

תרגיל בית מספר 1 - להגשה עד 26/10/2021 בשעה 23:55

קיראו בעיון את הנחיות העבודה וההגשה המופיעות באתר הקורס, תחת התיקיה assignments. חריגה מההנחיות תגרור ירידת ציון / פסילת התרגיל.

הגשה:

- תשובותיכם יוגשו בקובץ pdf ובקובץ py בהתאם להנחיות בכל שאלה.
- התשובות בקובץ ה pdf חייבות להיות מוקלדות ולא בכתב יד.
- השתמשו בקובץ השלד skeleton1.py כבסיס לקובץ ה py אותו אתם מגישים.
לא לשכוח לשנות את שם הקובץ למספר ת"ז שלכם לפני ההגשה, עם סיומת py.
- בסה"כ מגישים שני קבצים בלבד. עבור סטודנטית שמספר ת"ז שלה הוא 012345678 הקבצים שיש להגיש הם hw1_012345678.pdf ו-hw1_012345678.py.
- בנוסף, את הפונקציה שבשאלה 6 מגישים בקובץ נפרד דרך רכיב משוב עמיתים במודל. את הקובץ יש להגיש ללא שינוי שמו: max_even_seq.py.
- הקפידו לענות על כל מה שנשאלתם.
- בכל שאלה, אלא אם מצוין אחרת באופן מפורש, ניתן להניח כי הקלט תקין.
- תשובות מילוליות והסברים צריכים להיות תמציתיים, קולעים וברורים.
להנחיה זו מטרה כפולה:
 1. על מנת שנוכל לבדוק את התרגילים שלכם בזמן סביר.
 2. כדי להרגיל אתכם להבעת טיעונים באופן מתומצת ויעיל, ללא פרטים חסרים מצד אחד אך ללא עודף בלתי הכרחי מצד שני. זוהי פרקטיקה חשובה במדעי המחשב.

דוגמה לפונקציה

בחלק מהשאלות בתרגיל זה הנכם מתבקשים להגיש תוכניות בפיתון. את התוכניות יהיה עליכם להגיש כפונקציות, נושא שילמד בהרחבה בשבוע השני של הסמסטר. אולם פתרון כל השאלות לא מחייב הבנה של נושא זה, ולכן אפשר וכדאי להתחיל לעבוד על התרגיל כבר עכשיו. כדי להקל עליכם, להלן דוגמה של פונקציה פשוטה שמקבלת מספר בודד כקלט ומחזירה כפלט באמצעות הפקודה return את ערכו של המספר כפול 2.

נשים לב למספר דרישות בכתיבת פונקציה:

1. הגדרת הפונקציה תתחיל במילה def ולאחריה שם הפונקציה
2. לאחר שם הפונקציה יפורטו הקלטים אותם היא מקבלת, מופרדים ע"י פסיק.
3. יש להקפיד על העימוד: קוד גוף הפונקציה יכתב Tab אחד פנימה ביחס לשורת def. הפונקציה תחזיר פלט ע"י כתיבת המילה return (לא print!!) ולאחריה הערך שיוחזר כאשר תופעל הפונקציה.

```
def double_my_num(x):  
    return 2*x
```

דוגמאות להפעלת הפונקציה הנ"ל:

```
>>> z = double_my_num(5) #won't work with print...  
>>> z  
10  
>>> double_my_num(10)  
20  
>>> a = 30  
>>> double_my_num(a)  
60
```

דוגמה נוספת לפונקציה שמקבלת שני פרמטרים מספריים x,y ומחזירה כפלט באמצעות הפקודה return את הערך $x*y$ (המכפלה של x ו y):

```
def mult_nums(x, y):  
    return x*y
```

דוגמאות להפעלת הפונקציה הנ"ל:

```
>>> y = mult_nums(5, 10)  
>>> y  
50  
>>> mult_nums(10, 3)  
30  
>>> a = 2  
>>> b = 6  
>>> mult_nums(a, b)  
12
```

שאלה 1

כפי שראיתם בהרצאה, ישנן בפייתון פונקציות שמשייכות למחלקה מסויימת, למשל למחלקת המחרוזות (str). באינטרפרטר IDLE, אם תכתבו "str." ותלחצו על המקש tab, תיפתח חלונית עם מגוון פונקציות המשייכות למחלקת המחרוזות. כמובן, אפשר למצוא תיעוד רב על פונקציות אלו ואחרות ברשת. כמו כן אפשר להשתמש בפונקציה help של פייתון. למשל הפקודה help(str.title) תציג הסבר קצר על הפונקציה str.title שראיתם בהרצאה.

הערה כללית:

פונקציות של מחלקות ניתן להפעיל בשני אופנים שקולים. אם נסמן ב-C את שם המחלקה (למשל המחלקה str), וב-c_obj אובייקט קונקרטי מהמחלקה C (למשל מחרוזת "abc"), אז שתי הדרכים הן:

- C.func(c_obj,...), כלומר הפרמטר הראשון הוא c_obj ואחריו יתר פרמטרים, אם דרושים.
- c_obj.func(...), כלומר האובייקט c_obj לא מופיע בתוך הסוגריים אלא לפני שם הפונקציה.

להלן הדגמה על המחלקה str:

```
>>> course_name = "introduction to computer science"
>>> str.title(course_name)
'Introduction To Computer Science'
>>> course_name.title()
'Introduction To Computer Science'
```

מצאו שלוש פונקציות הקיימות במחלקה str שאינן קיימות במחלקה list, הדגימו אותן על המחרוזת "xyzw", כלומר צרפו לקובץ ה-pdf שלכם העתק (או צילום מסך) של הפקודות שהצתם ב-IDLE. כעת, מצאו שלוש פונקציות הקיימות במחלקה list שאינן קיימות במחלקה str. הדגימו אותן באופן דומה על הרשימה ['x', 'y', 'z', 'w']. הפעילו כל פונקציה בשתי השיטות (1) ו-(2).

הערה: המושגים "מחלקה" ו"אובייקט" יוסברו יותר לעומק בהמשך הקורס

שאלה 2

בכיתה ראיתם קוד בפייתון לחישוב ספרת ביקורת בתעודת זהות.
האלגוריתם לחישוב ספרת ביקורת בת"ז ישראלית מתואר [בקישור הזה](#). להלן הקוד שראיתם:

```
def control_digit(ID):  
    """ compute the check digit in an Israeli ID number,  
        given as a string of 8 digits  
    """  
  
    assert isinstance(ID, str)  
    assert len(ID) == 8  
  
    total = 0  
    for i in range(8):  
        val = int(ID[i]) # converts char to int  
        if i%2 == 0:      # even index (0,2,4,6)  
            total += val  
        else:             # odd index (1,3,5,7)  
            if val < 5:  
                total += 2*val  
            else:  
                total += ((2*val)%10) + 1 # sum of digits in 2*val  
                                           # 'tens' digit must be 1  
  
    total = total%10 # 'ones' (rightmost) digit  
    check_digit = (10-total)%10 # the complement modulo 10 of total  
                                # for example 42->8, 30->0  
  
    return str(check_digit)
```

- א. הריצו את הפונקציה על קלט לא תקין משני סוגים: (1) טיפוס שאינו מחרוזת, (2) מחרוזת באורך שונה מ-8. צפו בהודעת השגיאה המתקבלת. הסבירו בקובץ ה-pdf בקצרה את מהות השגיאה.
- ב. הוסיפו לקובץ ה-pdf שתי טבלאות מעקב אחר המשתנים בתוכנית המופיעה מעלה, טבלה עבור כל אחד משני הקלטים הבאים:
- a. "87654321" (כלומר הרצת הפקודה `(control_digit("87654321"))`).
- b. מספר תעודת הזהות האישי שלכם.
- הטבלה תיראה כך (בעמוד הבא):

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2022

iteration	i	ID[i]	val	total
1				
2				
...				
8				

שימו לב: בכל שורה יש לרשום את ערכי המשתנים בסוף האיטרציה הרלוונטית. למשל בשורה הראשונה (iteration 1) יש לרשום את ערכי המשתנים ברגע סיום האיטרציה הראשונה של לולאת ה-for. לפיכך בשורה 8 יופיעו ערכי המשתנים בסיום הלולאה ("רגע לפני" ביצוע הפקודה שמופיעה אחרי הלולאה).
ראו דוגמה בקובץ סיכום תרגול מספר 1 באתר הקורס. אין צורך להסביר כיצד הפונקציה פועלת.

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2022

שאלה 3

נדון בבעיה החישובית הבאה (שראינו בתרגול): בהינתן מספר שלם חיובי num , נרצה לדעת כמה פעמים מופיעה בו הספרה 0. למשל עבור הקלט 10030 הפלט המתאים הוא 3. מטרתנו בשאלה היא להשוות את זמני הריצה של שלושה פתרונות אפשריים לבעיה זו (הערה: אנו נדון בבעיה הנ"ל ובשלושת הפתרונות הללו גם בתרגול הראשון/שני, אבל אפשר להתחיל לפתור את השאלה כבר לאחר התרגול הראשון). לפניכם שלוש פונקציות הפותרות את הבעיה:

פתרון ראשון:

```
def zeros(num): #1st solution
    m = num
    cnt = 0
    while m > 0:
        if m % 10 == 0:
            cnt = cnt + 1
        m = m // 10
    return cnt
```

פתרון שני:

```
def zeros2(num): #2nd solution
    cnt = 0
    snum = str(num) #num as a string
    for digit in snum:
        if digit == "0":
            cnt = cnt + 1
    return cnt
```

פתרון שלישי:

```
def zeros3(num): #3rd solution
    cnt = str.count(str(num), "0")
    return cnt
```

שלושת הפונקציות מחזירות את התשובה כפלט ולכן נוכל להדפיס את הפתרון (למשל, של הפונקציה הראשונה) ע"י הפקודות:

```
num = 2**127
result = zeros(num)
print(num, "has", result, "zeros")
```

כדי למדוד זמן ריצה של פקודה או סדרת פקודות, נשתמש במעין "סטופר":

- נוסף בראש התוכנית שלנו את הפקודה `import time`
- נוסף מייד לפני קטע הקוד שאת זמן הריצה שלו ברצוננו למדוד את הפקודה:
`t0 = time.perf_counter()`
- נוסף מייד לאחר קטע הקוד הנ"ל את הפקודה:
`t1 = time.perf_counter()`
- זמן הריצה של קטע הקוד הוא ההפרש `t1-t0`. נוו להציגו למשל כך:
`print("Running time: ", t1-t0, "sec")`

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2022

הסבר קצר: time היא מחלקה של פייתון המאפשרת ביצוע פקודות שונות הקשורות לזמנים. הפקודה import הכרחית על מנת להשתמש במחלקה (היא "מיבאת" אותה. ניתן במהלך הקורס בדוגמאות רבות ל"ייבוא" של מחלקות). למידע נוסף על המחלקה time: <https://docs.python.org/3/library/time.html>

- א. מדדו את זמן הריצה של 2 הפתרונות הראשונים עבור המספרים: 2^{1400} , 2^{600} , 2^{250} , 2^{100} .
(תזכורת: האופרטור $**$ הוא אופרטור החזקה, כלומר $x**y$ משוערך ל- x בחזקת y). ציינו מה היו זמני הריצה בטבלה שבה תהיה עמודה לכל אחד מהקלטים הנ"ל, וכן שורה עבור כל פתרון. הסבירו בקצרה את התוצאות (ובפרט התייחסו לקצב הגידול כתלות בגודל הקלט). ניתן, אם רוצים, להציג את התוצאות בגרף על מנת להקל על ההסבר.
שימו לב: על המדידה למדוד את זמן הריצה של הקריאה לפונקציה בלבד, ובפרט אין למדוד את הזמן של פקודות נוספות כגון הדפסת הפלט.
- ב. פונקציות מובנות של פייתון, כמו למשל str.count, ממומשות בד"כ באופן יעיל למדי, לעיתים אף באמצעות אלגוריתמים מסובכים יחסית. חיזרו על סעיף א' עבור הפתרון השלישי. מבלי להיכנס לפרטי המימוש של str.count, האם היא אכן יעילה יותר מבחינת זמן ריצה, בהשוואה לשני הפתרונות הראשונים?
- ג. עבור קלטים בעלי מספר ספרות דומה, האם יש לפלט עצמו, כלומר למספר האפסים בקלט, השפעה כלשהי על זמן הריצה של כל אחד מהפתרונות? ביחרו קלטים מתאימים לבדיקת הסוגייה, ציינו מהם הקלטים בהם השתמשתם, הראו את תוצאות המדידות, והסבירו מה היא מסקנתכם.
- ד. להלן לולאה פשוטה:

```
num = 2**1000
cnt = 0
for i in range(num):
    cnt = cnt + 1
```

תנו הערכה גסה לזמן שיקח ללולאה להסתיים. ציינו כל הנחה עליה התבססתם בהערכתכם. איך אתם מסבירים זאת, לאור העובדה שבסעיף א' לולאת ה-for של הפתרון השני רצה בזמן קצר באופן משמעותי?

שאלה 4

בשאלה זו נעבוד על ניתוח בסיסי של מחרוזות. בשאלה חמישה סעיפים, ובכל סעיף יש לממש פונקציה אחת. בכל הסעיפים הקלט לפונקציה מכיל את המחרוזת text. בסעיף ב' ישנם קלטים נוספים אשר יפורטו בסעיף עצמו. שימו לב, בסעיף א' תידרשו גם לענות בקובץ ה pdf. לאורך כל השאלה ניתן להניח כי המחרוזת text מכילה ספרות (0 עד 9), אותיות קטנות באנגלית (a, b, c וכו') ורווחים בלבד. כמו כן, ניתן להניח כי בין כל שתי מילים במחרוזת מפריד רווח אחד בדיוק (מלבד המילה הראשונה במחרוזת שלפניה לא מופיע רווח והמילה האחרונה במחרוזת שאחריה לא מופיע רווח). רמז – בחלק מהסעיפים כדאי להשתמש בפונקציה split של המחלקה str. נסו להבין כיצד היא פועלת וכיצד היא יכולה לסייע לכם. ניתן לקרוא על הפונקציה ואופן השימוש בה על ידי חיפוש זריז בגוגל. שימו לב – בכל אחד מהסעיפים יתכן כי text היא המחרוזת הריקה. ודאו כי הפתרון שלכם מטפל גם במקרה קצה זה.

סעיף א'

הפונקציה num_different_letters(text) תחזיר כפלט את מספר האותיות (a,b,c וכו') השונות אשר מופיעות במחרוזת. שימו לב לא לספור גם ספרות (0 עד 9) שעשויות להופיע. רמז: חשבו איך להשתמש במשתנה chars אשר ניתן לכם בקובץ השלד. דוגמאות הרצה:

```
>>> num_different_letters("aa bb cccc dd ee fghijklmnopqrstuvwxyz")
26
>>> num_different_letters("aaa987654321000000000")
1
```

בנוסף, הסבירו בקובץ ה pdf כיצד הייתם משנים את הפתרון שלכם כדי לבדוק כמה אותיות (a,b,c וכו') וספרות (0 עד 9) שונות מופיעות במחרוזת.

סעיף ב'

הפונקציה replace_char(text, old, new) תקבל כקלט מחרוזת text ושני תווים בודדים במשתנים old ו-new. הפונקציה תחזיר מחרוזת זהה ל-text כאשר כל מופע של התו old מוחלף במופע של התו new. הנחיה: בסעיף זה אין להשתמש בפונקציות מובנות של המחלקה str. בפרט, במחלקה str יש פונקציה בשם replace שמבצעת בדיוק את הנ"ל, ואין להשתמש בה. דוגמאות הרצה:

```
>>> replace_char('abcdabcd', 'a', 'x')
'xbcdxbcd'
>>> replace_char('xyzwxyz', 'a', 't')
'xyzwxyz'
```

ציינו בקובץ ה-pdf כיצד ניתן להשתמש במתודה זו כדי למחוק את כל המופעים של תו מסויים ממחרוזת:

סעיף ג'

הפונקציה `longest_word(text)` תחזיר כפלט את אורך המילה הארוכה ביותר במחרוזת `text`. לצורך הפשטות, נניח כי מילה הוא כל רצף תווים המופרד ברווח במחרוזת (בין אם הוא מכיל תווים באנגלית או מספרים).
דוגמאות הרצה:

```
>>> longest_word('abcd xyz')
4
>>> longest_word('a 12b 34cd 5678efg zzz')
7
```

סעיף ד'

הפונקציה `to_upper(text)` תחזיר מחרוזת זהה ל-`text` כאשר כל מופע של אות אנגלית יוחלף באות הרישית המקבילה לה.

הנחיה: בסעיף זה אין להשתמש בפונקציות מובנות של המחלקה `str`. בפרט, במחלקה `str` יש פונקציה בשם `upper` שמבצעת בדיוק את הנ"ל, ואין להשתמש בה.

רמז: מומלץ להשתמש במחרוזות `lower` ו-`upper` המופיעות בגוף הפונקציה בקובץ השלד.

דוגמאות הרצה:

```
>>> to_upper('abc')
'ABC'
>>> to_upper('abc123')
'ABC123'
>>> to_upper('123')
'123'
```

שאלה 5

בשאלה זו נממש "מחשבון מחרוזות" בסיסי. הפונקציה calc תקבל כקלט מחרוזת expression המכילה ביטוי מהצורה הבאה:

$$a_0 \oplus a_1 \oplus a_2 \oplus \dots$$

כאשר \oplus היא אחת מבין הפעולות: $+$, $*$ (כלומר: שכפול מחרוזת מספר שלם חיובי של פעמים או שרשור של זוג מחרוזות) וכל a_i יפוענח באופן הבא:

- הוא מתחיל ונגמר בתו ' ' (גרש בודד).
- אם לפניו מופיעה הפעולה $+$ אז נפרש אותו כמחרוזת תווים (יכולה להכיל אותיות, מספרים, $+$, $*$, ורווחים)
- אם לפניו מופיעה הפעולה $*$ אז נפרש אותו כמספר שלם חיובי
- a_0 תמיד יפורש כמחרוזת (במידה שהוא קיים)

שערוך הביטוי expression יהיה התוצאה של הפעלת הפעולות החשבוניות על תתי הביטויים לפי סדר הופעתם (שימו לב – בתרגיל זה אין לחשב את הפעולות על פי סדר הפעולות של פייתון אלא משמאל לימין).

לדוגמא, הביטוי "'a' * '2' + 'b' * '2'" ישוערך למחרוזת "aabaab" (ולא "aabb") לפי הלוגיקה הבאה: נתחיל מהמחרוזת "a", נשכפל אותה פעמיים, נשרשר לה את "b" ולבסוף נשכפל את התוצאה פעמיים.

שימו לב: יש בפייתון מספר דרכים (שקולות) ליצור מחרוזות. בתרגיל זה נשתמש במרכאות כפולות "...". כיוון שאלו יאפשרו לנו להשתמש במרכאות יחידות '...' ללא שימוש בתווים מיוחדים. אתם מוזמנים לקרוא [כאן](#) על דרכים נוספות ליצור מחרוזות ולהתנסות בהבדלים (הקטנים) ביניהם בעצמכם.

דוגמאות הרצה:

```
>>> calc("'123321'*'2'")
"123321123321"
>>> calc("'Hi there '*3'+you2'")
"Hi there Hi there Hi there you2"
>>> calc("'hi+fi'*'2'*'2'")
"hi+fihi+fihi+fihi+fi"
```

הנחיות:

- ניתן להניח כי אין רווחים בין הפעולות וה- a_i
- פלט הפונקציה צריך להיות מטיפוס str
- ניתן להניח כי המחרוזת expression תקינה (מקיימת את הפורמט שמוגדר בשאלה)
- אין להשתמש בספריות חיצוניות או בפקודות שיערוך מובנות (eval)

שאלה 6

שאלה זו תשמש אתכם לצורך תהליך של משוב עמיתים (peer review) שיתבצע במהלך הסמסטר.

מיומנויות הקשורות לתכנות לא מסתכמות רק בכתיבת קוד "שעובד", אלא כוללות גם קריאת קוד והבנתו, איתור שגיאות ותיקונן, בניית מערך טסטים מקיף, הערכת יעילות, תכנון מקדים של הפתרון, וכו'. **המוטיבציה למשוב עמיתים היא להפגיש אתכם עם מיומנויות אלו בשלב מוקדם של הלימודים ולהיעזר לשם כך בחבריכם וחברותיכם לספסל הלימודים.**

בשלב זה של התהליך, אתם נדרשים לממש את הפונקציה שמפורטת להלן. כדי שתהליך משוב העמיתים יהיה מועיל ואף מהנה, אתם מתבקשים לעבוד על השאלה באופן עצמאי וללא עזרה מחברים וחברות לקורס או מסגל הקורס. על מנת לעודד זאת, **הניקוד להגשה של שאלה זו יינתן על עצם הגשת פתרון שנכתב עצמאית אפילו אם אינו נכון לגמרי או אם יש בו בעיות מסוגים שונים**. במילים אחרות, אנחנו מבקשים שתכתבו ותגישו את הפתרון הטוב ביותר שביכולתכם בשלב זה על בסיס מה שלמדתם עד כה.

בשאלה זו נכתוב פונקציה שבהינתן מספר שלם $n \geq 0$ מחשבת מהו האורך המקסימלי של רצף ספרות זוגיות ב- n . למשל, עבור $n = 23300247524689$ האורך המקסימלי של רצף ספרות זוגיות הוא 4 (ישנם שני רצפים שמתאימים לאורך זה: הרצף 0024 שמתחיל באינדקס 3 והרצף 2468 שמתחיל באינדקס 9).

הערות:

- במקרה שהמספר אינו מכיל ספרות זוגיות האורך המקסימלי הינו 0
- ניתן להניח כי הקלט תקין ואין צורך לבדוק זאת

דוגמת הרצה:

```
>>> max_even_seq(23300247524689)
```

4

ממשו את הפונקציה `max_even_seq(n, k)` שבקובץ השלד `max_even_seq.py` (ולא בקובץ השלד שבו נמצאות שאר שאלות הקוד) על פי ההנחיות לעיל. לאחר מכן, הגישו אותו ברכיב משוב עמיתים במודל (ולא בתיבת ההגשה של תרגיל בית 1), כאשר כותרת ההגשה היא **ארבע הספרות האחרונות בתעודת הזהות שלכם**. במהלך העבודה ניתן לערוך את הקובץ כרצונכם (למשל, כדי לבדוק את נכונות הקוד שלכם), אבל על ההגשה הסופית לכלול אך ורק את המימוש של הפונקציה ללא קטעי קוד נוספים. כמו כן, כדי לאפשר אונימיות בהמשך תהליך משוב העמיתים אין להוסיף להגשה הסופית פרטים מזהים מלבד כותרת ההגשה הנ"ל (בפרט, שם הקובץ המוגש צריך להישאר `max_even_seq.py`).

סוף.