הנחיות להגשה:

- בעבודה זו שני תרגילים, יש להגיש את שניהם
- עליכם להגיש **zip** הכולל שני קבצים:
  - הקובץ **py** המצורף לאחר השלמת הפונקציות כנדרש בעבודה
  - קובץ **readme** הכולל הסברים על המימוש וכן שמות ותעודות זהות של המגישים באופן הזהה לקובץ הדוגמא המצורף לעבודה זו
- ההגשה היא בזוגות
- רק בן זוג אחד מגיש את העבודה, יש לציין שם ותעודת זהות של שני המגישים ב**readme**
- **ניתן להיעזר בכלי בינה מלאכותית לצורך חיפוש מידע, העמקה והבמה <u>בלבד</u>. אין להגיש פתרון שנכתב על ידי** chat GPT **ודומיו .הגשה של עבודה שנכתבה על ידי מערכת בינה מלאכותית מהווה עבירה על כללי האתיקה ועשויה להוביל לצעדים משמעתיים**.
- **<u>לעבודה זו מצורפות שתי פונקציות עזר לנוחיותם. אתן רשאים להשתמש בהם לצורך כתיבת העבודה, אולם אין להגיש את הפונקציות בהגשה הסופית, ויש לוודא שמחקתם את הספריות שיבאתם בשבילם. הגשת הפונקציות עלולה לגרור בעייה בבדיקה.</u>**
- בהצלחה!

שיטת המיתר

באנליזה נומרית, שיטת המיתר היא שיטה איטרטיבית למציאת שורשים של פונקציה רציפה של משתנה אחד.

השיטה

ניקח שני ערכים התחלתיים, x_0 ו-x_1 , הקרובים יחסית לשורש המבוקש.

נבנה את הישר העובר דרך הנקודות $(x_0, f(x_0))$ ו-$(x_1, f(x_1))$. משוואת הישר היא:

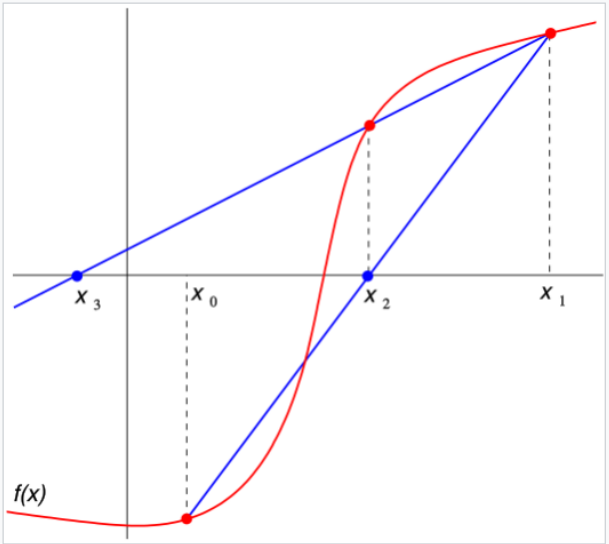
$$y = \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_1) + f(x_1).$$ נבחר

את x_2 , הערך הבא באיטרציה, להיות החיתוך של ישר זה עם ציר ה-x. $$x_2 = x_1 - f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)}$$

כעת נבנה ישר בין הנקודות $(x_1, f(x_1))$ ו-$(x_2, f(x_2))$, נחפש את החיתוך שלו בציר ה-x וכן הלאה. התהליך מוגדר בנוסחת הנסיגה הבאה:

$$x_n = x_{n-1} - f(x_{n-1})\frac{x_{n-1} - x_{n-2}}{f(x_{n-1}) - f(x_{n-2})}$$

ככל שנחשב יותר איטרציות של נוסחת הנסיגה, כך נקבל ערך קרוב יותר לערך השורש האמיתי.



שיטת המיתר: העקומה באדום מייצגת את הפונקציה והקווים הכחולים את הישרים שמחשבים בכל איטרציה של השיטה

לקריאה נוספת:
https://he.wikipedia.org/wiki/%D7%A9%D7%99%D7%98%D7%AA_%D78%94%D7%9E%D7%99%D7%AA%D7%A

השלימו את הפונקציה

```
def Secant_method(function: str, x0: float, x1: float) -> float:
```

אשר מחשבת שורשי פונקציה בשיטת המיתר.

הפונקציה תקבל את הפרמטרים הבאים :

1. **function** - מחרוזת המייצגת את הפונקציה המתמטית אשר יש לחשב עבורה את השורשים.
2. X0 מטיפוס float- מספר קרוב לשורש
3. X1 מטיפוס float- מספר קרוב לשורש

הפונקציה תקבל את שני הפרמטרים ותחשב את הקירוב לשורש עבור אפסילון 0.001 ו מקסימום של 30 אטרציות.

לרשותכם הפונקציה **str_to_func** אשר ממירה מחרוזת לפונקציה איתה תוכלו להשתמש.

דירוג מטריצות

שימו לב! בשאלה זו אנו דנים בפעולות במטריצות. ייתכן כי תתקלו בחומר שטרם למדתם, אבל השאלה מכילה את כל המידע החיוני. אם יש שאלות, מוזמנים לפנות לסגל הקורס.

שימו לב, בשאלה זו אין להשתמש בספרייה Numpy או כל ספריית מטריצות אחרת. ניתן להשתמש רק בפונקציות מובנות של השפה. שימוש בפונקציות אסורות יגרור פסילה .

במתמטיקה, מטריצה היא טבלה ממימד mxn - כלומר מכילה m שורות ו n עמודות) של ערכים. ניתן להתייחס אליה כאל רשימה של רשימות בשפת פייתון, כאשר כל תתי הרשימות באותו גודל. בדוגמא זו מדובר ברשימה בת m רשימות כך שכל רשימה מכילה n ערכים

למטריצות חשיבות עצומה במתמטיקה ומדעי המחשב.

ישנן מספר פעולות אלמנטריות במתמטיקה אשר הפעלתן על המטריצה אינה משנה את "מרחב הפתרונות" שלה , דהיינו אם קיים וקטור (מטריצה של עמודה אחת) x עבורו $Ax = 0$ (כאשר A מטריצה ו x אינו וקטור של אפסים), גם $Bx = 0$ כאשר B המטריצה המתקבלת על ידי ביצוע פעולות אלמנטריות.

הפעולות (המכונות "פעולות אלמנטריות") הן :

1. החלפת שורות
2. כפל שורה בסקלר
3. חיבור או חיסור שורות זו מזו

אחד התהליכים החשובים באלגברה לינארית הוא דירוג מטריצות. בתהליך זה, מבצעים את אותן פעולות אלמנטריות על המטריצה עד לקבלת מטריצה שחלקה השמאלי התחתון אפסים.

לדוגמא:

$$\begin{pmatrix} 2 & 4 & -2 & 2 \\ 4 & 9 & -3 & 8 \\ -2 & -3 & 7 & -1 \\ 6 & 15 & -7 & 9 \end{pmatrix} - - - - - - \begin{pmatrix} 1 & 2 & -1 & 1 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 1 & -3/4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

שימו לב שה"משולש" השמאלי התחתון מורכב מאפסים)

תהליך האלימינציה של גאוס הוא אחד האלגוריתמים המשמשים לדרוג המטריצה.

להלן שלבי האלגוריתם :

עבור מטריצה בת m שורות ו n עמודות כך ש $m \leq n$:

- עבור $i = 0$ עד n:
  - אם האיבר בשורה ה i בעמודה ה i הוא 0, החלף בין השורה ה i לשורה כלשהי j כך ש $i < j$ והאיבר בשורה הj בעמודה ה i שונה מ 0.
  - חלק את השורה ה i באיבר בשורה ה i בעמודה i
  - עבור $j = i + 1$ עד n:
    - החסר מהשורה ה j את השורה ה i מוכפלת באיבר בשורה ה j בעמודהi

שימו לה כי מדובר בלולאה. התהליך ממשיך עבור שאר השורות

דוגמא:

$$\begin{pmatrix} 2 & 4 & -2 & 2 \\ 4 & 9 & -3 & 8 \\ -2 & -3 & 7 & -1 \end{pmatrix}$$

בשלב הראשון נביט על הספרה הראשונה בשורה הראשונה. היא אינה 0 ולכן נחלק את השורה הראשונה בערך (2)

$$\begin{pmatrix} 1 & 2 & -1 & 1 \\ 4 & 9 & -3 & 8 \\ -2 & -3 & 7 & -1 \end{pmatrix}$$

בשלב הבא, נביט על הספרה הרשרונה בשורה השנייה (4). נחסר מהשורה השנייה 4 פעמים את השורה הראשונה. נעשה כך עבור כל השורות עם הערכים המתאימים להם (2- ו 6)

$$\begin{pmatrix} 1 & 2 & -1 & 1 \\ 0 & 1 & 1 & 4 \\ 0 & 1 & 5 & 1 \end{pmatrix}$$

נחזור על הפעולות עבור כל השורות

$$\begin{pmatrix} 1 & 2 & -1 & 1 \\ 0 & 1 & 1 & 4 \\ 0 & 1 & 5 & 1 \end{pmatrix} \to \begin{pmatrix} 1 & 2 & -1 & 1 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 4 & -3 \end{pmatrix} \to \begin{pmatrix} 1 & 2 & -1 & 1 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 1 & -3/4 \end{pmatrix}$$
$$\to \begin{pmatrix} 1 & 2 & -1 & 1 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 1 & -3/4 \end{pmatrix}$$

השלימו את הפונקציה

```
def ge(matrix: list[list[float]]) -> list[list[float]]:
```

אשר מקבלת רשימה של רשימות של מספרים המייצגת מטריצה, ומחזירה את המטריצה המדורגת ע"פ האלגורים המתואר בעבודה זו.

- אין צורך לבדוק תקינות קלט – הניחו כי קלט הפונקציה תקין.
- ניתן להניח כי מספר השוקות גדול ב 1 ממספר העמודות
- לרשותכם הקובץ helpers.py עם שתי פונקציות עזר לבדיקה:
  - Test_ge הבודקת האם שתי מטריצות שקולות
  - Print_mat המד]יסה מטריצה

אתם רשאים להשתמש בהם לצורך בדיקה, אבל **שימו לב שאינכן מגישים אותם או מייבאים את הספריות שלהם בגרסאת ההגשה**

דוגמאות:

קלט	פלט
[[2,4,-2,2],[4,9,-3,8],[-2,-3,7,-1]]	[[1,2,-1,1],[0,1,1,4],[0,0,1,-0.75]]
[[4,8,12],[5,16,20]]	[[1,2,3],[0,1,0.83333]]
[[5,8]]	[[1,1.6]]