תרגיל 4 חלק א: רקורסיה קלאסית

תאריך פרסום: 21/11

תאריך הגשה: 1/12 בשעה 23:59

מתרגל אחראי: נוי סלומון

משקל תרגיל: 2 נקודות

מטרות העבודה: הכרות עם רקורסיות בסיסיות, המרת לולאות לרקורסיות, עבודה עם פונקציות מעטפת.

הנחיות ספציפיות לעבודה:

עבודה זו היא עבודה ראשונה מתוך זוג בנושא של רקורסיה ועוסקת בהמרת לולאה לרקורסיה ופונקציות מעטפת. חלק גדול מן הבעיות ניתן לפתור בקלות בלולאה ואת חלקן אף פתרנו כך בכיתה אך הדרישה מכם היא לפתור באמצעות רקורסיה.

אלא אם נאמר אחרת:

- מטרת התרגיל היא תרגול רקורסיה. אין לממש באמצעות לולאות. שימו לב כי גם פקודות על אוספים כגון in, max, min או sum מהוות לולאות ואנחנו אוסרים עליכם להשתמש בהן.
- ניתן לממש פונקציות מעטפת ופונקציות עזר.
- בסעיף בו אתם בוחרים לכתוב פונקציית מעטפת הקפידו ששם פונקציית המעטפת יהיה זהה לשם הפונקציה אותה אתם מתבקשים לממש – פונקציית העבודה (הפונקציה הרקורסיבית שפונקציית המעטפת קוראת לה) יכולה להיות בכל שם (משמעותי) אחר שבחרתם לנכון.
- אין להשתמש בפרמטרי ברירת מחדל בהגדרה של פונקציה.

שאלה 1

בהרצאה ראיתם כיצד לחשב באמצעות רקורסיבית עצרת (Factorial). כעת נרצה לחשב, באמצעות רקורסיה, ***עצרת למקוטעין***: בהנתן מספר שלם, מבצעת את הפעולות המתמטיות הבאות עבור כל המספרים השלמים מ-1 ועד למספר (החיובי) הנתון:

- חיבור בין מס' אי-זוגי למספר העוקב.
- כפל בין מס' זוגי למספר העוקב.

לדוגמא - ערך העצרת למקוטעין של 6 מסומן באמצעות $, וערכו 33. להלן החישוב -

6$ = 1 + 2 * 3 + 4 * 5 + 6 = 33

ממשו את הפונקציה: $semi\_factorial(num)$

אשר מקבלת מספר מטיפוס שלם גדול מ-0 ומחזירה את ערך העצרת למקוטעין המתאים.

<u>דוגמאות:</u>

```
>> print(semi_factorial(3))
7

>> print(semi_factorial(6))
33

>> print(semi_factorial(1))
1
```

שאלה 2

עליכם לממש פונקציה לזיהוי חפיפות מרובות בין מחרוזות.

נגדיר *חפיפה יחידה* בין מחרוזת ליבה (core) למחרוזת חזרה (repeat), אם קיימת תת מחרוזת יחידה בליבה השווה למחרוזת החזרה. למשל, עבור מחרוזת הליבה 'saba balagan', קיימת חפיפה יחידה עם מחרוזת החזרה 'aba', מכיוון שתת מחרוזת הליבה באינדקסים 1:3 שווה למחרוזת החזרה.

s	a	b	a		b	a	l	a	g	a	n

עבור אותה מחרוזת ליבה, ומחרוזת חזרה 'ba', קיימת *חפיפה כפולה* – תתי מחרוזת הליבה באינדקסים 2:3 ו-6:7 שוות למחרוזת החזרה.

s	a	b	a		b	a	l	a	g	a	n

שני מופעים (רצף אינדקסים במחרוזת הליבה) של מחרוזת החזרה יספרו כחזרות שונות, אם לפחות אינדקס אחד אינו משותף לשני המופעים.

באופן דומה, ניתן להרחיב את ההגדרה של חפיפה בין מחרוזת למספרי חזרות גדולים מ-2.

שימו לב: בשאלה זו חל איסור להשתמש במתודות של מחרוזות.

<u>סעיף א'</u>

ממשו את הפונקציה $string\_overlap\_detector(core\ ,\ rep\ ,\ repetition)$

המקבלת שתי מחרוזות ומספר שלם אי-שלילי repetition, ומחזירה בוליאני שערכו אמת אם גודל החפיפה בין core ו-rep היא repetition. ניתן להניח שמחרוזת rep איננה ריקה.

<u>דוגמאות הרצה</u>:

```
>> print(string_overlap_detector('saba sababa', 'aba', 2))
False
>>print(string_overlap_detector('saba sababa', 'aba', 3))
True
```

סעיף ב'

הפתרון שלכם עובד כל כך טוב, שכעת תרצו להרחיב אותו ולהחזיר True גם אם מחרוזת הליבה היא תת מחרוזת במחרוזת החזרה מספר נדרש של חפיפות.

ממשו את פונקציית המעטפת $string\_overlap\_dual(string\_1\ ,\ string\_2\ ,\ repetition\ )$ המקבלת שתי מחרוזות ומספר שלם, ומחזירה האם אחת המחרוזות היא תת-מחרוזת של המחרוזת השניה מספר repetition של פעמים.

קלט:

- **string_1,string_2 : string** – מחרוזת כלשהי, עלינו לבדוק האם היא מחרוזת ליבה כאשר השניה היא מחרוזת חזרה או האם היא מחרוזת חזרה כאשר השניה היא מחרוזת ליבה.
- **repetition : int -** מספר שלם, מייצג את כמות החפיפות המבוקשת של אחת המחרוזות בשניה.

פלט: **בוליאני** – True אם אחת ממחרוזות הקלט מופיעה מספר השווה ל repetition של פעמים במחרוזת הקלט השניה.

הנחות קלט:

- שתי המחרוזות אינן ריקות.

דוגמאות הרצה: - כל הקלטים והפלטים של הסעיף הקודם, ובנוסף :

```
>> string_overlap_dual('wa' , 'which witch watch which watch' , 2)
True
>> string_overlap_dual( 'which witch watch which watch' ,'wa' , 2)
True
>> string_overlap_dual('ch' , 'which witch watch which watch' , 4)
False
```

שאלה 3

נגדיר את המושג "סופר רצף", סדרה של מספרים כך שכל מספר אינו קטן מסכום כל האיברים שקדמו לו.

לדוגמא, הסדרה 53 , 25 , 13 , 12 , 0 מייצגת סופר רצף.

ממשו את הפונקציה:

longest_super_subsequence (input_list)

המקבלת רשימת מספרים input_list, ומחזירה את אורך הסופר- רצף הארוך ביותר בתוך הרשימה.

אם רצף כזה אינו קיים, הפונקציה תחזיר 0.

<u>דוגמאות:</u>

```
>>>longest_super_subsequence([0 , 3 , 6 , 9 , 11])
4

>>>longest_super_subsequence([ 22 , 23, 24,  25 , 26 ,27])
2
```

שאלה 4

ראו את הפונקציה האיטרטיבית הבאה המקבלת מספר שלם אי שלילי בבסיס עשר dec_num ומספר שלם base_num בטווח [2,10] המייצג את הבסיס החדש אליו נרצה לעבור, ומחזירה מחרוזת שהינה ההמרה של המספר העשרוני לבסיס המבוקש.

```
def decimal_to_base_iter(dec_num, base_num):
    """
    converts a decimal number to the required base
    :param dec_num: int, a decimal number to be converted
    :param base_num: int, the base we wish to convert to. between 2 and 10
    :return: string, represents the converted number in the required base.
    """
    output_string = ''
    while dec_num > 0:
        output_string= str(dec_num % base_num) + output_string
        dec_num = dec_num // base_num
    return output_string
```

<u>סעיף א'</u> -

ממשו את הפונקציה הרקורסיבית

decimal_to_base(dec_num, base_num)

המקבלת מספר שלם עשרוני ומספר שלם המייצג את הבסיס אליו נרצה להמיר – ומחזירה מחרוזת המייצגת את המספר העשרוני בבסיס החדש באמצעות רקורסיה.

<u>דוגמאות:</u>

```
>>>decimal_to_base(10, 4)
'22'
>>>decimal_to_base(10, 5)
'20'
>>>decimal_to_base(10, 10)
'10'
```

סעיף ב' -

כעת ראו פונקציה איטרטיבית נוספת. הפונקציה מקבלת קלט מטיפוס מחרוזת string_to_convert אשר הינו פלט תקין של הסעיף הקודם וקלט base_num כפי שהוגדר בסעיף הקודם. הפלט של הפונקציה הוא מספר שלם המייצג המרה של מחרוזת הקלט מבסיס הקלט לבסיס עשרוני.

```
def base_to_decimal_iter(string_to_convert, base_num):
    """
    converts string_to_convert from base = base_num to decimal base iteratively
    :param string_to_convert: string, represents a valid number
    :param base_num: int, the base from which to convert string_to_convert into decimal base
    :return: int, the convertion of string_to_convert from base_num to decimal
    """
    output_dec_num = 0
    for i in range(len(string_to_convert)):
        cur_digit = int(string_to_convert[i])
        power_multiplier = len(string_to_convert) - i -1
        output_dec_num+= cur_digit * (base_num ** power_multiplier)

    return output_dec_num
```

ממשו את הפונקצייה הרקורסיבית

base_to_decimal(string_to_convert, base_num)

המקבלת מחרוזת המייצגת מספר בבסיס כלשהו בטווח [2,10] (השמור במשתנה base_num), מבצעת המרה שלו לבסיס עשרוני ומחזירה מספר המייצג את תוצאת ההמרה.

דוגמאות:

```
>>>base_to_decimal('121' , 9)
100
>>>base_to_decimal('' , 9)
0
```

בהצלחה !