פייתון – מטלה 1

משקל המטלה = 4 נקודות. כל שאלה = נקודה אחת. אפשר לענות על חלק מהשאלות ולקבל ניקוד חלקי.

שאלה 1: קריאה בטוחה

כתבו פונקציה בשם safe_call, המקבלת כקלט פונקציה אחרת וארגומנטים עם שמות, וקוראת לפונקציה עם הארגומנטים, אבל רק הם מתאימים בדיוק לסוגים המוגדרים ב-annotation של הפונקציה. אם הסוגים לא מתאימים, יש לזרוק חריגה (אם לחלק מהארגומנטים אין annotation, אז לא צריך לבדוק אותם).

דוגמה:

```
def f(x: int, y: float, z):
        return x+y+z


safe_call(f, x=5, y=7.0, z=3)       # returns 15.0
safe_call(f, x=5, y="abc", z=3)   # raises an exception
```

יש להגיש קישור לגיטהאב, הכולל קוד + תיעוד + בדיקות + דוגמאות הרצה.

שאלה 2: סידור עמוק

כתבו פונקציה בשם print_sorted, המקבלת כקלט מבנה-נתונים עמוק כלשהו המורכב מרשימות (list), טאפלים (tuple), קבוצות (set), ומילונים (dict), ומדפיסה אותו כאשר הוא מסודר בכל הרמות (הערכים ברשימות, טאפלים וקבוצות מסודרים בסדר עולה; הערכים במילון מסודרים בסדר עולה של המפתחות). פורמט ההדפסה לבחירתכם. הפונקציה שלכם לא צריכה לשנות את הקלט.

דוגמה:

```
x = {"a": 5, "c": 6, "b": [1, 3, 2, 4]}
print_sorted(x)  # prints e.g. {"a":5, "b":[1,2,3,4], "c":6}
```

בדוגמה זו המבנה בעומק 2; הפונקציה שלכם צריכה לטפל במבנים בעומק כלשהו.

יש להגיש קישור לגיטהאב, הכולל קוד + תיעוד + בדיקות + דוגמאות הרצה.

שאלה 3: מציאת שורש

שיטת ניוטון-רפסון היא שיטה למציאת שורש של פונקציה ממשית כלשהי. אפשר למצוא בויקיפדיה הסבר על אופן פעולת הפונקציה.

כתבו פונקציה find_root, המקבלת פונקציה ממשית כלשהי ושני מספרים ממשיים, ומוצאת שורש שלה בתחום המוגדר ע"י המספרים. אפשר להניח שלפונקציה אכן יש שורש בתחום זה.

דוגמה:

find_root(lambda x: x**2-4, 1, 3) # should return 2 (approximately).

יש להגיש קישור לגיטהאב, הכולל קוד + תיעוד + בדיקות + דוגמאות הרצה.

שאלה 4: משחק תיכנות

פתחו חשבון באתר Coding Game:

https://www.codingame.com

בחרו שאלה אחת ברמה קלה (easy): https://www.codingame.com/training

ופתרו אותה בעזרת פייתון.

יש להגיש קישור לפתרון שלכם באתר codingame + צילום מסך.