מבוא לתכנות בשפת פייתון
88-6960-01
88-6960-11
80-235-01

DSI.6960
Programming in Python

שאלון בחינה

מועד:	א׳
גרסה:	1
תאריך:	2 בפברואר 2020
משך הבחינה:	3 שעות
שאלון:	רגיל
חומר עזר:	מותר ללא הגבלה.
מחשבון:	השימוש מותר.
בחירה:	יש לענות על 3 מ-4 השאלות.
קריטריונים:	ציון טוב בבחינה יינתן למי שמדגים ״חשיבה חישובית״ ויכולת לייצר אלגוריתם טוב לפתרון הבעיה. **הקוד לא חייב להיות תקין במאה אחוז, ואפילו לא חייב לרוץ או להחזיר את התוצאה הנכונה!**
היקף נדרש:	אלא אם נכתב אחרת, יש להגיש רק את הקוד של הפונקציות/המחלקות המבוקשות ושל פונקציות/מחלקות עזר, ולא את הקוד של כל התוכנית. ניתן להניח שהקלט תמיד חוקי. אין צורך לבצע בדיקות קלט.
אופן הכתיבה:	את הפתרון הסופי יש לרשום בכתב יד ברור, גדול ולא מחובר בלבד. יש לתעד בקצרה את הקוד באמצעות הערות לפי הצורך.

1] תווים רקורסיביים

ממש את הפונקציה countChar המקבלת מחרוזת ותו, ומחזירה את מספר הפעמים בהם התו מופיע במחרוזת. על פונקציה זאת לבצע את החישוב בפועל באמצעות תכנות רקורסיבי, וללא שימוש בלולאות או בפונקציות/באופרטורים אחרים הסופרים מופעים.

דוגמה: בהינתן המחרוזת "These pretzels are making me thirsty" והתו 't', הפונקציה תחזיר 3.

בהינתן המחרוזת "These pretzels are making me thirsty" והתו 'A', הפונקציה תחזיר 0.

2] קורס מוכוון עצמים

ממש את הבנאי של המחלקה Student, המייצגת סטודנט באמצעות שמו ומספר הזהות שלו. ממש גם את הבנאי של המחלקה Course, המייצגת קורס באמצעות שמו, היום בשבוע בו הוא מתקיים, השעה בה הוא מתחיל ורשימת הסטודנטים הרשומים אליו (כרשימת אובייקטים מסוג Student).

ממש ב-Course את המתודה overlapExists, אשר מקבלת אובייקט אחר מסוג Course, ומחזירה True אם שני הקורסים מתקיימים באותו זמן (יום ושעה) וגם קיים סטודנט מסוים (לפחות אחד), הרשום לשני הקורסים.

דוגמה א': בהינתן הקורס Math, המתקיים ביום שני בשעה 10, ואשר אליו רשומים -
- שם: Jerry מ"ז: 1234
- שם: George מ"ז: 5678

ובהינתן הקורס Prog, המתקיים ביום שני בשעה 10, ואשר אליו רשומים -
- שם: Jerry מ"ז: 1234
- שם: Elaine מ"ז: 2468

המתודה תחזיר True.

דוגמה ב': בהינתן הקורס Chem, המתקיים ביום רביעי בשעה 9, ואשר אליו רשומים -
- שם: Jerry מ"ז: 1234
- שם: George מ"ז: 5678

ובהינתן הקורס Bio, המתקיים ביום רביעי בשעה 9, ואשר אליו רשומים -
- שם: Jerry מ"ז: 2244
- שם: George מ"ז: 3579

המתודה תחזיר False.

דגשים: שני אובייקטים מסוג Student מייצגים את אותו הסטודנט אם גם השם וגם מספר הזהות שלו שווים.

עזרה: ניתן להשתמש במתודה __eq__ במחלקה Student, אך לא חובה.
ניתן להשתמש באופרטור in במתודה overlapExists, אך לא חובה.

3] מיון היררכי

ממש את הפונקציה hierarchicalSort המקבלת רשימה, וממיינת אותה בשתי היררכיות. ראשית, יופיעו רק המספרים הזוגיים - ממוינים בינם לבין עצמם - ולאחר מכן, המספרים האי-זוגיים - גם כן ממוינים בינם לבין עצמם.

דוגמה: בהינתן הרשימה [7,4,8,3,6], צורתה הממוינת תהיה [4,6,8,3,7].

דגשים: מימוש ע"י חלוקה לשתי רשימות (מספרים זוגיים ומספרים אי-זוגיים), מיונן וחיבורן יחדיו, אינו יעיל ואינו אלגנטי. הוא לא יזכה את הפותר במלוא הנקודות.

עזרה: להלן המימוש של bubbleSort, כפי שהופיע בהרצאה. ניתן להיעזר בשיטת מיון זאת, או בכל שיטה אחרת.

```
def bubbleSort(lst):
    n = len(lst)
    for i in range(n):
        for j in range(0, n-i-1):
            if lst[j] > lst[j+1]:
                lst[j], lst[j+1] = lst[j+1], lst[j]
```

4] ניתוח נתוני ציונים

להלן דוגמת ריצה של קוד, המבצע ניתוחי נתונים שונים על ציוני בחינה ותרגיל של סטודנטים בקורס מסוים:

```
exam_grades = [50, 60, 72, 80, 100, 99, 42, 58, 82, 85]
exercise_grades = [60, 82, 74, 92, 100, 92, 68, 88, 90, 100]

print ("Exam:")
avg_exam_grade = average_score(exam_grades)
min_exam_grade, max_exam_grade = min_max(exam_grades)
print ("min exam grade = " + str(min_exam_grade))
print ("average exam grade = " + str(avg_exam_grade))
print ("max exam grade = " + str(max_exam_grade))
print ()
print ("Exercise:")
avg_exercise_grade = average_score(exercise_grades)
min_exercise_grade, max_exercise_grade = min_max(exercise_grades)
print ("min exercise grade = " + str(min_exercise_grade))
print ("average exercise grade = " + str(avg_exercise_grade))
print ("max exercise grade = " + str(max_exercise_grade))
```

א] ממש את הפונקציות average_score ו-min_max, המקבלות רשימת מספרים, ומחזירות את הממוצע או את המינימום והמקסימום שלהם, בהתאמה, כך שבהרצת הקוד הנ"ל יתקבל הפלט שלהלן.

Exam:
min exam grade = 42
average exam grade = 72.8
max exam grade = 100

Exercise:
min exercise grade = 60
average exercise grade = 84.6
max exercise grade = 100

דגשים: כמובן, אסור להשתמש בפונקציות המובנות sum, min, max ו-mean.
ניתן להניח שהרשימות אינן ריקות, כלומר, שהן מכילות לפחות איבר אחד.

ב] בהינתן שהציון הסופי מורכב מ-60% ציון הבחינה ו-40% ציון התרגיל, ממש את הפונקציה divideStudents, המקבלת את רשימת שמות סטודנטים, רשימת ציוני המבחן ורשימת ציוני התרגיל. על הפונקציה להחזיר שלוש רשימות, המכילות את שמות הסטודנטים המצטיינים, שמות הנכשלים ושאר השמות, לפי הכללים הבאים -

- הצטיינות: ציון סופי מעל 90 וגם ציון בחינה מעל 85.
- כישלון: ציון סופי מתחת ל-50 או ציון בחינה מתחת ל-60.

עזרה: ניתן להניח ששלוש הרשימות (שמות הסטודנטים, ציוני המבחן וציוני התרגיל) שוות בגודלן.

דוגמה: בהינתן רשימת שמות הסטודנטים והקוד הבאים -

```
student_names =  ['Guy', 'Amit', 'Shai', 'Moris', 'Zeev', \
                  'Chen', 'Michael', 'Tali', 'Inna', 'Ruth']
excls, fails, rest =  divide_students(student_names, \
                  exam_grades, exercise_grades)
print("Excellent students =", excls)
print("Failed students =", fails)
print("Rest of students =", rest)
```

יתקבל הפלט הבא -

```
Excellent students = ['Zeev', 'Chen']
Average studs = ['Amit', 'Shai', 'Moris', 'Inna', 'Ruth']
Failed studs = ['Guy', 'Michael', 'Tali']
```

בהצלחה!