# עקרונות שפות תוכנה – עבודת הגשה מספר 3

עבודה זו מורכבת משלוש תוכניות (קצרות ופשוטות יחסית) שתכתבו כך שהפלט של התוכנית הראשונה הוא הקלט של התוכנית השניה והפלט של התוכנית השניה הוא הקלט של התוכנית השלישית. הרעיון הוא לשרשר את התוכניות יחדיו באופן של conventional interface כדי לקבל מערכת גדולה יותר.

בהגשת העבודה יש להגיש כל תוכנית כקובץ פייתון נפרד (תיבת ההגשה תקבל מספר קבצים). בשם הקובץ יש לציין את המספור של התוכנית.

(Example: HW3_1_Joe_Doe.py)

כל תוכנית תיבדק בנפרד מול קלט שמתאים לה. עם זאת, מומלץ שתבדקו את כל התוכניות ביחד כמערכת כדי לוודא שהכל עובד תקין.

חלק מהדברים שתידרשו לעשות בתרגיל ייתכן ולא למדתם בהרצאה/תרגול – לדוגמה לקרוא מקובץ או לכתוב לקובץ. לכן מצופה ממכם לחיפוש באינטרנט (או כל מקור אחר).

## תוכנית ראשונה

קלט התוכנית הוא מספר קבצי טקסט (יש למצוא את הקבצים האלה בתיבת ההגשה):

colors.txt, fruit.txt, cities.txt, names.txt

כל קובץ טקסט מכיל מילים מסוימות המופרדות על ידי רווחים/שורות חדשות אשר נמצאות באותה קטגוריה – צבעים, פירות, ערים ושמות. לדוגמה המילה black תופיע בקובץ colors.txt. ייתכנו מילים שנמצאות ביותר מקטגוריה אחת, לדוגמה orange תימצא גם בfruit.txt וגם בcolors.txt.

עליכם ליצור **מילון** (dictionary) שהמפתח שלו הוא מילה מסוימת והערך שלו הוא **קבוצה** (set) של קטגוריות. אתם בונים את המילון בהתאם לקבצים הנ"ל.

לדוגמה עבור המפתח orange הערך המתאים יהיה {colors,fruit}

כמו כן, שימו לב שהמילון יכיל רק מילים עם אותיות קטנות וכן התוכנית איננה רגישה לאותיות גדולות/קטנות – לדוגמה Orange וorange הן מילים שקולות.

דבר נוסף, אין להכניס מילים שמכילות תווים שהם אינם אותיות אנגליות לתוך המילון. לדוגמה Tel-Aviv מכילה מקו ועל כן לא תיכנס למילון.

הקפידו על בנייה נכונה של סדר הפעולות בבניית המילון ועשו שימוש יעיל בכלים המובנים בפייתון (filter,map,reduce) וכו'.

לאחר בניית המילון יש לשמור את ה**ייצוג** (repr) שלו בקובץ טקסט שיקרא **dictionary.txt**. שימו לב שrepr היא פונקציה שמחזירה מחרוזת שאם נפעיל עליה את הפונקציה **eval** נקבל בחזרה עותק של האובייקט המתאים.

## תוכנית שנייה

קלט התוכנית הוא קובץ טקסט מסוים בשם **text.txt** שמגיע כחלק מתיבת ההגשה וכן המילון שיצרנו בתוכנית הקודמת **dictionary.txt**.

התוכנית קודם כל תחלץ את המילים מתוך קובץ הטקסט (שיכול להכיל כל דבר – ספרות, סימני פיסוק וסימנים כאלה ואחרים). כל מילה היא רצף של תווים (מחרוזת) שמכילה אותיות אנגליות קטנות בלבד. יש לזרוק מילים שמכילות ספרות וסימנים אחרים וכן להמיר מילים לאותיות קטנות בלבד. עם זאת, אם מילה מתחילה או מסתיימת בסימן פיסוק כלשהו, היא תיחשב כמילה, רק צריך לשים לב לא להכיל בתוכה את סימני הפיסוק.

הניחו כי סימני הפיסוק יכולים להיות רק פסיק ',' נקודה '.' סימן שאלה '?' וסימן קריאה '!'

לדוגמה, עבור הטקסט:

"Hello and w3lcome, please take a seat."

המחרוזת שתחולץ היא:

"hello and please take a seat"

לאחר ניקוי הטקסט, התוכנית תיצור **רשימה** ממוינת של **טאפלים** עם שלושה איברים כך שכל טאפל מורכב מ:

1. מילה
2. כמות הפעמים שאותה מילה מופיעה בטקסט.
3. קבוצת הקטגוריות – במידה והמילה לא שייכת לאף קטגוריה, אז קבוצה זו תהיה קבוצה ריקה (ולא None).

את הטאפלים יש למיין **בסדר יורד** (reverse = True) לפי הקריטריונים הבאים:

1. לפי כמות הפעמים שאותה מילה מופיעה בטקסט.
2. לאחר מכן לפי מיון לקסיקוגרפי על המילה.

את הייצוג של הרשימה יש לשמור בקובץ טקסט בשם **words.txt**.

שימו לב שאסור שמילה מסוימת תופיע ביותר מטאפל אחד!

תוכנית שלישית

התוכנית תקבל כקלט את הטאפלים מהקובץ words.txt וקודם כל תסנן את הטאפלים כך שהטאפל חייב לקיים **לפחות אחד** מהתנאים הבאים:

1. מילים שמופיעות לפחות חמש פעמים.
2. מילים שנמצאות ביותר מקטגוריה אחת.

לאחר הסינון, התוכנית **תדפיס** כל מילה שעברה את הסינון תוך ציון למה היא עברה את הסינון (תחשבו טוב איך ניתן להימנע מהתניות בקוד בשביל לממש את זה).

לדוגמה:

"'orange' because it belongs to at least two categories"

"'a' because it appears at least five times"

"'black' because it belongs to at least two categories and because it appears at least five times"

הנחיות כלליות וטיפים:

- כל שאלה הקשורה לתרגיל יש להפנות למתרגל האחראי על התרגיל – אביהוא אילן – AvihooI@Gmail.com.
- הניקוד מתחלק שווה בשווה בין שלושת התוכניות.
- התרגיל ייבדק באופן אוטומטי על ידי סקריפט בדיקה וכל תוכנית תיבדק באופן עצמאי (כלומר, תוך שימוש בקלט מוכן מראש עבור כל תוכנית ולא על ידי שרשור קלטים). לכן זה קריטי שתבדקו את התוכנית שלכם בנפרד מהשנייה ורק בסוף תבדקו את כל המערכת שלכם במלואה מול הקלט המקורי.
- הפלט של תוכנית 3 ייבדק ידנית – אבל תקפידו על ניסוח קריא של ההדפסה.
- במידה וסקריפט הבדיקה נכשל על שגיאת הרצה, הניקוד על אותה תוכנית יהיה 0. במידה והוא נכשל בעקבות פלט שלא תואם את הפלט המצופה – תתבצע השוואה ידנית בין הפלט המצופה לפלט של התוכנית ויורדו נקודות בהתאם למהות ההבדל. לכן הכי טוב שתקפידו לעשות בדיוק מה שנדרש – לא פחות ולא יותר.
- ניתן להניח שהקלטים תקינים ברמת הrepr והeval והפורמט של טיפוסי הנתונים. כלומר, אין צורך לבדוק את תקינות הפורמט.
- גם אם התוכניות עוברות את הבדיקה – יורדו נקודות על קוד בלתי ייעיל ובלתי קריא במיוחד. זה גם לטובתכם וחוסך לכם זמן לצמצם ולייעל את הקוד ולעשות שימוש בכלי conventional interface שלמדתם.
- את העבודה ניתן (ואף מומלץ) להגיש בזוגות. בן זוג אחד בלבד מגיש את העבודה במעטפה במודל. צריך לציין את שמות הסטודנטים שהשתתפו בעבודה בראש כל קובץ פייתון.
- כפי שכבר צוין למעלה - יש להשתמש בreprו eval בשביל להעביר טיפוסי נתונים בין תוכנית לתוכנית (ע"י קובץ). זאת הדרך הכי פשוטה והכי יעילה ובמידה ויצרתם הכל כפי שנדרש, היא מבטיחה שהבדיקה תעבור בהצלחה.
- בשביל תוכנית 2, חפשו בגוגל איך להשתמש בפונקציה sort של list.

קלטים ופלטים לדוגמה:

$cities.txt$:Toronto Brussels Zurich

$colors.txt$:Red Green Blue Orange

$fruit.txt$:Hazelnut Coconut Orange

$names.txt$: Jesse James Harry Potter

הקלטים הנ"ל ייגררו את הפלט הבא:

$dictionary.txt$:{'zurich': {'cities'}, 'brussels': {'cities'}, 'toronto': {'cities'},

'coconut': {'fruit'}, 'hazelnut': {'fruit'}, 'orange': {'fruit', 'colors'}, 'james': {'names'},

'jesse': {'names'}, 'harry': {'names'}, 'potter': {'names'}, 'red': {'colors'}, 'green': {'colors'}, 'blue': {'colors'}}

ועבור קובץ הטקסט הנ"ל והמילון שנוצר נקבל (שימו לב לאופן שבו הרשימה ממוינת):

$words.txt$:[('orange', 1, {'fruit', 'colors'}), ('loves', 1, {}), ('jesse', 1, {'names'})]

בהצלחה 😊