Register for Certification exam (https://examform.nptel.ac.in/2022_04/exam_form/dashboard)

Course outline

Week 3 Programming Assignment

Due on 2022-08-18, 23:59 IST

Write three Python functions as specified below. Paste the text for all three functions together into the submission window. Your function will be called automatically with various inputs and should return values as specified. Do not write commands to read any input or print any output.

- You may define additional auxiliary functions as needed.
- In all cases you may assume that the value passed to the function is of the expected type, so your function does not have to check for malformed inputs.
- For each function, there are normally some public test cases and some (hidden) private test cases.
- "Compile and run" will evaluate your submission against the public test cases.
- "Submit" will evaluate your submission against the hidden private test cases. There are 15 private test cases, with equal weightage. You will get feedback about which private test cases pass or fail, though you cannot see the actual test cases.
- Ignore warnings about "Presentation errors".

1. Write a function `contracting(l)` that takes as input a list of integer `l` and returns `True` if the absolute difference between each adjacent pair of elements strictly decreases.

   Here are some examples of how your function should work.

```
>>> contracting([9,2,7,3,1])
True

>>> contracting([-2,3,7,2,-1])
False

>>> contracting([10,7,4,1])
False
```

2. In a list of integers `l`, the neighbours of `l[i]` are `l[i-1]` and `l[i+1]`. `l[i]` is a *hill* if it is strictly greater than its neighbours and a *valley* if it is strictly less than its neighbours.

Week 3 Programming Assignment (/noc22_cs70/progassignment?name=125)

Week 4: Sorting, Tuples, Dictionaries, Passing Functions, List Comprehension ()

Week 4 Quiz ()

Week 4 Programming Assignment ()

Week 5: Exception handling, input/output, file handling, string processing ()

Week 5 Programming Assignment ()

Week 6: Backtracking, scope, data structures; stacks, queues and heaps ()

Week 6 Quiz ()

Week 7: Classes, objects and user defined datatypes ()

Week 7 Quiz ()

Week 8: Dynamic programming, wrap-up ()

Week 8 Programming

Write a function `counthv(l)` that takes as input a list of integers `l` and returns a list `[hc,vc]` where `hc` is the number of hills in `l` and `vc` is the number of valleys in `l`.

Here are some examples to show how your function should work.

```
>>> counthv([1,2,1,2,3,2,1])
[2, 1]

>>> counthv([1,2,3,1])
[1, 0]

>>> counthv([3,1,2,3])
[0, 1]
```

3. A square n×n matrix of integers can be written in Python as a list with n elements, where each element is in turn a list of n integers, representing a row of the matrix. For instance, the matrix

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

would be represented as `[[1,2,3], [4,5,6], [7,8,9]]`.

Write a function `leftrotate(m)` that takes a list representation m of a square matrix as input, and returns the matrix obtained by rotating the original matrix counterclockwize by 90 degrees. For instance, if we rotate the matrix above, we get

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

Your function should *not* modify the argument m provided to the function `rotate()`.

Here are some examples of how your function should work.

```
>>> leftrotate([[1,2],[3,4]])
[[2, 4], [1, 3]]

>>> leftrotate([[1,2,3],[4,5,6],[7,8,9]])
[[3, 6, 9], [2, 5, 8], [1, 4, 7]]

>>> leftrotate([[1,1,1],[2,2,2],[3,3,3]])
[[1, 2, 3], [1, 2, 3], [1, 2, 3]]
```

Assignment ()

Text Transcripts ()

Books ()

Download Videos ()

Problem Solving Session ()

Private Test cases used for evaluation	**Input**	**Expected Output**	**Actual Output**	**Status**
Test Case 1	`contracting([98,21,77,35,11])`	`True\n`	`True\n`	`Passec`
Test Case 2	`contracting([-36,25,79,38,11])`	`True\n`	`True\n`	`Passec`
Test Case 3	`contracting([100,77,33,11])`	`False\n`	`False\n`	`Passec`
Test Case 4	`contracting([12,-11,10,-9,8,-7,6,-5,4,-3,2,-1])`	`True\n`	`True\n`	`Passec`
Test Case 5	`contracting([-32,-11,10,-9,8,-7,6,-5,4,-3,2,-1])`	`False\n`	`False\n`	`Passec`
Test Case 6	`counthv([23,44,22,1,26,10])`	`[2, 1]\n`	`[2, 1]\n`	`Passec`
Test Case 7	`counthv([23,44,22,1,5,1])`	`[2, 1]\n`	`[2, 1]\n`	`Passec`
Test Case 8	`counthv([1,10,2,11,3,12,4,13,5,14,6])`	`[5, 4]\n`	`[5, 4]\n`	`Passec`
Test Case 9	`counthv([1,10,2,11,3,12,4,13,5,14,23])`	`[4, 4]\n`	`[4, 4]\n`	`Passec`
Test Case 10	`counthv([12,55,22,88,40])`	`[2, 1]\n`	`[2, 1]\n`	`Passec`
Test Case 11	`leftrotate([[1,1,1,1],[2,2,2,2],[3,3,3,3],[4,4,4,4]])`	`[[1, 2, 3, 4], [1, 2, 3, 4], [1, 2, 3, 4], [1, 2, 3, 4]]\n`	`[[1, 2, 3, 4], [1, 2, 3, 4], [1, 2, 3, 4], [1, 2, 3, 4]]\n`	`Passec`

Test Case 12	`leftrotate([[1,1,1,1,1],[2,2,2,2,2],[3,3,3,3,3],[4,4,4,4,4],[5,5,5,5,5]])`	`[[1, 2, 3, 4, 5], [1, 2, 3, 4, 5], [1, 2, 3, 4, 5], [1, 2, 3, 4, 5], [1, 2, 3, 4, 5]]\n`	`[[1, 2, 3, 4, 5], [1, 2, 3, 4, 5], [1, 2, 3, 4, 5], [1, 2, 3, 4, 5], [1, 2, 3, 4, 5]]\n`	Passec
Test Case 13	`leftrotate([[1,1,1,1,1,1],[2,2,2,2,2,2],[3,3,3,3,3,3],[4,4,4,4,4,4],[5,5,5,5,5,5],[6,6,6,6,6,6]])`	`[[1, 2, 3, 4, 5, 6], [1, 2, 3, 4, 5, 6], [1, 2, 3, 4, 5, 6], [1, 2, 3, 4, 5, 6], [1, 2, 3, 4, 5, 6], [1, 2, 3, 4, 5, 6]]\n`	`[[1, 2, 3, 4, 5, 6], [1, 2, 3, 4, 5, 6], [1, 2, 3, 4, 5, 6], [1, 2, 3, 4, 5, 6], [1, 2, 3, 4, 5, 6], [1, 2, 3, 4, 5, 6]]\n`	Passec

Test Case 14	`leftrotate([[1,1,1,1,1,1,1],[2,2,2,2,2,2,2],[3,3,3,3,3,3,3],[4,4,4,4,4,4,4],[5,5,5,5,5,5,5],[6,6,6,6,6,6,6], [7,7,7,7,7,7,7]])`	` [[1, 2, 3, 4, 5, 6, 7], [1, 2, 3, 4, 5, 6, 7], [1, 2, 3, 4, 5, 6, 7], [1, 2, 3, 4, 5, 6, 7], [1, 2, 3, 4, 5, 6, 7], [1, 2, 3, 4, 5, 6, 7], [1, 2, 3, 4, 5, 6, 7]]\n`	`[[1, 2, 3, 4, 5, 6, 7], [1, 2, 3, 4, 5, 6, 7], [1, 2, 3, 4, 5, 6, 7], [1, 2, 3, 4, 5, 6, 7], [1, 2, 3, 4, 5, 6, 7], [1, 2, 3, 4, 5, 6, 7], [1, 2, 3, 4, 5, 6, 7]]\n`	`Passed`

Test Case 15	leftrotate([[1,1,1,1,1,1,1,1],[2,2,2,2,2,2,2,2],[3,3,3,3,3,3,3,3],[4,4,4,4,4,4,4,4],[5,5,5,5,5,5,5,5], [6,6,6,6,6,6,6,6],[7,7,7,7,7,7,7,7], [8,8,8,8,8,8,8,8]])	[[1, 2, 3, 4, 5, 6, 7, 8], [1, 2, 3, 4, 5, 6, 7, 8], [1, 2, 3, 4, 5, 6, 7, 8], [1, 2, 3, 4, 5, 6, 7, 8], [1, 2, 3, 4, 5, 6, 7, 8], [1, 2, 3, 4, 5, 6, 7, 8], [1, 2, 3, 4, 5, 6, 7, 8], [1, 2, 3, 4, 5, 6, 7, 8]]\n	[[1, 2, 3, 4, 5, 6, 7, 8], [1, 2, 3, 4, 5, 6, 7, 8], [1, 2, 3, 4, 5, 6, 7, 8], [1, 2, 3, 4, 5, 6, 7, 8], [1, 2, 3, 4, 5, 6, 7, 8], [1, 2, 3, 4, 5, 6, 7, 8], [1, 2, 3, 4, 5, 6, 7, 8], [1, 2, 3, 4, 5, 6, 7, 8]]\n	Passed

The due date for submitting this assignment has passed.

15 out of 15 tests passed.

You scored 100.0/100.

Assignment submitted on 2022-08-18, 21:09 IST

Your last recorded submission was :

```
# Contracting function
def contracting(l):
  for i in range(0,len(l)-3):
    if (abs(l[i+2]-l[i+1]) < abs(l[i+1]-l[i])):
      continue
    else:
      return(False)
  return(True)


# Counting function
def counthv(l):
  hc=0
  vc=0
  if len(l) > 2:
    for i in range(1,len(l)-1):
      if l[i] > l[i-1] and l[i] > l[i+1]:
        hc+=1
      elif l[i] < l[i-1] and l[i] < l[i+1]:
        vc+=1
  return [hc,vc]

```

```

# Rotation function
def leftrotate(m):
    b=[]
    for i in range(len(m)):
        b.append([])
        for j in range(len(m)):
            b[i].append(m[j][len(m)-i-1])
    return(b)


import ast

def parse(inp):
  inp = ast.literal_eval(inp)
  return (inp)

fncall = input()
lparen = fncall.find("(")
rparen = fncall.rfind(")")
fname = fncall[:lparen]
farg = fncall[lparen+1:rparen]

if fname == "contracting":
  arg = parse(farg)
  print(contracting(arg))

if fname == "counthv":
  arg = parse(farg)
  print(counthv(arg))

if fname == "leftrotate":
  arg = parse(farg)
  savearg = arg
  ans = leftrotate(arg)
  if savearg == arg:
    print(ans)
  else:
    print("Side effect")

```

Sample solutions (Provided by instructor)

```
def contracting(l):
    if len(l) < 3:
        return(True)
    return((abs(l[1]-l[0]) > abs(l[2]-l[1])) and contracting(l[1:]))

###################

def contracting_iterative(l):
    if len(l) < 3:
        return(True)
    for i in range(len(l)-2):
        diff = abs(l[i+1]-l[i])
        if diff <= abs(l[i+2]-l[i+1]):
            return(False)
    return(True)

###################

def counthv(l):
    hills = 0
    valleys = 0
    for i in range(1,len(l)-1):
        if l[i] > l[i-1] and l[i] > l[i+1]:
            hills = hills + 1
        if l[i] < l[i-1] and l[i] < l[i+1]:
            valleys = valleys + 1
    return([hills,valleys])

###################

def leftrotate(m):
    size = len(m)
    rotated_m = []
    for i in range(size):
        rotated_m.append([])
    for c in range(size-1,-1,-1):
        for r in range(size):
            rotated_m[size-(c+1)].append(m[r][c])
    return(rotated_m)

###################
```

```
import ast

def parse(inp):
  inp = ast.literal_eval(inp)
  return (inp)

fncall = input()
lparen = fncall.find("(")
rparen = fncall.rfind(")")
fname = fncall[:lparen]
farg = fncall[lparen+1:rparen]

if fname == "contracting":
  arg = parse(farg)
  print(contracting(arg))

if fname == "counthv":
  arg = parse(farg)
  print(counthv(arg))

if fname == "leftrotate":
  arg = parse(farg)
  savearg = arg
  ans = leftrotate(arg)
  if savearg == arg:
    print(ans)
  else:
    print("Side effect")
```