בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

מבוא למדעי המחשב 67101

תרגיל 7 - רקורסיה

להגשה בתאריך 5/12/2018 בשעה 22:00

בתרגיל זה נתרגל מבני רקורסיה שונים. התרגיל מורכב ממספר משימות בלתי תלויות.

שימו לב: בפתרון תרגיל זה אין לעשות שימוש באף מודול חיצוני של **python** – כלומר, אין לעשות import לאף מודול לצורך הפתרון! בפרט, אין לעשות שימוש במודול math או במודול itertools.

חלק ראשון: רקורסיה פשוטה

את המשימות בחלק זה יש לפתור ע"י שימוש בפונקציות רקורסיביות, ללא שימוש בלולאות מכל סוג שהוא (גם לא בעקיפין)!

1. **הפונקציה print_to_n(n)**
עליכם לממש את הפונקציה print_to_n, המקבלת את המספר n (int) , ומדפיסה את המספרים (השלמים) מ-1 עד n בסדר עולה (הכוונה כל מספר בשורה נפרדת).
במקרה של קלט קטן מ-1 אין להדפיס דבר.

2. **הפונקציה print_reversed(n)**
עליכם לממש את הפונקציה print_reversed_n, המקבלת את המספר n (int), ומדפיסה את המספרים (השלמים) מ-n עד 1 בסדר יורד.
במקרה של קלט קטן מ-1 אין להדפיס דבר.

3. **הפונקציה is_prime(n)**
עליכם לממש את הפונקציה is_prime, המקבלת את המספר n (int), ומחזירה True אם הוא ראשוני ו- False אחרת. **מספר ראשוני הוא שלם גדול מ- 1**, שהשלמים היחידים המחלקים אותו ללא שארית הם 1 והמספר עצמו.
רמז: תוכלו לממש ולהיעזר בפונקציה has_divisor_smaller_than(n, i), הבודקת האם ל- n מחלק (שונה מ- 1) קטן מ- i.
הערה: מכאן ואילך עליכם להבין בעצמכם האם נדרשות פונקציות עזר ואילו.

4. **הפונקציה exp_n_x(n, x)**
עליכם לממש את הפונקציה exp_n_x, המקבלת את המספר n (int **אי שלילי**), והמספר x (float), ומחזירה את $exp_n(x)$ - פונקציית הסכום האקספוננציאלי, כאשר:

$$exp_n(x) = \sum_{i=0}^{n} \frac{x^i}{i!} \approx e^x$$

חלק שני: רקורסיה מתקדמת

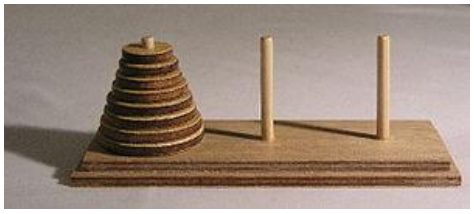
(לצורך פתרון המשימות בחלק זה, תוכלו להיעזר גם בלולאות)

5. הפונקציה play_hanoi(hanoi, n, src, dest, temp)

עליכם לממש את הפונקציה play_hanoi, הפותרת משחק "מגדלי הנוי".
משחק "מגדלי הנוי" כולל:

- שלושה מוטות אנכיים ("המגדלים").
- מספר דיסקיות בגדלים שונים שניתן להשחיל על המוטות, כאשר כל דיסקית - בגודל שונה.

בתחילת המשחק, הדיסקיות מסודרות על פי גודלן על אחד המוטות, כשהגדולה ביותר למטה והקטנה ביותר למעלה. מטרת המשחק היא להעביר את כל הדיסקיות ממוט זה אל אחד משני המוטות הנותרים, בכפוף לשני חוקים:



- מותר להזיז רק דיסקית אחת בכל פעם - מראש מוט אחד לראש מוט אחר.
- אסור להניח דיסקית אחת על דיסקית שקטנה ממנה.

לקריאה והסברים נוספים:
מגדלי האנוי

הפונקציה נקראת עם הפרמטרים הבאים:
hanoi – אובייקט מורכב שהוא המשחק הגרפי בו מתבצע השינוי (אותו יש להעביר בכל קריאה לפונקציה כפרמטר הראשון).
n – מספר (int) הדיסקיות אותן על הפונקציה להעביר.
src – אובייקט מורכב המייצג את המוט ממנו מעוניינים להעביר את הדיסקיות.
dest – אובייקט מורכב המייצג את המוט אליו מעוניינים להעביר את הדיסקיות.
temp – אובייקט מורכב המייצג את המוט הנותר.

על מנת להעביר דיסקית במשחק hanoi ממוט למוט, יש להשתמש בפקודה:

```
hanoi.move(src, dest)
```

כאשר שני הפרמטרים הם אובייקטים המייצגים מוטות במשחק. שימו לב כי אם src הוא מוט ריק או שהדיסקית העליונה בו, גדולה יותר מהדיסקית העליונה במוט dest תקבלו שגיאה. אחרת, קריאה לפקודה זו תעביר את הדיסקית העליונה מ-src לראש המוט dest.

כדי לבדוק את הפונקציה שכתבתם ולראות מימוש גרפי של המשחק יש למקם את הקובץ hanoi_game.py(קובץ עזר שסיפקנו לכם במודל) באותה תיקייה שבה נמצא הקובץ ex7.py אותו אתם ממשים. הרצה של הקובץ (של hanoi_game.py) תפעיל את המשחק ותקרא לפונקציה שכתבתם. **שימו לב שאין לייבא את הקובץ hanoi_game.py לתוך הקוד שלכם!**

תוכלו להניח כי בזמן הקריאה הראשונית לפונקצייה מצב המשחק תקין (כלומר, ישנן בדיוק n דסקיות על המוט src מסודרות בצורה חוקית, ואין דסקיות על שאר המוטות). לא ניתן להניח דבר על מימוש האובייקטים המורכבים.
שימו לב! הקובץ hanoi_game.py יקרא לפונקציה שלכם רק עם ערכי n חיוביים. אך על הפונקציה שלכם להתמודד עם כל ערך שלם - עבור n שלילי, על הפונקציה להתנהג כאילו התקבל 0.

6. **הפונקציה print_sequences(char_list, n)**
עליכם לממש את הפונקציה print_sequences, המקבלת רשימה של תווים char_list, ומדפיסה את כל הצירופים האפשריים באורך n של תווים מהרשימה, כאשר אותו תו יכול להופיע יותר מפעם אחת.
תוכלו להניח כי מתקבלת רשימה חוקית ולא ריקה של תווים (char) שונים אחד מהשני, וכן כי n אי שלילי.

7. **הפונקציה print_no_repetition_sequences(char_list, n)**
עליכם לממש את הפונקציה print_no_repetition_sequences, המקבלת רשימה של תווים char_list, ומדפיסה את כל הצירופים האפשריים באורך n של תווים מהרשימה, ללא חזרות (כלומר, אותו תו לא יכול להופיע יותר מפעם אחת).
תוכלו להניח כי מתקבלת רשימה חוקית ולא ריקה של תווים (char) שונים אחד מהשני, כי len(char_list)≥n, וכן כי n אי שלילי.

8. **הפונקציה parentheses(n)**
אנו נגיד שבמחרוזת יש n זוגות חוקיים של סוגריים אם המחרוזת מכילה אך ורק את התווים '('ו-')' ואם רצף הסוגריים הזה היה יכול להיכתב כך בנוסחה מתמטית (כלומר, כל פתיחת סוגריים נסגרת וסוגריים לא נסגרים לפני שהם נפתחים). דרך יותר מדויקת לתאר זאת היא שבכל רישא של המחרוזת מספר התווים ')' גדול או שווה למספר התווים '(' - **ומספר התווים מכל סוג במחרוזת כולה שווה**.

כתבו פונקציה המקבלת מספר שלם חיובי n ומחזירה רשימה עם כל המחרוזות עם n זוגות חוקיים של סוגריים.

9. **הפונקציה up_and_right(n, k)**
אתם נמצאים על לוח משבצות במשבצת 0,0. ברצונכם להגיע למשבצת n,k (כלומר n צעדים ימינה ו-k צעדים למעלה). בכל צעד מותר לכם רק להתקדם צעד אחד ימינה או צעד אחד למעלה.

כתבו פונקציה המקבלת זוג מספרים n,k המדפיסה את כל הדרכים להגיע למשבצת n,k אך ורק בעזרת צעדים ימינה או למעלה. צעד למעלה ייוצג על ידי התו 'u' וצעד ימינה ייוצג על ידי התו 'r'. על כל מסלול להיות רצף של התווים u,r ויש להדפיס כל אחד בשורה נפרדת.

למשל, עבור המספר 2,1 ניתן ללכת שני צעדים ימינה ואז צעד אחד למעלה, צעד אחד למעלה ואז שני צעדים ימינה או צעד אחד ימינה, צעד אחד למעלה ואז צעד אחרון ימינה. לכן, במקרה זה הפונקציה תדפיס את השורות הבאות:

rru

urr

rur

10. **הפונקציה flood_fill(image, start)**
הפונקציה תקבל מערך דו מימדי image ובו רק התווים "*" ו-". ". התו "*" מסמל משבצת מלאה והתו "." מסמל משבצת ריקה. ניתן להניח שכל הרשימות בתוך image הן מאותו אורך וש"מסגרת התמונה" מלאה בתווים "*" (כלומר השורה הראשונה, השורה האחרונה, והסוף וההתחלה של כל שורה מלאים בתו "*").

כמו כן, הפונקציה תקבל מיקום במשתנה start שהינו tuple בו האיבר הראשון הוא אינדקס עבור שורה ב-image והאיבר השני הוא אינדקס עבור העמודה ב-image. כלומר, start מתייחס לתא

```
image[start[0]][start[1]]
```

ניתן גם להניח שתא זה מלא בתו "."

אנו נרצה להתחיל להפוך חלק מהתאים הריקים (המסומנים ב-".") לתאים מלאים (המסומנים ב-"*") באופן הבא: בכל רגע נתון, מותר לנו למלא תא ריק רק אם הוא סמוך לתא שהתמלא קודם לכן, או אם הוא התא המסומן ב-start. אנו נגדיר תאים כסמוכים רק אם הם מימין, משמאל, מעל או מתחת אחד לשני (ולא אם הם באלכסון). יש להבדיל בין תאים שהתחילו מלאים לבין תאים שהתמלאו תוך כדי - אין למלא תאים שסמוכים לתאים שהתחילו מלאים אם הם לא סמוכים לתא שהתמלא תוך כדי. ניתן לדמיין זאת כמים המתחילים לזרום בתוך מבוך וממלאים את כולו. באופן טבעי, משבצת שאינה סמוכה למים לא תתמלא במים בעצמה.

עליכם לכתוב פונקציה שמחליפה את התווים "." בתווים "*" החל מהנקודה start באופן המפורט לעיל. על הפונקציה לשנות את התמונה המקורית ולמלא בה את התו "*" במקומות הנכונים ולא להחזיר דבר. יש לציין כי יתמלאו רק התאים שאפשר להגיע אליהם מנקודת ההתחלה רק דרך תאים עם התו ".". לכן, אם נקודת ההתחלה נמצאת בתוך צורה המוקפת בכוכביות, כל יתר התאים לא יתמלאו בכוכביות. לדוגמא במערך הבא:

```
[['*','*','*','*','*'],
 ['*','.','*','.','*'],
 ['*','.','*','.','*'],
 ['*','*','*','*','*']]
```

עם נקודת ההתחלה (1,1), בסוף הריצה המערך ייראה כך:

```
[['*','*','*','*','*'],
 ['*','*','*','.','*'],
 ['*','*','*','.','*'],
 ['*','*','*','*','*']]
```

חלק שלישי: ניתוח זמני ריצה

כתבו בקובץ הרידמי את ניתוח זמני הריצה של הפונקציות הבאות:

- is_prime
- print_no_repetition_sequences
- flood_fill

נתחו את זמני הריצה ב-big O notation, כפי שנלמד בשיעור.

נהלי הגשה

הלינק להגשה של התרגיל הוא תחת השם: 7ex

בתרגיל זה עליכם להגיש את הקבצים הבאים:

1. ex7.py – עם המימושים שלכם לפונקציות.
2. README (על פי פורמט ה-README לדוגמא שיש באתר הקורס, ועל פי ההנחיות לכתיבת README המפורטות בקובץ נהלי הקורס).

יש להגיש קובץ zip הנקרא ex7.zip המכיל בדיוק את שני הקבצים הנ"ל.

בהצלחה