X

Programming Assignment 2

Due on 2021-11-12, 23:59 IST

Write four Python functions as specified below. Paste the text for both 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 16 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. Function: `matched(s)`

   Write a function `matched(s)` that takes as input a string `s` and checks if the brackets `"("` and `")"` in `s` are matched: that is, every `"("` has a matching `")"` after it and every `")"` has a matching `"("` before it. Your function should ignore all other symbols that appear in `s`. Your function should return `True` if `s` has matched brackets and `False` if it does not.

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

```
>>> matched("zb%78")
True

>>> matched("(7)(a")
False

>>> matched("a)*(?")
False

>>> matched("((jkl)78(A)&l(8(dd(FJI:),):)?)")
True
```

2. Function: `splitsum(l)`

   Write a Python function `splitsum(l)` that takes a nonempty list of integers and returns a list `[pos,neg]`, where `pos` is the sum of squares all the positive numbers in `l` and `neg` is the sum of cubes of all the negative numbers in `l`.

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

```
>>> splitsum([1,3,-5])
[10, -125]

>>> splitsum([2,4,6])
[56, 0]

>>> splitsum([-19,-7,-6,0])
[0, -7418]

>>> splitsum([-1,2,3,-7])
[13, -344]
```

3. Function: `matrixflip(m,d)`

   A two dimensional matrix can be represented in Python row-wise, as a list of lists: each inner list represents one 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]]`.

   A horizonatal flip reflects each row. For instance, if we flip the previous matrix horizontally, we get

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

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

   A vertical flip reflects each column. For instance, if we flip the previous matrix that has already been flipped horizontally, we get

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

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

   Write a Python function `matrixflip(m,d)` that takes as input a two dimensional matrix `m` and a direction `d`, where `d` is either `'h'` or `'v'`. If `d == 'h'`, the function should return the matrix flipped horizontally. If `d == 'v'`, the function should retun the matrix flipped vertically. For any other value of `d`, the function should return `m` unchanged. In all cases, the argument `m` should remain undisturbed by the function.

   Here are some examples to show how your function should work. You may assume that the input to the function is always a non-empty matrix.

   ```
   >>> myl = [[1,2],[3,4]]

   >>> myl
   [[1, 2], [3, 4]]

   >>> matrixflip(myl,'h')
   [[2, 1], [4, 3]]

   >>> myl
   [[1, 2], [3, 4]]
   ```

```
>>> matrixflip(myl,'v')
[[3, 4], [1, 2]]

>>> myl
[[1, 2], [3, 4]]

>>> matrixflip(matrixflip(myl,'h'),'v')
[[4, 3], [2, 1]]

>>> myl
[[1, 2], [3, 4]]

>>> matrixflip(matrixflip(myl,'h'),'v')
[[4, 3], [2, 1]]

>>> myl
[[1, 2], [3, 4]]
```

4. Function: rainaverage(l)

   We have a list of annual rainfall recordings of cities. Each element in the list is of the form (c,r) where c is the city and r is the annual rainfall for a particular year. The list may have multiple entries for the same city, corresponding to rainfall recordings in different years.

   Write a Python function rainaverage(l) that takes as input a list of rainfall recordings and computes the avarage rainfall for each city. The output should be a list of pairs (c,ar) where c is the city and ar is the average rainfall for this city among the recordings in the input list. Note that ar should be of type float. The output should be sorted in dictionary order with respect to the city name.

   Here are some examples to show how rainaverage(l) should work.

```
>>> rainaverage([(1,2),(1,3),(2,3),(1,1),(3,8)])
[(1, 2.0), (2, 3.0), (3, 8.0)]

>>> rainaverage([('Bombay',848),('Madras',103),('Bombay',923),('Bangalore',201),('Madr
as',128)])
[('Bangalore', 201.0), ('Bombay', 885.5), ('Madras', 115.5)]
```

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Stat
Test Case 1	matched("a3qw3;4w3(aasdgsd)((agadsgdsgag)agaga)")	True\n	True\n	Pa
Test Case 2	matched("(ag(Gaga(agag)Gaga)GG)a)33)cc(")	False\n	False\n	Pa
Test Case 3	matched("((((((((((()))))))))))")	True\n	True\n	Pa
Test Case 4	matched("(adsgdsg(agaga)a")	False\n	False\n	Pa
Test Case 5	splitsum([1,2,3,4,5,6])	[91, 0]\n	[91, 0]\n	Pa
Test Case 6	splitsum([1,4,-9,16,-25,36,-49,64])	[5665, -134003]\n	[5665, -134003]\n	Pa
Test Case 7	splitsum([0,1,-1,0,2,-2,3,-3])	[14, -36]\n	[14, -36]\n	Pa

Test Case 8	splitsum([-1,-2,-3,-4,-5,-6])	[0, -441]\n	[0, -441]\n	Pa
Test Case 9	matrixflip([[1,2,3],[4,5,6],[7,8,9]],'h')	[[3, 2, 1], [6, 5, 4], [9, 8, 7]]\n	[[3, 2, 1], [6, 5, 4], [9, 8, 7]]\n	Pa
Test Case 10	matrixflip([[1,2,3],[4,5,6],[7,8,9]],'v')	[[7, 8, 9], [4, 5, 6], [1, 2, 3]]\n	[[7, 8, 9], [4, 5, 6], [1, 2, 3]]\n	Pa
Test Case 11	matrixflip([[1,2,3]],'h')	[[3, 2, 1]]\n	[[3, 2, 1]]\n	Pa
Test Case 12	matrixflip([[1,2,3]],'v')	[[1, 2, 3]]\n	[[1, 2, 3]]\n	Pa
Test Case 13	rainaverage([(1,0),(1,3),(2,3),(1,1),(3,8),(3,-8)])	[(1, 1.3333333333333333), (2, 3.0), (3, 0.0)]\n	[(1, 1.3333333333333333), (2, 3.0), (3, 0.0)]\n	Pa
Test Case 14	rainaverage([('Bombay',848),('Madras',103),('Bombay',923),('Bangalore',201),('Madras',128)])	[('Bangalore', 201.0), ('Bombay', 885.5), ('Madras', 115.5)]\n	[('Bangalore', 201.0), ('Bombay', 885.5), ('Madras', 115.5)]\n	Pa
Test Case 15	rainaverage([('Bombay',1848),('Madras',103),('Bombay',923),('Bangalore',201),('Madras',128),('Madras',103),('Bombay',948),('Bangalore',323)])	[('Bangalore', 262.0), ('Bombay', 1239.6666666666667), ('Madras', 111.33333333333333)]\n	[('Bangalore', 262.0), ('Bombay', 1239.6666666666667), ('Madras', 111.33333333333333)]\n	Pa
Test Case 16	rainaverage([('Bombay',1848),('Bombay',923),('Bombay',201),('Bombay',128),('Bombay',103),('Bombay',948),('Bangalore',323)])	[('Bangalore', 323.0), ('Bombay', 691.8333333333334)]\n	[('Bangalore', 323.0), ('Bombay', 691.8333333333334)]\n	Pa

The due date for submitting this assignment has passed.
16 out of 16 tests passed.
You scored 100.0/100.

Assignment submitted on 2021-11-11, 00:15 IST

Your last recorded submission was :

```
def matched(s):
    list_str = list(s)
    len_str = len(list_str)
    result = False

    c_open = list_str.count("(")
    c_close = list_str.count(")")
    if c_open!= c_close:
        return False

    if ('(' not in list_str) and (')' not in list_str):
        return True

    for i in range(len_str):
        if list_str[i]=="(":
            for j in range(i+1,len_str):
                if list_str[j]==")":
                    result = True
                    break
            else:
                result = False

    return result

def splitsum(l):
    pos = [i for i in l if i>=0]
    neg = [i for i in l if i<0]
    pos_sum = 0
    neg_sum = 0
    for i in pos:
        pos_sum+= (i*i)
    for j in neg:
        neg_sum+=(j*j*j)
    return([pos_sum,neg_sum])

```

```

def matrixflip(m,d):
    flip_m=[]
    copy_m = m[:]
    if d == 'h':
        for i in copy_m:
            i = i[::-1]
            flip_m.append(i)
    elif d == 'v':
        for i in range(-1,-len(copy_m)-1,-1):
            flip_m.append(copy_m[i])
    else:
        flip_m = copy_m
    return flip_m


def rainaverage(l):
    length = len(l)
    city=[]
    ar_l =[]
    for i in range(length):
        ar=[]
        if l[i][0] not in city:
            ar.append(l[i][1])
            city.append(l[i][0])
            for j in range(i+1,length):
                if l[i][0] == l[j][0]:
                    ar.append(l[j][1])
            ar_l.append((l[i][0],float(sum(ar)/len(ar))))
    ar_l.sort()
    return ar_l
import ast

def tolist(inp):
  inp = "["+inp+"]"
  inp = ast.literal_eval(inp)
  return (inp[0],inp[1])

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 == "matched":
   arg = parse(farg)
   print(matched(arg))
elif fname == "splitsum":
   arg = parse(farg)
   print(splitsum(arg))
elif fname == "matrixflip":
  (arg1,arg2) = parse(farg)
  savearg1 = []
  for row in arg1:
    savearg1.append(row[:])
  myans = matrixflip(arg1,arg2)
  if savearg1 == arg1:
    print(myans)
  else:
    print("Illegal side effect")
elif fname == "rainaverage":
   arg = ast.literal_eval(farg)
   print(rainaverage(arg),end="\n")
else:
   print("Function", fname, "unknown")

```

Sample solutions (Provided by instructor)

```
def matched(s):
  nesting = 0
  for c in s:
    if c == '(':
      nesting = nesting + 1
    elif c == ')':
      nesting = nesting - 1
    if nesting < 0:
      return(False)
  return(nesting == 0)

###########

def splitsum(l):
    pos = 0
    neg = 0
    for x in l:
        if x > 0:
            pos = pos + x**2
        if x < 0:
            neg = neg + x**3
    return([pos,neg])

```

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

def matrixflip(l,d):
  outl = []
  for row in l:
    outl.append(row[:])
  if d == 'h':
    for row in outl:
      row.reverse()
  elif d == 'v':
    outl.reverse()
  return(outl)

###########

def rainaverage(l):
  raindata = {}
  for (c,r) in l:
    if c in raindata.keys():
      raindata[c].append(r)
    else:
      raindata[c] = [r]
  outputlist = []
  for c in sorted(raindata.keys()):
    thisaverage = sum(raindata[c])/len(raindata[c])
    outputlist.append((c,thisaverage))
  return(outputlist)

###########

import ast

def tolist(inp):
  inp = "["+inp+"]"
  inp = ast.literal_eval(inp)
  return (inp[0],inp[1])

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 == "matched":
   arg = parse(farg)
   print(matched(arg))
elif fname == "splitsum":
   arg = parse(farg)
   print(splitsum(arg))
elif fname == "matrixflip":
  (arg1,arg2) = parse(farg)
  savearg1 = []
  for row in arg1:
    savearg1.append(row[:])
  myans = matrixflip(arg1,arg2)
  if savearg1 == arg1:
    print(myans)
  else:
    print("Illegal side effect")
elif fname == "rainaverage":
   arg = ast.literal_eval(farg)
   print(rainaverage(arg),end="\n")
else:
   print("Function", fname, "unknown")

```