חשיבה מחשובית ותכנות בשפת פייתון
תרגיל בית 10

הנחיות כלליות:

- קראו **היטב** את השאלות והקפידו שהתכניות שלכם פועלות בהתאם לנדרש.
- את התרגיל יש לפתור לבד!
- הקפידו על הוראות ההגשה המפורסמות במודל. בפרט, יש להגיש את כל השאלות יחד בקובץ ex#_012345678.py המצורף לתרגיל, לאחר החלפת ה# במספר התרגיל והחלפת הספרות 012345678 במספר ת.ז שלכם, כל 9 הספרות כולל ספרת הביקורת. למשל, אם מספר תעודת הזהות שלי הינו 112233445 ואני מגיש את תרגיל מספר 1, שם הקובץ שאגיש יהיה ex1_112233445.py.
- מועד אחרון להגשה: כמפורסם באתר.
- בדיקה עצמית: כדי לוודא את נכונותן ואת עמידותן של התוכניות לקלטים שגויים, בכל שאלה, הריצו את תוכניתכם עם מגוון קלטים שונים, אלה שהופיעו כדוגמאות בתרגיל וקלטים נוספים עליהם חשבתם (וודאו כי הפלט נכון וכי התוכנית אינה קורסת).
- היות ובדיקת התרגילים עשויה להיות אוטומטית, **יש להקפיד על פלטים מדויקים על פי הדוגמאות (כן כן, עד לרמת הרווח).**
- אופן ביצוע התרגיל: בתרגיל זה עליכם להשלים את הקוד בקובץ המצורף.
- **יש לעבוד עם המשתנים שמופיעים בשלד התרגיל ואין לשנות את שמם.** על קטע הקוד של כל שאלה לעבוד ולספק את התוצאה הדרושה עבור קלט שיוזן במשתנים שמופיעים בשלד (המשתנים שלידם סימני שאלה ומחכים לקלט כפי שמופיע בדוגמאות בתרגול). יחד עם זאת, אתם רשאים להוסיף משתנים נוספים כראותם עינכם.
- מומלץ להתעדכן בפורום לגבי שאלות של סטודנטים אחרים וכמובן, במידה ועדיין משהו לא ברור, לשאול בעצמכם.

<u>שימו לבים! (תקף החל מתרגיל בית מספר 5 ואילך)</u>

- ✓ שימו לב 1, החתימה של כל אחת מהפונקציות שעליכם לממש מופיעה כבר בשלד התרגיל ואין לשנותה. עליכם לממש את <u>גוף הפונקציה בלבד</u>. יש למחוק את הפקודה pass (היא נמצאת שם רק כדי שתוכלו להריץ חלקי קוד בלי לשים בהערה את המימושים החסרים) ולהחליף אותה במימוש המתאים.
- ✓ שימו לב 2, בשאלה בה נתבקשתם לממש פונקציה, אין לעשות שום דבר מעבר לכך. ניתן ואף רצוי לקרוא לפונקציה שכתבתם עם מגוון קלטים על מנת לוודא את תקינותה אך אין להשאיר את הקריאות הללו בהגשת התרגיל.
- ✓ שימו לב 3, אף פונקציה לא מדפיסה דבר אלא רק מחזירה ערך כלשהו.

בתרגיל זה ננתח את נתוני השימוש במתקני "לונה פארק תל חביב". קראו את התיאור בקפידה והשלימו את הקוד לחישוב של חיתוכים שונים טבלת סיכום נתוני השימוש במתקני "לונה פארק תל חביב". שימו לב שהחבילה Numpy כבר מיובאת בתחילת הקוד עבורכם ואין צורך לייבא אותה שוב.

נתון קובץ בשם luna_park.csv שהינו קובץ טקסט (בפורמט CSV) המכיל טבלת סיכום השימוש במתקני "לונה פארק תל חביב" באחד הימים במהלך החופש הגדול. כותרות השורות (עמודה ראשונה) מייצגות את שם הילד. כל שורה מתעדת את מספר השימושים של ילד ספציפי בכל אחד מהמתקנים. כותרות העמודות (שורה ראשונה) מייצגות את שמות המתקנים בלונה פארק. כל עמודה מתעדת את מספר השימושים של כל אחד מהילדים במתקן ספציפי. כאמור, השדות השונים בקובץ מופרדים על ידי פסיק ועל כן אם נסתכל ישירות בקובץ. למשל, שלוש השורות הראשונות בו יראו כך :

```
Individual,Anaconda,Bumper_cars,Water_slides,Pirate_ship,Top_spin,Sky_loop,Roller_coaster
Oren,3,7,9,4,3,2,1
Jonathan,6,9,5,6,0,6,7
```

מבין השורות הנ"ל, למשל, אפשר לראות שיהונתן הכי אהב את מתקן המכוניות המתנגשות (שכנעו את עצמכם שהבנתם למה, ובאופן כללי, שהבנתם את מבנה הדאטה, <u>לפני</u> התחלת העבודה. גישה טובה באופן כללי, לא רק בתרגיל זה).

*הדמויות הומצאו לצרכי השאלה. כל קשר בינן לבין המציאות מקרי בהחלט (אני דווקא מעדיף רכבות הרים).

נתונה לכם הפונקציה load_luna_park_data_to_lists(csv_file) (אתם מוזמנים לנסות ולממש אותה בעצמכם) המקבלת מחרוזת המייצגת נתיב לקובץ CSV של נתוני שימוש מתקנים בלונה פארק (במבנה הנ"ל) הפונקציה מחזירה 3 רשימות (רגילות) המייצגות את טבלת הסיכום לפי הפירוט הבא:

- הרשימה הראשונה מכילה את שמות המתקנים (באותו סדר כמו בקובץ המקורי).
- הרשימה השנייה מכילה את שמות הילדים (באותו סדר כמו בקובץ המקורי).
- הרשימה השלישית מכילה את נתוני השימוש (באותו סדר כמו בקובץ המקורי).

להלן דוגמת הרצה:

```
>>> load_luna_park_data('luna_park.csv')
(['Anaconda', 'Bumper_cars', 'Water_slides', 'Pirate_ship', 'Top_spin',
'Sky_loop', 'Roller_coaster'], ['Oren', 'Jonathan', 'Gail', 'Yossi', 'Dana',
'Yair'], [[3, 7, 9, 4, 3, 2, 1], [6, 9, 5, 6, 0, 6, 7], [4, 5, 0, 3, 4, 9, 4], [5, 2, 6, 9,
1, 23, 10], [0, 7, 8, 0, 14, 0, 4], [0, 0, 0, 0, 0, 0, 0]])
```

חידוד: יש לכתוב קוד <u>כללי</u> שיפעל עבור טבלת סיכום השימוש במתקני כל לונה פארק (עם מספר כלשהו של מתקנים) לכל כמות של ילדים, קרי, אין להסתמך על הנתונים הספציפיים המובאים בדוגמא (נכון תמיד אך כאן בפרט). כמוכן, **אין להשתמש בלולאות** בקוד שאתם כותבים. לצורך מניעת חזרה בהוראות מטה, נניח ש המשתנה names הינו מערך המכיל את שמות הילדים, המשתנה rides הינו מערך המכיל את שמות המתקנים, והמשתנה data מכיל את נתוני השימוש.

<u>שאלה 1</u> – ממשו את הפונקציה load_luna_park_data_to_arrays(csv_file) המקבלת נתיב לקובץ ומחזירה 3 מערכי Numpy כאשר הראשון מכיל את שמות המתקנים, השני את שמות הילדים ואילו השלישי את נתוני השימוש. להלן דוגמת הרצה:

```
>>> rides, names, data = load_luna_park_data_to_arrays('luna_park.csv')
>>> print(rides)
['Anaconda' 'Bumper_cars' 'Water_slides' 'Pirate_ship' 'Top_spin'
 'Sky_loop' 'Roller_coaster']
>>> print(names)
['Oren' 'Jonathan' 'Gail' 'Yossi' 'Dana' 'Yair']
>> print(data)
[[ 3  7  9  4  3  2  2]
 [ 6  9  5  6  0  6  7]
```

```
[ 4 -1  0  3  4  9  4]
[ 5  2  6  9 -1 23 10]
[-1  7  8  0 14  0  4]
[ 0  0  0  0  0  0  0]]
```

<u>שאלה 2</u> – ממשו את הפונקציה max_use(data) שמחזירה מספר המייצג את כמות השימוש המקסימלית שבוצעה במתקן (יחיד) כלשהו ע"י ילד (יחיד) כלשהו. במקרה של קובץ הדוגמא, יוסי עלה על הsky_loop 23 פעמים. אף ילד לא עלה על אף מתקן יותר מ23 פעמים. להלן דוגמת הרצה:

```
>>> print(max_use(data))
23
```

<u>שאלה 3</u> – על מנת לחשב את רווחיות כרטיסי הכניסה לפארק, הנהלת "לונה פארק תל חביב" מעוניינת לדעת מה כמות השימוש הממוצעת לכל ילד במתקני הפארק. ממשו את הפונקציה average_use_per_kid(data) שמחזירה מערך שהאיבר הi שלו הינו כמות השימוש הממוצעת של ילד i על פני כל המתקנים בפארק (כמות האיברים במערך אמורה כמובן להיות ככמות הילדים שהופיעו בטבלה). להלן דוגמת הרצה:

```
>>> print(average_use_per_kid(data))
[4.28571429 5.57142857 3.28571429 7.71428571 4.57142857 0.]
```

<u>שאלה 4</u> – ממשו את הפונקציה usage_variance_per_ride(data) שמחזירה מערך שהאיבר הi שלו הינו שונות (var) כמות השימוש במתקן הi (כמות האיברים במערך אמורה כמובן להיות ככמות המתקנים שהופיעו בטבלה). להלן דוגמת הרצה:

```
>>> print(usage_variance_per_ride(data))
[ 6.47222222 14.66666667 12.55555556 10.22222222 25.88888889
63.88888889 10.58333333]
```

<u>שאלה 5</u> – ממשו את הפונקציה no_use(data) שמחזירה True אם קיים מתקן (אחד או יותר) שלא היה בו אף שימוש, וFalse אם כל המתקנים היו בשימוש. במקרה של קובץ הדוגמא, כל המתקנים היו בשימוש. להלן דוגמת הרצה:

```
>>> print(no_use(data))
False
```

שאלה 6 – ממשו את הפונקציה more_than_25(data) שמחזירה מספר המייצג את כמות המתקנים בהם היו למעלה מ25 שימושים. במקרה של קובץ הדוגמא, יש 3 כאלה- Water_slides, Sky_loop, Roller_coaster (עלו עליהם 28, 40, 27 פעמים, בהתאמה). להלן דוגמת הרצה:

```
>>> print(more_than_25(data))
3
```

שאלה 7 – ממשו את הפונקציה heavy_user(data, names) שמחזירה מחרוזת המייצגת את שם הילד שעלה סך הכל הכי הרבה פעמים על מתקנים בלונה פארק. ניתן להניח שיש רק אחד כזה. במקרה של קובץ הדוגמא, יוסי הינו הילד שעלה על הכי הרבה מתקנים (54). להלן דוגמת הרצה:

```
>>> print(heavy_user(data, names))
Yossi
```

שאלה 8 – לכבוד שנת 2020 רוצה הנהלת "לונה פארק תל חביב" להוסיף מתקן חדש לפארק. שטח הפארק קטן ולא ניתן להכניס מתקן נוסף. הנהלת הפארק החליטה לכנס ועדה שתדון האם כדאי להוריד את המתקן הכי פחות פופולרי. ממשו את הפונקציה least_popular_ride(data, rides) שמחזירה מחרוזת המייצגת את שם המתקן הכי פחות פופולרי. ניתן להניח שיש רק אחד כזה. במקרה של קובץ הדוגמא, המתקן Anaconda היה בשימוש הכי מעט פעמים (17). להלן דוגמת הרצה:

```
>>> print(least_popular_ride(data, rides))
Anaconda
```

שאלה 9 – התברר ששכחו להזין את הנתונים של הילד האחרון בטבלה ולכן מופיע 0 בכל השורה שלו. אותו ילד עלה על כל אחד מהמתקנים בדיוק 5 פעמים. עלינו לתקן את הנתונים שבידינו. ממשו את הפונקציה fix_last_kid(data) שמחזירה מערך זהה בגודלו לזה של מערך הקלט כאשר ההבדל בינו לבין מערך הקלט הוא השורה

האחרונה, המכילה את המספר 5 בכל אחד מהאיברים שלה. תזכורת, יש לכתוב קוד כללי, שלכל כמות של ילדים, זה שיתוקן יהיה הילד האחרון. להלן דוגמת הרצה:

```
>>> print(fix_last_kid(data))
[[ 3  7  9  4  3  2  2]
 [ 6  9  5  6  0  6  7]
 [ 4 -1  0  3  4  9  4]
 [ 5  2  6  9 -1 23 10]
 [-1  7  8  0 14  0  4]
 [ 5  5  5  5  5  5  5]]
```

שימו לב, למען נוחיותכם, החל מכאן נניח בדוגמאות שהשורה האחרונה בdata מכילה 5-ים בלבד (אחרת נאלץ אחרי כל קריאה לפונקציה זו לטעון מחדש את הנתונים).

```
>>> print(data)
[[ 3  7  9  4  3  2  2]
 [ 6  9  5  6  0  6  7]
 [ 4 -1  0  3  4  9  4]
 [ 5  2  6  9 -1 23 10]
 [-1  7  8  0 14  0  4]
 [ 5  5  5  5  5  5  5]]
```

שאלה 10 – התברר שישנם אי דיוקים נוספים בטבלה. מסתבר שבכל תא בטבלה שהוזן 1- אמור היה להיות מוזן 1. כלומר, כל מופע של 1- בטבלה הינו שגוי וצריך להפוך ל1. עלינו לתקן את הנתונים שבידינו. ממשו את הפונקציה fix_wrong_minus(data) שמחזירה מערך זהה בגודלו לזה של מערך הקלט כאשר ההבדל בינו לבין מערך הקלט הוא תיקון מופעי 1- ל1. להלן דוגמת הרצה:

```
>>> print(fix_wrong_minus(data))
[[ 3  7  9  4  3  2  2]
 [ 6  9  5  6  0  6  7]
 [ 4  1  0  3  4  9  4]
 [ 5  2  6  9  1 23 10]
 [ 1  7  8  0 14  0  4]
```

```
 [ 5  5  5  5  5  5  5]]
```

שימו לב, למען נוחיותכם, החל מכאן נניח בדוגמאות שהמופעים של 1- בטבלה הפכו ל1 (אחרת נאלץ אחרי כל קריאה לפונקציה זו לטעון מחדש את הנתונים).

```
>>> print(data)
[[ 3  7  9  4  3  2  2]
 [ 6  9  5  6  0  6  7]
 [ 4  1  0  3  4  9  4]
 [ 5  2  6  9  1 23 10]
 [ 1  7  8  0 14  0  4]
 [ 5  5  5  5  5  5  5]]
```

<u>שאלה 11</u> – ההנהלה ב"לונה פארק תל חביב" מאמינה ש11 הוא מספר המסמל מזל רע. אם יש מתקן שעלו עליו כמות פעמים שמתחלקת ב11 (ללא שארית) הוא נחשב "מתקן של מזל רע" ולכן, סוגרים את המתקן ביום למחרת (אבל אל דאגה, יומיים אח"כ הוא חוזר לפעול כרגיל). ממשו את הפונקציה bad_luck_rides(data, rides) שמחזירה מערך המכיל את שמות כל המתקנים שנחשבים "מתקן של מזל רע". במקרה של קובץ הדוגמא (לאחר תיקון הנתונים בשני הסעיפים הקודמים), המתקן Water_slides נחשב "מתקן של מזל רע" (שכן עלו עליו 33 פעמים). להלן דוגמת הרצה:

```
>>> print(bad_luck_rides(data,rides))
['Water_slides']
```

<u>שאלה 12</u> – ממשו את הפונקציה sort_users_descending(data, names) שמחזירה מערך המכיל את שמות הילדים ממויינים בסדר יורד לפי כמות הפעמים שעלו על מתקנים בסך הכל. במקרה של קובץ הדוגמא (לאחר תיקון הנתונים בשני הסעיפים הקודמים), גייל עלתה הכי מעט פעמים סה"כ (25) ולכן תופיע אחרונה במערך המוחזר. להלן דוגמת הרצה:

```
>>> print(sort_users_descending(data, names))
['Yossi' 'Jonathan' 'Yair' 'Dana' 'Oren' 'Gail']
```