מבחן בית – פייתון הלכה למעשה – 2020

את המבחן יש להגיש עד ה29.7 בשעה 12:00 בבוקר. אין הארכות זמן.

המבחן מחולק לשני חלקים, בחלק א' 3 שאלות (60 נקודות, 10 לשאלה 1 ו-25 לכל השאר) ובחלק ב' שאלה אחת (40 נקודות). עליכם לענות על כל השאלות במבחן. כל הקוד הרלוונטי לחלק הראשון יוגש יחד בקובץ הנקרא partA.py, הקוד לחלק השני יוגש בקובץ בשם partB.py ואת התשובות המילוליות תגישו בקובץ pdf אחד בשם answers_file. את כל הקבצים הללו עליכם להגיש במודל.

כמו בתרגילים עליכם לרשום בשורה הראשונה את תעודת הזהות שלכם ובשורה השנייה את מספר השעות שהשקעתם במבחן (כהערות כמובן). בנוסף, עליכם לתעד את הקוד על ידי הוספה של דוקומנטציה לכל פונקציה \ מתודה \ class.

חלק א':

1. פלינדרום הוא מילה או משפט (ובמקרה שלנו מחרוזת) שקריאתו מימין לשמאל ומשמאל לימין היא זהה. למשל: "אבא", "ילד כותב בתוך דלי", "פרשנו רעבתן שבדבש נתבער ונשרף".

כתבו רקורסיה הנקראית is_palindrome הבודקת בצורה רקורסיבית האם מחרוזת היא פולינדרום. שימו לב כי אין חשיבות האם האותיות הן uppercase או lowercase. כלומר, שתי המחרוזות aba ו-Aba יחזירו True.
הפונקציה מקבלת מחרוזת והיא מחזירה האם המחרוזת היא פלינדרום או לא (True\False). אסור להוסיף פונקציות נוספות מעבר לפונקציה זו ואסור להשתמש בלולאות.

2. בשאלה זו עליכם לממש שני classים הראשון למספרים זוגיים בשם EvenNumber עבור מספרים זוגיים והשני בשם MultiplesOfSix עבור מספרים אותם ניתן לחלק ב-6 ללא שארית.
ה-class בשם EvenNumber יכול לקבל רק מספרים שלמים הקטנים מ-100 (לא כולל 100) שהינם זוגיים. במידה והפרמטר אינו תקין עליכם להקפיץ שגיאה מתאימה.
ה-class בשם MultiplesOfSix יכול לקבל רק מספרים הקטנים מ-100 שהם כפולה של 6 (לא כולל 100). במידה ומתקבל ערך לא חוקי עליכם להקפיץ שגיאה מתאימה (סוג שגיאה וטקסט בהתאם).
שני ה-classים יהיו עם attribute בשם value של הערך עצמו (במידה והוא חוקי).
שני ה-classים צריכים לתמוך במתודה add_number המקבלת מספר נוסף ומחזירה True אם הסכום של המספרים עונה על התנאים של אותו class. במידה והסכום עונה על התנאים המתודה מחזירה True ומעדכנת את ערך המספר (value) לסכום המספרים. במידה ולא המתודה מחזירה False (ללא שגיאה וללא עדכון של הערך).
לדוגמה:

```
num = EvenNumber(3) -> return error
num = EvenNumber(4)
num.add_number(4) -> return True (value = 8)
num.add_number(3) -> return False (value is still 8)
```

3. בשאלה זו תנתחו נתונים הנלקחו מתוך נתוני NBA אמיתיים. הנתונים נמצאים בקבצי csv המצורפים למבחן. בעונת NBA יש שני שלבים מרכזיים: עונה רגילה (regular season) ופלייאוף (playoff).

הקבצים הנתונים לכם הם הקבצים הבאים:

- קבצים בשם Playoff_XXX.csv – כל קובץ כזה מכיל את הפירוט אודות קבוצה מסוימת בשלב הפלייאוף.
- קובץ בשם Regular_season.csv – קובץ המכיל את נתוני כל הקבוצות עבור שלב העונה הרגילה.

בכל הקבצים יש שלוש עמודות בלבד:

- FULL NAME – השם המלא של השחקן
- TEAM – שלוש אותיות שמייצגות את הקבוצה בה השחקן משחק
- POINTS PER GAME – ממוצע הנקודות של השחקן במשחק

עליכם לממש פונקציה בשם traded_players_points_per_game המקבלת כקלט שני פרמטרים: מיקום התיקייה בה נמצאים הקבצים ומחרוזת המייצגת שם של קבוצה (3 אותיות ראשונות כפי שמופיע בשמות הקבצים שקיבלתם). מטרת הפונקציה היא לחשב את ממוצעי הנקודות בשלב הפלייאוף בלבד רק עבור השחקנים של הקבוצה (המיוצגת על ידי שלוש האותיות) אשר שיחקו בעונה הרגילה ביותר מקבוצה אחת.

הפונקציה תחזיר dataframe עם שתי עמודות (נוסף על עמודת האינדקסים): שם השחקן (עמודה ראשונה) וממוצע הנקודות (עמודה שנייה). ה-dataframe יהיה ממוין מהשחקן עם הממוצע הגבוהה ביותר לנמוך ביותר. שימו לב שאתם מאפסים את עמודת האינדקסים לאחר המיון (השחקן עם הממוצע הגבוהה ביותר צריך להיות עם אינדקס 0). אם אין שחקנים העונים על ההגדרות הפונקציה תחזיר dataframe עם אותן עמודות רק ריק (ללא שורות).

בשאלה זו אסור להשתמש בלולאות ואסור לשנות את שמות העמודות או התאים.
אין לתקן או לשנות את הדאטה עצמו. למשל, אם נגיד יש שם של שחקן אשר מופיע פעם בצורה אחת ופעם בצורה אחרת, מבחינתכם זה שני שחקנים שונים.

חלק ב':

בשאלה זו תממשו קוד אשר ינהל הושבת אורחים בשולחנות של מסעדה על ידי class שנקרא Restaurant. המסעדה תוגדר על ידי רשימה של מספרים. כל מספר ברשימה מגדיר את מספר מקומות הישיבה בשולחן הספציפי. למשל הרשימה [2,2,3] מתארת מסעדה עם שלושה שולחנות – שני שולחנות עם שני מושבים ושולחן אחד עם שלושה מושבים. למען הפשטות בכל שולחן הכיסאות מסודרים בשורה אחת, כלומר אי אפשר להושיב חלק מהקבוצה בתחילת השולחן וחלק אחר בסוף השולחן (ראו איורים להמחשה למטה). המסעדה היא מסעדה חברותית וקבוצות אנשים שונות המגיעות יכולות לשבת יחדיו באותו שולחן. כמובן שאין מושיבים אנשים מאותה קבוצה בשולחנות נפרדים.

להלן המתודות אשר עליכם לממש ב-class:

- הושבה של קבוצה למסעדה – group_seating – המתודה מקבלת tuple עם שם הקבוצה כאיבר הראשון ומספר הנפשות כאיבר השני. המתודה תחזיר True אם ניתן היה להושיב את הקבוצה במסעדה ו-False אחרת. כלומר, במידה והיה מקום במסעדה הפונקציה גם תוסיף את האנשים לרשימות המסעדה וגם תחזיר True.
- הסרה של קבוצה מהמסעדה – group_removal – המתודה מקבלת שם של קבוצה ומחזירה True אם הצליחה לפנות את מקומות הישיבה של הקבוצה ו-False אם לא הייתה קיימת קבוצה בשם זה במסעדה.
- מצב המסעדה – status – מתודה המחזירה את התפוסה של המסעדה, מספר המושבים התפוסים.
- פינוי כל הסועדים – reset – מתודה אשר מרוקנת את המסעדה. לאחר הפעלת מתודה זו המתודה של מצב המסעדה מחזירה 0.

השקיעו מחשבה במימוש המסעדה והסבירו בקובץ התשובות מדוע החלטתם לממש את הפתרון בצורה שבחרתם.

הערות:

- במידה וקבוצה מגיעה ובאחד השולחנות יש מספר מקומות פנויים בדיוק כגודל הקבוצה, עליכם להושיב את הקבוצה בשולחן זה. בשאר המקרים אתם יכולים להושיבם באיזה צורה שתחפצו.
- אין מקרה בו יש שתי קבוצות עם אותו שם ואין מספרים בשמות. לכן אין צורך לטפל במקרים אלו.
- נסו לגרום לקוד להיות כמה שיותר ברור בצורה המאפשרת גם לאחרים להבינו ולהשתמש בו.
- אין חשיבות ליעילות הקוד ואין צורך להשתמש באלגוריתמים לשיפור התפוסה במסעדה.
- אין צורך לבדוק קלט לא תקין.

דוגמה להרצה (עם איורים להמחשה):

```
rest = Restaurant([3, 2])
```

```
rest.group_seating(("GROUP1", 2))
print(rest.status())  # 2
```


```
rest.group_seating(("GROUP2", 1))
print(rest.status())  # 3
```


```
rest.group_removal("GROUP1")
print(rest.status())  # 1
```


```
rest.reset()
print(rest.status())  # 0
```

בהצלחה!