אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2017

תרגיל בית מספר 1 - להגשה עד 30 במרץ בשעה 23:55

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

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.
- 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

להלן תוכנית קצרה בשפת Python:

```
res = 0
num = int(input("Enter a positive integer: "))
while num > 0:
    res = res + (num % 10)
    num = num // 10
print(res)
```

הסבר קצר: תפקיד הפונקציה input() הוא לקלוט ממשתמש התוכנית ערכים (קלט) הנחוצים להמשך ביצוע התוכנית. הביטוי בתוך הסוגריים **"Enter a positive integer: "** הוא מחרוזת שתודפס למשתמש לפני שיידרש להכניס קלט. הפונקציה input() קולטת את הערך שהכניס המשתמש כמחרוזת. ערך זה עובר המרה מטיפוס מחרוזת לטיפוס מספר שלם (ע"י הפעלת הפונקציה int() שראינו בתרגול), ואז המשתנה num מקבל ערך זה.

א. מה תדפיס התוכנית עבור הקלט 542227?

ב. באופן כללי, מה מדפיסה התוכנית בהינתן מספר שלם חיובי כלשהו n?

ג. מה תדפיס התכנית עבור מספר שלילי n ומדוע?

ד. כאשר רוצים לתת לתוכנית קלט כביטוי אריתמטי, כמו למשל 3**222 + 47, מתעוררת בעייה כלשהי. נסו להריץ, ציינו מהי בדיוק הפקודה שנכשלה וגרמה לבעייה, והסבירו בקצרה מה הבעייה.

ה. הפונקציה `eval` של פייתון יכולה לסייע בקלות לפתרון הבעייה הנ"ל. נסו להבין מה עושה הפונקציה (מומלץ ע"י ניסוי וטעייה וקצת חיפוש. הפונקציה מקבלת בסוגריים פרמטר אחד מטיפוס מחרוזת), והיעזרו בה כדי להריץ את התוכנית הנ"ל על 3**121 + 17. מה מודפס?

ו. מה קורה כאשר משנים את התנאי `while num > 0` ל- `while num >= 0`? הסבירו בקצרה.

שאלה 2

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.

<u>הערה כללית</u>:

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).

<u>הערה</u>: המושגים "מחלקה" ו"אובייקט" יוסברו יותר לעומק בהמשך הקורס

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.

השלימו את הקוד בקובץ השלד תחת הפונקציה `avg_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 הממוצע של מילה באותה השורה בקובץ הנתון `filename`. אין לשנות את השורות 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 .

הערה: בשאלה זו נניח כי בין כל שתי מילים מפריד לפחות רווח אחד.

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 `avg_word_len("dorian_gray.txt")`, 5 השורות הראשונות של output.txt יהיו:

```
4.4
0
2.0
0
5.0
```

כמו כן, 5 השורות האחרונות של output.txt יהיו:

```
4.75
4.3076923076923075
4.5
0
4.833333333333333
```

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")
```

(המשך השאלה בעמוד הבא)

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.clock()`
- נוסיף מייד לאחר קטע הקוד הנ"ל את הפקודה: `t1 = time.clock()`
- זמן הריצה של קטע הקוד הוא ההפרש `t1-t0`. נוח להציגו למשל כך:

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

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). הפונקציה clock של המחלקה מחזירה את הזמן שחלף מאז הפעלתה בפעם הראשונה.
(למידע נוסף: http://docs.python.org/release/1.5.1/lib/module-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 (אפשר להציג זאת בגרף), והסבירו בקצרה את התוצאות (התייחסו לקצב הגידול כתלות בגודל הקלט).
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 והדפסת 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 לצעוק בִּמְקום "boom!". כמו כן, אם מספר גם מתחלק ב- 7 וגם מכיל את הספרה 7 (למשל המספר 7) יש לצעוק "boom-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(n, k)`, שמחזירה מחרוזת של כל המספרים בין 1 ל- n (כולל) בהתאם לחוקי המשחק "k בום!", כאשר n ו- k הינם קלטים של הפונקציה. כל ערך יופרד בתו רווח " " מהערך הבא אחריו.

לדוגמה, עבור k=7 ו- n=15, הפעלת `k_boom(15, 7)` תחזיר <u>בדיוק</u> את המחרוזת הבאה:

"1 2 3 4 5 6 boom-boom! 8 9 10 11 12 13 boom! 15"

כלומר:

```
>>> k_boom(15, 7)
"1 2 3 4 5 6 boom-boom! 8 9 10 11 12 13 boom! 15"
```

<u>עזרה</u>:

דרך פשוטה לבדוק האם ספרה מופיעה במספר היא המרת המספר למחרוזת, ושימוש בפקודה:

```
if x in y:
```

כאשר x ו- y הן שתי מחרוזות, והביטוי x in y שווה True אם x מוכלת ב- y, אחרת False.

<u>הנחיות הגשה</u>:

- הוסיפו את קוד הפתרון במקום המתאים בקובץ ה- py אותו אתם מגישים.
- הפונקציה תחזיר אך ורק את הערכים הדרושים, <u>בדיוק</u> 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 n טבעי וכל $1 \leq k \leq 9$ טבעי.
- בעת בדיקת התרגילים הבודק עשוי לבדוק את הפונקציה עם ערכי n ו- k שונים.

שאלה 6

בשאלה זו נכתוב פונקציה שמחשבת מהו האורך של רצף מקסימלי של ספרות זוגיות במספר נתון כלשהו.

למשל עבור המספר 23300247524689 האורך המקסימלי הוא 4 (ישנם שני רצפים שמתאימים לאורך זה: הרצף 0024 שמתחיל באינדקס 3 והרצף 2468 שמתחיל באינדקס 9).

במקרה שהמספר אינו מכיל ספרות זוגיות האורך המקסימלי הינו 0.

כדי לענות על השאלה השלימו את הפונקציה `max_even_seq(n)` בקובץ השלד skeleton1.py.

הנחיות: המספר השלם האי-שלילי ייקלט לפונקציה שמופיעה כבר בקובץ השלד. בסיום יוחזר האורך הרצף המקסימלי.

דוגמאות הרצה:

```
>>> max_even_seq(233002475246 89)
4
>>> max_even_seq(1357)
0
```

סוף.