שלום

כאן יש את מדריך משתמש לקוד שלנו כדי לענות על שאלת חקר שלנו.

שאלת החקר שלנו: איך ביצועים של מודלי LLM שונים (כמו gpt-4o, gpt-4o-min) יהיו שצריכים לעבוד עם ספריות פרטיות שלא מכירים מעבר, ב python ו Javascript , ואם הביצועים שלהם מספיק טובים כדי לסמוך עליהם ?

כדי לענות על השאלה אז החלטנו לכתוב קוד בשתי שפות תכנות Python ו Javascript כדי להעריך ביצועים מודלי LLM עם ספריות פרטיות בשפות אלה.

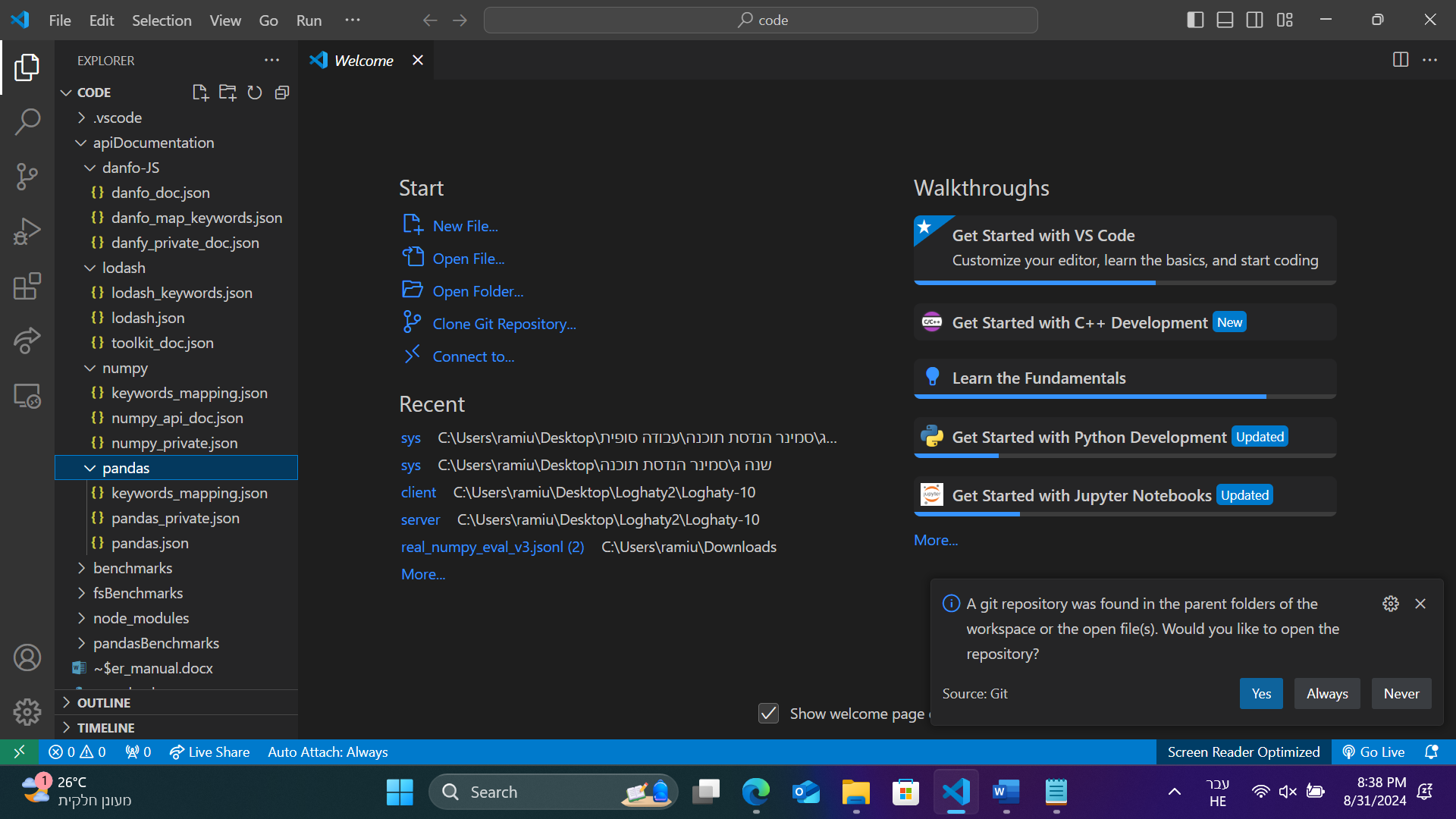
בחרנו להתחבר ל OpenAI API ולכן שלמנו עבור קבלת api שדרכו ניתן לעבוד עם המודלים השונים של החברה.

איך עשינו זאת ?

קודם כל בחרנו 2 ספריות ב Python (Pandas, NumPy) ו 2 ספריות ב Javascript (Lodash, Danfojs) והפכנו אותן לספריות פרטיות על ידי מיפוי מילות מפתח בכל אחת מהספריות.

בקובץ **apiDocumentation** יש תיעוד לכל אחת מהספריות ה 4 , התיעוד הוא חלקי ולא לכל הספרייה מאחר שספריות אלו ענקיות ולא צריך כל מה יש בהן כדי להעריך את הביצועים למודל.

בעזרת מילות מפתח עשינו מיפוי כך כדי להתאים לתיעוד של ספרייה פרטית (כך בעצם שינוי שמות פונקציות ותיאור שלהן, ברור גם שינוי שם הספרייה עצמה, כך המודל יתייחס אליה כספריה פרטית שלא מכיר אותה, אף פעם).

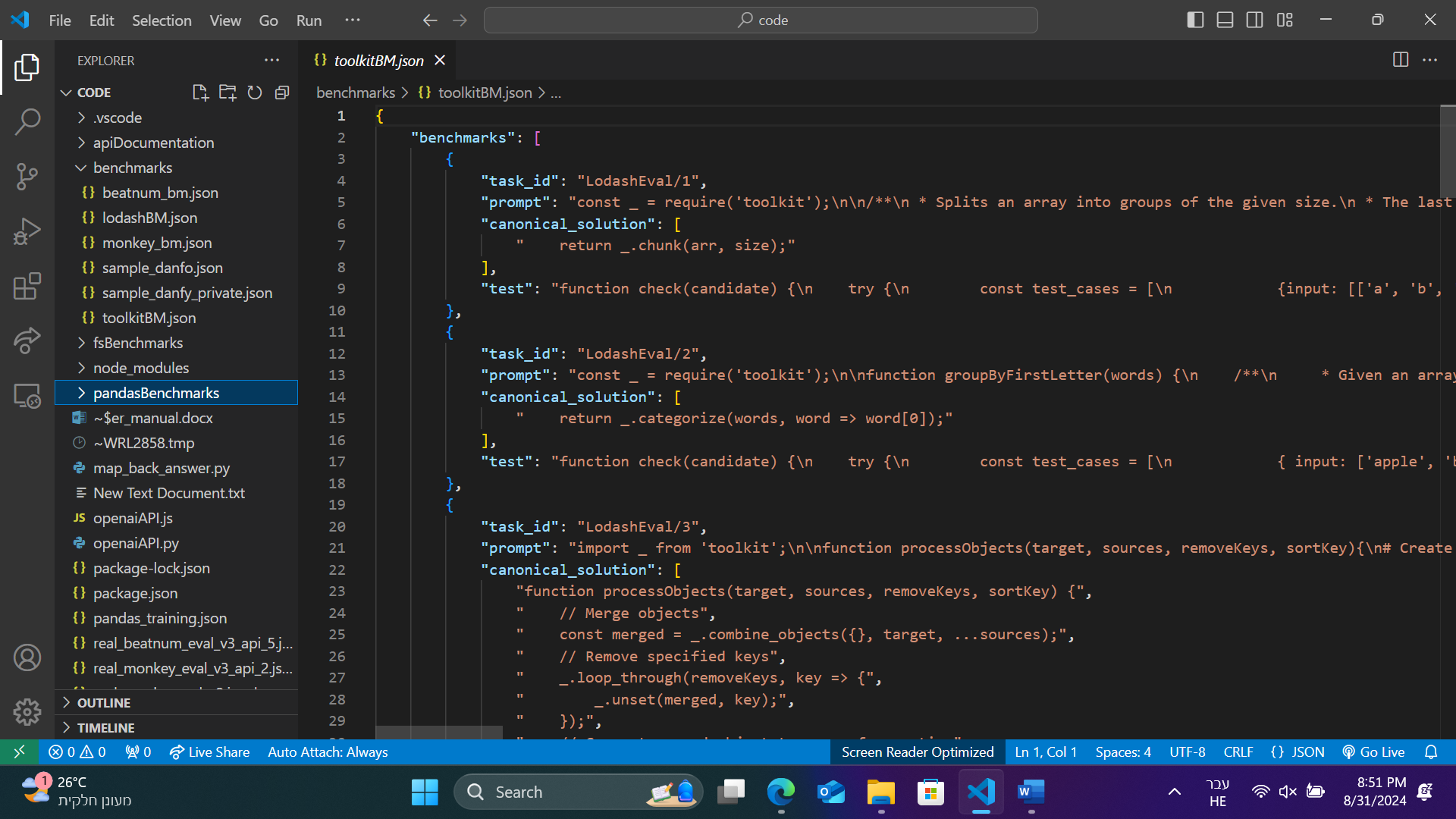


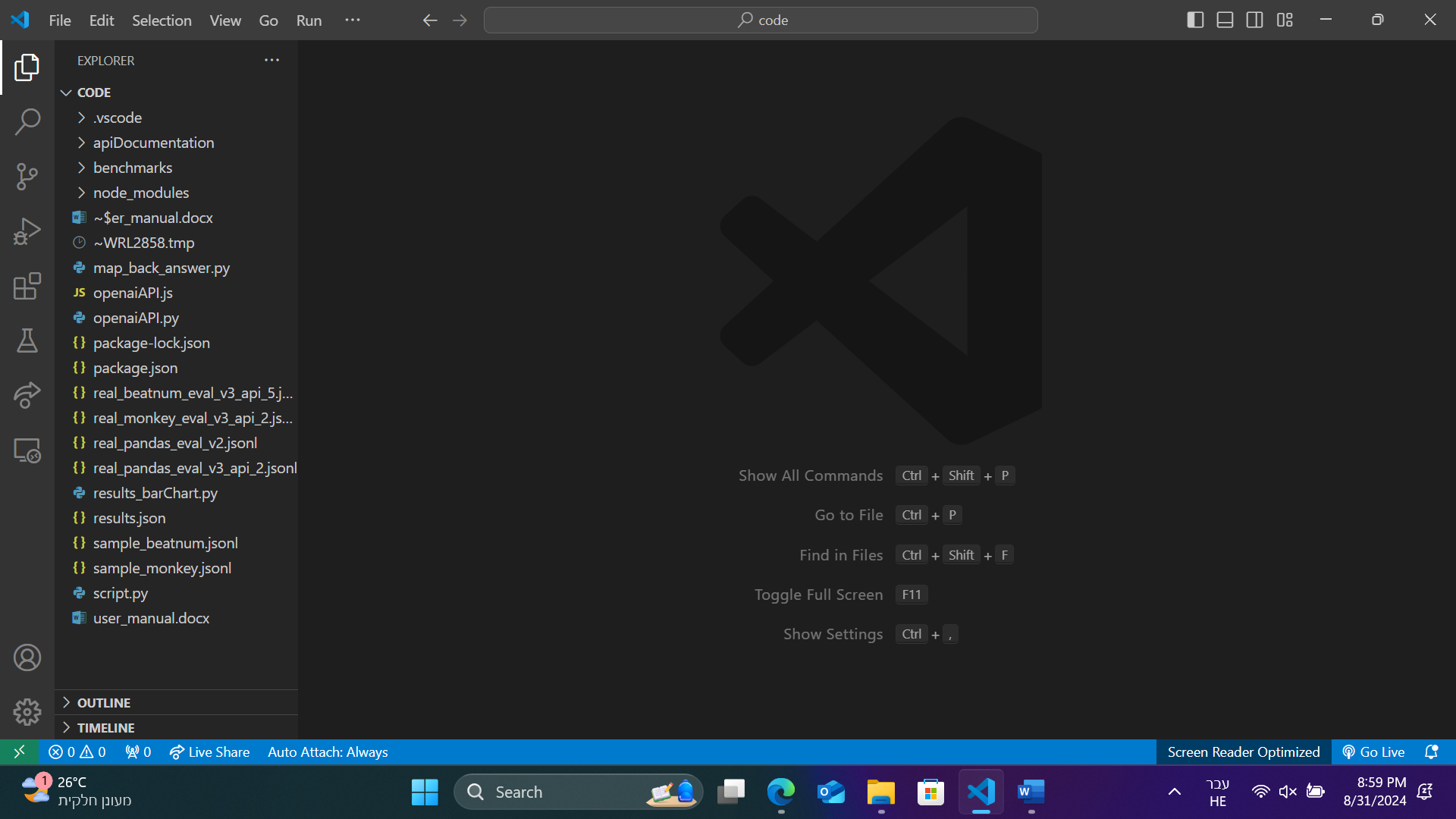
בקובץ **benchmarks** יש לנו את ה benchamrks לכל אחת מ 4 ספריות כדי להעריך ביצועים, בהתחלה עבור pandas ו numpy היו benchmarks זמנים מ github למאמר השני:  
[PyCodeGPT/apicoder/private-eval/data at main · microsoft/PyCodeGPT · GitHub](https://github.com/microsoft/PyCodeGPT/tree/main/apicoder/private-eval/data)

לכן עשינו סינון לקחנו הרלוונטיים ביותר ביחס לתיעוד שיצרנו.

עבור שתי ספריות ב JS לא מצאנו benchmarks לכן יצרנו בעזרת prompts רלוונטים לספרייה.

בסוף המרנו את benhmarks לרלוונטים לספריות פרטיות ע''י מיפוי מילות מפתח, וכך נוכל להשתמש בהם להעריך ביצועים.



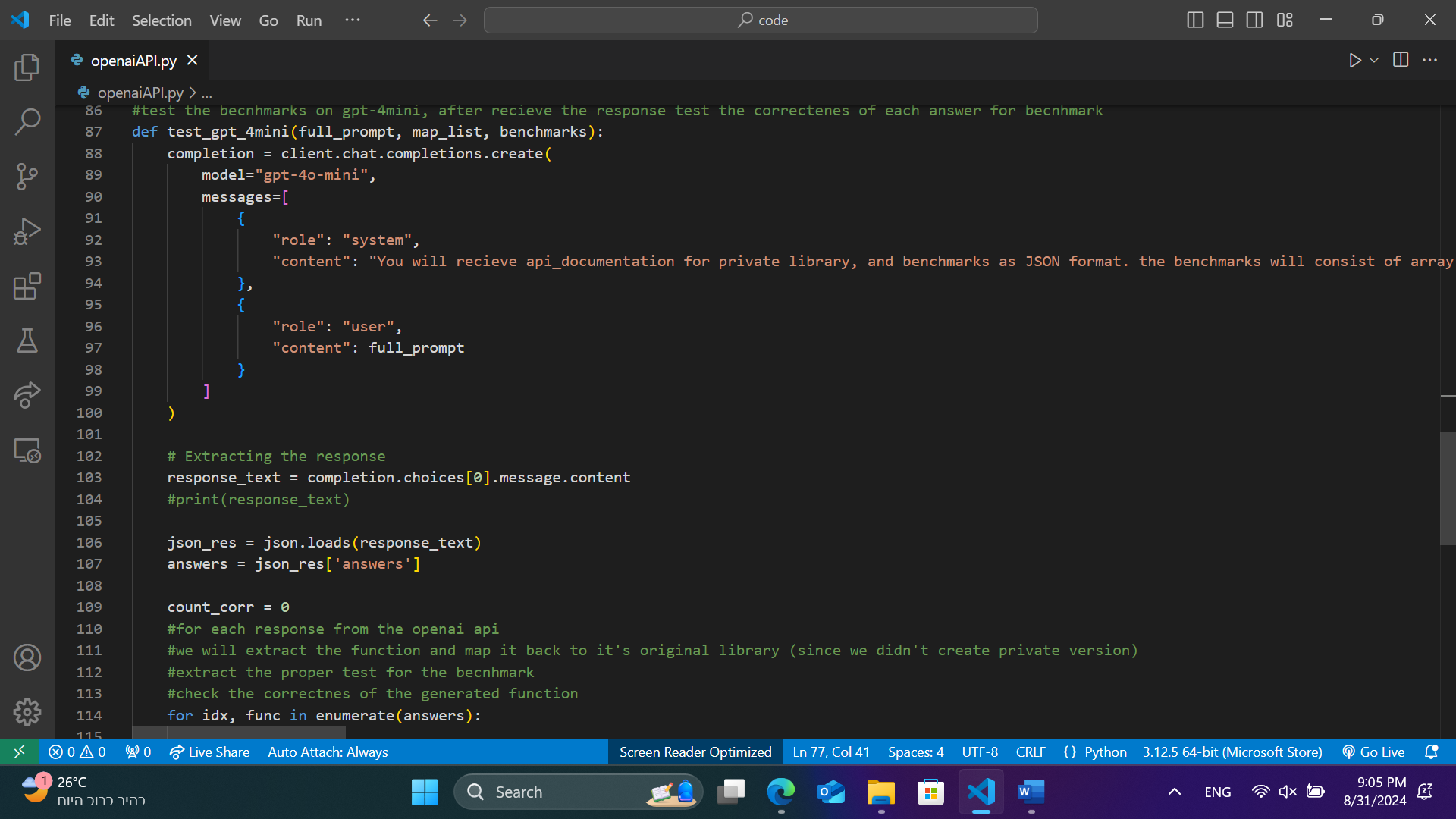
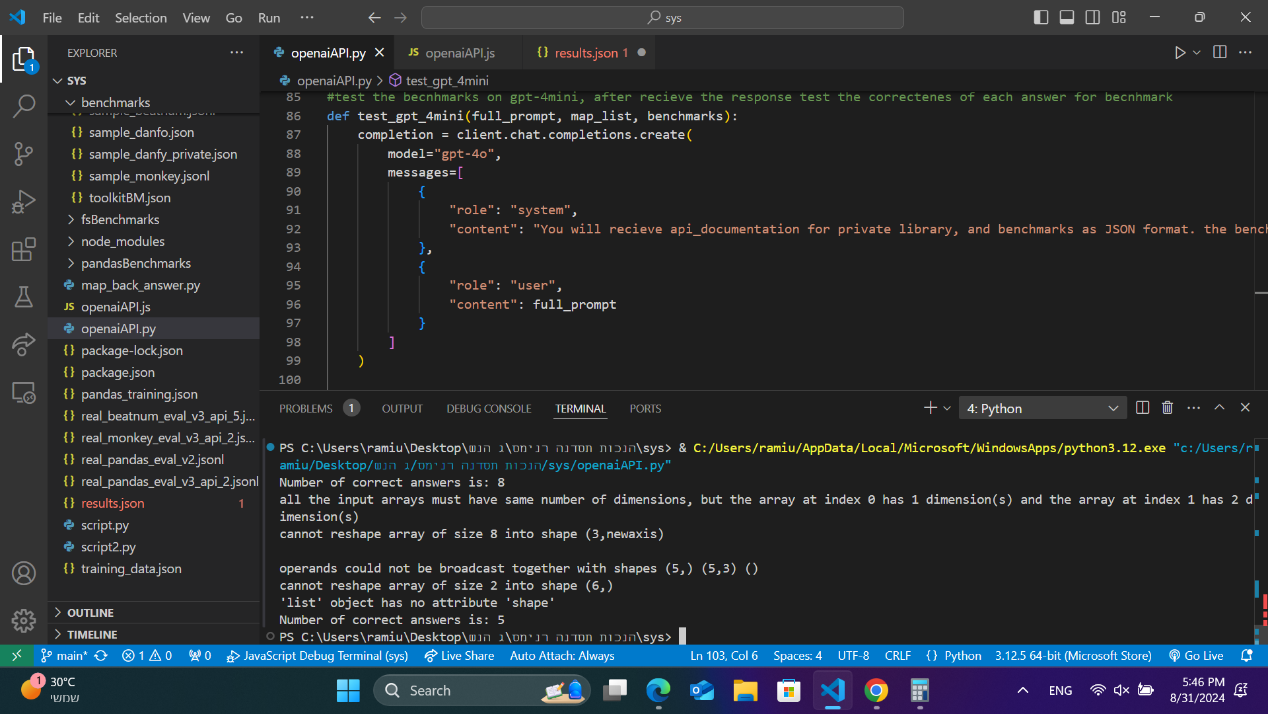
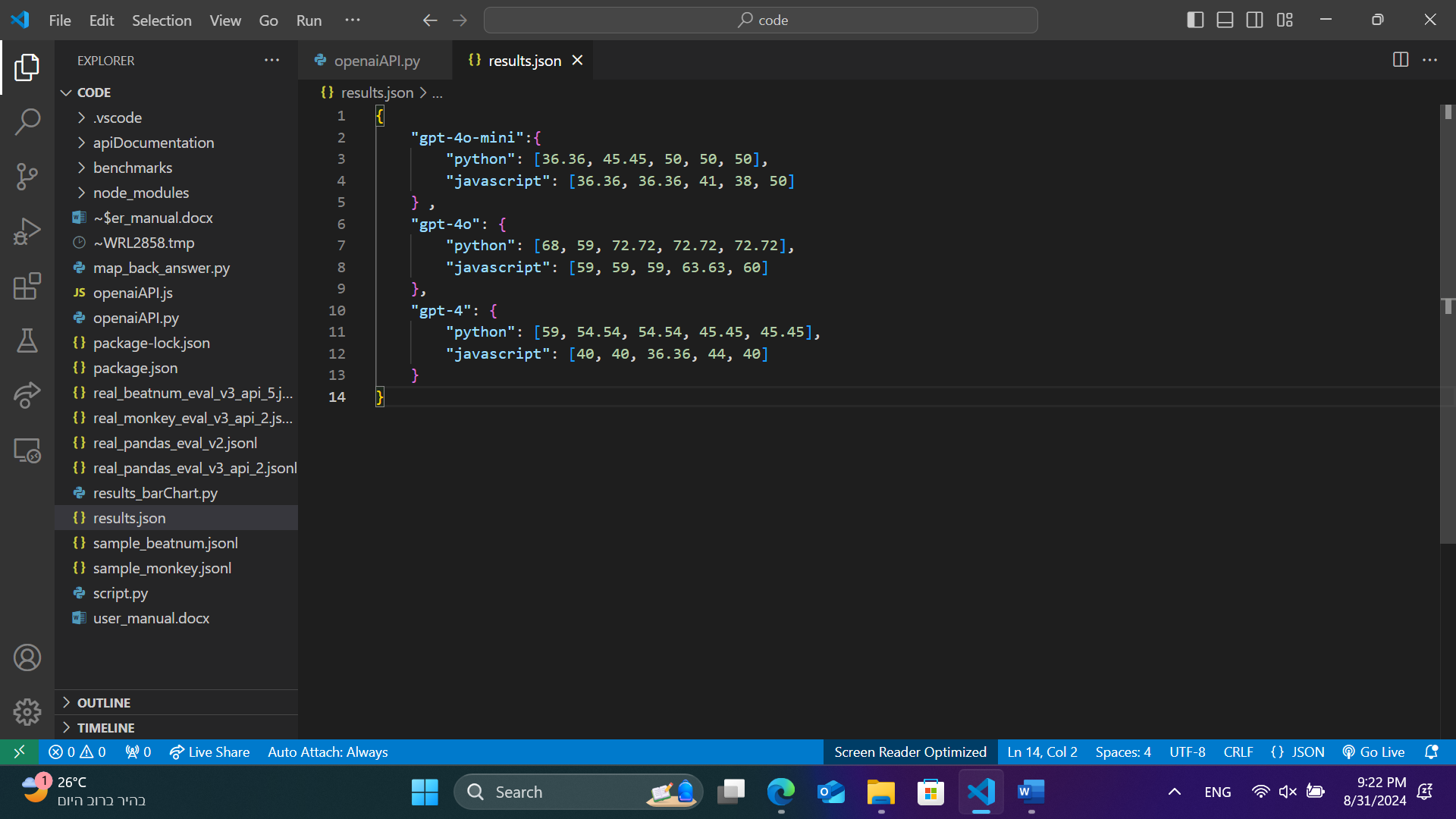
עכשיו יש לנו 2 קבצים חשובים:  


המטרה אחת לשני הקבצים.

לפייתון המטרה להעריך הביצועים של 2 ספריות פרטיות – beatnum ו monkey

ל JS כדי להעריך את הביצועים של Toolkit ן Danfy

אז מה בדיוק מטרת הקוד בכל אחד מהקבצים ?

1. להתחבר ל OpenAI API
2. החיבור בעזרת api key שמשלמים עליו
3. העלאת קבצים תיעוד documentation, benchmark ו map key words list
4. עבור כל ספרייה נשלח את התיעוד עם benchmarks כדי שיענה עליהם ונוכל להעריך את הקוד שהחזיר
5. 
6. בקוד למעלה מראה חיבור למודל gpt-4o-mini
7. **עם קבלת התשובה לכל אחד מה benchmarks** נעבור עליהם אחד אחד, וננפה אותה בחזרה ל syntax של הספרייה הרגילה, למשל מ syntax של monkey נחזור ל syntax של pandas
8. דבר זה מאחר שהיה **מאוד קשה להפוך ספרייה רגילה לפרטית**, מצאנו קוד שהעלו ל github שעושה את בעזרת מיפוי מילות מפתח בספרייה, אך בהמשך עשה המון bugs ובעיות מאחר שצריך לעדכן קבצי .build של כל ספרייה וזה קשה ומסוכן. (הקוד נמצא בקוד script.py).
9. המיפוי הזה מאפשר לנו להריץ הקוד בספריה המקורית ולהעריך את הביצועים. למודל LLM לא ישפיע מאחר שקיבל תיעוד עם שמות פו' ותיאור שוני לגמרי מהספרייה המקורית.
10. המיפוי בחזרה נעשה בעזרת פו': map\_back\_keywords
11. הרצנו את הקוד ובדקנו ביצועים, לכל אחת מהספריות היה מדפיס על כמה benchmarks עבורו בהצלחה למשל 5 מ 10 עבור ספריית monkey
12. 
13. ניתן לראות שענה נכון על 8 לספריה ראשונה + 5 לספרייה שנייה = 13  
    מסך הכל 22
14. הרצנו את שני הקבצים מס' פעמים, על כל מודל הרצנו 5 פעמים על בדיקה ספריות פייתון ו 5 פעמיים לבדיקה JS.
15. כל פעם מקבלים מס' שונה. (יש לנו 22 benchmarks לשתי ספריות ב פייתון, ואותו מס' 22 לשתי ספריות ב JS)
16. עשינו את זה ל 3 המודלים gpt-4o gpt-4o-mini gpt-4
17. בכל פעם חשבנו ממוצע ההצלחה לכל מודל ולכל שפה.
18. התוצאות בקובץ results.json
19. 
20. למשל אם הרצנו gpt-4o כל פייתון וענה נכון על 11 מ 22 (סך הכל benchamarks לשתי הספריות) אז יקבל 50. ונחזור על זה 5 פעמים..
21. קובץ results\_barChart.py מייצר bar chart שמראה ההבדל בביצועים בין 3 המודלים. (לאחר שמחשבים הממוצע לכל אחת מהשפות עבור כל מודל).

כדי שהקוד יעבוד תקין:

1. צריך להתקין openai api לפייתון ו js   
   ל python -> pip install openai  
   ל JS -> npm I openai
2. להתקין גם 4 הספריות  
   ל פייתון -> pip install pandas  
   pip install numpy  
   ל npm Install lodash  
   npm Install danfojs
3. כדי להריץ קובץ openaiAPI.js ניתן בעזרת ה terminal לרשום: node .\openaiAPI.js