חלק ג' – עיבוד נתוני המעבדה (מתחילים לעבוד)

2. ביצוע חישוב על וקטורי נתונים, כתיבת פונקציה (10 סעיפים וגם אתם כותבים פונקציות)

לצורך התרגיל, נניח שהווקטור times מכיל זמנים בהם מדדנו נפילה חופשית של גוף כלשהו, תחת תאוצה הכבידה. כעת נחשב את המיקום (הגובה) של הגוף בזמנים הנתונים שכביכול מדדנו במעבדה.

1. **פתחו** cell חדש **ורשמו** לו את הכותרת vector calculations
2. **צרו** משתנה x0=0.8 **וציינו** יחידות של m. **צרו** משתנה v=1.6 **וציינו** יחידות m/sec. **צרו** משתנה a=-9.8 **וציינו** יחידות m/sec**2. (כן, יחידות זה חשוב. "משיקולי יחידות" וכו')
3. **חשבו** את הווקטור distances לפי משוואת התנועה, עבור המיקום ההתחלתי x0, המהירות v, התאוצה a והזמנים times. יש לכתוב שורת קוד אחת. **ציינו** את היחידות בסוף השורה.

כפי שראיתם, הפעולות המתמטיות הבסיסות פועלות על numpy.array כצפוי. כדי לבצע פעולות יותר מסובכות, נשתמש מפורשות בספריית numpy. באופן כללי, כל פונקציה מוכרת (כזאת שאתם יכולים להקליד למנוע חיפוש של wolfram alpha או למחשבון) קיימת בnumpy. דוגמאות: np.arctan(), np.sqrt(), np.abs().

4. **חשבו** את השגיאה הנגררת של וקטור המרחקים distances_err לפי הנוסחה:

$$distances_err = \left|\frac{dx}{dt}\right| \cdot \delta t = |v + a \cdot t| \cdot 0.01$$

5. **עגלו** את השגיאה לספרה הדומיננטית. לשם כך **הקלידו** בconsole "np.round?" ולפי התיעוד של הפונקציה **כתבו** את הפקודה המתאימה.

---

6. **פתחו** cell חדש **וקראו** לו defining functions.

נדגים כעת כתיבה של פונקציה. בכל שפת תכנות, לפונקציות יש 3 תכונות:

- שם הפונקציה (למשל functionName)
- הארגומנטים שהיא מקבלת - input
- הערכים שהיא מחזירה – output

והשימוש בפונקציה ("הקריאה לפונקציה") הוא תמיד באמצעות הפקודה

the_corresponding_output = functionName(some_specific_input)

הסינטקס שבו מגדירים את הפונקציות משתנה משפה לשפה. פונקציה בפיתון כותבים באופן הבא:

```
def functionName(argument1, argument2, argument3):
    # some calculation on the arguments, for example
    temp1 = np.max(argument1) + 50 * argument2
    return np.abs(np.sqrt(argument3)) – temp1
```

נשים לב:

- פונקציה נפתחת במילית השמורה def (קיצור של define)

- הארגומנטים לפונקציה מוגדרים בסוגריים עגולים "()".
- שמות הארגומנטים מוגדרים בשורת ה-def וקיימים רק בתוך הקונטקסט של הפונקציה. אם אתם משתמשים בשם של משתנה שמוגדר כבר מחוץ לפונקציה, ערך הארגומנט "ידרוס" את הערך הזה בתוך הקונטקסט של הפונקציה.
- השורה def מסתיימת בנקודותיים שמסמנות פתיחה של code block והשורות שבתוך הblock מוזחות (indented) בtab בודד או 4Xspace במקלדת (ביחס לשורה של ה-":"). שורות שאינן בתוך הפונקציה ייכתבו בהזחה רגילה – כלומר ללא הזחה. (אפשר לכתוב code block שהוא בתוך code block אחר וכן הלאה. ההזחה מסמנת את ההיררכיה.)
- הערך שמוחזר מהפונקציה נכתב אחרי המילית return.
- אפשר לכתוב פונקציה בתוך פונקציה. או slicing אחרי פונקציה, בתוך פונקציה. או slicing של slicing בתוך פונקציה של פונקציה. בכללי בפיתון, התשובה לשאלה "האם אפשר..." היא בדרך כלל "כן, תנסו!" וכדאי לצאת מנקודת ההנחה הזאת כשכותבים קוד בפיתון =)
- בניגוד ל-list או פעולות slicing שמשתמשים בהן בסוגריים מרובעים "[]", סוגריים עגולים יוצרים אובייקט שנקרא tuple – כמו רשימה, אבל אי אפשר לשנות את התוכן של tuple אחרי שהוא נוצר. אם רוצים להחזיר מהפונקציה מספר ערכים, מחזירים אותם כ-tuple.

לדוגמה:

```
def func1(arg1, arg2):
    return (arg1/2, arg2+5)
(A,B) = func1(2.2,15)
```

7. שאלה: אילו משתנים יופיעו ב-variable explorer אחרי הרצה של הקוד הזה ומה יהיו הערכים שלהם?

8. נניח שנרצה לדעת מתי הגוף הנופל עובר גובה מסוים. **כתבו** פונקציה בשם timeOfCrossing שמקבלת וקטור זמן t, וקטור מרחק d וערך חצייה cross_value ומחזירה את הזמן בו המרחק של הגוף היה הכי קרוב לערך החצייה (הזמן שמתאים לאיבר ב-d שהוא הקרוב ביותר ל- cross_value בערך מוחלט). ניתן להשתמש בפונקציה np.argmin().
9. אם נריץ את cross_time = timeOfCrossing(times, distances, 0.5),
מה הערך שיתקבל ב-cross_time? _____
10. **בדקו** שאתם אכן מקבלים את הערך הרצוי._____