

מבוא למדעי המחשב 67101 - סמסטר ב' 2022

תרגיל 7 - רקורסיה

להגשה בתאריך **12/05/2022** 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.

הנחיות כלליות לתרגיל:

- התרגיל מורכב ממספר משימות, אשר אינן בהכרח קשורות אחת לשנייה. יש לממש את כל הפונקציות המפורטות. המימוש של כל הפונקציות צריך להיות **רקורסיבי**, וללא שימוש בלולאות, וכן בפעולות שזמן הריצה שלהן הוא $O(n)$ בגודל הקלט. בפרט, אין להשתמש ב-**list comprehension**, **slicing**, שימוש באופרטור **in** על **list** (או כל **sequence** אחר), פונקציות על **list** שדורשות מעבר על כל הרשימה (למשל הפונקציה **sum**), וכו'.
בפתרון תרגיל זה אין לעשות שימוש באף מודול חיצוני של **python**, מלבד המודול **typing** - כלומר, אין לעשות **import** לאף מודול, שאינו המודול **typing**, לצורך הפתרון. בפרט, אין לעשות שימוש במודול **math** או במודול **itertools**.
- בתרגיל אתם נדרשים לבצע בדיקות טיפוסים בכל הפונקציות. כלומר בהרצת הפקודה

```
python3 -m mypy --strict ex7.py
```

במחשבי בית הספר על הקוד, ההרצה צריכה לעבור בצורה תקינה וללא שגיאות. בנוסף, תהיה בדיקה שהטיפוסים שהגדרתם אינם רחבים מדי. אין להשתמש באף מקום בטיפוס **Any**, אלא אם צוין אחרת.
אם במשימה מסוימת לא כתוב במפורש להחזיר ערך - אין להחזיר דבר.
- לאורך כל התרגיל, מותר להשתמש בפונקציות עזר במידת הצורך.
- לאורך כל התרגיל, הניחו שכל הקלטים הינם מטיפוסים (types) תקינים.
- לאורך כל התרגיל, לצורך השוואת טיפוסים, השתמשו בפונקציה **isinstance()**, ולא בפונקציה **type()**.
- לתרגיל מצורפים קבצי עזר בשם **ex7_helper.py** ו-**hanoi_game.py** המכילים פונקציות עזר בהן תצטרכו להשתמש למימוש חלק מהמשימות. **אין להגיש קבצים אלו בהגשת התרגיל.**
- ייתכן ולחלק מהמשימות יש מימוש יעיל יותר שאינו רקורסיבי. מטרת התרגיל היא לתרגל שימוש ברקורסיה, גם עבור משימות שעבורן רקורסיה אינה הפתרון האידיאלי.

חלק ראשון: רקורסיה פשוטה

את המשימות בחלק זה יש לפתור ע"י שימוש בפונקציות רקורסיביות, ללא שימוש בלולאות מכל סוג שהוא, כולל לולאות הנמצאות במימוש של פונקציית עזר, או לחלופין פונקציות או פעולות של פייתון המבצעות מעבר על אוסף של איברים (למשל שימוש ב-slicing, קריאה לפונקציה sum או שימוש באופרטור in על רשימות):

1. הפונקציה:

```
mult(x: float, y: int) -> float
```

עליכם לממש את הפונקציה `mult`, המקבלת מספר `x` (float) ומספר `y` שהינו שלם אי-שלילי (int), ומחזירה את תוצאת החישוב של `x` כפול `y`. המימוש של הפונקציה צריך להיות רקורסיבי. במשימה זו, במימוש שלכם מותר להשתמש רק בפונקציות הבאות שסיפקנו לכם בקובץ העזר `ex7_helper.py`:

- `add(x: float, y: float)` – המקבלת שני מספרים מטיפוס float ומחזירה את סכומם.

- `subtract_1(x: int)` – מקבלת מספר שלם `x` ומחזירה את המספר הקודם לו (`x-1`).

אסור להשתמש באף אחד מהאופרטורים המתמטיים בצורה ישירה (כלומר אסור להשתמש באף אחד מבין האופרטורים `+`, `-`, `*`, `/`, `%`). המימוש שלכם צריך לעבוד בזמן ריצה לינארי (ולא פחות מכך). לדוגמא, עבור הקלטים `x=3, y=4` יתקבל הפלט 12.

2. הפונקציה:

```
is_even(n: int) -> bool
```

עליכם לממש את הפונקציה `is_even`, המקבלת את המספר השלם `n` (int) אי-שלילי, ומחזירה `True` אם `n` הוא מספר זוגי, ו-`False` אחרת. במשימה זו, במימוש שלכם מותר להשתמש רק בפונקציה `subtract_1(x: int)` שסיפקנו לכם בקובץ העזר, המקבלת המספר מטיפוס `int` ומחזירה את המספר הקודם לו (למשל עבור הקלט 7 יוחזר המספר 6). בפרט, אסור להשתמש באף אחד מהאופרטורים המתמטיים בצורה ישירה (כלומר אסור להשתמש באף אחד מבין האופרטורים `+`, `-`, `*`, `/`, `%`). לדוגמא, עבור הקלט 479: הפלט יהיה `False`, ועבור הקלט 8: הפלט יהיה `True`.

3. הפונקציה:

```
log_mult(x: float, y: int) -> float
```

עליכם לממש את הפונקציה `log_mult`, הפועלת בדומה לפונקציה `mult` מסעיף 1, אך הפעם המימוש שלה חייב להיות בזמן ריצה לוגריתמי. במשימה זו, במימוש שלכם מותר להשתמש רק בפונקציות העזר הבאות שסיפקנו לכם בקובץ העזר `ex7_helper.py`:

- **is_odd(n: int) -** מקבלת מספר שלם **n** ומחזירה **True** אם **n** הוא מספר אי-זוגי, ומחזירה **False** אחרת.
 - **add(x: float, y: float) -** מקבלת שני מספרים מטיפוס **float** ומחזירה את סכומם.
 - **divide_by_2(n: int) -** מקבלת מספר שלם **n** ומחזירה את תוצאת החלוקה **השלמה** של המספר ב-2 (כלומר חלוקה ב-2 ללא שארית).
- אסור להשתמש בפונקציה **mult** שמימשם סעיף 1. אסור להשתמש באף אחד מהאופרטורים המתמטיים בצורה ישירה (כלומר אסור להשתמש באף אחד מבין האופרטורים **+, -, *, //, %**). רמז: הזכרו במימוש לפונקציה **power** עם זמן ריצה לוגריתמי, שראיתם בהרצאה.

4. הפונקציה:

is_power(b: int, x: int) -> bool

עליכם לממש את הפונקציה **is_power**, המקבלת שני מספרים שלמים אי-שליליים **b** ו-**x**, ובודקת האם קיים מספר שלם אי-שלילי **n** כך ש-**b** בחזקת **n** שווה ל-**x**. הפונקציה תחזיר **True** במידה וזה מתקיים, ו-**False** אחרת. בשאלה זו אתם נדרשים לכתוב מימוש שזמן הריצה שלו הוא $O(\log(b) * \log(x))$. אסור להשתמש באף אחד מבין האופרטורים המתמטיים בצורה ישירה (כלומר אסור להשתמש באף אחד מבין האופרטורים **+, -, *, //, %**), אך מותר לכם להשתמש בכל אחת מהפונקציות שסיפקנו לכם בקובץ העזר, או לחלופין, בפונקציות שמימשתם בעצמכם במשימות הקודמות. לדוגמא, עבור הקלטים **b=2** ו-**x=16**, הפונקציה תחזיר **True** כי עבור **n=4** מתקיים 2 בחזקת 4 שווה ל-16. לעומת זאת, עבור הקלטים **b=3** ו-**x=17**, הפונקציה תחזיר **False** כי לא קיים מספר שלם אי-שלילי **n** שעבורו 3 בחזקת **n** שווה ל-17.

5. הפונקציה:

reverse(s: str) -> str

עליכם לממש את הפונקציה **reverse**, המקבלת מחרוזת תווים **s** (str), ומחזירה את המחרוזת המכילה את אותם התווים של **s**, אך בסדר הפוך. במשימה זו, במימוש שלכם מותר להשתמש רק בפונקציה **append_to_end(s: str, c: str)** שסיפקנו לכם בקובץ העזר, המקבלת מחרוזת **s** (str) ותו בודד **c** שגם הוא מטיפוס מחרוזת **s** (str) אך באורך 1, ומחזירה מחרוזת חדשה **s** (str) בה התו **c** נוסף לסוף המחרוזת **s**. במימוש אסור לכם להשתמש באף פעולה אחרת של פייתון על מחרוזות (כולל שרשור מחרוזות עם האופרטור "+"). כמו כן, אסור לכם לקרוא לפונקציה **list**, וכן אין לבצע היפוך למחרוזת על-ידי **slicing** באופן הבא **s[::-1]**. לדוגמא, עבור הקלט **s="intro"**, הפונקציה תחזיר את המחרוזת **"ortni"**.

חלק שני: רקורסיה מתקדמת

6. הפונקציה:

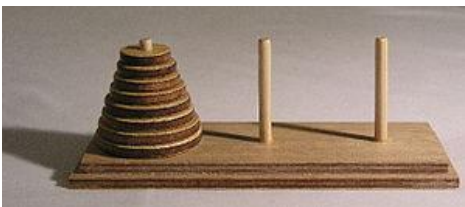
```
play_hanoi(Hanoi: Any, n: int, src: Any, dst: Any, temp: Any)
```

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.

שימו לב (2) - 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` דיסקיות על המוט `src` מסודרות בצורה חוקית, ואין דיסקיות על שאר המוטות). לא ניתן להניח דבר על מימוש האובייקטים המורכבים.

שימו לב (3) - הקובץ `hanoi_game.py` יקרא לפונקצייה שלכם רק עם ערכי `n` חיוביים. אך על הפונקצייה שלכם להתמודד עם כל ערך שלם! עבור `n` שלילי, על הפונקצייה להתנהג כאילו התקבל 0.

שימו לב (4) - במשימה זו, אתם רשאים להשתמש בטיפוס `Any` לצורך בדיקת הטיפוסים של הפונקצייה `play_hanoi`. אין להשתמש באופרטור `Any` במימוש של אף פונקצייה אחרת בתרגיל!

7. הפונקצייה:

`number_of_ones(n: int) -> int`

עליכם לממש את הפונקצייה `number_of_ones` המקבלת מספר טבעי `n (int)`, ומחזירה את מספר הפעמים שהספרה '1' מופיעה בכל המספרים מ-1 עד `n` (כולל), לרבות כפילויות של הספרה '1' באותו המספר. לצורך הפתרון שלכם, מותר להשתמש באופרטורי המודולו % והחילוק השלם // וכן בחיבור (+) ובחיסור (-). אסור לבצע המרה ל-`str` בשום שלב. לדוגמא, עבור הקלט `n=13`, יוחזר הפלט 6 כי הספרה '1' מופיעה פעם אחת במספרים: 1, 10, 12, 13, וכן עוד פעמיים במספר 11.

8. הפונקצייה:

`compare_2d_lists(l1: List[List[int]], l2: List[List[int]]) -> bool`

עליכם לממש את הפונקצייה `compare_2d_lists` המקבלת שתי רשימות דו-מימדיות של מספרים (כל אחת מהן מיוצגת על-ידי רשימה של רשימות מספרים). הפונקצייה תחזיר `True` אם שתי הרשימות הדו-מימדיות זהות על כל ערכיהן, ו-`False` במידה ויש לפחות איבר אחד שבה הן נבדלות. הרשימה החיצונית וכן הרשימות הפנימיות, יכולות להיות ריקות, וכן ייתכן שאורכן של הרשימות הפנימיות בכל אחת מהרשימות הדו-

מימדיות אינו זהה. כמו כן, במידה ואורכן של הרשימות החיצוניות $l1$ ו- $l2$ שונה, או לחלופין, אורכן של רשימות פנימיות הממוקמות באותו מיקום ברשימות החיצוניות $l1$ ו- $l2$ שונה, יש להחזיר גם כן **False**. אין להשתמש באופרטור $==$ על פני רשימות (חד-מימדיות או דו-מימדיות), אך מותר להשתמש באופרטור $==$ כדי להשוות בין מספרים.

לדוגמא, עבור הקלט $l1=[[1,2],[4,5,6]]$ ו- $l2=[[1,2],[4,5,8]]$ הפונקציה תחזיר **False** כי הן נבדלות בקואורדינטה המסומנת **באדום**.

9. הפונקציה:

`magic_list(n: int) -> List[Any]`

עליכם לממש את הפונקציה `magic_list` המקבלת מספר n שלם אי-שלילי (מטיפוס `int`) ומחזירה

רשימה בגודל n , המייצגת את האיבר ה- n בסדרה הבאה:

- עבור $n=0$, הפונקציה תחזיר רשימה ריקה `[]`
- עבור $n=1$, הפונקציה תחזיר רשימה בגודל 1, הכוללת באינדקס ה-0 את האיבר ה-0 בסדרה: `[[]]`
- עבור $n=2$, הפונקציה תחזיר רשימה בגודל 2, אשר כוללת באינדקס ה-0 את האיבר ה-0 בסדרה, ובאינדקס ה-1 את האיבר ה-1 בסדרה: `[[], [[]]]`
- עבור $n=3$, הפונקציה תחזיר רשימה בגודל 3, אשר כוללת את שלושת האיברים הראשונים בסדרה: `[[], [[]], [[[], []]]]`
- ובאותו אופן, עבור n כלשהו, תוחזר רשימה בעלת n איברים, כך שבכל אינדקס i ממוקם האיבר ה- i בסדרה (עבור על i בין 0 ל- $(n-1)$)

שימו לב: עליכם לוודא שהעותקים השונים של הרשימות הפנימיות ברשימת הפלט הם עותקים **עמוקים**

(deep copy), ובפרט לוודא שעל אף רשימה פנימית (מכל עומק) לא יהיו הצבעות משני מקומות שונים

ברשימת הפלט החיצונית (בכל עומק). שימו לב גם **שאסור** לכם לייבא את הספריה `copy` ולהשתמש

בפונקציה `copy.deepcopy`.

שימו לב (2): גם במשימה זו אתם רשאים להשתמש בטיפוס `Any` לצורך בדיקת הטיפוסים.

שימו לב (3): מותר להשתמש עד פעם אחת (בכל קריאה רקורסיבית) בפונקציה `append` הפועלת על

רשימות.

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** בלבד.

בהצלחה!