**《Python程序设计》第二次作业（条件、循环与函数）**

1. 假设你是某门课的助教，负责批改平时作业并登记平时成绩。列表Files记录了你已收到作业的文件名。作业命名要求为“第**x**次作业+姓名+学号”（**x**为一、二或三）。列表Scores则记录了Files中对应作业的分数，例如，Files中第二个作业**'第一次作业+李宇鸣+2022000552 '**的分数为88。请完成以下任务：

Files = [**'第一次作业+刘彦俊+2022000406'**,**'第一次作业+李宇鸣+2022000552'**,**'第一次作业+王云桐+2022000941'**,**'第一次作业+刘云桐+2022000456'**,**'第一次作业+刘含梓+2022000356'**,**'第一次作业+刘诗诗+2022000584'**,**'第一次作业+张云桐+2022000656'**,**'第一次作业+张宇鸣+2022000593'**,**'第一次作业+张诗诗+2022000675'**,**'第一次作业+李诗诗+2022000344'**,**'第一次作业+王彦俊+2022000156'**,**'第一次作业+赵宇鸣+2022000406'**,**'第一次作业+李云桐+2022000411'**,**'第三次作业+刘含梓+2022000356'**,**'第三次作业+刘云桐+2022000456'**,**'第三次作业+刘彦俊+2022000406'**,**'第三次作业+刘诗诗+2022000584'**,**'第三次作业+周彦俊+2022000789'**,**'第三次作业+张含梓+2022000728'**,**'第三次作业+张宇鸣+2022000593'**,**'第三次作业+张诗诗+2022000675'**,**'第三次作业+李云桐+2022000411'**,**'第三次作业+王云桐+2022000941'**,**'第三次作业+王彦俊+2022000156'**,**'第三次作业+赵宇鸣+2022000406'**,**'第三次作业+张云桐+2022000656'**,**'第三次作业+李宇鸣+2022000552'**,**'第二次作业+刘彦俊+2022000406'**,**'第二次作业+李宇鸣+2022000552'**,**'第二次作业+刘云桐+2022000456'**,**'第二次作业+刘含梓+2022000356'**,**'第二次作业+周彦俊+2022000789'**,**'第二次作业+张云桐+2022000656'**,**'第二次作业+张含梓+2022000728'**,**'第二次作业+张宇鸣+2022000593'**,**'第二次作业+李云桐+2022000411'**,**'第二次作业+李诗诗+2022000344'**,**'第二次作业+王彦俊+2022000156'**,**'第二次作业+张诗诗+2022000675'**,**'第二次作业+王云桐+2022000941'**]  
Scores = [85, 88, 95, 71, 86, 94, 76, 75, 90, 78, 88, 96, 87, 93, 87, 79, 81, 70, 88, 88, 74, 76, 72, 72, 86, 99, 93, 76, 81, 99, 89, 88, 85, 76, 96, 98, 78, 74, 80, 87]

1. 整理平时作业数据：对于Files中的每个作业，从其文件名中提取出**作业编号**、**姓名**、**学号**信息，并从Scores中匹配对应的**分数**。最后将所有作业的这四项信息存储到字典Homework中。
2. 计算平时成绩并排名：对于每位同学，将其三次平时作业的平均分作为平时成绩（注：缺交的作业记零分），再按如下格式将所有同学平时成绩的排名打印输出：

**第1：刘含梓(2022000356) 89.3分  
第2：李宇鸣(2022000552) 87.3分  
第3：李云桐(2022000411) 87.0分  
...**

参考答案：

*# a.*Homework = {}

*# 方法1*  
**for** File,Score **in** zip(Files,Scores):  
 wid,name,sid = File.split(**"+"**)  
 **if** sid **not in** Homework.keys():  
 Homework[sid] = {**"姓名"**:name,**"平时作业"**:{}}  
 Homework[sid][**"平时作业"**][wid] = Score

*# 方法2*

for i in range(0, len(Files)):

c = Files[i].split('+')

score = Scores[i]

…

*# 方法3*

for i, filename in enumerate(Files):

info = filename.split("+")

…

score = Scores[i]

*# 方法4*

for i, filename in enumerate(Files):

regex = r'^第([一二三])次作业\+([\u4e00-\u9fa5]+)\+(\d{10})$'

match = re.match(regex, filename)

*# 方法5*

for filename in Files:

…

score = Scores[Files.index(filename)]

*# 方法6*

for file in Files:

i = file.find('+')

j = file.find('+', i + 1)

Numbers.append(file[0:i])

Names.append(file[i + 1:j])

IDs.append(file[j + 1:])

…

*# 方法7-将姓名和学号组合作为key*

for i in range(len(Scores)):

key = Names[i]+"("+IDs[i]+")"

if not Homework.\_\_contains\_\_(key):

value = {Numbers[i]: Scores[i]}

Homework[key] = value

*# 方法8-提取元素方法*

info = [file.split('+') for file in Files]

order = [file[0] for file in info]

names = [file[1] for file in info]

ids = [file[2] for file in info]

…

*# 方法9-将姓名和学号元组作为key*

for i in range(0,len(Files)):

j=Files[i].find('+')

m=Files[i].find('+',j+1)

Name=(Files[i][j+1:m])

Number=(Files[i][:j])

ID=(Files[i][m+1:])

score=Scores[i]

if (Name,ID) in Homework.keys():

Homework[(Name,ID)][Number]=score

else:

Homework[(Name,ID)]={Number:score}

*# b.***for** sid,\_ **in** Homework.items():  
 scores = list(\_[**"平时作业"**].values())  
 **while** len(scores) < 3:  
 scores.append(0)  
 Homework[sid][**"平时成绩"**] = sum(scores)/3

*# 方法1*  
**for** i,item **in** enumerate(sorted(Homework.items(),key=**lambda** x:-x[1][**"平时成绩"**])):  
 print(**"第%d：%s(%s) %.1f分"** % (i+1,item[1][**"姓名"**],item[0],item[1][**"平时成绩"**]))

*# 方法2*

score\_rank = dict(sorted(score\_list.items(), key=lambda x: x[1], reverse=True))

for i in range(0, len(score\_rank)):

print(f'第{i+1}：'+list(score\_rank.keys())[i], '{}'.format(list(score\_rank.values())[i]))

*# 方法3*

rank = 1

for name, score in sorted(scores.items(), key=lambda x: x[1], reverse=True):

print(f"第{rank}：{name}({student\_num}) {score:.1f}分")

rank += 1

*# 方法4-小数点保留位数方法*

average\_score = round(sum(scores.values())/3,1)

1. 循环语句可用于打印输出如下由字符“\*”填充的、高度（即行数）为7的菱形图案：

**\*  
\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*  
\***

请定义一个函数DiamondPrint(Char,N,Dir)，用于按指定方向（Dir）连续打印输出N个由指定字符（Char）填充的、高度为7的菱形图案。其中，Dir的值默认为X（横向），也可为Y（纵向）；相邻菱形通过“角”（指定方向上最边缘的单个字符）相连。例如，分别执行DiamondPrint(**"%"**,2,**"Y"**)和DiamondPrint(**"c"**,6)可打印输出如下两个图案，其中标红部分为“角”：

**%**

**%%%**

**%%%%%**

**%%%%%%% c c c c c c**

**%%%%% ccc ccc ccc ccc ccc ccc**

**%%% ccccc ccccc ccccc ccccc ccccc ccccc**

**% ccccccccccccccccccccccccccccccccccccc**

**%%% ccccc ccccc ccccc ccccc ccccc ccccc**

**%%%%% ccc ccc ccc ccc ccc ccc**

**%%%%%%% c c c c c c**

**%%%%%**

**%%%**

**%**

参考答案：

*# 方法1*

**def** DiamondPrint(Char,N,Dir=**"X"**):  
 **if** Dir == **"Y"**:  
 k = 0  
 **while** k < N:  
 **for** i **in** range(4):  
 print(**" "**\*(3-i)+Char\*(2\*i+1))  
 **for** i **in** range(2):  
 print(**" "**\*(i+1)+Char\*(5-2\*i))  
 k += 1  
 print(**" "**\*3+Char+**" "**\*3)  
  
 **elif** Dir == **"X"**:  
 **for** i **in** range(3):  
 print(**" "**\*(3-i)+(Char\*(2\*i+1)+**" "**\*(5-2\*i))\*N)  
 print(Char\*(6\*N+1))  
 **for** i **in** range(3):  
 print(**" "**\*(i+1)+(Char\*(5-2\*i)+**" "**\*(2\*i+1))\*N)  
 **return**DiamondPrint(**"%"**,2,**"Y"**)  
DiamondPrint(**"c"**,6)

*# 方法2-跟方法1相似，但循环位置不同*

…

elif Dir == 'Y':

print(3\*" " + Char)

for i in range(N):

for j in range(2, 4, 1):

print((4-j)\*" " + (2\*j-1)\*Char)

for j in range(4, 0, -1):

print((4-j)\*" " + (2\*j-1)\*Char)

*# 方法3-直接计算重复次数*

def DiamondPrint(Char, N, Dir):

if Dir == 'Y':

…

else:

print(" ", ("%s " % (Char)) \* N)

print(" ", ("%s" %(Char) \*3+" ")\*N)

print(" ", ("%s" % (Char)\*5+" ") \* N)

print("",("%s" % (Char)\*6) \* N+"%s"%(Char))

print(" ", ("%s" % (Char)\*5+" ") \* N)

print(" ", ("%s" %(Char) \*3+" ")\*N)

print(" ", ("%s " % (Char)) \* N)

*# 方法4-创建菱形模板*

def DiamondPrint(Char, N, Dir='X'):

diamond = [[' ' for i in range(7)] for j in range(7)]

for i in range(4):

diamond[i][3-i:4+i] = [Char]\*(2\*i+1)

diamond[6-i][3-i:4+i] = [Char]\*(2\*i+1)

…

elif Dir == 'X':

for row in diamond:

for i in range(N):

print(''.join(row), end='')

print()

*# 方法5-利用center和reversed函数*

def DiamondPrint(Char, N, Dir='X'):

'打印内容为字符（Char），重复N次，Dir为X时呈横向的菱形（Y时为纵向）'

if Dir == 'Y':

for i in range(1, 7, 2):

print((Char \* i).center(7))

for i in reversed(range(1, 8, 2)):

print((Char \* i).center(7))

for j in range(1, N):

for i in range(3, 7, 2):

print((Char \* i).center(7))

for i in reversed(range(1, 8, 2)):

print((Char \* i).center(7))

elif Dir == 'X':

for i in range(1, 6, 2):

print(' '+((Char \* i).center(6)) \* N)

print((Char\*(6\*N+1)))

for i in reversed(range(1, 6, 2)):

print(' '+((Char \* i).center(6)) \* N)

*# 方法6*

def DiamondP(Char, Y, X, Diamond):

b = ' ' + Char + ' '

Diamond = list(Diamond)

for i in range(len(Diamond)):

Diamond[i] = X \* Diamond[i] + b[i]

ll = len(Diamond)

for i in range(Y - 1):

for j in range(ll):

Diamond.append(Diamond[j])

Diamond.append(X \* (' ' + Char + ' '))

for i in range(len(Diamond)):

print(Diamond[i])

def DiamondPrint(Char, N, Dir='X'):

Diamond = [

" " + Char + ' ', " " + 3 \* Char + " ", " " + 5 \* Char, 6 \* Char,

" " + 5 \* Char, " " + 3 \* Char + " "

]

if Dir == 'X':

DiamondP(Char, 1, N, Diamond)

elif Dir == 'Y':

DiamondP(Char, N, 1, Diamond)

*# 方法7*

if Dir == 'Y':

for j in range(N):

i = 1

while i < 7:

if i < 5:

print((Char\*(2\*i-1)).center(7))

else:

print((Char\*(15-2\*i)).center(7))

i += 1

print(Char.center(7))

*# 方法8*

def DiamondPrint(Char, N, Dir = "X"):

if Dir == "X":

for i in range(0, 3):

line[i] = Char\*char\_num[i] + " "\*between\_num[i]

line[i] = " "\*side\_num[i] + line[i]\*(N-1) + Char\*char\_num[i] + " "\*side\_num[i]

line[3] = Char \* (6 \* N + 1)

line[4] = line[2]

line[5] = line[1]

line[6] = line[0]

for i in range(0, 6+1):

print(line[i])

…

*# 方法9-利用join函数*

if Dir=='Y':

…

else:

for j in range(1,8):

if j <=3:

print(' '\*(4-j),Char\*(2\*j-1),''.join((' '\*(-2\*j+7),Char\*(2\*j-1))\*(N-1)),sep='')

elif j==4:

print(Char\*7,(Char\*6)\*(N-1),sep='')

else:

print(' '\*(j-4),Char\*(-2\*j+15),''.join((' '\*(2\*j-9),Char\*(-2\*j+15))\*(N-1)),sep='')

return ''

*# 方法10-利用rjust函数*

def DiamondPrint(Char,N,Dir='X'):

if Dir == 'X':

print(Char.rjust(4)+(Char.rjust(6))\*(N-1))

print((Char\*3).rjust(5)+((Char\*3).rjust(6))\*(N-1))

print(((Char\*5).rjust(6))\*N)

print(Char\*(6\*N+1))

print(((Char\*5).rjust(6))\*N)

print((Char\*3).rjust(5)+((Char\*3).rjust(6))\*(N-1))

print(Char.rjust(4)+(Char.rjust(6))\*(N-1))

*# 方法11-在空格基础上，填充符号生成结果*

def DiamondPrint(Char, N, Dir = 'X'):

if Dir != 'X' and Dir != 'Y':

return

row = 0

if Dir == 'X':

row = 7

col = N

else:

row = 7 \* N

col = 1

i = 0

line = [' ' for j in range(6 \* N + 7)]

while i < row:

l = i % 7 + 1

start = abs(4 - l)

delta = 5 - 2 \* (3 - start)

length = 2 \* (3 - start) + 1

for j in range(col):

for k in range(length):

line[start + length \* j + delta \* j + k] = Char

print(''.join(line))

line = [' ' for j in range(6 \* N + 7)]

i += 1

if (i % 7 == 0):

i += 1

1. 请定义一个函数Distance(Mode,Co)，该函数可利用指定距离度量方法（Mode）计算输入的**任意多个**坐标（Co）中任意两个坐标之间的距离，并将结果存入字典Results并返回。Mode的值可为Manhattan（曼哈顿距离）或Euclidean（欧式距离），这两种方法的示意图及计算公式如下：

|  |  |
| --- | --- |
|  |  |
|  |  |

其中，X和Y分别代表一个坐标，如(1.5,2)和(2,1.5)。X*i*和Y*i*则对应坐标第*i*位的数值，如若X=(1.5,2)则X1=1.5。*k*为坐标的长度，如(1.5,2)的*k*为2。另外，返回结果Results的键为形如((1.5,2),(2,1.5))的坐标对，值为这对坐标的距离。例如，执行Distance(**"Euclidean"**,(1.5,2),(2,1.5),(1.5,1.5))返回的Results为：

{((1.5,2),(2,1.5)):0.7071067811865476,((1.5,2),(1.5,1.5)):0.5,((2,1.5),(1.5,1.5)):0.5}

参考答案：

*# 方法1*

**def** Distance(Mode,\*Co):  
 Results = {}  
 **for** i **in** range(len(Co)-1):  
 **for** j **in** range(i+1,len(Co)):  
 Co\_i,Co\_j = Co[i],Co[j]  
 **if** Mode == **"Euclidean"**:  
 D = sum([(Co\_i[k]-Co\_j[k])\*\*2 **for** k **in** range(len(Co\_i))])\*\*0.5  
 **elif** Mode == **"Manhattan"**:  
 D = sum([abs(Co\_i[k]-Co\_j[k]) **for** k **in** range(len(Co\_i))])  
 Results[(Co\_i,Co\_j)] = D  
 **return** Results  
print(Distance(**"Euclidean"**,(1.5,2),(2,1.5),(1.5,1.5)))  
print(Distance(**"Manhattan"**,(2,3),(19,1),(7,3),(-1,9),(0,1)))

*# 方法2*

import math

def Distance(Mode, \*Co):

…

elif Mode == "Euclidean":

…

distance = math.sqrt(sum(pow(a - b, 2) for a, b in zip(Co[i], Co[j])))

*# 方法3*

def Distance(Mode,\*Co):

…

elif Mode=='Euclidean':

…

distance+=pow(Co\_List[i][k]-Co\_List[j][k],2)

*# 方法4 (3班：胡湘文)*

import numpy as np

import itertools

def Distance(Mode, \*Co):

…

pairlist = list(itertools.combinations(colist, 2))

…

elif Mode == "Euclidean":

…

Results[i] = np.sqrt(np.sum(np.square(np.array(i[0])-np.array(i[1]))))

*# 方法5*

import math

dist = math.sqrt(sum([(co[i][k] - co[j][k])\*\*2 for k in range(len(co[i]))]))

*# 方法6*

if Mode =='Euclidean':

…

counting\_2=pow(sum(counting\_1),0.5)

*# 方法7 -不同距离公式用函数封装*

def Distance(Mode, \*Co):

def Euclidean(x, y):

return math.sqrt(sum([(a-b)\*\*2 for a, b in zip(x, y)]))

def Manhattan(x, y):

return sum([abs(a-b) for a, b in zip(x, y)])

if Mode == "Euclidean":

dist\_func = Euclidean

elif Mode == "Manhattan":

dist\_func = Manhattan

Results = {}

for i in range(len(Co)):

for j in range(i+1, len(Co)):

d = dist\_func(Co[i], Co[j])

Results[(Co[i], Co[j])] = d

*# 方法8*

for i in range(len(Co)):

for j in range(i+1, len(Co)):

if Mode == "Manhattan":

dist = sum(abs(a-b) for a, b in zip(Co[i], Co[j]))

elif Mode == "Euclidean":

dist = pow(sum(pow(a-b, 2) for a, b in zip(Co[i], Co[j])), 0.5)

…

**作业要求**：

1. **作业提交截止时间**：2023年3月31日晚24:00前（迟交会酌情扣分）
2. **作业提交方式**：发送至助教邮箱。发送邮件标题命名为：Python第二次作业+姓名+学号。
3. **文件命名方式**：每道题目的程序单独建一个python文件（.py或.ipynb），例如本次作业建3个文件，分别命名为1.py，2.py，3.py或1.ipynb，2.ipynb，3.ipynb。

本次作业统一打包成一个压缩文件(.zip或.rar)，以邮件附件的形式提交给助教，压缩文件命名方式为“Python第二次作业+姓名+学号.zip，或者“第二次作业+姓名+学号.rar”