תרגיל בית מס' 1

שאלה 1

נרצה למצוא פונקציות 3 פונקציות של str שאינן קיימות בlist ו3 פונקציות שקיימות בlist ולא בstr.
ראשית, נגדיר משתנים:

```
>>> string = "abcd"
>>> lst = ['a', 'b', 'c', 'd']
>>> |
```

כעת, נמצא 3 פונקציות שנמצאות על str ולא על list, נפעיל אותן בשתי ה"שיטות":

```
>>> str.islower(string)
True
>>> string.islower()
True
>>> str.capitalize(string)
'Abcd'
>>> string.capitalize()
'Abcd'
>>> str.center(string, 20, '#')
'########abcd########'
>>> string.center(20, '#')
'########abcd########'
```

נראה שהן אכן לא קיימות על list:

```
>>> list.islower(lst)
Traceback (most recent call last):
  File "<pyshell#131>", line 1, in <module>
    list.islower(lst)
AttributeError: 'list' object has no attribute 'islower'
>>> list.capitalize(lst)
Traceback (most recent call last):
  File "<pyshell#132>", line 1, in <module>
    list.capitalize(lst)
AttributeError: 'list' object has no attribute 'capitalize'
>>> list.center(lst, 20, '#')
Traceback (most recent call last):
  File "<pyshell#133>", line 1, in <module>
    list.center(lst, 20, '#')
AttributeError: 'list' object has no attribute 'center'
```

כעת, נמצא 3 פונקציות שקיימות על list ולא על str:

נפעיל על הרשימה שהגדרנו את append, pop, reverse בשתי השיטות:

```
>>> list.append(lst, "z")
>>> lst
['a', 'b', 'c', 'd', 'z']
```

```
>>> lst.append("z")
>>> lst
['a', 'b', 'c', 'd', 'z']
```

```
>>> list.pop(lst, 0)
'a'
>>> lst
['b', 'c', 'd']
```

```
>>> lst.pop(0)
'a'
>>> lst
['b', 'c', 'd']
```

```
>>> list.reverse(lst)
>>> lst
['d', 'c', 'b', 'a']
```

```
>>> lst.reverse()
>>> lst
['d', 'c', 'b', 'a']
```

וכעת נראה כי לא ניתן להפעיל פעולות אלו על str:

```
>>> string.append('z')
Traceback (most recent call last):
  File "<pyshell#15>", line 1, in <module>
    string.append('z')
AttributeError: 'str' object has no attribute 'append'
>>> string.pop(1)
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    string.pop(1)
AttributeError: 'str' object has no attribute 'pop'
>>> string.reverse()
Traceback (most recent call last):
  File "<pyshell#17>", line 1, in <module>
    string.reverse()
AttributeError: 'str' object has no attribute 'reverse'
```

שאלה 2

הוספתי שורת after loop בטבלה בהתאם לדוגמא שבתרגול

טבלת מעקב עבור הקלט "12345678":

Iteration	i	ID[i]	val	total
1	0	“1”	1	**1**
2	1	“2”	2	**5**
3	2	“3”	3	**8**
4	3	“4”	4	**16**
5	4	“5”	5	**21**
6	5	“6”	6	**24**
7	6	“7”	7	**31**
8	7	“8”	8	**38**
After loop	7	“8”	8	**8**

הפונקציה תחזיר 2

טבלת מעקב עבור הת.ז שלי: 207704842

Iteration	i	ID[i]	val	total
1	0	“2”	2	**2**
2	1	“0”	0	**2**
3	2	“7”	7	**9**
4	3	“7”	7	**14**
5	4	“0”	0	**14**
6	5	“4”	4	**22**
7	6	“8”	8	**30**
8	7	“4”	4	**38**
After loop	7	“4”	4	**8**

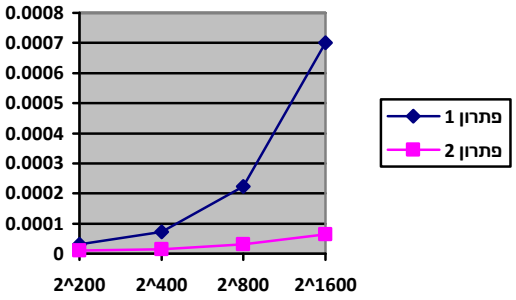
הפונקציה תחזיר 2

שאלה 4

א.

	2**200	2**400	2**800	2**1600
פתרון 1	3.069999999999809e-05 (0.00003069999999999809)	7.219999999999796e-05 (0.00007219999999999796)	0.00022280000000000216	0.0007010999999999996
פתרון 2	1.0099999999998999e-05 (0.00001009999999998999)	1.5500000000001624e-05 (0.00001550000000001624)	3.019999999999759e-05 (0.00003019999999999759)	6.460000000000146e-05 (0.00006460000000000146)

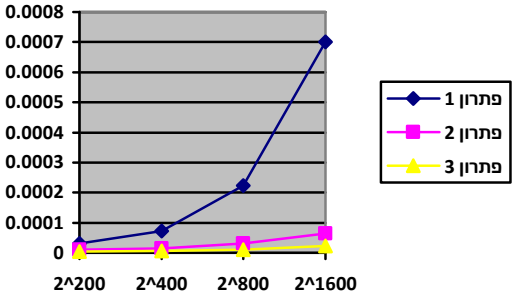
נשתמש בגרף כדי להסביר את הקשר בין הקלט וזמן הריצה עבור כל פתרון:



כפי שניתן לראות בגרף, זמן הריצה של שני הפתרונות מושפע מגודל הקלט, אך זמן הריצה של הפתרון הראשון גדל מהר יותר מזמן הריצה של הפתרון השני.

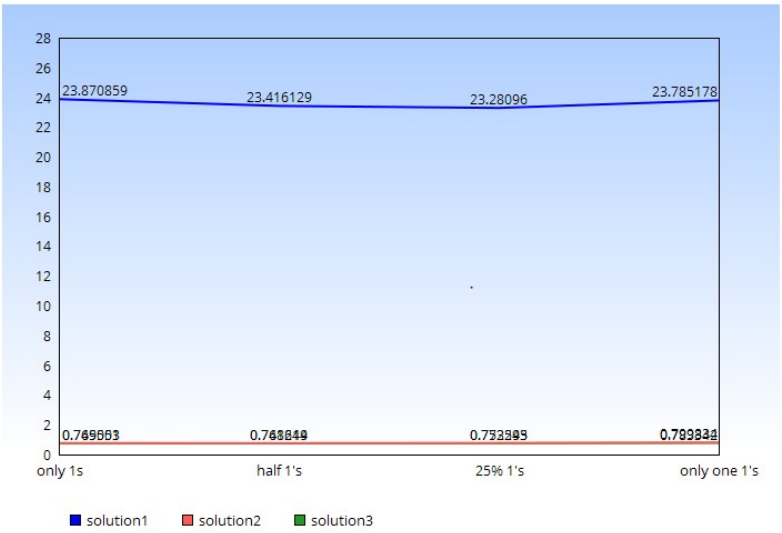
ב. נחזור על סעיף א והפעם נכלול את פתרון 3:

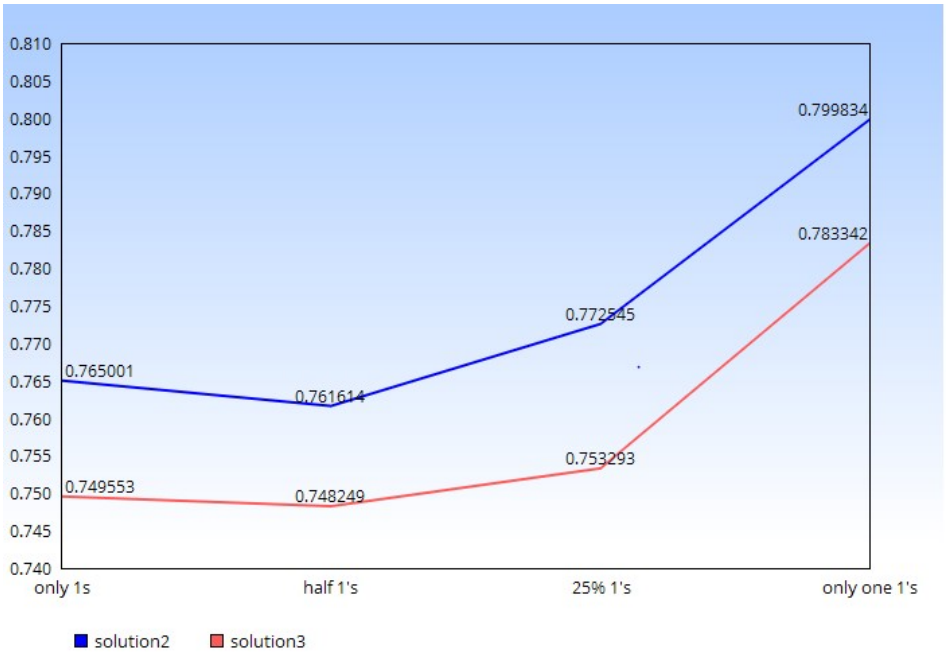
	2**200	2**400	2**800	2**1600
פתרון 1	3.069999999999809e-05 (0.00003069999999999809)	7.219999999999796e-05 (0.00007219999999999796)	0.00022280000000000216	0.0007010999999999996
פתרון 2	1.0099999999998999e-05 (0.00001009999999998999)	1.5500000000001624e-05 (0.00001550000000001624)	3.019999999999759e-05 (0.00003019999999999759)	6.460000000000146e-05 (0.00006460000000000146)
פתרון 3	3.3000000000012186e-06 (0.00000330000000000012186)	5.200000000000343e-06 (0.00000520000000000343)	9.699999999997905e-06 (0.00000969999999997905)	2.3599999999998622e-05 (0.00002359999999998622)



ניתן לראות כי זמן הריצה של פתרון 3, שעושה שימוש בפונקציה מובנית של פייתון, הוא אכן מהיר יותר מזה של שני הפתרונות האחרים, עם זאת, גם במקרה זה ניתן לראות בעיקר בקלט האחרון שגודל שקלט משפיע על זמן הריצה.

ג. מהסעיף הקודם ניתן לראות שההבדלים בין הפתרונות מובהקים יותר ככל שהקלט גדל, לכן בחרתי בקלט גדול- מספר בן 100000 ספרות שכולן 1, חצי מהן 1, 75% הן 0 והשאר 1, הספרה הראשונה 1 והשאר 0. להלן הגרף:





הדרף הראשון מתאר את השפעת מספר האפסים על שלושת הפתרונות, ומכיוון שהפתרון השני והשלישי יצאו קרובים זה לזה צירפתי גרף נוסף שמתאר רק אותם.

נשים לב כי לא ניתן להגיד שמספר האפסים משפיע על מהירות הריצה- ההבדל בין זמני הגרף אינו מונוטוני- כלומר, כאשר יש רק אחדים שלושת הפתרונות רצים מהר יותר מכשחצי מהספרות הן 1, וכשחצי מהספרות הן 1 הפתרונות רצים יותר מכשיש יותר אפסים מאחדים

ד. הלולאה המתוארת תרוץ 100**2 פעמים. כדי לדעת כמה זמן זה ייקח, נבדוק כמה זמן ייקח לריצה אחת של הלולאה, כלומר num=1. יצא שריצה אחת לוקחת 0.00000929999999846582 שניות, נכפיל את זה פי 100**2 ונקבל שזה ייקח שנים- אפילו אלפי שנים.

הסיבה לכך שהלולאה בסעיף א רצה בזמן קצר בהרבה היא שהלולאה בסעיף א רצה על ספרות המספר, ולא על המספר עצמו: עבור הקלט 9 לדוגמא, הלולאה שבסעיף א תרוץ פעם אחת, והלולאה הנוכחית תרוץ 9 פעמים

6. ב. זמן הריצה של הפונקציה עבור limit=10000 ורשימת הראשוניים מסעיף א הוא 9.8192495שניות