אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2020

תרגיל בית מספר 1 - להגשה עד 25/03/2020 בשעה 23:55

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

הנחיה כללית: אין להגיש תשובות בכתב יד (סרוקות)

הנחיות והערות ספציפיות לתרגיל זה:

- אפשר וכדאי להתחיל לעבוד על התרגיל כבר בשבוע הראשון לסמסטר, לאחר ההרצאה + התרגול הראשונים.
- הגשה:
  - תשובות לשאלות 1, 2 , 4 יש להגיש בקובץ pdf יחיד. **אין לסרוק תשובות מילוליות ולצרפן לקובץ ה pdf.**
  - קוד משאלות 3, 5 ו- 6 יש לממש בקובץ השלד (skeleton1.py) המצורף לתרגיל זה. אין לצרף לקובץ ה-py את הקוד ששימש לפתרון יתר השאלות.
- לא לשכוח לשנות את שם הקובץ לשם הדרוש לפני ההגשה, עם סיומת py.
- בקובץ השלד בשאלות 5 ו-6 מופיע הפקודה pass בגוף הפונקציה. **יש למחוק פקודה זו** ולכתוב במקומה את הקוד.
- 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.
- הקפידו לענות על כל מה שנשאלתם.
- את הקוד שתידרשו לכתוב בקובץ השלד תכתבו בצורת פונקציות על מנת להקל על בדיקת התרגיל. נושא הפונקציות יוסבר בהמשך באופן מעמיק ומסודר. דוגמה לפונקציה תופיע בתחילת התרגיל.
- תשובות מילוליות והסברים צריכים להיות תמציתיים, קולעים וברורים.
  להנחיה זו מטרה כפולה:
  - על מנת שנוכל לבדוק את התרגילים שלכם בזמן סביר.
  - 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 בפייתון. 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 ייכתב 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
```

הערה חשובה לגבי שאלה 3: פונקציה שאינה צריכה להחזיר ערך (כמו זו משאלה 3) יכולה שלא לכלול פקודת return כלל.

שאלה 1

כפי שראיתם בהרצאה, ישנן בפייתון פונקציות שמשויכות למחלקה מסויימת, למשל למחלקת המחרוזות (str).

באינטרפרטר IDLE, אם תכתבו "str." ותלחצו על המקש tab, תיפתח חלונית עם מגוון פונקציות המשויכות 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 "abcd", כלומר צרפו לפתרון שלכם העתק (או צילום מסך) של הפקודות שהרצתם ב- IDLE.

כעת, מצאו שלוש פונקציות הקיימות במחלקה list שאינן קיימות במחלקה str. הדגימו אותן באופן דומה על הרשימה ['a', 'b', 'c', 'd'].

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:

```
def control_digit(ID):
    """ compute the check digit in an Israeli ID number,
        given as a string """

    total = 0
    for i in range(8):
        val = int(ID[i]) #converts a char to its numeric integer value
        if i % 2 == 0:
            total = total+val
        else:
            if val < 5:
                total += 2*val
            else:
                total += (2*val % 10) + 1 # sum of digits in 2*val
    total = total % 10
    check_digit= (10 - total) % 10  # the complement mod 10 of sum

    return str(check_digit)
```

האלגוריתם לחישוב ספרת ביקורת בת״ז ישראלית מתואר בקישור הזה.

הוסיפו לקובץ ה pdf שתי טבלאות מעקב אחר המשתנים בתוכנית המופיעה מעלה, טבלה עבור כל אחד משני הקלטים הבאים:

1. "12345678" (כלומר ביצוע הפקודה control_digit("12345678"))
2. מספר תעודת הזהות האישי שלכם

הטבלה תיראה כך:

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 את הפונקציה.

שאלה 3

במשימה זו תכירו פעולות בסיסיות על קבצים, כגון פתיחת קובץ, סגירת קובץ, קריאה וכתיבה לקובץ.

היעזרו בתיעוד שמופיע בקישור הבא (בסעיף 7.2): https://docs.python.org/3/tutorial/inputoutput.html

ובהסבר על עבודה עם קבצים המופיע בהמשך.

השלימו את הקוד בקובץ השלד תחת הפונקציה `max_word_len(filename)` שמקבלת שם קובץ שנמצא באותה תיקייה שמכילה את קובץ השלד ויוצרת קובץ פלט חדש בתיקייה זו בשם output.txt שבו כל שורה מכילה את האורך המקסימלי של מילה בשורה זו בקובץ הנתון `filename`. אין לשנות את שורות הקוד שכבר מומשו עבורכם.

רמז: היעזרו בפונקציה split של המחלקה str. מומלץ לקרוא את התיעוד שלה עד תומו. תיעוד רלוונטי נמצא, למשל, בקישור הבא: https://docs.python.org/3.6/library/stdtypes.html. בפרט, קראו על התנהגות הפונקציה split כאשר מפעילים אותה על מחרוזת ללא פרמטרים נוספים.

הערות:

- יש להקפיד שכל תוצאה תופיע בשורה נפרדת בקובץ הפלט
- בשאלה זו נניח כי בין כל שתי מילים מפריד לפחות רווח אחד. לדוגמא, המשפט הבא:

  "He is a self-centered person."

  מכיל חמש מילים, והן: "He", "is", "a", "self-centered", "person.".

  שימו לב – למספר רב של רווחים (שניים או יותר) נתייחס כאילו היו רווח בודד, כלומר, לא נתייחס ל"מילים הריקות" שבין שני רווחים כמילה. לדוגמא: "Hello world" היא מחרוזת המכילה שתי מילים בלבד, על אף הרווחים בין שתי המילים והרווחים בסוף המחרוזת.

  כדוגמא נוספת, המשפט "Hello . world" מכיל שלוש מילים, והן: "Hello", ".", "world", כלומר, גם סימן פיסוק המופרד **בין שני רווחים** יחשב כמילה.
- עבור שורה שלא מכילה מילים כלל נכתוב לקובץ 0

מצורף לתרגיל קובץ טקסט dorian_gray.txt שמכיל את תוכן הספר "The Picture of Dorian Gray" מאת אוסקר ווילד. שימרו קובץ זה באותה תיקיה בה שמרתם את קובץ השלד skeleton1.py.

וודאו שעבור קובץ זה, המספרים שאתם מקבלים זהים למספרים הנתונים בהמשך.

לדוגמא, לאחר הפעלת הפונקציה באופן הבא:

```
max_word_len("dorian_gray.txt")
```

חמש השורות הראשונות של output.txt תהיינה:

```
7
2
5
7
7
```

כמו כן, חמש השורות האחרונות של output.txt תהיינה:

```
9
9
10
0
11
```

<u>הסבר קצר על עבודה עם קבצים:</u>

1. על מנת לפתוח קובץ קיים בשם myfile.txt <u>לקריאה</u> נשתמש בפקודה:
```
f = open("myfile.txt", "r")
```
שימו לב שההנחה היא שהקובץ myfile.txt נמצא בתיקיה הנוכחית.
המשתנה f מחזיק כעת אובייקט מטיפוס קובץ.
על מנת לפתוח קובץ בשם myfile.txt <u>לכתיבה</u> נשתמש בפקודה:
```
f = open("myfile.txt", "w")
```
אם הקובץ אינו קיים בתיקיה הנוכחית יווצר קובץ כזה.
2. בסיום העבודה עם קובץ חובה עליכם לסגור את הקובץ. קובץ אליו כתבתם שלא יסגר עלול להכיל רק חלק מהמידע שנכתב אליו, אם בכלל. סגירת קובץ שמוחזק ע"י המשתנה f תתבצע באופן הבא:
```
f.close()
```
3. על מנת לקרוא שורות מקובץ שנפתח לקריאה ניתן להשתמש בלולאה הבאה. שימו לב שהקטע הקוד הבא קורא שורה שורה של הקובץ, לתוך משתנה בשם line (line היא מטיפוס מחרוזת) ואז מדפיס את המשתנה.
```
for line in f:
    print(line)
```
תו זה מייצג ירידת שורה (ומיוצג במקלדת שלכם ע"י מקש ה- Enter).
4. על מנת לכתוב שורה לקובץ שנפתח לכתיבה ניתן להשתמש בפקודה הבאה. שימו לב להוסיף את התו "\n" בתום המחרוזת אותה תרצו לכתוב לקובץ כדי לסמן את סיום השורה.
```
f.write("An example\n")
```

שאלה 4

נדון בבעייה החישובית הבאה: בהינתן מספר שלם חיובי num, נרצה לדעת כמה פעמים מופיעה בו הספרה 0. למשל עבור הקלט 10030 הפלט המתאים הוא 3.

הקלט יינתן באמצעות הפקודה input:

```
num = int(input("Please enter a positive integer: "))
```

(לאחר ביצוע פקודה זו, המשתנה num יכיל את המספר אותו הכניס המשתמש.)

מטרתנו בשאלה היא להשוות את זמני הריצה של שלושה פתרונות אפשריים לבעייה זו (הערה: אנו נדון בבעייה הנ"ל ובשלושת הפתרונות הללו גם בתרגול הראשון/שני, אבל אפשר להתחיל לפתור את השאלה כבר לאחר התרגול הראשון):

פתרון ראשון:

```
#1st solution
m = num
cnt = 0
while m > 0:
    if m % 10 == 0:
        cnt = cnt + 1
    m = m // 10
```

פתרון שני:

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

פתרון שלישי:

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

בשלושת הפתרונות הפלט הרצוי יימצא לבסוף במשתנה cnt:

```
print(num, "has", cnt, "zeros")
```

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

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

```
print("Running time: ", t1-t0, "sec")
```

(המשך השאלה בעמוד הבא)

הסבר קצר: time היא מחלקה של פייתון המאפשרת ביצוע פקודות שונות הקשורות לזמנים. הפקודה import הכרחית על מנת להשתמש במחלקה (היא "מיבאת" אותה. ניתקל 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**200, 2**400, 2**800, 2**1600.
ציינו מה היו זמני הריצה בטבלה שבה תהיה עמודה לכל אחד מהקלטים הנ"ל, וכן שורה עבור כל פתרון.
הסבירו בקצרה את התוצאות (התייחסו לקצב הגידול כתלות בגודל הקלט). ניתן, אם רוצים, להציג את התוצאות בגרף על מנת להקל על ההסבר.
שימו לב: כדי לנטרל השפעות של פקודות שקשורות להשגת הקלט והצגת הפלט, ואינן חלק מהפתרון עצמו, זמן הריצה לא יכלול את שורת ה- input בהתחלה ואת הדפסת הפלט בסוף.

ב. פונקציות מובנות של פייתון, כמו למשל str.count, ממומשות בד"כ באופן יעיל למדיי, 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**100
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 5

במשחק הילדים הנודע "7 בום!" המשתתפים צריכים לנקוב במספרים טבעיים בסדר עולה, אך בכל פעם שמגיעים למספר שמתחלק ב- 7 או שמופיעה בו הספרה 7, יש לנהוג כך:

- אם המספר מתחלק ב-7 אך אין בו מופע של הספרה 7 יש לצעוק בִּמְקומו "boom!" (למשל, יש לצעוק "boom!" במקום 14)
- אם המספר מכיל מופע אחד או יותר של הספרה 7 אך הוא לא מתחלק ב-7 יש לצעוק "boom-boom-...-boom!" כמספר המופעים של הספרה 7 במספר (למשל, יש לצעוק "boom-boom-boom-boom!" במקום 77717 ויש לצעוק "boom-boom-boom!" במקום 7717)
- אם מספר גם מתחלק ב- 7 וגם מכיל מופע אחד או יותר של הספרה 7 (למשל המספר 7 או המספר 770) יש לצעוק "bada-boom!".

הגירסה המוכללת של המשחק (שהולכת ונהיית פופולרית יותר ויותר במקום נידח כלשהו) קרויה "k בום!" והגדרתה זהה לנ"ל כאשר k הינו מספר שלם כלשהו בין 1 ל 9 כולל.

השלימו בקובץ השלד את הפונקציה `k_boom(start, end, k)`, שמחזירה מחרוזת של כל המספרים בין start ל-end (כולל) בהתאם לחוקי המשחק "k בום!", כאשר *start, end* ו- *k* הינם קלטים של הפונקציה. כל ערך יופרד בתו רווח " " מהערך הבא אחריו.

להלן מספר דוגמאות הרצה:

```
>>> k_boom(10,20, 7)
'10 11 12 13 boom! 15 16 boom! 18 19 20'

>>> k_boom(70,80, 7)
'bada-boom! boom! boom! boom! boom! boom! boom! bada-boom! boom!
boom! 80'

>>> k_boom(797,802, 7)
'boom-boom! bada-boom! boom! 800 801 802'
```

עזרה:

דרך פשוטה לבדוק את מספר המופעים של ספרה במספר היא המרת המספר למחרוזת, ושימוש בפקודה:

```
num_string.count(digit_string)
```

כאשר num_string ו-digit_string הן שתי מחרוזות, הביטוי יחזיר את מספר המופעים של digit_string ב-num_string (ואפס אם הספרה לא מופיעה במספר בכלל)

הנחיות הגשה:

- הוסיפו את קוד הפתרון במקום המתאים בקובץ ה- py אותו אתם מגישים.

- הפונקציה תחזיר אך ורק את הערכים הדרושים, בדיוק לפי הפורמט המצויין בדרישות השאלה, ללא שום תוספות או הודעות. בפרט, המנעו מרווחים מיותרים ומסימני פיסוק. על מנת לקבל את מלוא הניקוד עליכם להקפיד על מילוי הנחיה זו.
- הפונקציה צריכה לעבוד נכון עבור כל $start \leq end$ טבעיים וכל $1 \leq k \leq 9$ טבעי.
- בעת בדיקת התרגילים הבודק עשוי לבדוק את הפונקציה עם ערכי *start, end* ו- k שונים.
- ניתן להניח כי בכל הבדיקות $1 \leq k \leq 9$ וכן כי $start \leq end$.

שאלה 6

בשאלה זו נעסוק בבדיקת השערת גולדבאך. השערת גודלבאך אומרת כי כל מספר זוגי גדול מ-2 הוא סכום של שני מספרים ראשוניים.

אחד הקבצים המצורפים לתרגיל הינו primes_lst.py שמכיל רשימה בשם primes (מטיפוס lst) של 10000 המספרים הראשוניים הראשונים החל מ-2. עליכם לדאוג שקובץ זה ישב באותה תיקייה בה יושב קובץ השלד. רשימה זו תהיה זמינה לכם בקובץ השלד ע"י גישה למשתנה primes.

א. השלימו בקובץ השלד את הפונקציה check_goldbach_for_num(n, primes_list) אשר מקבלת מספר זוגי גדול מ-2 בשם n ורשימת מספרים ראשוניים primes_list, ומחזירה True אם המספר מקיים את ההשערה עבור הקבוצה (כלומר, ישנם שני ראשוניים ברשימה אשר סכומם הוא n) ו- False אחרת. ניתן להניח כי הקלט תקין.

דוגמת הרצה:

```
>>> check_goldbach_for_num(10, [2, 3])
False
>>> check_goldbach_for_num(10, [2, 3, 5, 7])
True
```

ב. השלימו את הפונקציה check_goldbach_for_range(limit, primes_list) אשר מקבלת מספר גדול מ-2 בשם limit ורשימת מספרים ראשוניים primes_list ובודקת את ההשערה עבור כל המספרים הזוגיים הגדולים מ-2 עד ל- limit (לא כולל). כלומר, הפונקציה תחזיר True אם כל מספר זוגי הגדול מ-2 וקטן מ-limit ניתן להצגה כסכום של שני ראשוניים ברשימה primes_list ו-False אחרת.

דוגמת הרצה:

```
>>> check_goldbach_for_range(20, [2, 3, 5, 7, 11])
True
```

- בדקו את ההשערה עם limit=10000 ורשימת הראשוניים שיצרתם בסעיף א'. **כתבו בקובץ ה- pdf מהו זמן הריצה של הפונקציה.**

ג. כעת נרצה לאסוף סטטיסטיקות על המספרים שמקיימים את ההשערה ובפרט, כמה זוגות ראשוניים יכולים להרכיב מספר מסוים. למשל, 7+23=30 אבל גם 11+19=30. כלומר יתכנו מספר זוגות ראשוניים שמרכיבים את אותו מספר זוגי. כמו בסעיפים הקודמים, הניחו כי הקלט primes_list הינו רשימה של מספרים ראשוניים.

השלימו בקובץ השלד את הפונקציה check_goldbach_for_num_stats(n, primes_list) אשר תחזיר את מספר זוגות הראשוניים ב-primes_list שיכולים להרכיב מספר n המתקבל כקלט.

הערות: שימו לב לא לספור זוגות פעמיים!

- <u>דוגמאות הרצה (משתמשות ברשימת הראשוניים ששמורה במשתנה primes)</u>:

```
>>> check_goldbach_for_num_stats(20, primes)
2 # שני זוגות ראשוניים מרכיבים את 20
>>> check_goldbach_for_num_stats(10, primes)
2  # שני זוגות ראשוניים מרכיבים את 10
```

<u>סוף.</u>