תרגילים בסיסיים בתכנות

שאלה 1:
בהינתן רשימה, יש לספור את מספר המופעים של כל אובייקט ברשימה על ידי מילון ולבסוף להדפיס את המילון. בשאלה זו אין להשתמש בפונקציה count.
לדוגמא:

```
input_list = [11, 45, 8, 11, 23, 45, 23, 45, 89]
```

הדפסה שתופיע למשתמש:

```
{11: 2, 45: 3, 8: 1, 23: 2, 89: 1}
```

פתרון שאלה 1:

```
sample_list = [11, 45, 8, 11, 23, 45, 23, 45, 89]
print("Original list ", sample_list)

count_dict = {}
for item in sample_list:
  if item in count_dict:
    count_dict[item] += 1
  else:
    count_dict[item] = 1
print("Printing count of each item  ", count_dict)
```

שאלה 2:
פונקציה המקבלת רשימת מספרים ומספר (הקטן מאורך הרשימה).
הפונקציה מחזירה את n המספרים המקסימליים של הרשימה.
לא ניתן להשתמש בפונקציית מיון sort ו- max, יש לבצע זאת בלולאה.

לדוגמא:

```
input_list = [2, 6, 41, 85, 0, 3, 7, 6, 10]
number_of_top_largest = 3
print(n_max_elements(input_list, number_of_top_largest))
```

נקבל:

```
[10 ,41 ,85]
```

עבור:

```
input_list = [2, 6, 41, 85, 0, 3, 7, 6, 10]
number_of_top_largest = 2
print(n_max_elements(input_list, number_of_top_largest))
```

נקבל:

```
[41 ,85]
```

פתרון שאלה 2:

```
def n_max_elements(input_list, number_of_top_largest):
    final_list = []
     for i in range(number_of_top_largest):  # The amount of max number we need to return
       largest_counter = 0

       for j in input_list:      # finding the max number in the list
          if j > largest_counter:
             largest_counter = j
       input_list.remove(largest_counter)
       final_list.append(largest_counter)
    return final_list

if __name__ == "__main__":
    input_list = [2, 6, 41, 85, 0, 3, 7, 6, 10]
    number_of_top_largest = 4
    print(n_max_elements(input_list, number_of_top_largest))
```

שאלת בונוס – הגנה על הקוד:

הוסף בדיקות לקלט (האובייקטים המתקבלים) של הפונקציה משאלה 3, והגן על הקוד מקריסה במהלך הפעולות אשר מתבצעות על הקלט.

ניתן לכתוב את הבדיקות בסעיף חדש, למספר כל בדיקה ולבצע הפניה מהבדיקה למקומה בפונקציה 3.

פקודות שימושיות: try, except, raise Exception, assert וכדומה.

פתרון שאלת בונוס – הגנה על הקוד:

```
def n_max_elements(input_list, number_of_top_largest):
    assert number_of_top_largest <= len(input_list), 'number_of_top_largest is bigger than input_list'
    final_list = []
    for i in range(0, number_of_top_largest):  # The amount of max number we need to return
       largest_counter = 0
       for j in input_list:      # finding the max number in the list
          try:
             if int(j) > largest_counter:
                largest_counter = int(j)
          except ValueError:
             print("input_list should contain only int elements, but got {}.".format(j))
             raise ValueError("input_list should contain only int elements, but got {}.".format(j))
       input_list.remove(largest_counter)
       final_list.append(largest_counter)
    return final_list

if __name__ == "__main__":
    input_list = [2, 6, 41, 85, 0, 3, 7, 6, 10]
    number_of_top_largest = 4
    print(n_max_elements(input_list, number_of_top_largest))
```

שאלת בונוס – רקורסיה – טיפוס מדרגות:
היו היה בית עם גרם מדרגות פנימי וילד שאוהב מאוד לטפס עליהן.
הילד יכול לבחור לטפס מדרגה-מדרגה, או לדלג שתי מדרגות או לדלג שלוש מדרגות.
הילד שונא לחזור על דרכים. אם הילד בוחר כל יום לטפס בדרך אחרת, תוך כמה ימים הוא יצטרך לעבור דירה? (כי לא יישארו לו עוד דרכים חדשות לטפס).

אם לדוגמא יש בבית 3 מדרגות אז האפשרויות של הילד די מוגבלות:

הוא יכול לטפס את כל שלושת המדרגות בצעד אחד. [3]
הוא יכול לטפס את שתי המדרגות הראשונות בצעד אחד, ואז עוד צעד למדרגה האחרונה. [2, 1]
הוא יכול לטפס את המדרגה הראשונה בצעד אחד, ואז את השתיים הנותרות בצעד הבא. [1, 2]
הוא יכול לטפס צעד-צעד את כל המדרגות ולעלות את שלושת המדרגות ב-3 צעדים. [1, 1, 1]
סך הכל אחרי 4 ימים הוא יצטרך למצוא בית חדש, אלא אם כן ההורים יצליחו לשכנע אותו שדווקא לא נורא לטפס כמה פעמים באותה שיטה.
כלומר התשובה ל- print(ways(3)) תהיה 4.

פתרון שאלת בונוס – רקורסיה – טיפוס מדרגות:
פתרון יותר אינטואיטיבי הוא פתרון רקורסיבי:

נסה לטפס מדרגה אחת.
נסה לטפס שתי מדרגות.
נסה לטפס שלוש מדרגות.
בכל אחד מהמקרים אם לא נשארו מדרגות לטפס הוסף 1 לסכום, אם טיפסת יותר מדי מדרגות אפשר למחוק את האפשרות, ואם נשארו מדרגות חזור על החישוב עם המדרגות שנותרו.

בדוגמא של שלוש המדרגות זה יראה בערך כך:

אם טיפסתי את כל שלושת המדרגות יחד הגעתי. נשארו 0 מדרגות לטפס. לכן ניתן לאפשרות זו את הערך 1.
אם טיפסתי שתי מדרגות נשארה מדרגה אחת בלבד: א. אני יכול לטפס מדרגה אחת נוספת ולהגיע (עוד 1). ב. אני יכול לטפס 2 או 3 מדרגות, אבל אז אגיע רחוק מדי לכן אפשר להתעלם מאפשרויות אלו.
אם טיפסתי מדרגה אחת נשארו שתי מדרגות לטפס: א. אני יכול לטפס עוד 2 מדרגות ולהגיע (עוד 1). ב. אני יכול לטפס עוד 3 מדרגות, אבל אז הגעתי רחוק מדי (להתעלם). ג. אני יכול לטפס עוד מדרגה אחת. עדיין לא הגעתי לסוף אז מנסים שוב:
אני יכול לטפס עוד מדרגה אחת (הגעתי! עוד 1)
אני יכול לטפס 2 או 3 מדרגות אבל אז הגעתי רחוק מדי ולכן נתעלם.
סך הכל, כשנסכום את כל האפשרויות נגיע ל-4, שזה גם מה שיצא בחישוב ידני. את התהליך קל לקודד בפייתון באמצעות רקורסיה. הפונקציה ways מחזירה את מספר האפשרויות לטפס גרם מדרגות בגובה h:

```
def ways(h):
   if h<0:
         return 0
   if h==0:
         return 1
   return ways(h-1) + ways(h-2) + ways(h-3)

print(ways(3))
```

ניתן לשפר זאת יותר, על ידי הוספת מילון ושמירת תוצאות של חישובים שכבר בוצעו.