Week 3 Programming Assignment

Due on 2019-08-22, 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 10 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 `expanding(l)` that takes as input a list of integer `l` and returns `True` if the absolute difference between each adjacent pair of elements strictly increases.

   Here are some examples of how your function should work.

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

Explanation: Differences between adjacent elements are `3-1 = 2`, `7-3 = 4`, `7-2 = 5`, `9-2 = 7`.

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

Explanation: Differences between adjacent elements are `3-1 = 2`, `7-3 = 4`, `7-2 = 5`, `2-(-3) = 5`, so not strictly increasing.

```
>>> expanding([1,3,7,10])
False
```

Explanation: Differences between adjacent elements are `3-1 = 2`, `7-3 = 4`, `10-7 = 3`, so not (strictly) increasing.

2. Write a function `accordian(l)` that takes as input a list of integer `l` and returns `True` if the absolute difference between each adjacent pair of elements alternates between increasing strictly and decreasing strictly.

   Here are some examples of how your function should work.

```
>>> accordian([1,5,1])
False
```

Explanation: Differences between adjacent elements are `5-1 = 4`, `5-1 = 4`, which are equal.

```
>>> accordian([1,5,2,8,3])
True
```

Explanation: Differences between adjacent elements are `5-1 = 4`, `5-2 = 3`, `8-2 = 6`, `8-3 = 5`, so the differences decrease, increase and then decrease.

```
>>> accordian([-2,1,5,2,8,3])
True
```

Explanation: Differences between adjacent elements are `1-(-2) = 3`, `5-1 = 4`, `5-2 = 3`, `8-2 = 6`, `8-3 = 5`, so the differences increase, decrease, increase and then decrease.

```
>>> accordian([1,5,2,8,1])
```

user defined datatypes

Week 7 Quiz

Week 8: Dynamic programming, wrap-up

Week 8 Programming Assignment

Download videos

Text Transcripts

Online Programming Test - Sample

Online Programming Test 1, 26 Sep 2019, 09:30-11:30

```
False
```

Explanation: Differences between adjacent elements are `1-(-2) = 3`, `5-1 = 4`, `5-2 = 3`, `8-2 = 6`, `8-1 = 7`, so the differences increase, decrease, increase and then increase again.

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

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

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

Here are some examples of how your function should work.

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

Explanation:

```
   1  2    becomes     3  1
   3  4                4  2
```

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

Explanation:

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

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

Explanation:

```
    1  1  1    becomes   3  2  1
    2  2  2              3  2  1
    3  3  3              3  2  1
```

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	expanding([11,35,77,21,98])	True\n	True\n	Passed
Test Case 2	expanding([11,38,79,25,-36])	True\n	True\n	Passed
Test Case 3	expanding([11,33,77,100])	False\n	False\n	Passed
Test Case 4	expanding([-1,2,-3,4,-5,6,-7,8,-9,10,-11,12])	True\n	True\n	Passed
Test Case 5	expanding([-1,2,-3,4,-5,6,-7,8,-9,10,-11,-32])	False\n	False\n	Passed
Test Case 6	accordian([23,44,22,1,26,10])	True\n	True\n	Passed

Test Case 7	accordian([23,44,22,1,5,1])	False\n	False\n	Passed
Test Case 8	accordian([1,10,2,11,3,12,4,13,5,14,6])	True\n	True\n	Passed
Test Case 9	accordian([1,10,2,11,3,12,4,13,5,14,23])	False\n	False\n	Passed
Test Case 10	accordian([12,55,22,88,40])	True\n	True\n	Passed
Test Case 11	rotate([[1,1,1,1],[2,2,2,2],[3,3,3,3],[4,4,4,4]])	[[4, 3, 2, 1], [4, 3, 2, 1], [4, 3, 2, 1], [4, 3, 2, 1]]\n	[[4, 3, 2, 1], [4, 3, 2, 1], [4, 3, 2, 1], [4, 3, 2, 1]]\n	Passed
Test Case 12	rotate([[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]])	[[5, 4, 3, 2, 1], [5, 4, 3, 2, 1], [5, 4, 3, 2, 1], [5, 4, 3, 2, 1], [5, 4, 3, 2, 1]]\n	[[5, 4, 3, 2, 1], [5, 4, 3, 2, 1], [5, 4, 3, 2, 1], [5, 4, 3, 2, 1], [5, 4, 3, 2, 1]]\n	Passed

Test Case 13	rotate([[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]])	[[6, 5, 4, 3, 2, 1], [6, 5, 4, 3, 2, 1], [6, 5, 4, 3, 2, 1], [6, 5, 4, 3, 2, 1], [6, 5, 4, 3, 2, 1], [6, 5, 4, 3, 2, 1]]\n	[[6, 5, 4, 3, 2, 1], [6, 5, 4, 3, 2, 1], [6, 5, 4, 3, 2, 1], [6, 5, 4, 3, 2, 1], [6, 5, 4, 3, 2, 1], [6, 5, 4, 3, 2, 1]]\n	Passed
Test Case 14	rotate([[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]])	[[7, 6, 5, 4, 3, 2, 1], [7, 6, 5, 4, 3, 2, 1], [7, 6, 5, 4, 3, 2, 1], [7, 6, 5, 4, 3, 2, 1], [7, 6, 5, 4, 3, 2, 1], [7, 6, 5, 4, 3, 2, 1], [7, 6, 5, 4, 3, 2, 1]]\n	[[7, 6, 5, 4, 3, 2, 1], [7, 6, 5, 4, 3, 2, 1], [7, 6, 5, 4, 3, 2, 1], [7, 6, 5, 4, 3, 2, 1], [7, 6, 5, 4, 3, 2, 1], [7, 6, 5, 4, 3, 2, 1], [7, 6, 5, 4, 3, 2, 1]]\n	Passed
Test Case 15	rotate([[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]])	[[8, 7, 6, 5, 4, 3, 2, 1], [8, 7, 6, 5, 4, 3, 2, 1], [8, 7, 6, 5, 4, 3, 2, 1], [8, 7, 6, 5, 4, 3, 2, 1], [8, 7, 6, 5, 4, 3, 2, 1], [8, 7, 6, 5, 4, 3, 2, 1], [8, 7, 6, 5, 4, 3, 2, 1], [8, 7, 6, 5, 4, 3, 2, 1]]\n	[[8, 7, 6, 5, 4, 3, 2, 1], [8, 7, 6, 5, 4, 3, 2, 1], [8, 7, 6, 5, 4, 3, 2, 1], [8, 7, 6, 5, 4, 3, 2, 1], [8, 7, 6, 5, 4, 3, 2, 1], [8, 7, 6, 5, 4, 3, 2, 1], [8, 7, 6, 5, 4, 3, 2, 1], [8, 7, 6, 5, 4, 3, 2, 1]]\n	Passed

Due Date Exceeded.

15 out of 15 tests passed.

You scored 100.0/100.

Your last recorded submission was :

```
def expanding(l):
    diff = []

    for i in range(1,len(l)):
        diff.append(abs(l[i]-l[i-1]))

    check = 0
    for i in range(1,len(diff)):
        if diff[i] > diff[i-1]:
            check+=1
        else:
            return False
    if check == len(diff)-1:
        return True

def accordian(l):
    if len(l) <= 2:
        return False

    diff = []

    for i in range(1,len(l)):
        diff.append(abs(l[i]-l[i-1]))

    rtrn = False
    for i in range(1,len(diff)-1):
        if diff[i] > diff[i-1]:
            if diff[i+1] < diff[i]:
                rtrn = True
            else:
                return False
        elif diff[i] < diff[i-1]:
            if diff[i+1] > diff[i]:
                rtrn = True
            else:
                return False
        else:
            return False
    return rtrn

def rotate(m):
    new_m = [e[:] for e in m]
    N = len(m)

    for i in range(0,N):
        for j in range(0,N):
            new_m[j][N-1-i] = m[i][j]
    return new_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 == "expanding":
  arg = parse(farg)
  print(expanding(arg))

if fname == "accordian":
  arg = parse(farg)
```

```
  print(accordian(arg))

if fname == "rotate":
  arg = parse(farg)
  savearg = []
  for row in arg:
    savearg.append(row[:])
  ans = rotate(arg)
  if savearg == arg:
    print(ans)
  else:
    print("Side effect")


```