תרגיל 10

מבוא

- בתרגיל זה תממשו מערכת לסריקה מערכת קבצים ובדיקת מופעי מילים בקבצים.
  - המשימה הראשונה בתרגיל היא מימוש המחלקה WordExtractor אשר נועדה לחלץ את כל המילים המופיעות בקובץ נתון .
  - המשימה השנייה היא מימוש המחלקה WordTracker אשר מאפשרת ביצוע מעקב על הופעות של מילים (לדוגמה בתוך קובץ) מתוך רשימת מילים קבועה.
  - המשימה השלישית מממשת מעבר רקורסיבי על מערכת קבצים (תיקיות והתוכן שלהן); בין השאר, במטרה למצוא קובץ (שאת מילותיו יש לחלץ בעזרת WordExtractor) אשר מכיל רשימה מוגדרת של מילים (את המעקב אחר אילו מילים מופיעות יש לממש בעזרת WordTracker).
- בכל פונקציה בה לא מאופיין הקלט של הפונקציה - הפונקציה לא מקבלת קלט.
- בכל פונקציה בה לא מאופיין הפלט של הפונקציה - הפונקציה לא צריכה להחזיר פלט באופן מפורש. באופן בלתי מפורש (ללא שתכתבו דבר), תחזיר הפונקציה None (כי כך ממומשות פונקציות ב – Python).
- מימושי האובייקטים והפונקציות בתרגיל צריכים להיות יעילים.
- הפונקציות השונות המוגדרות לכל מחלקה משפיעות על העיצוב של המחלקה (אילו פונקציות עזר ומשתנים על המחלקה להחזיק). מומלץ טרם תחילת בניית המחלקה לקרוא ביסודיות את כל הדרישות, להגדיר היטב את העיצוב ורק אז לגשת למימוש.
- בכל פונקציה אשר עושה שימוש בקבצים, הקפידו לשחרר את משאב הקובץ לאחר סיום השימוש בו.
- אין לעשות שימוש בפונקציות מיון מובנות של פייתון.
- עדכונים לתרגיל –
  - מהותיים (במידה ויהיו) יפורסמו בפורום ההודעות באתר.
  - שוטפים – שאלות חוזרות בפורום או הבהרות יופיעו כמסמך מסודר בכתובת הזו. המעקב אחרי עמוד זה אינו מחייב אך נועד להקל (למעוניינים) על קריאה מקיפה של הפורום.

משימה 1 – מחלץ המילים

- במשימה זו תממשו את המחלקה WordExtractor. קובץ השלד WordExtractor מכיל את הגדרות המחלקה והפונקציות שלה.
- בעת אתחול המחלקה יוגדר לה קובץ עבודה (***קובץ המקור***) והמחלקה תמשש פונקציונאליות של איטראטור המחזיר את כל המילים בקובץ.
- במימוש עליכם למלא את הקוד של הפונקציות המפורטות להלן. אתם רשאים להוסיף למחלקה פונקציות ו - data members כראות עיניכם אך לא לא לשנות את ההגדרות הקיימות.
- על מימושי הפונקציות להיות יעילים (מבחינת זמן הריצה שלהם) ככל הניתן.
- באופן כללי יש להניח כי גודל הקבצים עשוי להיות גדול מאוד, לכן **אין לשמור את תוכן כל הקובץ** במחלקה. באופן פורמאלי, דרישות סיבוכיות המקום למימוש המחלקה צריכות להיות ב - O(1) כלומר, לתפוס מקום קבוע בזיכרון שהוא קטן מקבוע כלשהו, ללא קשר לגודל הקובץ.

- לצורך תכנון סיבוכיות המקום של המימוש, ניתן להניח כי גודל כל המילים וגודל כל השורות בכל קובץ הנו קטן מקבוע מסויים. כלומר, אין בעייה מבחינת דרישת הסיבוכיות לשמור מילים או שורות שלמות בזיכרון (אבל יש רבות מהן לכן אי אפשר לשמור את כולן).

1. קונסטרקטור המחלקה WordExtractor

a. <u>מקבל</u>

- כפרטמר מחרוזת המציינת את מיקומו האבסולוטי או היחסי (ביחס לתיקייה ממנה מורץ הקובץ) של ***קובץ המקור***.
- הנחות על הקלט :
  1. אפשר להניח כי ***קובץ המקור*** קיים במקום הנתון.

2. הפונקציה __iter__

a. <u>מחזירה</u>

- את האובייקט עצמו (self).

3. הפונקציה __next__

a. <u>המימוש</u>

- בכל קריאה לפונקציה מוחזרת מילה יחידה מתוך הקובץ. מילה מוגדרת כרצף תווים (מחרוזת) המופרדת מרצף תווים אחר על ידי מספר חיובי כלשהו של רווחים או על ידי תחילת או סוף שורה.
- לדוגמה המילים בקובץ שתוכנו הוא (התו 'X' מסמלת תחילת שורה והתוים 'XX' סוף שורה):

```
X   Are -  you pondering      whatXX
XI am    pondering pinky?  XX
```

- הן:

['Are','-','you','pondering','what','I','am','pondering','pinky?']

a. אין צורך להתחשב בצורה מיוחדת בתווים שאינם אלפאביתיים (לדוגמה סימני פיסוק או מספרים), ויש להפריד בין מילים על פי רווחים בלבד (סימנים מיוחדים צריכים להכלל בתוך המילה).
b. בפרט, מילה יכולה להיות מורכבת מתווים מיוחדים (לדוגמה מספרים או סימני פיסוק) בלבד.

b. <u>מחזירה</u>

- מילה יחידה מתוך הקובץ.

1. סדר המילים המוחזרות על ידי הפונקציה אינו בעל חשיבות. בפרט, אפשר להחזיר מילים לפי סדר הופעתן בקובץ.
2. קריאות חוזרות לפונקציה צריכות להחזיר כל **מופע** של מילה פעם אחת לכל היותר. לדוגמה עבור קובץ הדוגמה שלעיל,

קריאות חוזרות צריכות להחזיר 9 מילים : כיוון והמילה 'pondering' מופיעה פעמיים, יש להחזיר אותה פעמיים.

3. אחרי שכל מופע של מילה בקובץ הוחזר, כל קריאה לפונקציה צריכה להעלות את הפסיקה StopIteration (כמוסבר בשיעור – שקף 7) על ידי שימוש בפקודה :

```
raise StopIteration
```

משימה 2 – מופעי המילים

- במשימה זו תממשו את המחלקה **WordTracker**. קובץ השלד **WordTracker** מכיל את הגדרות המחלקה והפונקציות שלה.
- במימוש עליכם למלא את הקוד של הפונקציות המפורטות להלן. אתם רשאים להוסיף למחלקה פונקציות ו - data members כראות עיניכם אך לא לא לשנות את ההגדרות הקיימות.
- על מימושי הפונקציות להיות יעילים (מבחינת זמן הריצה שלהם) ככל הניתן.
- מחלקה מוגדרת על רשימת מילים (נכנה – מילון) ומתחזקת מערכת ייצוג להופעות של המילים ברשימה. אחד השימושים של המחלקה הזו הוא לעקוב אחרי מופעים של מילים מקובץ חיצוני כדי לבדוק האם כל המילים המופיעות במילון נמצאות בקובץ זה.
  - שימו לב, רשימת המילים מכונה מילון אך היא אינה קשורה לאובייקט dictionary (אשר לא נלמד עדיין) של Python.

4. קונסטרקטור המחלקה WordTracker

a. <u>מקבל</u>

- רשימת מילים שתשמש בכל קריאה נוספת לפונקציה כמילון.
  - אפשר להניח כי ברשימה אין כפילויות. כלומר, כל מילה מופיעה לכל היותר פעם אחת ברשימה.

b. <u>המימוש</u>

- רשימת המילים, כפי שהיא מתקבלת בקלט, צריכה היא והיא בלבד לשמש כמילון המחלקה. כאשר מועברת הרשימה למחלקה מועבר *מצביע* לרשימה, כך שאם הרשימה תשתנה (מחוץ למחלקה, על ידי מצביע אחר אליה) תשתנה גם רשימת המחלקה (יש למעשה רק רשימה אחת ושני מצביעים אליה). הדרישה במימוש היא ליצור **עותק** של רשימת הקלט כך ששינויים שייעשו מבחוץ לרשימת הקלט לא ישפיעו על המחלקה.

5. הפונקציה __contains__

a. <u>מקבלת</u>

- מחרוזת.

b. <u>המימוש</u>

- השימוש המיועד במחלקה WordTracker עושה שימוש רב בפונקציה זו עבור אותו המילון, לפיכך על המימוש להיות יעיל מאוד מבחינת זמן הריצה. עבור – n מילים במילון כל קריאה לפונקציה צריכה לרוץ במקרה הגרוע בסיבוכיות של O(log(n)).

- על מנת לעמוד בהגבלת סיבוכיות הזו, יש לממש חיפוש בינארי של מחרוזת הקלט במילון המחלקה. החיפוש הבינארי צריך להיות ממומש על ידיכם, אין להשתמש בפונקציות מובנות של פייתון.

c. <u>מחזירה</u>
- True – אם מילת הקלט מופיעה במילון;
- False אחרת.

d. הוסיפו בקובץ ה - **README** אם מימוש פונקציה זו הביא להחלטות נוספות בעיצוב המחלקה (לדוגמה פעולות נוספות בפונקציות אחרות) ציינו מה הן ולמה בחרתן במימוש הזה.

הערה לבחירת שם הפונקציה הנ"ל : שם הפונקציה __contains__ מאפשר שימוש במילה השמורה **<u>in</u>** בקוד לצורך בדיקת שייכות, בדומה לבדיקה המוקרת לנו לשייכות ברשימה (אין חובה לעשות שימוש בפונקציונאליות זו) :

```
lst = ['1','2','3']
if '2' in lst:
        print('2 is in the list')
```

```
wt = WordTracker(lst)
if '2' in wt:
        print('2 is in the wordtracker')
```

6. הפונקציה encounter

a. <u>מקבלת</u>
- מחרוזת.

b. <u>המימוש</u>
- אם מחרוזת הקלט נמצאת במילון, המימוש משנה את הייצוג הפנימי של המחלקה כדי "לזכור" שמחרוזת הקלט הופיעה בשלב כלשהו (לדוגמה לצורך בדיקה מאוחרת האם כל המילים במילון הופיעו בקובץ חיצוני כלשהו).

c. <u>תחזיר</u>
- True אם מילת הקלט נמצאת ברשימת המילים;
- False אחרת.

d. הוסיפו בקובץ ה - **README** מהו זמן הריצה של הפונקציה (ציינו בפירוש מה הם הפרמטרים שקובעים זמן ריצה זה).

7. הפונקציה encountered_all

a. <u>מחזירה</u>
- True אם כל המילים במילון "נראו" (כלומר לכל מילה במילון הפונקציה encounter נקראה עם מילה זו לפחות פעם אחת);
- False אחרת.

b. הוסיפו בקובץ ה - **README** מהו זמן הריצה של הפונקציה (ציינו בפירוש מה הם הפרמטרים שקובעים זמן ריצה זה).

8. הפונקציה reset

a. <u>המימוש</u>

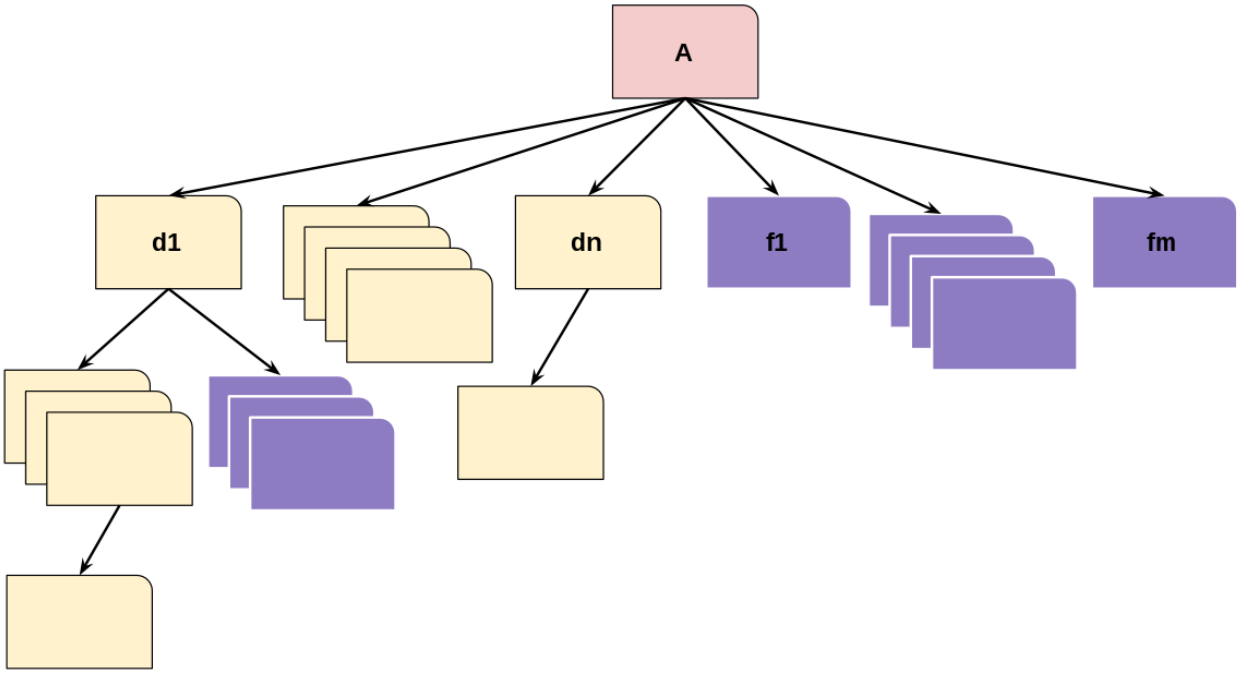
- משנה את הייצוג הפנימי של המחלקה כך שהמחלקה "שוכחת" את כל המילים שנראו עד עתה. כך לדוגמה, לאחר הקריאה לפונקציה זו לא משנה מה היא היסטוריית הקריאה ל – encounter, על מנת שהפונקציה encountered_all תחזיר True, לכל מילה במילון יש לקרוא לפונקציה encounter עם מילה זו.

b. הוסיפו בקובץ ה - **README** מהו זמן הריצה של הפונקציה (ציינו בפירוש מה הם הפרמטרים שקובעים זמן ריצה זה).

משימה 3 - מעבר על מערכת קבצים

- משימה זו תמומש בקובץ **PathScanner.py**. קובץ השלד **PathScanner.py** מכיל את חתימות והגדרות הפונקציות שלהלן.
- במימוש עליכם למלא את הקוד של הפונקציות המפורטות להלן. אתם רשאים להוסיף פונקציות ומחלקות כראות עיניכם אך לא לא לשנות את ההגדרות הקיימות.
- על מימושי הפונקציות להיות יעילים (מבחינת זמן הריצה שלהם) ככל הניתן.

- באופן כללי, פונקציות המחלקה מקבלות נתיב אבסולוטי או יחסי (נכנה - הנתיב), עוברות **<u>בצורה רקורסיבית</u>** על כל הקבצים והתיקיות המופיעים תחת נתיב זה עד ל*<u>עומקם המלא</u>* (ראו הסבר להלן) ובהתאם להגדרות הפונקציות מבצעות עבור כל קובץ פעולה מסויימת.
- עבור כל המימוש בשאלה זו, ניתן להניח כי במערכת הקבצים לא קיימות "לולאות". כלומר לכל תיקייה A, על ידי כניסה לתת התיקיות של - A ותת התיקיות שלה ותתי התיקיות שלהן וכן הלאה, לעולם לא נגיע ל - A.
- הגדרת *<u>עומק מלא</u>* של נתיב :
  - יהא נתיב המצביע לקובץ או תיקייה A.
  - אם A היא תיקייה אזי נניח כי היא המכילה את התיקיות $d1 \ldots dn$ והקבצים $f1 \ldots fm$ (עבור n ו - m שלמים אי-שליליים).
  - התיקייה d1 מכילה גם היא את התיקיות $d1_1 \ldots d1_n$ והקבצים $f1_1 \ldots fm_1$ ובאופן דומה כך גם שאר התיקיות $d2 \ldots dn$ וכך גם כל התיקיות המוכלות ב – d1 וכל תתי התיקיות המוכלות בהן וכן הלאה.
  - כל הקבצים ותת התיקיות של A ותתי התיקיות של תת התיקיות של - A נכללות *<u>בעומק המלא</u>* של הנתיב של A.
  - <u>באופן פורמאלי</u> נגדיר בצורה רקורסיבית כי ה*<u>עומק המלא</u>* של נתיב A מכיל את:
    - התיקיות והקבצים הנמצאים ב - A.
    - התיקיות והקבצים הנמצאים ב*<u>עומק המלא</u>* של התיקיות הנמצאות A.
    - רפלקסיביות : *<u>עומק מלא</u>* הוא יחס לא רלפקסיבי, כלומר תיקייה וקובץ לא נמצאים ב*<u>עומק המלא</u>* של עצמם.
  - כל הקבצים והתיקיות המופיעים באיור שלהלן (מלבד התיקייה - A), הם ב*<u>עומק המלא</u>* של התיקייה - A.



9. הפונקציה path_iterator

a. <u>מקבלת</u>

- נתיב אבסולוטי או יחסי.

1. ניתן להניח כי הנתיב מתייחס לתיקייה קיימת.

b. <u>מחזירה</u>

- אובייקט מסוג PathIterator:

1. שלד המחלקה PathIterator אך לא את הפונקציות שלה (עליכם לממש אותן).
   a. שימו לב להגדרות שניתנו בשיעור (שקף 7) על הפונקציות המגדירות איטרטור.
2. אובייקט ה – PathIterator המוחזר הנו איטראטור אשר צריך לאפשר איטראציה על כל הקבצים והתיקיות בנתיב הנתון (שימו לב – בנתיב בלבד, ולא ב*עומק המלא*) על ידי החזרת הנתיב המלא שלהם.
   a. אין חשיבות לסדר הקבצים והתיקיות המוחזרים על ידי האיטראטור.
3. לדוגמה הפלט עבור קטע הקוד הבא, עבור מערכת הקבצים הנתונה לדוגמה בסעיף הקודם :

```
it = path_iterator ('/A/F')
for e in it:
        print(e)
/A/F/G

/A/F/H

/A/F/I

/A/F/J
```

10. הפונקציה print_tree

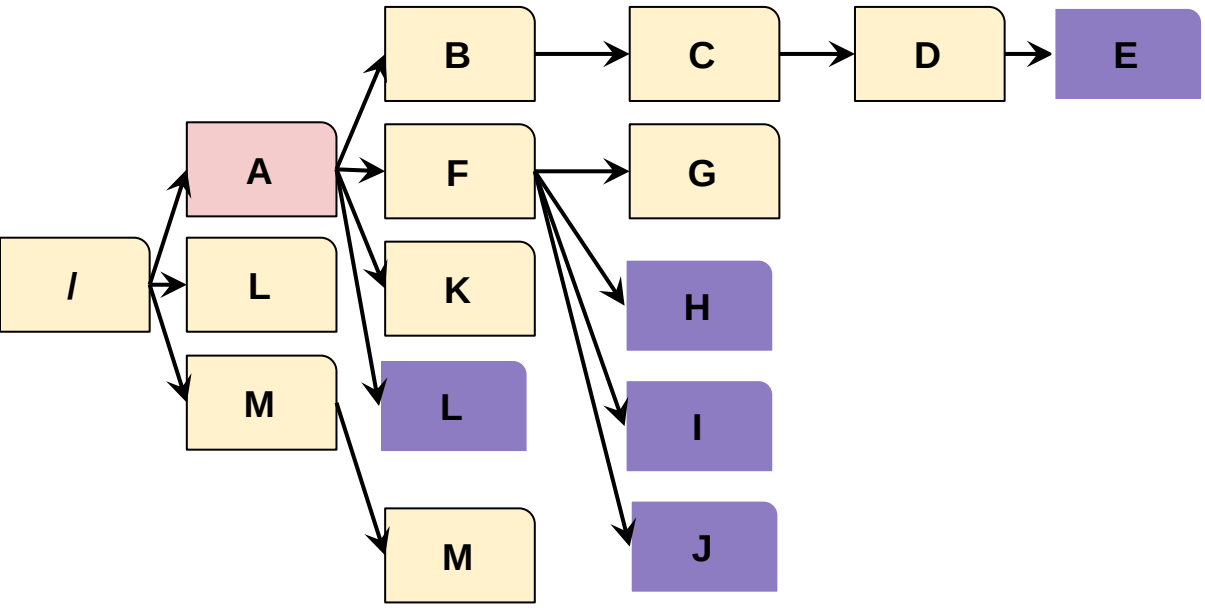
c. <u>מקבלת</u>

- נתיב אבסולוטי או יחסי.
  1. ניתן להניח כי הנתיב מתייחס לתיקייה קיימת.
- מפריד ההיררכייה : פרמטר אופציונאלי המגדיר את פורמאט ההדפסה של ה<u>*עומק המלא*</u> של הנתיב.
  1. ערך ברירת המחדל של מפריד ההיררכייה הוא שני רווחים.

d. <u>המימוש</u>

- ידפיס את הנתיב וכל הקבצים והתיקיות ב<u>*עומק המלא*</u> של הנתיב כך שכל קובץ ותיקייה מודפסים בשורה משל עצמם, בשמם בלבד, כלומר **ללא השורש שלהם** (הנתיב האבסולוטי/יחסי שלהם), ועם תחילית מתאימה.

1. התחילית של נתיב הקלט (בדוגמה להלן /A) היא **מחרוזת ריקה**.
2. התחילית של כל קובץ או תיקייה הוא התחילית של התיקייה בה הם מוכלים בתוספת מפריד ההיררכייה.
3. עבור כל זוג A ו - B (זוג קבצים, זוג תיקיות או קובץ ותיקייה) המוכלים באותה תיקייה :
   a. אין חשיבות לסדר ההדפסה של A ו - B.
   b. אם A ו - B תיקיות. נניח כי בלי הגבלת הכלליות A מודפסת לפני B, כל ה<u>*עומק המלא*</u> של A צריך להיות מודפס לפני B וכל הפרטים ב<u>*עומק המלא*</u> שלה.
4. לדוגמה עבור מפריד ההיררכייה '__', מערכת הקבצים שבאיור (חץ מסמל את היחס הכלה), והנתיב/A , ההדפסה למסך צריכה להיות :




```
A
__B
____C
______D
________E
__F
____G
____H
____I
____J
__K
__L
```

11. הפונקציה file_with_all_words

e. <u>מקבלת</u>

- נתיב אבסולוטי או יחסי.
  1. ניתן להניח כי הנתיב מתייחס לתיקייה קיימת.
- רשימת מילים.

f. <u>המימוש</u>

- עובר על ה<u>*עומק המלא*</u> של הנתיב הנתון ומחפש קובץ אשר מכיל את כל המילים ברשימת המילים הנתונה.
- אם נמצא קובץ אשר מכיל את כל המילים יש להחזיר את הנתיב אליו ולסיים את החיפוש. בפרט :
  1. אין צורך להמשיך לסרוק את מערכת הקבצים.
  2. אם נמצא יותר מקובץ אחד המקיים את הדרישה ב<u>*עומק המלא*</u> של הנתיב הנתון, יש להחזיר אחד בלבד ולא משנה איזה מהקבצים המקיים את הדרישה הוא זה המוחזר.
- המימוש צריך לעשות שימוש ב**רקורסיה** כדי לעבור על מערכת הקבצים.
- המימוש צריך לעשות שימוש במחלקות שהוגדרו בתרגיל:
  1. WordExtractor כדי לחלץ את המילים מכל קובץ אשר פוגשים במערכת הקבצים.
  2. WordTracker כדי לבצע מעקב אחר מופעי המילים בקובץ ביחס לרשימת המילים הנתנת כקלט לפונקציה זו.

g. <u>מחזירה</u>

- את הנתיב לקובץ כלשהו תחת הנתיב הנתון אשר מכיל את כל המילים בקלט ; או
- None אם לא קיים קובץ כזה.

פונקציות שימושיות לתרגיל

קריאה/כתיבה לקובץ

להסברים המלאים, ראו <u>תרגול 10</u> (שקפים 5-27).

הפקודה open :

```
f = open(file_name)
```

מקבלת כמשתנה (file_name) נתיב יחסי או אבסולוטי ומחזירה (בדוגמה, לתוך f) אובייקט קובץ.

פתיחת קובץ נעשית תמיד במצב מוגדר של קריאה/כתיבה/גם וגם. על מנת לציין את מצב הפתיחה הרצוי לקובץ יש להוסיף לקריאה ל - open מחרוזת המציינת את מצב פתיחת הקובץ:

- f = open(file_name,'r') : Read only (default mode)
- f = open(file_name,'w') : Write only (overwrite)
- f = open(file_name,'a') : Write only (append)
- f = open(file_name,'r+') : Read and write

מתוך קובץ הפתוח לקריאה, ניתן לקרוא את כל תוכנו שורה אחרי שורה בעזרת הפקודות הבאות:

```
with open(file_name) as f:

    current_line = f.readline()
```

פונקציות על מערכת קבצים

המודול os (לתיעוד רלוונטי ראו כאן וכאן) מציע פונקציות רבות ושימושיות לעבודה עם קבצים. בפרט:

- os.**listdir**(*path='.'*)¶ : Return a (randomly ordered) list of all the items contained in the specified path.
- os.path.**isfile**(*path*)¶ :Return True if the given path is an existing file and false otherwise.
- os.path.**isdir**(***path***)¶ Return True if the given path is an existing directory and false otherwise.

על מנת להשתמש במודול יש לייבא אותו (import os).

הפרדת מחרוזת למילים בודדות

הפונקציה split פועלת על מחרוזת ומחזירה את רשימת המילים מתוכה, כלומר מפרידה רצף תווים למספר רצפים על פי מפריד קבוע (ברירת המחדל היא התו רווח).

לדוגמה :

```
>>'1 2 3'.split()

['1','2','3']
```

הוראות הגשה

יש להגיש קובץ tar יחיד המכיל את שלושת קבצי השלד עם מימושם ואת קובץ ה – README.