## שאלה 1

להלן שלוש פונק' הקיימות במחלקה str שאינן קיימות במחלקה

תחילה בצורת הקריאה דרך מופע של האובייקט.

```
>>> s = "xyzw"
>>> s.upper()
'XYZW'
>>> s.lower()
'xyzw'
>>> s.capitalize()
'Xyzw'
```

לאחר מכן באמצעות קריאה דרך המחלקה עצמה.

```
>>> str.upper(s)
'XYZW'
>>> str.lower(s)
'xyzw'
>>> str.capitalize(s)
'Xyzw'
```

כנ"ל עם list:

תחילה בקריאה דרך מופע של האובייקט.

```
>>> 1 = ['x', 'y', 'z', 'w']
>>> 1.reverse()
>>> 1
['w', 'z', 'y', 'x']
>>> 1 = ['x', 'y', 'z', 'w']
>>> 1.pop()
'w'
>>> 1
['x', 'y', 'z']
>>> 1 = ['x', 'y', 'z', 'w']
>>> 1.sort()
>>> 1
['w', 'x', 'y', 'z']
```

לאחר מכן קריאה דרך המחלקה.

```
>>> l = ['x', 'y', 'z', 'w']
>>> list.reverse(l)
>>> l
['w', 'z', 'y', 'x']
>>> l = ['x', 'y', 'z', 'w']
>>> list.pop(l)
'w'
>>> l
['x', 'y', 'z']
>>> l = ['x', 'y', 'z', 'w']
>>> list.sort(l)
>>> l
['w', 'x', 'y', 'z']
```

## שאלה 2

- א. קלט לא תקין:
- assert קלט שאינו מחרוזת שגיאת Assertion, סוג הקלט נבדק ע"י תנאי בתוך פקודת a בתחילת הפונק' וזורק שגיאה לפני הרצת הפונק' במקרה והתנאי לא מתקיים.

- b. קלט באורך שאינו תקין שוב שגיאת Assertion, אורך הקלט נבדק לאחר שנבדק סוגו בתחילת הפונק', שוב בתוך פקודת assert, במידה אורך הקלט שונה מ-8 התנאי לא מתקיים ונזרקת שגיאה.
  - ב. טבלאות מעקב:
  - :"87654321" <u>בעבור ה</u>קלט.a

iteration	i	ID[i]	val	Total
1	0	<b>'</b> 8'	8	8
2	1	'7'	7	13
3	2	<b>'</b> 6'	6	19
4	3	<b>'</b> 5'	5	20
5	4	<b>'4'</b>	4	24
6	5	<b>'</b> 3'	3	30
7	6	'2'	2	32
8	7	<b>'1'</b>	1	34

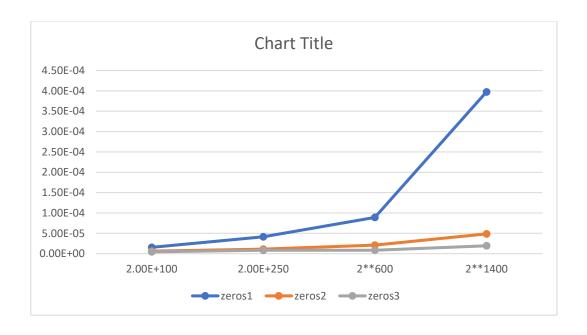
b. בעבור הקלט – "31851682":

	l		l	•
iteration	i	ID[i]	val	Total
1	0	'3'	3	3
2	1	'1'	1	5
3	2	'8'	8	13
4	3	'5'	5	14
5	4	'1'	1	15
6	5	'6'	6	18
7	6	'8'	8	26
8	7	'2'	2	30

## שאלה 3

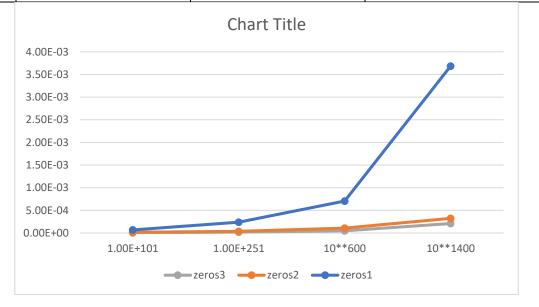
א. נראה כי יש הבדל ניכר בין שני המימושים. בראשון הזמן הריצה נראה כתלות מעריכית בגודל הקלט בעוד שהפונק' השנייה ספק אם התלות היא לינארית או מעט גרועה יותר, בכל מקרה מנקודת מבט zeros2 = o(zeros1).

	zeros1	zeros2	zeros3
2**100	1.56000000000045e-05	6.300000000007494e-06	4.60000000000437e-06
2**250	4.170000000000562e-05	1.129999999998811e-05	8.39999999998686e-06
2**600	8.88999999999593e-05	2.110000000000306e-05	8.49999999998092e-06
2**1400	0.0003973999999999914	4.869999999998744e-05	1.93999999999928e-05



- ב. הפנק' השלישית אשר משתמשת בפתרון מובנה של פייתון אכן יעילה יותר משתי הפונק' שלנו, אמנם ניתן להתנחם כי בעבור הקלטים שאנו בדקנו הפונק' השנייה לא משמעותית יותר גרועה, אין בכלל מה לדבר על הפונק' הראשונה.
  - ג. נבדוק את זמני הריצה עם בסיס 10, כך מובטח מספר מקסימלי של ספרות 0. ניכר שמיכוון שבכל מקרה עלינו לעבור על כל ספרות הקלט על מנת לספור תמיד את כל האפסים, מספר האפסים במספר לא משפיע על זמן הריצה, למעט אולי בצורה זניחה ביחס לסדר הגודל במקרים בהם יש כמות לא שווה של פעולות בין מציאת אפס לדילוג לספרה הבאה.

		_	_
	zeros1	zeros2	zeros3
10**100	6.75000000000159e-05	1.630000000000342e-05	4.50000000001031e-06
10**250	0.0002372999999999932	3.68999999999943e-05	1.829999999998873e-05
10**600	0.000704099999999999	0.00010680000000000064	4.619999999999714e-05
10**1400	0.003684299999999998	0.00032350000000000087	0.0002092999999999997



ד. אנחנו יכולים להשוות את הביצוע של פעולת הסכימה של count = count + 1 בתוך לולאה הרצה לכל איבר ב-range לעומת לולאה שעושה אותו מספר איטרציות אבל באמצעות המרת המספר למחרוזת ומעבר על כל תו במחרוזת על מנת לראות את ההבדל. נשים לב כי בעוד המקרה השני יעיל באותו

סדר גודל כמו zeros2, המקרה הראשון לא נגמר גם לא אחרי מספר שניות. הסיבה היא range. עוד לא הגענו לנושא זה בקורס בצורה מפורשת, אבל המימוש של range בפייתון 3 הוא למעשה המימוש של range מפייתון 2, וזו למעשה פונקציה גנרטיבית שמספקת לנו את האיברים המבוקשים לפי xrange של פרשימה שהזנו לפונק' מבלי לעשות אלוקציה מראש של כל האיברים שהרשימה הזו הייתה מכילה אילו היינו צריכים לשמור את כולה בזיכרון. המחיר לפי דעתי במצב זה הוא למעשה לולאה בתוך לולאה – יעילות ביחס ריבועי לגודל הקלט. בעוד שכאשר אנו דורשים להמיר את המספר למחרוזת, פייתון שומר את כל המחרוזת לזיכרון ורק לאחר מכן מתחילות האיטרציות, ובעוד שמדובר במחרוזת גדולה למדי, אנחנו מקבלים כביכול את סדר גודל הפעולות שלו היינו מצפים בלולאה מסוג זה.

## <u>שאלה 4</u>

- א. בשביל לספור את ההופעות של ספרות שונות בנוסף לאותיות עלינו פשוט להוסיף את הספרות שאנו מעוניינים למצוא ל-chars.
  - ב. על מנת להסיר תו עלינו להזין למשתנה new תו ריק.