הוראות לפרוייקט תיכנות

הפרוייקט נועד להשיג כמה מטרות:

1. לבנות ספריית קוד פתוח של אלגוריתמים לחלוקה הוגנת, כדי לעזור לסטודנטים וחוקרים בעתיד.
2. לאמן אתכם הסטודנטים בקריאת מאמרים מחקריים והבנתם.
3. לאמן אתכם בתיכנות אלגוריתמים מתקדמים בשפת פייתון.

שלבי העבודה:

שלב א: הכנה (כ-30 דקות)

1. ל"מזלג" (fork) את הפרוייקט הזה: https://github.com/erelsgl/fairpy , להוריד למחשב שלכם, לוודא שאתם יכולים להריץ את הדוגמאות cut_and_choose_demo.py ו last_diminisher_demo.py.

2. לקרוא ולוודא שאתם מבינים את הקוד של האלגוריתמים בקבצים cut_and_choose.py ו last_diminisher.py (את האלגוריתמים עצמם אתם אמורים כבר להכיר מהשיעור הראשון).

- בכל הערה או שאלה אפשר לפנות אליי בדואל או לפני/אחרי השיעור.

3. להיכנס לדף https://github.com/erelsgl/fairpy/blob/master/references.md ולבחור אחד מהמאמרים תחת הכותרת Future Work.

- חלק מהמאמרים מסומנים כמתאימים למתכנת יחיד, וחלק מסומנים כמתאימים לזוג (מאמרים ארוכים יותר). ניתן לעבוד לבד או בזוג; יש לבחור מאמר לפי גודל הצוות.
- אנא שלחו לי דואל עם שם המאמר שבחרתם ושמות חבר/י הצוות, כדי לוודא שהוא לא תפוס.

שלב ב: קריאה (יום-יומיים, תלוי בשליטה שלכם באנגלית ומתמטיקה)

4. לקרוא את המאמר שבחרתם, ולהבין את האלגוריתמים.

- יש להתמקד בחלקים הדרושים להבנת האלגוריתמים: המבוא (פרק 1), הסימונים (בד"כ פרק 2), והאלגוריתם עצמו.
- הוכחת הנכונות של האלגוריתם – יש להבין באופן כללי, כדי להבין איך האלגוריתם עובד. אין צורך להבין את כל הפרטים הטכניים.
- לא חובה להבין את הפרקים שאינם קשורים לאלגוריתמים, כגון: הוכחת קושי חישובי.
- בכל הערה או שאלה אפשר לפנות אליי בדואל או לפני/אחרי השיעור.

שלב ג: תיכנות (יומיים-שלושה, תלוי בשליטה שלכם בתיכנות בפייתון)

5. ב"מזלג" שלכם, ליצור קובץ חדש ששמו כשם המאמר / המחברים / האלגוריתמים (מה שנראה לכם הכי מתאים), ולשים בו רק את הדברים הבאים:

- **כותרת לקובץ –** עם שם האלגוריתם, המאמר שבו הופיע (אפשר להעתיק מהקובץ references.md), שמ/ות המתכנת/ים, והתאירך.
- **כותרות של פונקציות** בהתאם לאלגוריתמים במאמר – פונקציה לכל אלגוריתם.
- **בדיקות-יחידה לכל פונקציה** – בעזרת **doctest**.

- כדי להבין איך עובד doctest, ראו כאן: https://docs.python.org/2/library/doctest.html וכן בדוגמאות בקבצים cut_and_choose.py, last_diminisher.py .
- ב-doctest יש לשים בדיקות פשוטות ובסיסיות שהתוצאה שלהן קלה לבדיקה ידנית, כגון בדיקת מקרים עם שניים-שלושה שחקנים.
- אם יש דוגמאות במאמר, אפשר ומומלץ להשתמש בהן ב-doctest.

אנא אל תתחילו לממש את האלגוריתמים עצמם בשלב זה – לפני כן אנא שילחו לי קישור לגיטהאב עם הכותרת והבדיקות, כדי לוודא שהבנתם נכון איך בדיוק זה אמור לעבוד.

6. אחרי שקיבלתם אישור על הכותרת והבדיקות – לממש את האלגוריתמים עצמם.
   - מומלץ להשתמש בספריות קיימות של פייתון במידת האפשר. למשל: cvxpy לפתרון בעיות אופטימיזציה, networkx לאלגוריתמים בגרפים, וכו'.
   - יש לכתוב קוד מתועד, ברור וקריא, כדי שסטודנטים וחוקרים בעתיד יוכלו להבין אותו בקלות.

7. כיוון שהאלגוריתמים אמורים לשמש גם ללימוד והדגמה לסטודנטים בעתיד, יש להוסיף הדפסות-ביניים המסבירות את שלבי הריצה, ע"י הפקודה **logger.info**. מה בדיוק להדפיס?
   - לפחות שורה אחת לכל שורה של האלגוריתם כפי שהוא כתוב במאמר. הטקסט לא צריך להיות זהה לטקסט שבמאמר, אלא להסביר במילים שלכם מה בדיוק האלגוריתם עושה.
   - אפשר ומומלץ להוסיף הדפסות נוספות כדי שהאלגוריתם יהיה ברור יותר.
   - כדי להבין איך זה עובד, ראו דוגמאות בקבצים cut_and_choose.py, last_diminisher.py.

8. ליצור קובץ חדש ששמו כשם הקובץ הראשון בתוספת _demo, ולכתוב בו תוכנית המדגימה את פעולת האלגוריתם במצבים שונים. איזה דוגמאות לשים?
   - כל דוגמה שהופיעה במאמר.
   - דוגמה אחת לכל מקרה מעניין שהאלגוריתם מטפל בו.
   - דוגמאות מורכבות יותר – יותר שחקנים, פונקציות ערך מורכבות יותר, וכד'.

כדי לקבל רעיונות, ראו בקבצים cut_and_choose_demo.py, last_diminisher_demo.py.

שלב ד: הגשה

8. להגיש את המימוש שלכם ב- pull request.

9. להתייחס להערות שאכתוב לכם (אם יהיו), לתקן לפי הצורך ולהגיש שוב.

10. לקבל 100 :)