

```
2
```

Expected Output — Download

```
2
```