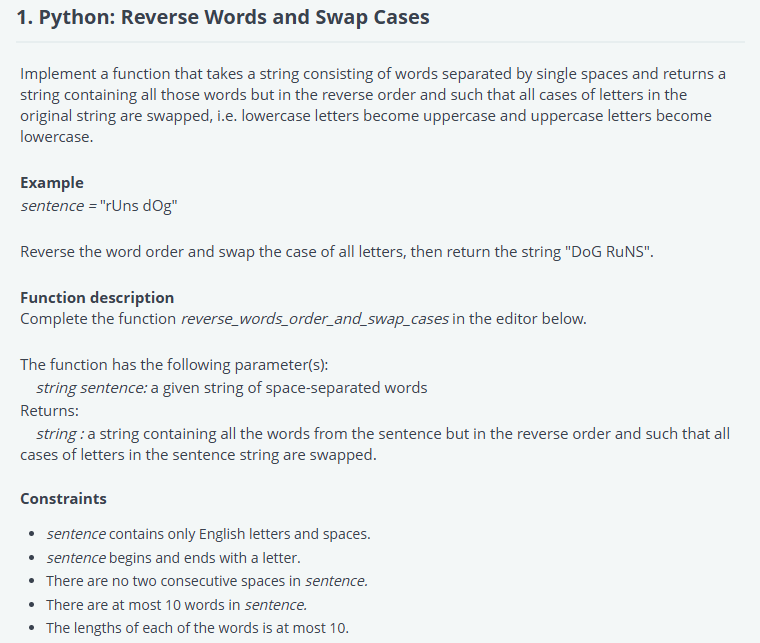
# Python: Reverse Words and Swap Cases



#!/bin/python

import math

import os

import random

import re

import sys

#

# 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

    # Split the sentence into words

    words = sentence.split()

    # Reverse the order of the words

    reversed\_words = words[::-1]

    # Join them back into a string with spaces and swap case

    result = ' '.join(reversed\_words).swapcase()

    return result

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

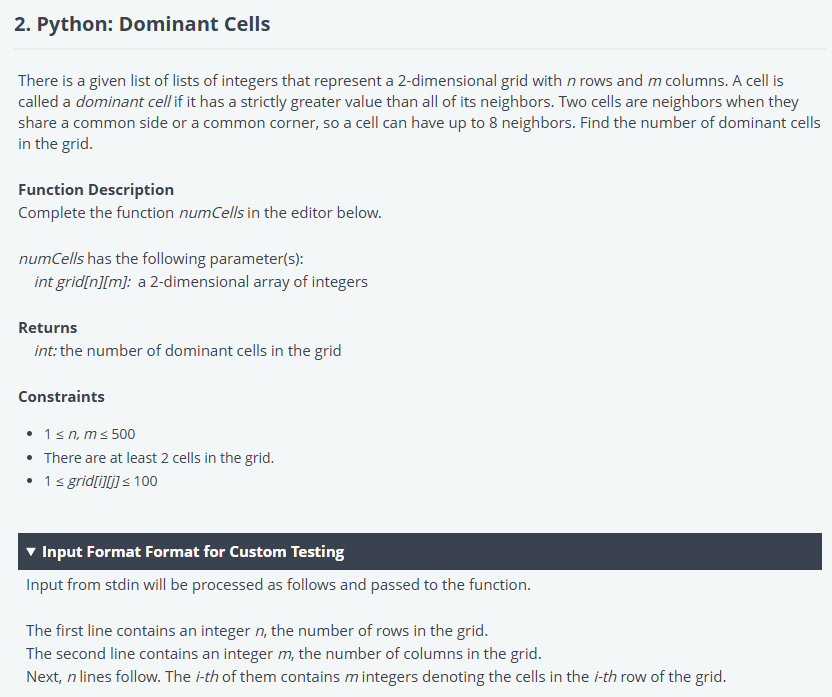
    sentence = raw\_input()

    result = reverse\_words\_order\_and\_swap\_cases(sentence)

    fptr.write(result + '\n')

    fptr.close()

# Python: Dominant Cells



#!/bin/python3

import math

import os

import random

import re

import sys

#

# 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):

    # Write your code here

    res = 0

    for i in range(len(grid)):

        for k in range (len(grid[0])):

            val = grid[i][k]

            flag = 1

            for ii in range (max(0,i-1),min(len(grid),i+2)):

                for kk in range(max(0,k-1),min(len(grid[0]),k+2)):

                    if (ii,kk)!=(i,k) and val<= grid[ii][kk] :

                         flag=0

                         break

                if flag == 0:

                     break

            else:

                res+=1

    return res

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    grid\_rows = int(input().strip())

    grid\_columns = int(input().strip())

    grid = []

    for \_ in range(grid\_rows):

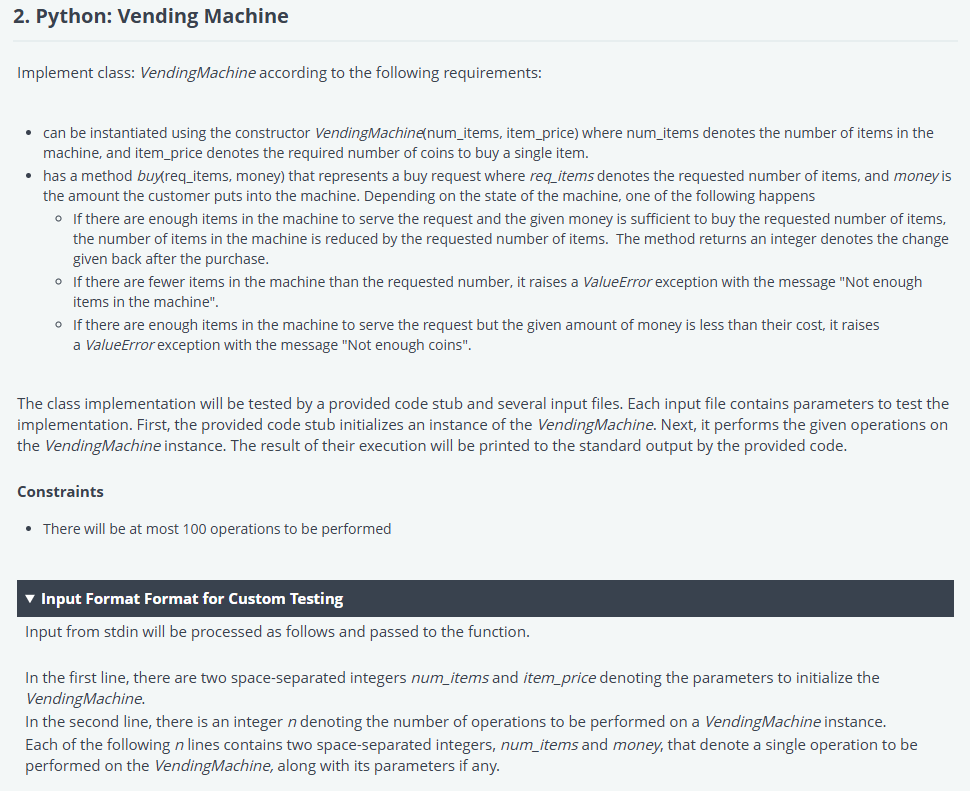
        grid.append(list(map(int, input().rstrip().split())))

    result = numCells(grid)

    fptr.write(str(result) + '\n')

    fptr.close()

# Python: Vending Machine



import os

class VendingMachine:

    def \_\_init\_\_(self, num\_items, item\_price):

        # Initialize the vending machine with the given number of items and item price

        self.num\_items = num\_items

        self.item\_price = item\_price

    def buy(self, req\_items, money):

        # Check if the requested items are available in the machine

        if req\_items > self.num\_items:

            raise ValueError("Not enough items in the machine")

        # Check if the provided money is enough to buy the requested items

        total\_price = req\_items \* self.item\_price

        if money < total\_price:

            raise ValueError("Not enough coins")

        # If both conditions are met, calculate the change and update the machine's inventory

        self.num\_items -= req\_items

        change = money - total\_price

        return change

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    # Read the number of items and item price

    num\_items, item\_coins = map(int, input().split())

    # Create the VendingMachine instance

    machine = VendingMachine(num\_items, item\_coins)

    # Read number of operations

    n = int(input())

    for \_ in range(n):

        # Read the requested items and the amount of money provided

        req\_items, money = map(int, input().split())

        try:

            # Try to make the purchase

            change = machine.buy(req\_items, money)

            fptr.write(str(change) + "\n")

        except ValueError as e:

            # If there's an error (e.g., not enough items or not enough money), write the error message

            fptr.write(str(e) + "\n")

    fptr.close()