מבוא למחשב – שפת פייתון (234128)

מועד א' סמסטר חורף תש"ף

20 בפברואר 2020

משך המבחן: 180 דקות (שלוש שעות).
חומר עזר: אין להשתמש בכל חומר עזר בכתב, מודפס או אלקטרוני.

הנחיות והוראות:

- בדקו שיש 20 עמודים (4 שאלות) במבחן, כולל עמוד זה.
  החל מעמוד 17 העמודים ריקים ומיועדים לטיוטה.
- כתבו את התשובות על טופס המבחן בלבד, במקומות המיועדים לכך. שימו לב שהמקום המיועד לתשובה אינו מעיד בהכרח על אורך התשובה הנכונה. אך ניתן לכתוב כל תשובה במעט שורות קוד.
- יש לכתוב באופן ברור, נקי ומסודר. **אסור להשתמש בעפרון.** את עמוד השער גם חובה למלא בעט. אין לכתוב באדום בטופס הבחינה.
- בכל השאלות, הנכם רשאים להגדיר (לממש) פונקציות עזר כרצונכם, תחת מגבלות השאלה. אלא אם צוין אחרת במפורש בשאלה.
- אין להשתמש במשתנים גלובליים ו/או סטטיים.
- **מותר** להשתמש בפונקציות שמופיעות בטבלה בעמוד הבא ובפונקציות קלט/פלט print(), input().
- **אין להשתמש ללא מימוש בפונקציות מובנות של פייתון** (למעט הפונקציות בטבלה) **או בפונקציות שמומשו בכיתה,** אלא אם צוין אחרת במפורש בשאלה.
- בכל שאלה ניתן להשתמש בפונקציות המוגדרות בסעיפים קודמים, גם אם לא פתרתם סעיפים אלו, אלא אם נדרש אחרת.
- יש לעמוד בעקרונות תכנות שנלמדו בקורס. שכפול קוד הוא סיבה להורדת ציון.
- אין צורך לבדוק תקינות קלט אלא אם צוין אחרת במפורש בשאלה.
- סדר כתיבת הפונקציות איננו חשוב.
- בשאלות בהן לא נדרשה במפורש סיבוכיות פתרון, הסיבוכיות לא תיבדק.

- נוהל **"לא יודע"**: אם תכתבו בצורה ברורה "**לא יודע/ת**" על שאלה (או סעיף) שבה אתם נדרשים לקודד, תקבלו 17% מהניקוד. דבר זה מומלץ אם אתם יודעים שאתם לא יודעים את התשובה.

צוות הקורס 234128
מרצים: פרופ' אהוד ריבלין, ד"ר ילנה נופברי, יעל ארז.
מתרגלים: גארי מטייב, רגב כהן, אוהד ברטה, רוג'ה חאיק, מוחמד נאסר, איימן אבו יונס, יונתן אלול, יותם אמיתי, רג'אאי חטיב.

בהצלחה!

רשימת פונקציות ופעולות :

פעולה	סיבוכיות זמן
my_list.append(x) / my_list += [x]	O(1)
my_list.extend(list2) / my_list += list2	O(k), k=len(list2)
new_list=my_list[start: stop: step]	O(m), m=len(new_list)
len(my_list)	O(1)
my_list[i]	O(1)
sum(my_list)	O(n)
min(my_list) / max(my_list)	O(n)
x **in** my_list	O(n)
range(start, stop, step)	O(m), m is the length of range
enumerate(my_list)	O(n)
int(x) / round(x) / bool(x)	O(1)
abs(x)	O(1)
ord(c) / chr(x)	O(1)
my_list.sort()	O(n log(n))

שאלה 1 (20 נק')

סעיף א' (7 נק')

מה יודפס כאשר תרוץ התוכנית הבאה?

```
def down_to_one(num):
    num += num**2
    while num > 1:
        if num % 2 == 0:
            num = num // 2
        else:
            num -= 1
        print(num, end = "#")
    print()

num = 4
down_to_one(num)
print(num)
```

אין צורך בהסבר.

```
10#5#4#2#1#
4
```

סעיף ב' (6 נק')

מה יודפס כאשר תרוץ התוכנית הבאה?

```
def my_expand(var, n):
    return var*n

A  = [1,2]
B  = "12"
A = my_expand(A,3)
B = my_expand(B,2)
print('A = ', A, end=", ")
print('B = ', B, end=".")
```

אין צורך בהסבר.

A = [1, 2, 1, 2, 1, 2], B = 1212.

סעיף ג' (7 נק')

מה יודפס כאשר תרוץ התוכנית הבאה?

```
def fun(string):
    lst = [s.upper() for s in string]
    lst2 = lst[0::2]
    lst[0::2] = lst[1::2]
    lst[1::2] = lst2
    return "".join(lst)

print(fun("ameks nees!!"))
```

אין צורך בהסבר.

MAKE SENSE!!

שאלה 2 (25 נק')

לצורך חיפוש **זוג פוטנציאלי** באתר ההיכרויות לכל שם משתמש מותאם מספר שלם.

הגדרה: **"זוג פוטנציאלי"** הוא זוג משתמשים אשר סכום המספרים שלהם שווה למספר כלשהו x הנתון כחלק מהקלט.

סעיף א' (10 נק')

עליכם לממש את הפונקציה `def name_to_num(username)`
אשר מקבלת את שם המשתמש כמחרוזת `username` הכוללת אותיות קטנות באנגלית בלבד.
הפונקציה ממירה כל תו במחרוזת לערך מספרי שהוא מיקום יחסי באלף-בית, כלומר 'a' מותאם ל-1,
'b' מותאם ל- 2, ... 'z' – 26. הפונקציה מחזירה מספר שלם שהוא סכום של הערכים של כל האותיות במחרוזת.

דוגמה:
עבור שם המשתמש **`'aya'`** הפונקציה מחזירה 27,
כי האות a מומרת ל-1, האות y מומרת ל-25 ולכן סכום המספרים הנו 1+25+1=27.

הערה:
- הקלט תקין והוא מחרוזת לא ריקה הכוללת אותיות קטנות באנגלית a-z בלבד.
- **אסור להשתמש ב-`if`.**

```
def name_to_num(username):
    return sum([ord(x)-(ord('a')-1) for x in username])
```

סעיף ב' (15 נק')

עליכם לממש את הפונקציה

```
def find_match(names_lst, x)
```

אשר מקבלת רשימה לא ממוינת `names_lst` של שמות שונים של המשתמשים ומספר שלם חיובי x.
הפונקציה מחזירה True אם ברשימה קיים **זוג פוטנציאלי** של משתמשים, דהיינו, זוג משתמשים אשר סכום המספרים שלהם שווה ל-x. אחרת הפונקציה מחזירה False.

דרישת סיבוכיות: על הפונקציה לעמוד בסיבוכיות $\boldsymbol{n\ Log\ (n)}$, כאשר n אורך הרשימה `names_lst`.

דוגמאות:

- עבור $\mathbf{x = 36, names\_lst = ['aya', 'ed']}$ הפנקציה מחזירה True כי 36=27+9.
- עבור $\mathbf{x = 36, names\_lst = ['eda', 'aya']}$ הפנקציה מחזירה False.

הערות:

- פתרון פחות יעיל מהנדרש **יקבל ניקוד חלקי**.
- אין איסור להשתמש בפונקציית פייתון ()sort.
- יש להשתמש בפונקציה מסעיף **א'** אפילו אם לא פתרתם אותו. סיבוכיות הפונקציה בסעיף א' - O(1).
- אנו מחפשים **זוג** פוטנציאלי, כלומר לא ייתכן שייבחר אותו איבר (אותו שם משתמש) פעמיים.
- הרשימה כוללת לפחות 2 שמות. כל השמות ברשימה כוללים אותיות קטנות באנגלית בלבד.

```
def find_match(names_lst, x):
    num_lst = [name_to_num(username) for username in name_lst]
    num_lst.sort()    # O(n log(n))

    low, high = 0, len(num_lst)-1
    while low < high:    # O(n)
        current_sum = num_lst[low]+num_lst[high]
        if current_sum < x:
            low +=1
        elif current_sum > x:
            high -= 1
        else:
            return True
    return False
```

שאלה 3 (30 נק')

סעיף א' (8 נק')

תזכורת: בהרצאה למדנו את מושג ה-gcd , המחלק המשותף הגדול ביותר בין שני מספרים.
ה-gcd של שני מספרים שלמים חיוביים הוא המספר הטבעי הגדול ביותר שמחלק את שניהם ללא שארית.

דוגמאות:

gcd(15, 20) = 5
gcd(28, 14) = 14
gcd(25, 16) = 1

עליכם לממש את הפונקציה `def gcd(n, m)` , אשר מקבלת שני מספרים טבעיים n ו-m ומחזירה את ה-gcd שלהם.

הערות:

- פתרון ש**לא** משתמש באופרטור % יקבל רק 4 נק'.
- פתרון **שכולל if** יקבל רק 4 נק'.
- הקלט תקין.

```
def gcd(n, m):
```

```
def gcd(n, m): # for n>m
    while  m !=0:
          m, n = n % m, m
    return n

OR

def gcd(n, m):
    return n if m==0 else gcd(m, n%m)
```

סעיף ב' (7 נק')

עליכם לממש את הפונקציה

```
def gcd_list(num_lst)
```

אשר מקבלת רשימה של מספרים טבעיים `num_lst` ומחזירה את "**מחלק הרשימה**",
כלומר המספר הגדול ביותר שמחלק את כל איברי הרשימה ללא שארית.

<u>דוגמה</u>:

gcd_list([10, 100, 15]) = 5
gcd_list([42, 6, 7]) = 1

<u>הערות</u>:

- **<u>חובה</u>** להשתמש בפונקציה מסעיף **א'** אפילו אם לא פתרתם אותו.
- הקלט תקין. הרשימה `num_lst` כוללת לפחות שני מספרים.

```
def gcd_list(num_lst):
    res = num_lst[0]
    for i in range(1, len(num_lst)):
.....   res = gcd(res, num_lst[i])
    return res
```

סעיף ג' (15 נק')

הגדרה: בהינתן מטריצה של מספרים טבעיים mat, צירוף שורה i ועמודה j ייקרא **"טוב"**,
אם "מחלק הרשימה" של כל המספרים בשורה i **זהה** ל-"מחלק הרשימה" של כל המספרים בעמודה j.
לדוגמה, במטריצה

10	100	120
50	8	7

צירוף השורה ***באינדקס 0*** [10, 100, 120] והעמודה ***באינדקס 0*** [50, 10] הוא "**טוב**", כי מחלק הרשימה של שתיהן שווה ל-10.
צירוף השורה ***באינדקס 1*** [50, 8, 7] והעמודה ***באינדקס 2*** [120, 7] "**טוב**", כי מחלק הרשימה של שתיהן שווה ל-1.
שאר צירופי עמודה-שורה בדוגמה לא מהווים צירוף שורה-עמודה "טוב".

עליכם לממש את הפונקציה

```
def count_good(mat)
```

אשר מקבלת מטריצה שמיוצגת על ידי רשימה של רשימות mat הכוללת מספרים טבעיים.
הפונקציה מחזירה את מספר צירופי שורה-עמודה טובים קיימים במטריצה.

דוגמה:
עבור הקריאה `count_good([[10, 100, 120], [50, 8, 7]])` הפונקציה מחזירה 2.

הערות:
- **חובה** להשתמש בפונקציה מסעיף **ב'** אפילו אם לא פתרתם אותו.
- ניתן להניח כי הקלט תקין.

```
def count_good(mat):
    cnt = 0
    for i in range(len(mat)):
        for j in range(len(mat[0])):
            tmp = [mat[k][j] for k in range(len(mat))]
            if gcd_list(tmp) == gcd_list(mat[i]):
                cnt += 1
    return cnt
```

שאלה 4 (25 נק')

עליכם לממש את הפונקציה **הרקורסיבית** `def sum_triangle(num_lst)`
אשר מקבלת רשימה של מספרים שלמים `num_lst`.
הפונקציה מדפיסה את "משולש סכום הקצוות" של הרשימה.

סדר החישוב של משולש סכום הקצוות:

- בבסיסו של המשולש (בשורה האחרונה של ההדפסה) נמצאת הרשימה המקורית,
- מחשבים סכום של אברי קצוות השורה (שניים משמאל, שניים מימין) ומדפיסים רשימה שקצרה בשני איברים (ר' דוגמאות),
- בראש המשולש יודפס רק איבר יחיד.

סדר הדפסה של משולש סכום הקצוות - רגיל:

- קודם כל ראש המשולש, אחר כך עוד שורות, ולבסוף את הבסיס (הרשימה המקורית).

דוגמה 1: עבור הרשימה `num_lst = [6,4,5,21]` הפונקציה מדפיסה

[36]
[10, 26]
[6, 4, 5, 21]

הסבר: חישוב האיברים בשורה השנייה

[6, 4, 5, 21] → (6+4 = 10, 5+21 =26) → **[10, 26]**

חישוב האיברים בשורה העליונה

[10, 26] → (10 + 26 = 36) → **[36]**

דוגמה 2:

אם לרשימה יש 3 איברים, אז רק האיבר השמאלי (בתחילת הרשימה) נסכם והימני נשאר ללא שינוי.

עבור הרשימה `num_lst = [6,4,18]` הפונקציה מדפיסה

[28]
[10, 18]
[6, 4, 18]

הסבר: חישוב האיברים בשורה השנייה

[6, 4, 18] → (6+4 = 10, 18) → **[10, 18]**

חישוב האיברים בשורה העליונה

[10, 18] → (10 + 18 = 28) → **[28]**

הערות:

- הפונקציה חייבת להיות **רקורסיבית! אסור** להשתמש בלולאות כולל list comprehension.
- **אסור קריאות לפונקציות פייתון** למעט len(), print().
- הרשימה `num_lst` איננה ריקה.

```
def sum_triangle(num_lst):
```

```
def sum_triangle(lst):
    n = len(lst)
    if n == 0:
        return
    if n == 1:
        print(lst)
        return
     if n == 2:
        sum_triangle([lst[0]+lst[1]])
        print(lst)
        return
     if n == 3:
        sum_triangle([lst[0]+lst[1]]+[lst[-1]])
        print(lst)
        return

    sum_triangle([lst[0]+lst[1]]+lst[2:-2]+[lst[-1]+lst[-2]])
    print(lst)
    return

num_lst = [6,4,18,5,6,7,9]
sum_triangle(num_lst)
```