

שאלה 1

להלן שלוש פונק' הקיימות במחלקה str שאינן קיימות במחלקה list:

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.

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

a. קלט שאינו מחרוזת – שגיאת Assertion, סוג הקלט נבדק ע"י תנאי בתוך פקודת assert

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:

a. בעבור הקלט – "87654321":

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

b. בעבור הקלט – "31851682":

| 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 אמפירית נראה כי $zeros2 = o(zeros1)$.

| | zeros1 | zeros2 | zeros3 |
|---------|------------------------|------------------------|------------------------|
| 2**100 | 1.560000000000045e-05 | 6.3000000000007494e-06 | 4.6000000000000437e-06 |
| 2**250 | 4.1700000000000562e-05 | 1.1299999999998811e-05 | 8.399999999998686e-06 |
| 2**600 | 8.889999999999593e-05 | 2.110000000000306e-05 | 8.499999999998092e-06 |
| 2**1400 | 0.0003973999999999914 | 4.869999999998744e-05 | 1.93999999999928e-05 |

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 בפיתון 3 הוא למעשה המימוש של xrange מפיתון 2, וזו למעשה פונקציה גנרטיבית שמספקת לנו את האיברים המבוקשים לפי דפוס הרשימה 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, פיתון 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.

שאלה 4

- 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.