אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2021

תרגיל בית מספר 1 - להגשה עד 18 במרץ בשעה 23:55

קיראו בעיון את הנחיות העבודה וההגשה המופיעות באתר הקורס, תחת התיקייה 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.
- מכיוון שניתן להגיש את התרגיל בזוגות, עליכם בנוסף למלא את המשתנה SUBMISSION_IDS שבתחילת קובץ השלד. רק אחת הסטודנטיות בזוג צריכה להגיש את התרגיל במודל.

- הקפידו לענות על כל מה שנשאלתם.

- 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 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
```

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 "xyzw", 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 ['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:

```
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. "87654321" (כלומר ביצוע הפקודה control_digit("87654321"))
2. מספר תעודת הזהות האישי של חבר הקבוצה שלכם שתעודת הזהות שלו מופיעה ראשונה במשתנה SUBMISSION_IDS בקובץ ה-py. במידה ואתם מגישים לבד, אז מספר תעודת הזהות 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 כיצד הפונקציה פועלת.

שאלה 3

נדון בבעיה 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 ולכן נוכל להדפיס את הפתרון ע"י הפקודה:

```
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**100, 2**300, 2**700, 2**1500.

תזכורת, המשמעות של ** היא חזקה. 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: 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**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 ארבעה סעיפים, ובכל סעיף יש לממש פונקציה אחת. בכל הסעיפים הקלט לפונקציה הוא מחרוזת text. 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 במתודה split של המחלקה str. נסו להבין כיצד היא פועלת וכיצד היא יכולה לסייע לכם.

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("aaa98765432100000000")
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 max_letter_count(text) תחזיר כפלט את מספר החזרות המקסימלי של אות (a,b,c וכו') ב text. שימו לב לא לספור גם ספרות (0 עד 9) שעשויות להופיע.

דוגמאות הרצה:

```
>>> max_letter_count("aa bb cccc dd ee fghijklmnopqrstuvwxyz")
4
>>> max_letter_count("aaa98765432100000000")
3
```

סעיף ג'

הפונקציה is_legal(text) תחזיר True אם text מייצגת משפט חוקי ו False אחרת. משפט חוקי הוא כזה שכל המילים בו חוקיות. מילה חוקית היא כזאת שמכילה או רק אותיות (a,b,c וכו') או רק ספרות (0 עד 9). כלומר, "hi" ו "34" הן מילים חוקיות אבל "hi34" אינה חוקית.

דוגמאות הרצה:

```
>>> is_legal("number 34 says hi to number 43")
True
>>> is_legal("454f says hi")
```

```
False
```

<u>סעיף ד'</u>

הפונקציה is_palindrome(text) תחזיר True אם text הוא פלינדרום ו False אחרת. פלינדרום היא מחרוזת טקסט אשר קריאתה משמאל לימין או מימין לשמאל היא זהה.

הנחיות:

- אין להשתמש ב slicing של מחרוזות (כלומר פקודה כגון text[::-1] המייצרת מחרוזת בה סדר התווים הפוך)
- יש להשתמש בלולאה אחת בלבד לצורך המימוש (while או for לבחירתכם)

דוגמאות הרצה:

```
>>> is_palindrome("1step on no pets1")
True
>>> is_palindrome("hello")
False
```

שאלה 5

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

$$a_0 \oplus a_1 \oplus a_2 \oplus \cdots$$

כאשר $\oplus$ היא אחת מבין הפעולות: $+,*$ (כלומר: כפל מחרוזת במספר שלם חיובי או חיבור של זוג מחרוזות) וכל a_i יפוענח באופן הבא:

- הוא מתחיל ונגמר בתו '.
- 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 משמאל לימין** (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", נכפיל אותה פי 2, נחבר לה את "b" ולבסוף נכפיל את התוצאה פי 2.

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 מחרוזת 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 (כמו eval)

שאלה 6

בשאלה זו נכתוב פונקציה שבהינתן מספר שלם אי-שלילי כלשהו $\boldsymbol{n}$ ומספר שלם $\boldsymbol{k}$ בין 1 ל-9 (כולל) מחשבת מהו האורך של רצף מקסימלי של ספרות ב-$\boldsymbol{n}$ אשר כל ספרה בו מתחלקת ב-$\boldsymbol{k}$ (ללא שארית).

למשל עבור $\boldsymbol{n} = \mathbf{23300247524689}$ ו-$\boldsymbol{k} = \mathbf{2}$, האורך המקסימלי של רצף ספרות שמתחלקות ב-2 הוא 4 (ישנם שני רצפים שמתאימים לאורך זה: הרצף 0024 שמתחיל באינדקס 3 והרצף 2468 שמתחיל באינדקס 9).

במקרה שהמספר אינו מכיל ספרות שמתחלקות ב-$\boldsymbol{k}$ האורך המקסימלי הינו 0.

דוגמאות נוספות:

- עבור $\boldsymbol{n} = \mathbf{1630860}$ ו-$\boldsymbol{k} = \mathbf{3}$, אורך הרצף המקסימלי הוא 3 (הרצף 630 שמתחיל באינדקס 1).
- עבור $\boldsymbol{n} = \mathbf{1630860}$ ו-$\boldsymbol{k} = \mathbf{8}$, אורך הרצף המקסימלי הוא 2 (הרצף 08 שמתחיל באינדקס 3).

ממשו את הפונקציה `max_div_seq(n,k)` שבקובץ השלד על פי ההנחיות לעיל.

<u>הערות</u>:

- שימו לב כי על כל ספרה ברצף להתחלק ב-k, כלומר, הרצף 122 איננו תקין עבור $k = 2$ מאחר ש-1 לא מתחלק ב-2 ללא שארית

<u>הנחיות</u>:

- הפונקציה מקבלת כקלט את המספר $\boldsymbol{n}$ והמספר $\boldsymbol{k}$
- ניתן להניח כי $\boldsymbol{n} \geq \mathbf{0}$ וכי $\mathbf{1} \leq \boldsymbol{k} \leq \mathbf{9}$
- הפונקציה תחזיר כפלט את אורך הרצף המקסימלי

<u>דוגמאות הרצה</u>:

```
>>> max_div_seq(23300247524689, 2)
4
>>> max_div_seq(1357, 2)
0
```

סוף.