# מטלת פייתון – תבניות עיצוב

אפשר לענות על חלק מהשאלות ולקבל ניקוד חלקי.

יש לבצע בדיקות אוטומטיות בעזרת דוקטסט. הגשה ע"י קישור לגיטהאב.

## שאלה 1: ייצור תת-קבוצות (2 נק').

א (1 נק'). כתבו איטרטור או גנרטור (לבחירתכם) בשם bounded\_subsets. הוא מקבל כקלט סדרה S של מספרים חיוביים שונים, ומספר חיובי כלשהו C, ומייצר סדרה של כל התת-קבוצות של S, שסכומן לכל היותר C.

הפתרון צריך להיות יעיל. בפרט:

* אין לייצר את כל הקבוצות ואז לבדוק את הסכום, אלא לייצר מלכתחילה רק את הקבוצות שסכומן לכל היותר C.
* אין לייצר רשימה של כל התת-קבוצות הרלבנטיות ואז להחזיר איטרטור לרשימה, אלא לייצר כל תת-קבוצה כשצריך אותה (כמו שלמדנו בהרצאה).

דוגמאות:

for s in bounded\_subsets([1,2,3], 4):

print(s) # prints: [], [1], [2], [3], [1,2], [1,3].

for s in bounded\_subsets(range(50,150), 103):

print(s) # prints: [], [50], [51],..., [103], [50, 51], [50, 52], [50, 53], [51, 52]

for s in zip(range(5), bounded\_subsets(range(100), 1000000000000)):

print(s) # prints: (0,[]), (1,[0]), (2,[1]), (3,[2]), (4,[3])

for s in bounded\_subsets(list(range(90,100)) + list(range(920,1000)), 1000)):

print(s) # prints 1024 subsets of [90,...,99] plus 80 singletons [920],..,[999]

הראו שהפונקציה שלכם אכן פותרת את הבעיה באופן יעיל.

ב (1 נק'). שנו את הפונקציה כך שתחזיר תת-קבוצות **בסדר עולה של הסכום שלהן** (הסדר בין תת-קבוצות עם אותו סכום לא משנה). בדוגמה השניה למעלה, יודפס:

[], [50], [51],..., [100], [101], [50, 51], [102], [50, 52], [103], [50, 53], [51, 52]

בדוגמה השלישית למעלה יודפס:

print(s) # prints: (0,[]), (1,[0]), (2,[1]), (3,[2]), (4,[3])

גם כאן הפתרון צריך להיות יעיל:

* אין לייצר רשימה של כל התת-קבוצות הרלבנטיות ואז לסדר אותה לפי סכום, אלא מלכתחילה לייצר תת-קבוצות לפי הסדר.

## שאלה 2: דגם-עיצוב אסטרטגיה (2 נק')

הגירסה הסופית של הקוד שהוצג בשיעור נמצאת בתיקיית הקוד, בקובץ בשם:

**50-flyweight.py**

כתבו מערכת דומה עבור בעיה אלגוריתמית אחרת כלשהי (לדוגמה: [בעיית הסוכן הנוסע](https://he.wikipedia.org/wiki/בעיית_הסוכן_הנוסע)). המערכת שלכם צריכה לטפל באופן יעיל, ובלי שיכפול קוד, ב:

* לפחות שני סוגי קלט (לדוגמה: מרחקים בלבד, או מרחקים + שמות של ערים).
* לפחות שני סוגי פלט (לדוגמה: המסלול כולו, או רק אורך המסלול).
* לפחות שני אלגוריתמים. אפשר לבחור אלגוריתמים פשוטים וקצרים – לא דווקא אופטימליים.

השתמשו בדגמי-העיצוב שלמדנו בשיעור.

## שאלה 3: משחק תיכנות (1 נק')

בחרו שאלה אחת מהרשימה הבאה: <https://www.codingame.com/training>

בדרגת קושי **קשה – Hard** לפחות. פתרו אותה בפייתון.

יש להגיש קישור לפתרון שלכם באתר codingame + צילום מסך.