# מבוא למחשב – שפת פייתון (234128)

## מועד ב' סמסטר אביב תש"ף

## 24 בספטמבר 2020

**משך המבחן: שעתיים וחצי.**
**חומר עזר:** אין להשתמש בכל חומר עזר בכתב, מודפס או אלקטרוני.

**הנחיות והוראות:**

1. יש לכתוב פתרונות בכתב-יד באופן ברור, נקי ומסודר. יש להתחיל את הפתרון לכל שאלה **בעמוד חדש**! בכל עמוד חדש יש לכתוב את מספר תעודת הסטודנט.
2. אין לכתוב תשובות **בעפרון.**
3. אין להפוך את דפי הבחינה שלכם בסריקה בכיוון "landscape".
4. מותר להשתמש בפונקציות שמופיעות **בטבלה** בעמוד הבא ובפונקציות קלט/פלט ()print, input().
5. **אין להשתמש ללא מימוש בפונקציות מובנות של פייתון** (למעט הפונקציות בטבלה) **או בפונקציות שמומשו בכיתה,** אלא אם צוין אחרת במפורש בשאלה.
6. בכל השאלות, הנכם רשאים להגדיר (לממש) פונקציות עזר כרצונכם, אלא אם צוין אחרת במפורש בשאלה. סדר כתיבת הפונקציות איננו חשוב.
7. בכל שאלה ניתן להשתמש בפונקציות המוגדרות בסעיפים קודמים, גם אם לא פתרתם סעיפים אלו, אלא אם נדרש אחרת.
8. אין להשתמש במשתנים גלובליים ו/או סטטיים.
9. אין להשתמש במילונים (dict).
10. יש לעמוד בעקרונות תכנות שנלמדו בקורס. **שכפול קוד הוא סיבה להורדת ציון**.
11. אין צורך לבדוק תקינות קלט אלא אם צוין אחרת במפורש בשאלה.
12. נוהל **"לא יודע":** אם התשובה לסעיף מסוים (או שאלה מסוימת) אינה ידועה, הנכם ראשים לכתוב בסעיף זה "**לא יודע/ת**" בצורה ברורה (!) ולקבל 20% מהניקוד לאותו סעיף.

| צוות הקורס 234128 |
|---|
| **מרצים**: ד"ר ילנה נופברי |
| **מתרגלים**: רוג'ה חאיק, אוהד ברטה, יותם אמיתי, אביגיל כהן רמון, מוחמד נאסר, יונתן אלול, תום אברך, וסים מח'ול. |

בהצלחה!

## רשימת פונקציות ופעולות :

| פעולה | סיבוכיות זמן |
|---|---|
| len(my_list) | O(1) |
| my_list[i] | O(1) |
| 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) |
| range(start, stop, step) | O(m), m is the length of range |
| sum(my_list) | O(n), n=len(my_list) |
| min(my_list) / max(my_list) | O(n) |
| x in my_list | O(n) |
| 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 (15 נק')** **יש לכתוב את הפתרון בעמוד חדש ולסמן את מספר הסטודנט**

עליכם להשלים קריאה לפונקציה הנתונה foo וגם לענות לשאלה "מה יודפס כאשר תרוץ התוכנית הבאה"?

```
def foo(lst):
    lst1 = lst
    lst1 = lst1 + [6]
    lst1[0] = 20
    length = len(lst)
    lst2 = lst + [lst[length-1-i]+2 for i in range(length)]
    print("lst2=", lst2)
    sum1 = sum(lst2[-1::-2])
    sum2 = sum(lst2[::2])
    return sum1, sum2



lst1=[1,2,3,4,5,6]

*********************
sum1, sum2 = foo(lst1)
```

**יש להשלים במקום הכוכביות קריאה לפונקציה** foo **עבור הרשימה** lst1

```
print("lst1=", lst1)
print("Odd" if sum1>sum2 else "Even")
```

**הנחיות לכתיבת הפתרון:**

בדף הבחינה שלכם יש לכתוב רק **4 שורות**:

1) קריאה לפונקציה foo – שורה אחת,

```
sum1,sum2=foo(lst1)
```

2) שלוש שורות של הדפסות בהתאם לקוד הנתון בשאלה.

```
lst2= [1, 2, 3, 4, 5, 6, 8, 7, 6, 5, 4, 3]
lst1= [1, 2, 3, 4, 5, 6]
Even

============================
```

בנוסף לידיעתכם (לא נדרש לעשות ולא נבדק כמובן):

```
print("sum1=", sum1)
print("sum2=", sum2)
sum1=27
sum2=27
```

**שאלה 2 (30 נק')**

**סעיף א' (15 נק')** **יש להתחיל את הפתרון בעמוד חדש**

הגדרה: בתחום הקריפטוגרפיה **צופן קיסר** הידוע גם כצופן היסט. זהו סוג של צופן החלפה שבו כל אות מוחלפת על ידי אות הנמצאת בהיסט כלשהו ממנה באלף-בית, כאשר היסט הוא מספר שלם: חיובי, שלילי או אפס.

לדוגמה, עבור ההיסט 3: האות A תוחלף באות D, האות B תוחלף באות E. תשימו לב, צריך להתייחס להיסט בצורה ציקלית, כלומר: אם האות היא A וההיסט הוא 1- אז יוחזר Z, ואם האות היא Z וההיסט הוא 2 יוחזר B.

עליכם לממש את הפונקציה

```
def caesar_cipher(text_str, offset)
```

אשר מקבלת מחרוזת `text_str` המכילה אותיות קטנות באנגלית בלבד
והיסט `offset`, מספר שלם לאו דווקא חיובי.
הפונקציה מחזירה את ההצפנה של המחרוזת באותיות גדולות, בצופן קיסר עם ההיסט הנתון.

דוגמאות:

- בהינתן היסט 2, עבור המחרוזת "abcd" המחרוזת המוצפנת היא "CDEF".
- בהינתן היסט 1-, עבור המחרוזת "azyx" המחרוזת המוצפנת שתוחזר היא "ZYXW".

הערות:

- אין להשתמש בפעולות פייתון lower\upper.
- אין להפוך מחרוזת לרשימה.
- אין להשתמש במילונים (dict).

```
def to_capital(text_str):
    new_str=""
    for ch in text_str:
        new_str+=chr(ord(ch)-(ord('A')-ord('a')))
    return new_str


def caesar_cipher(text_str, offset):
    new_str=""
    for ch in text_str:
        new_str+=chr(ord('a') + (ord(ch)-ord('a')+offset) %26 )
    new_str_capital=to_capital(new_str)
    return new_str_capital


print(caesar_cipher("abcd",2))
print(caesar_cipher("abcd",2-26))

print(caesar_cipher("azyx",-1))
print(caesar_cipher("azyx",-1-26))
```

**סעיף ב' (15 נק')** **יש להתחיל את הפתרון בעמוד חדש**

**סעיף א' וסעיף ב' בלתי תלויים**

הגדרה: **צופן אתב"ש,** הוא צופן החלפה, כאשר כל אות במחרוזת מוחלפת באות שנמצאת באותו המרחק מהקצה השני של אלף-בית אנגלי בטבלת Unicode. כלומר, האות a מוחלפת באות האחרונה באלף-בית (כלומר, ב- z), האות b מוחלפת ב- y, האות z מוחלפת ב-a, האות y ב- b וכן הלאה.


עליכם לממש את הפונקציה

```
def atbash_cipher(text_str)
```

אשר מקבלת מחרוזת `text_str` המכילה אותיות גדולות באנגלית בלבד ומחזירה את ההצפנה של המחרוזת בצופן אתב"ש באותיות קטנות.

דוגמאות:

- עבור המחרוזת "ZYXW" הפונקציה מחזירה "abcd".
- עבור "ZABC" הפונקציה מחזירה "azyx".

הערות:

- אין להשתמש בפעולות פייתון lower\upper.
- אין להפוך מחרוזת לרשימה.
- **אין** להגדיר מחרוזת או רשימה הכוללת את אלף-בית!
- אין להשתמש במילונים (dict).

```
def to_small(text_str):
    new_str=""
    for ch in text_str:
        new_str+=chr(ord(ch)-( ord('a')- ord('A')))

    return new_str


def atbash_cipher(text_str):
    new_str=""
    for ch in text_str:
        diff=ord(ch)-ord('A')
        new_ch=ord('Z')-diff
        new_str+=chr(new_ch)

    new_str_small=to_small(new_str)
    return new_str_small

print(atbash_cipher("ZYXW"))
print(atbash_cipher("ZABC"))
```

**שאלה 3 (25 נק')** **יש להתחיל את הפתרון בעמוד חדש**

נתונה מטריצה mat של מספרים שלמים, כך שהיא מקיימת את התנאים הבאים:

1) **כל שורה** בנפרד ממוינת בסדר עולה,
2) עבור **כל שורה i** (i>=1), נתון כי האיבר הראשון שלה גדול או שווה לאיבר האחרון בשורה **i-1.**

עליכם לממש את הפונקציה `def find_elem(mat, num)`
אשר מקבלת מטריצה mat המכילה מספרים שלמים ועוד מספר שלם num.
הפונקציה מחזירה את האינדקסים **i** ו-**j** של האיבר ששווה ל-num.
אחרת (אם איבר כזה לא קיים במטריצה), הפונקציה מחזירה 1-.

**<u>דרישת סיבוכיות זמן</u>**: O(log(M) + log(N)), כאשר N מספר שורות, M מספר עמודות במטריצה.

<u>דוגמה:</u>

- עבור num=14 בהינתן המטריצה הבאה mat הפונקציה מחזירה (1,0)

| | | |
|---|---|---|
| 1 | 5 | 9 |
| 14 | 20 | 21 |
| 30 | 34 | 43 |

- עבור num=42 בהינתן המטריצה הבאה mat הפונקציה מחזירה 1-

| | | | |
|---|---|---|---|
| 1 | 5 | 9 | 11 |
| 14 | 20 | 21 | 26 |
| 30 | 34 | 43 | 50 |

**<u>הנחיות:</u>**

1. **אין להגדיר רשימת עזר.**
2. **אין לשנות** את מטריצת הקלט mat בשום שלב של פעילות הפונקציה.
3. אין להגדיר פונקציות עזר.
4. **אין להשתמש באף פונקציה למעט** len(), **האיסור כולל גם את הפונקציות מהטבלה במבחן.**
5. במטריצה קיימות לפחות 2 שורות.
6. פתרון נכון שמגדיר **רשימת עזר אחת** יקבל *<u>ניקוד חלקי</u>*. יש להסביר בקצרה מה סיבוכיות הפתרון שלכם. ההנחיות (2,3,4,5) בתוקף גם עבור הפתרון החלופי הזה.
7. להזכירכם, log(x*y)=log(x)+log(y)

```
def find_elem(mat, num):
    N = len(mat)
    M = len(mat[0])
    if(N == 0):
        return -1
    low, high = 0, N * M - 1
    while(low <= high):
        mid = (low+high)//2
        Xindex = mid//M
        Yindex = mid%M
        if(mat[Xindex][Yindex] == x):
            return Xindex,Yindex
        if(mat[Xindex][Yindex] > x):
            high = mid - 1
        else:
            low = mid + 1
    return -1
```

**שאלה 4 (30 נק')** **יש להתחיל את הפתרון בעמוד חדש**

הגדרה: **פרמוטציה** של מחרוזת זה סידור מחדש של אותן האותיות.
לדוגמה, 'ABD' היא פרמוטציה של 'ADB'.

**סעיף א' (15 נק')**

עליכם לממש את הפונקציה

```
def perm(s1, s2)
```

אשר מקבלת שתי מחרוזות `s1, s2` המכילות אותיות גדולות באנגלית בלבד,
ומחזירה True אם המחרוזת `s1` היא פרמוטציה של המחרוזת השנייה `s2`, או False אחרת.

**הנחיות ודרישות:**

1) סיבוכיות הפונקציה חייבת להיות **לינארית.** בנוסף, מותר לעבור דרך כל אחת מהמחרוזות רק **פעם אחת** בלבד. פתרון שלא עומד בדרישות האלה לא מתקבל.
2) מומלץ בחום להגדיר פונקציית עזר כדי להימנע משכפול קוד. שכפול קוד יגרום להורדת נקודות בציון.
3) **אין להפוך מחרוזת לרשימה.** אין להשתמש במילונים (dict).
4) הפונקציה לא רקורסיבית.

דוגמאות:

- 'AJBD' היא לא פרמוטציה של 'ADB' בגלל האות J.
- 'ABDB' היא לא פרמוטציה של 'ADB'.
- 'A' היא פרמוטציה של 'A'.

```
def hist_fun(string):
    hist = [0] * (ord('z')-ord('a')+1)
    for s in string:
        hist[ord(s)-ord('A')] += 1
    return hist

def perm(s1, s2):
    return hist_fun(s1) == hist_fun(s2)
```

**סעיף ב' (15 נק')** **יש להתחיל את הפתרון בעמוד חדש**

עליכם לממש את **הפונקציה הרקורסיבית**

```
def check_combination(str_buckets, s, temp)
```

אשר מקבלת רשימה של רשימות של זוגות המחרוזות str_buckets,
כלומר אורך של כל רשימה פנימית שווה 2. לדוגמה, str_buckets = [['AB','A'],['DA','E']].
בנוסף, הפונקציה מקבלת מחרוזת s, ופרמטר עזר temp.

הפונקציה מחזירה True אם אפשר לבנות פרמוטציה של המחרוזת s מהמחרוזות של str_buckets בתנאים הבאים:

1. מכל רשימה שנמצאת בתוך str_buckets מותר לקחת רק מחרוזת אחת לכל היותר.
2. במחרוזת שנלקחה מ-str_buckets, חובה להשתמש בכל האותיות שלה.

אחרת (אם לא ניתן לבנות פרמוטציה) הפונקציה מחזירה False.

שימו לב: הפונקציה לא בונה פרמוטציה. היא רק בודקת האם אפשר לעשות זאת.

דוגמאות:

1. עבור הקלט הבא הפונקציה מחזירה True כי אפשר פשוט לקחת את 'A' מהרשימה הראשונה.
   str_buckets = [['AB','A'], ['D','E']]
   s = 'A'
2. עבור הקלט הבא
   str_buckets = [['AB','A'], ['E','D'], ['KL','M']]
   s = 'ADB'
   הפונקציה מחזירה True. ניקח את 'AB' מהרשימה הראשונה ואת 'D' מהרשימה השנייה, וכך קיבלנו את כל האותיות של 'ADB'. שימו לב, לוקחים רק מחרוזת אחת בלבד מכל רשימה אבל כל עוד לוקחים אותה אז את כל האותיות שלה!
3. עבור הקלט הבא הפונקציה מחזירה False כי צריכים לקחת את 'A' מהרשימה הראשונה (זה עומד בתנאיי השאלה) ועוד את 'D' ואת 'E' מהרשימה השנייה (אבל זה כבר ***שתי מחרוזות***, מה שאסור לעשות).
   str_buckets = [['AB','A'], ['D','E']]
   s = 'ADE'

הנחיות:

- הפונקציה חייבת להיות **רקורסיבית**. פתרון הכולל לולאה (אפילו אחת) **לא מתקבל**.
- חובה להשתמש בפונקציה **מסעיף א'** (לא משנה שהיא פונקציה לא רקורסיבית) אפילו אם לא פתרתם אותו.
- בקריאה הראשונה לפונקציה, הערך שמעבירים לפרמטר עזר temp הוא מחרוזת ריקה.
- בכל הקריאות הרקורסיביות בקוד שלכם הפרמטר s נשאר ללא שינוי.
- התוים של כל המחרוזות הם אותיות אנגליות גדולות בלבד.

```
def check_combination(str_buckets, s, temp):
    if perm(s, temp):
        return True
    if str_buckets == []:
        return False

    return check_combination(str_buckets[1:],s,temp+str_buckets[0][0])\
                or check_combination(str_buckets[1:],s,temp+str_buckets[0][1])\
                or check_combination(str_buckets[1:],s,temp)
```