

תרגיל בית 6

הנחיות כלליות:

- קראו **היטב** את השאלות והקפידו שהתוכניות שלכם פועלות בהתאם לנדרש.
- את התרגיל יש לפתור לבד!
- הקפידו על כללי ההגשה המפורסמים באתר. בפרט, יש להגיש את כל השאלות יחד בקובץ `ex6_012345678.py` המצורף לתרגיל, לאחר החלפת הספרות 012345678 במספר ת.ז. שלכם, כל 9 הספרות כולל, ספרת ביקורת.
- אופן ביצוע התרגיל: בתרגיל זה עליכם לכתוב את הפונקציות במקומות המתאימים בקובץ השלד.
- מועד אחרון להגשה: כמפורסם באתר.
- בדיקה עצמית: כדי לוודא את נכונותן ואת עמידותן של התוכניות בכל שאלה הריצו את תוכניתכם עם מגוון קלטים שונים וודאו כי הפלט נכון. לנוחיותכם מצורפות בדיקות בסוף הקובץ השלד (אך ייתכנו עוד מקרים שיש לבדוק).
- ניתן להניח כי הקלט שמקבלות הפונקציות הינו כפי שהוגדר בכל שאלה ואין צורך להתייחס לקלט לא תקין, אלא אם כן נאמר אחרת.
- אין למחוק את ההערות שמופיעות בשלד .
- אין לשנות את החתימות (שורות ההגדרה) של הפונקציות שמופיעות בקובץ השלד של התרגיל. עם זאת, אתם רשאים להוסיף פונקציות נוספות ולקרוא להן.
- אין להשתמש בקריאה לספריות חיצוניות (אסור לעשות `import`).

הנחיות מיוחדות לתרגיל זה:

- **על כל הפונקציות שתכתבו להיות רקורסיביות - פתרונות לא רקורסיביים לא יתקבלו.**
- **אין להשתמש בלולאות (`for`, `while`) בשום מקום בפתרון.**
- **בחלק מהשאלות יש איסור על שימוש בפונקציות מובנות מסוימות.**

שאלה 1

כתבו פונקציה רקורסיבית `reverse_string` אשר מקבלת כקלט מחרוזת ומחזירה את המחרוזת בהיפוך סדר התווים.

הבהרות:

- ניתן להשתמש ב `slicing` בשביל הקריאה הרקורסיבית, אך אסור לכם להפוך את המחרוזת באמצעות `slicing` (ע"י `step` שלילי). כמו כן, אין להשתמש בשום פונקציה אחרת שמבצעת היפוך מחרוזות, כדוגמת `.reverse` הפונקציה `.reverse`.

דוגמאות הרצה:

```
>>> reverse_string("abc")
'cba'

>>> reverse_string("Hello!")
'!olleH'

>>> reverse_string("")
''
```

שאלה 2

כתבו פונקציה רקורסיבית `find_maximum` שמקבלת רשימה של מספרים אי-שליליים (אפס ומעלה) ומחזירה את הגדול מביניהם. במידה והרשימה ריקה, הפונקציה תחזיר את המספר -1.

הבהרה: אין להשתמש בפונקציה המובנית `max` בשום שלב.

דוגמאות הרצה:

```
>>> find_maximum([9,3,0,10])
```

```
10
```

```
>>> find_maximum([9,3,0])
```

```
9
```

```
>>> find_maximum([])
```

```
-1
```

שאלה 3

מחרוזת נקראת פלינדרום אם קריאתה מהסוף להתחלה ומהתחלה לסוף תניב את אותו רצף תווים. נגדיר שבמחרוזת שהינה פלינדרום יש לפחות תו אחד, התו הראשון שלה זהה לתו האחרון, התו השני זהה לתו שלפי האחרון וכן הלאה. הזרות בין התווים צריכה להיות מוחלטת, Aba אינו פלינדרום כיוון שהתו A אינו זהה לתו a. לדוגמה, המחרוזות 'abba', 'a' הינן פלינדרומים ואילו 'abcab' אינה פלינדרום.

עליכם לכתוב את הפונקציה הרקורסיבית `is_palindrome` המקבלת מחרוזת לא ריקה `s` ומחזירה `True` אם המחרוזת הינה פלינדרום, ו-`False` אחרת.

הבהרה: אין להפוך את מחרוזת הקלט או חלקים ממנה כחלק מהפתרון.

דוגמאות הרצה:

```
>>> is_palindrome("aa")
True
>>> is_palindrome("aa ")
False
>>> is_palindrome("caca")
False
>>> is_palindrome("abcbbcbba")
True
```

הדרכה: מחרוזת (לא ריקה) היא פלינדרום אם מתקיימים שני התנאים הבאים:

- התו הראשון זהה לתו האחרון.
- תת המחרוזת המורכבת מכל התווים החל מהתו והשני ועד התו לפני האחרון הינה פלינדרום.

שאלה 4

עליכם לטפס במעלה גרם מדרגות בן n (מספר שלם גדול מאפס) מדרגות. בכל צעד טיפוס אתם יכולים לבחור אם לעלות מדרגה אחת בלבד או שתי מדרגות בבת אחת. בכמה דרכים שונות ניתן לטפס לקצה גרם המדרגות?

כתבו את הפונקציה הרקורסיבית `climb_combinations` שמקבלת מספר שלם גדול מאפס n ומחזירה את מספר האפשרויות לעלות גרם המדרגות בו יש n מדרגות.

דוגמאות הרצה:

```
>>>climb_combinations(3)
3
```

הסבר: יש בדיוק 3 דרכים לעלות 3 מדרגות:

- לעלות מדרגה אחת 3 פעמים
- לעלות שתי מדרגות בבת אחת ואז לעלות מדרגה אחת
- לעלות מדרגה אחת ואז לעלות שתי מדרגות בבת אחת

```
>>>climb_combinations(10)
89
```

שאלה 5

מחרוזת תקינה של סוגריים היא מחרוזת שבה לכל סוגר שמאלי '(' מתאים סוגר ימני ')' בהמשך המחרוזת ולהפך. כלומר, בכל פעם שפותחים סוגריים (עם סוגר שמאלי) צריך לסגור אותם בהמשך (עם סוגר ימני) ולא ניתן לסגור סוגריים במידה ולא הופיע קודם סוגר שמאלי שנשאר פתוח. למשל, המחרוזת '((()))' תקינה ואילו המחרוזות הללו אינן תקינות:

```
' )'  
' ('  
' ) ('
```

גם המחרוזות הבאות לא תקינות:

```
' ( ( ) '
```

הסבר: שימו לב שכל סוגר ימני מבצע סגירה רק לסוגר שמאלי אחד, בדיוק בצורה שבה אנחנו רגילים להשתמש בסוגריים. לכן, נשאר בדוגמה זו סוגר שמאלי פתוח, וזה לא תקין.

```
' ( ) ) ('
```

הסבר: הסוגר הימני המסומן לא תקין, כי הסוגר השמאלי היחיד לפניו כבר סגור.

כתבו פונקציה רקורסיבית `is_valid_paren` שתקבל מחרוזת המורכבת מתווים שונים (לא רק סוגריים) ותחזיר `True` אם המחרוזת תקינה מבחינת הסוגריים ואחרת `False`. הפונקציה תקבל ערך נוסף, `cnt`, אשר יעזור במניית הסוגריים השמאליים והימניים (ראו הדרכה בהמשך).

חתימת הפונקציה כפי שכתובה בקובץ השלד ולהלן, מגדירה שערך ברירת המחדל (`default`) עבור הפרמטר `cnt` הוא 0. כלומר, אם לא קוראים לפונקציה עם ערך מפורש ל-`cnt`, ערכו יהיה שווה ל 0.

```
def is_valid_paren(s, cnt=0):
```

דוגמאות הרצה - במהלך בדיקת השאלה יסופק לפונקציה רק ארגומנט אחד שהוא מחרוזת, כמו בדוגמאות הבאות:

```
>>> is_valid_paren("(.(a)")  
False
```

```
>>> is_valid_paren("p((()r((0)))")  
True
```

```
>>> is_valid_paren("")  
True
```

שימו לב שהקריאה האחרונה שקולה לקריאה:

```
is_valid_paren("p((()r((0)))", 0)
```

הדרכה: כשעוברים על מחרוזת, ברגע שפגשנו פתיחה של סוגריים, כלומר '(', צריך לוודא שבהמשך הם נסגרים ע"י ')'. צריך לשים לב גם למקרה שבו פוגשים סוגר ימני מבלי שיש סוגר שמאלי שהוא מתאים לו, למשל, התו

פייתון למהנדסים 0509-1820 , סמסטר א' תשפ"א 2021

השלישי במחרוזת הבאה: "()).". במחרוזת זו מספר הסוגריים הפותחים זהה למספר הסוגריים הסוגרים, אך מדובר בתבנית לא תקינה. השתמשו ב- cnt על מנת לעקוב אחר מספר הסוגריים שנפתחו וטרם נסגרו.