0.1 הכנת סביבת העבודה

1. הריצו בהצלחה את הקוד הבא:

```
import numpy as np
import scipy
import matplotlib.pyplot as plt
import pandas as pd
```

2. הריצו בהצלחה את הקוד הבא:

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array(range(50))
plt.plot(x)
plt.show()
```

אם הקוד קורס, נסו להריץ:

```
import matplotlib
help(matplotlib.use)
```

בחרו את ה- backend המתאים למחשב שלכם והריצו בהצלחה (החליפו את TkAgg בשם הסביבה המתאימה ביותר למחשב שלכם) :

```
import matplotlib
matplotlib.use('TkAgg')

import numpy as np
import matplotlib.pyplot as plt

x = np.array(range(50))
plt.plot(x)
plt.show()
```

0.2 הכרת ndarray

1. הוסיפו את numpy לסביבת העבודה
2. צרו מערך בצורה הבאה:

```
v = np.array(5)
```

3. העזרו בפונקציה type לבדיקת הטיפוס של v.
4. הדפיסו את פלט התכונה (attribute) shape של v.
5. הדפיסו את v.
6. הגדירו מחדש את המערך בצורה הבאה:

```
v = np.array([1, 2, 3, 4, 5])
```

7. הדפיסו את פלט התכונה shape של v.
8. הדפיסו את v.
9. הגדירו מחדש את המערך בצורה הבאה:

```
v = np.array([[1, 2, 3, 4, 5]])
```

10. הדפיסו את פלט התכונה shape של v.

11. הדפיסו את v.
12. צרו משתנה חדש u על ידי קריאה למתודה reshape של v עם הפרמטר (1 ,1-).
13. הדפיסו את פלט התכונה shape של u.
14. הדפיסו את u.
15. צרו את המשתנה החדש m על ידי הכפלת u ב- v בעזרת האופרטור *.
16. הדפיסו את פלט התכונה shape של m.
17. הדפיסו את m.
18. חזרו על ההכפלה, הפעם היפכו את הסדר של u ו v. האם הצורה השתנתה ? מה קרה ?
19. צרו את m מחדש, הפעם בצעו את ההכפלה עם האופרטור @. הדפיסו את המשתנה, הדפיסו את ה- shape שלו. שנו את הסדר של u ו- v ונסו שוב.
20. נסו להכפיל את v בעצמו בעזרת *. נסו להכפיל את v בעצמו בעזרת @.
21. נסו להכפיל את v ב-10, מה קרה ?
22. הוסיפו את הספריה time לסביבת העבודה.

    צרו רשימה (list) באורך 1000, תוכלו להעזר ב- range.

    הוסיפו שמירה של הזמן הנוכחי בעזרת time.time לתוך המשתנה start.

    צרו לולאה שתרוץ 10000 פעמים ותריץ את הפונקציה add_loop על הרשימה שיצרתם.

    הדפיסו את זמן הריצה על ידי הדפסת ההפרש בין הזמן הנוכחי ל- start.

    כמה זמן זה לקח ?
23. חזרו על התרגיל, הפעם צרו מערך של numpy והגדילו אותו בעזרת אופרטור ההצבה =+.

    כמה זמן זה לקח ?

0.3 תרגיל הגשה עם ציון

ממשו את הפונקציה עם החתימה הבאה:

```
def measure_run_diff(vec_size: int, num_tests: int) -> float:
```

הפונקציה תיצור פעם רשימה של פייתון ופעם מערך של NumPy עם vec_size איברים. הפונקציה תרוץ בלולאה ו- num_tests פעמים תכפיל את כל המערך / רשימה ב-3.
הפונקציה תחזיר את ההפרש בין זמן הריצה הממוצע של ביצוע הפעולה עם NumPy לביצוע הפעולה עם פעולות נקיות של פייתון.

0.4 תרגול עצמי בשעות הפנאי

1. ממשו בפייתון, ללא ספריות, פונקציה היודעת לבצע מכפלת מטריצות. אין צורך לבדוק נכונות קלט.

   בדקו את נכונות הפלט על ידי השוואה ל- numpy. כדאי לפחות מכפלה אחת לוודא גם על נייר.
2. כתבו פונקציה שיודעת לקבל שתי מטריצות, פונקציה להכפלת מטריצות ומשתנה מניה n . העזרו בפונקציה

   time מהספריה time ומדדו כמה זמן לוקח לבצע את המכפלה בין שתי המטריצות n פעמים.
3. צרו זוגות מטריצות אשר ניתן להכפיל. צרו מטריצות בגדלים שונים. הגדירו את המטריצות בעזרת רשימות.

   בחרו n מתאים.
4. הכפילו את כל זוגות המטריצות, ומצאו את הזמן הממוצע. העזרו בפונקציה שכתבתם בסעיף הקודם בשביל לבחור, הכפילו פעם אחת עם המימוש שלכם לכפל מטריצות, ופעם אחת בצעו את המכפלה בעזרת numpy.

שימו לב לבצע את ההכפלות ב- numpy בעזרת @.