מבוא למדעי המחשב 67101 – סמסטר ב' 2020
תרגיל 7 – רקורסיה
להגשה בתאריך **10/12/2020** בשעה 22:00

הקדמה

בתרגיל זה נתרגל מבני רקורסיה שונים. התרגיל מורכב ממספר משימות בלתי תלויות. בתרגיל זה ניתן להניח שהקלט חוקי והגיוני אלא אם צוין מפורשות אחרת.

בתרגיל אתם נדרשים לבצע בדיקות טיפוסים בכל הפונקציות.

כלומר בהרצת הפקודה python3 -m mypy –strict filetocheck במחשבי בית הספר על הקוד, ההרצה צריכה לעבור בצורה תקינה וללא שגיאות. בנוסף, תהיה בדיקה שהטיפוסים שהגדרתם אינם רחבים מדי.

אין להשתמש ב-Any**,** אלא אם צוין אחרת.

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.

בנוסף, אם לא כתוב במפורש להחזיר ערך – אין להחזיר דבר.

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 (כולל n) בסדר עולה.

במקרה של קלט קטן מ-1 אין להדפיס דבר.

2. הפונקציה digit_sum(n)

עליכם לממש את הפונקציה digit_sum(n), המקבלת את המספר האי-שלילי n (int) ומחזירה את הסכום של הספרות של n. לדוגמא, עבור הקלט 479, הפלט יהיה הסכום של 4+7+9, כלומר 20.

אין להשתמש במחרוזות בסעיף זה.

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.

חלק שני: רקורסיה מתקדמת

לצורך פתרון המשימות בחלק זה, תוכלו להיעזר גם בלולאות.

בשאלה 4 **בלבד** מותר להשתמש ב-Any כשזהו הטיפוס המתאים.

4. הפונקציה play_hanoi(hanoi, n, src, dst, temp)

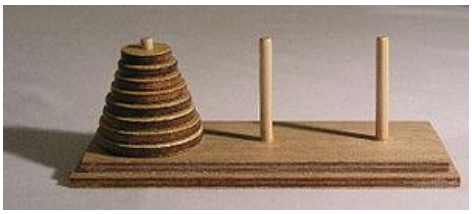
עליכם לממש את הפונקציה play_hanoi, הפותרת משחק "מגדלי הנוי".

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 / Tower of Hanoi / برج هانوي

כדי לבדוק את הפונקציה יש למקם את הקובץ hanoi_game.py באותה תיקייה שבה נמצא הקובץ ex7.py ולהריץ את hanoi_game.py. **שימו לב #1 אין לייבא את הקובץ hanoi_game לתוך הקוד שלכם!**

על מנת שהפונקציה אותה אתם כותבים תבצע שינויים במשחק הגרפי, הפונקציה נקראת עם הפרמטרים הבאים:

hanoi – אובייקט מורכב שהוא המשחק הגרפי בו מתבצע השינוי.

n – מספר (int) הדיסקיות אותן על הפונקציה להעביר.

src – אובייקט מורכב המייצג המוט ממנו מעוניינים להעביר את הדיסקיות.

dest – אובייקט מורכב המייצג את המוט אליו מעוניינים להעביר את הדיסקיות.

temp – אובייקט מורכב המייצג את המוט השלישי במשחק.

<u>שימו לב #2</u>: האובייקטים המורכבים ניתנים לכם בקריאה המקורית לפונקצייה play_hanoi שמתבצעת בקובץ hanoi_game.py.

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

hanoi.move(src, dest)

כאשר שני הפרמטרים הם מוטות במשחק. קריאה לפקודה זו תעביר את הדיסקית העליונה מהמוט src לראש המוט dest. שימו לב כי אם תנסו להזיז דיסקית ממוט ריק תקבלו שגיאה.

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.

<u>שימו לב #3</u>: 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.

5. הפונקציה print_sequences(char_list, n)

עליכם לממש את הפונקציה print_sequences, המקבלת מחרוזת של תווים char_list ומדפיסה את כל הצירופים האפשריים באורך n של תווים מהרשימה, כאשר אותו תו יכול להופיע יותר מפעם אחת. ניתן להדפיס את הצירופים בכל סדר שהוא.

תוכלו להניח כי מתקבלת רשימה חוקית (רשימה ריקה – גם היא חוקית) של תווים (char) שונים אחד מהשני, וכן כי n אי שלילי. לדוגמא, עבור הקלט ([“a”,”b”,”c”], 2), ההדפסה תהיה של הצירופים הבאים :

ba

bc

cc

ca

aa

ac

cb

ab

bb

6. 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). ניתן להדפיס את הצירופים בכל סדר שהוא.

תוכלו להניח כי מתקבלת רשימה חוקית (רשימה ריקה – גם היא חוקית) של תווים (char) שונים אחד מהשני, וכן כי n אי שלילי.

לדוגמא, עבור הקלט ([“a”,”b”,”c”], 2), ההדפסה תהיה של הצירופים הבאים :

ba

bc

ca

ac

cb

ab

רשימה ריקה היא חוקית

7. הפונקציה parentheses(n)

עליכם לממש את הפונקציה parentheses, המקבלת מספר n (int). הממשת את החוקיות הבאה: במחרוזת יש n זוגות חוקיים של סוגריים אם המחרוזת מכילה אך ורק את התווים '(',')' 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 אי שלילי n ומחזירה רשימה עם כל המחרוזות עם n זוגות חוקיים של סוגריים. המחרוזות יכולות להופיע בכל סדר שהוא ברשימה.

לדוגמא, עבור הקלט 3, הפלט יכול להיות:

['()()()','(())()','()(())','((()))','(()())']

8. 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 נמצא בתוך המערך וכי הוא 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 התא המסומן ב-start, או אם הוא סמוך לתא שהתמלא קודם לכן. אנו נגדיר תאים כסמוכים רק אם הם מימין, משמאל, מעל או מתחת אחד לשני (ולא אם הם באלכסון). חשוב לציין שאין למלא תאים שסמוכים לתאים שהתחילו מלאים אם הם לא סמוכים לתא שהתמלא קודם לכן. 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:

[['*','*','*','*','*'],

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

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

['*','*','*','*','*']]

נהלי הגשה

הלינק להגשה של התרגיל הוא תחת השם: ex7

בתרגיל זה עליכם להגיש את הקובץ:

ex7.py – עם המימושים שלכם לפונקציות.

יש להגיש קובץ zip הנקרא ex7.zip המכיל את ex7.py.

בהצלחה!