**סטגנוגרפיה | מבוא לאבטחת המרחב המקוון**

רקע: Steganography היא שיטה להסתרת מידע בתוך קובץ, הודעה תמונה או וידאו.

עבודה זו עוסקת בהסתרת טקס בתוך תמונה, ובפענוח של טקסט מוסתר מתוך תמונה.

**חלק א**

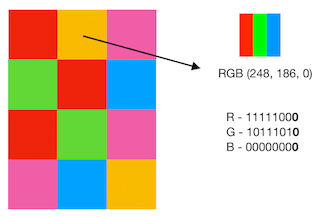
בחלק זה התבקשנו לממש הסתרה של טקסט בתוך תמונה מסוג PNG.

איך הטקסט מוסתר? לפני כן נקפוץ רגע להסבר קצר:

תמונה מורכבת מהמון פיקסלים.

כל פיקסל מורכב משלושה בתים בני 8 ביטים (ערכים 0-255), הנקראים RGB

(Red, Green Blue). כל שלושה בתים שכאלו, מייצגים את העוצמה של כל צבע בפיקסל יחיד. כלומר הפיקסל נצבע בצבע שהוא שילוב של אדום ירוק וכחול בעוצמה שונה.



אז איפה אנחנו מסתירים את הטקסט?

כמו שאמרנו - תמונה מורכבת מפיקסלים, וכל פיקסל מורכב משלושה צבעים.

צבע, מיוצג על ידי בית (Byte) שמורכב מ-8 ביטים (Bits).

ייצוג העוצמה של כל צבע בתוך הבית, הוא כמובן בינארי.

אנחנו מסתירים את הטקסט בתוך הביטים הנמוכים (LSB) שבכל בית.

איך עושים זאת?

לכל אות בשפה האנגלית (וגם לתווים אחרים, כמו רווחים וסימני פיסוק) יש ייצוג ascii שונה. נניח לאות H יש את הייצוג 72, או בבינארי 01001000.

ההסתרה בתמונה נעשית ככה שאנו עוברים על כל הביטים בייצוג הבינארי של ערך הascii של האות , ושמים אותם בביט הנמוך (LSB)של כל בית בתמונה.

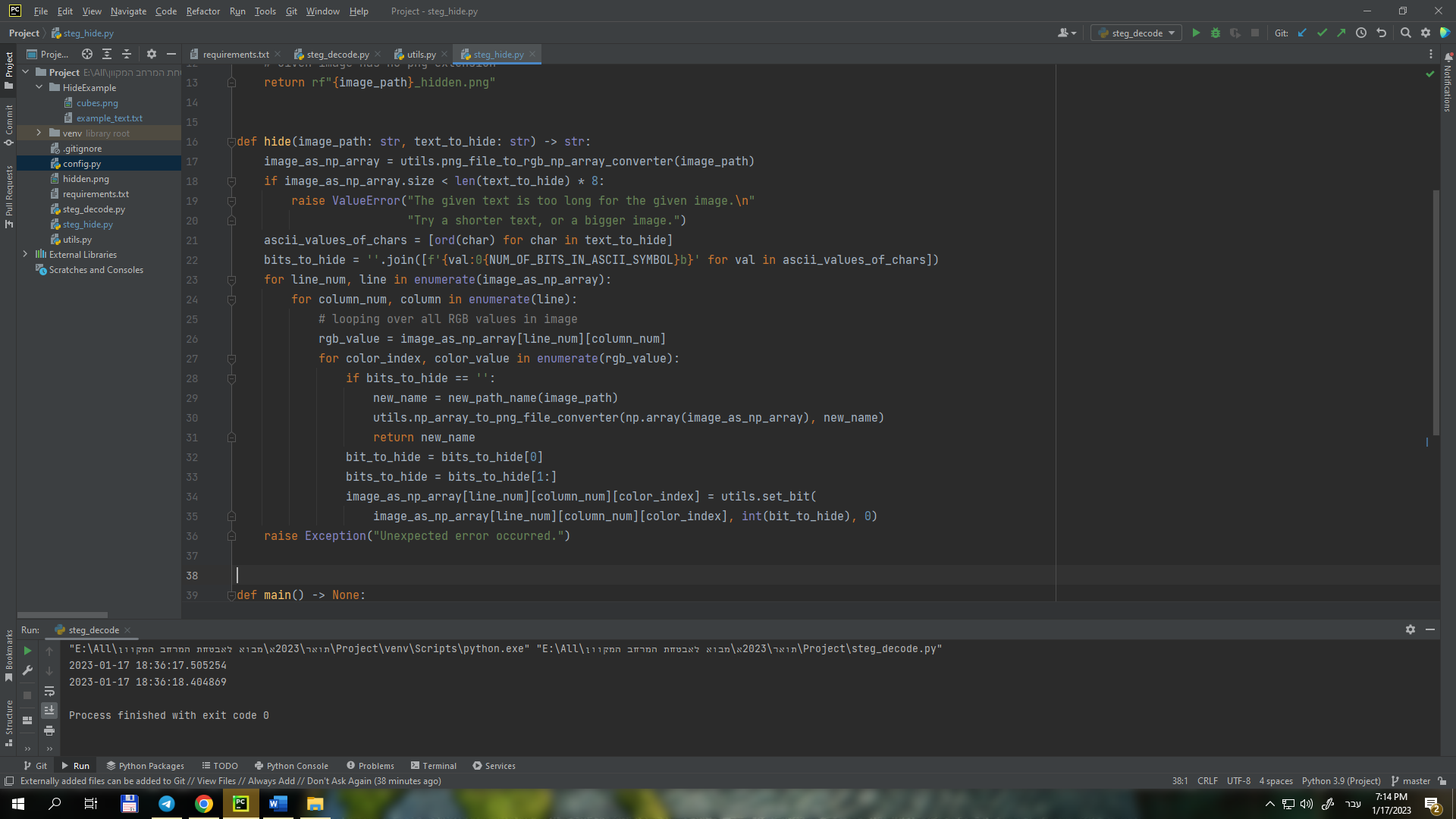
לדוגמה, אם ערך ה-ascii של האות H הוא 72, או בבינארי 01001000. אז המעבר יהיה:

0->1->0->0->1->0->0->0

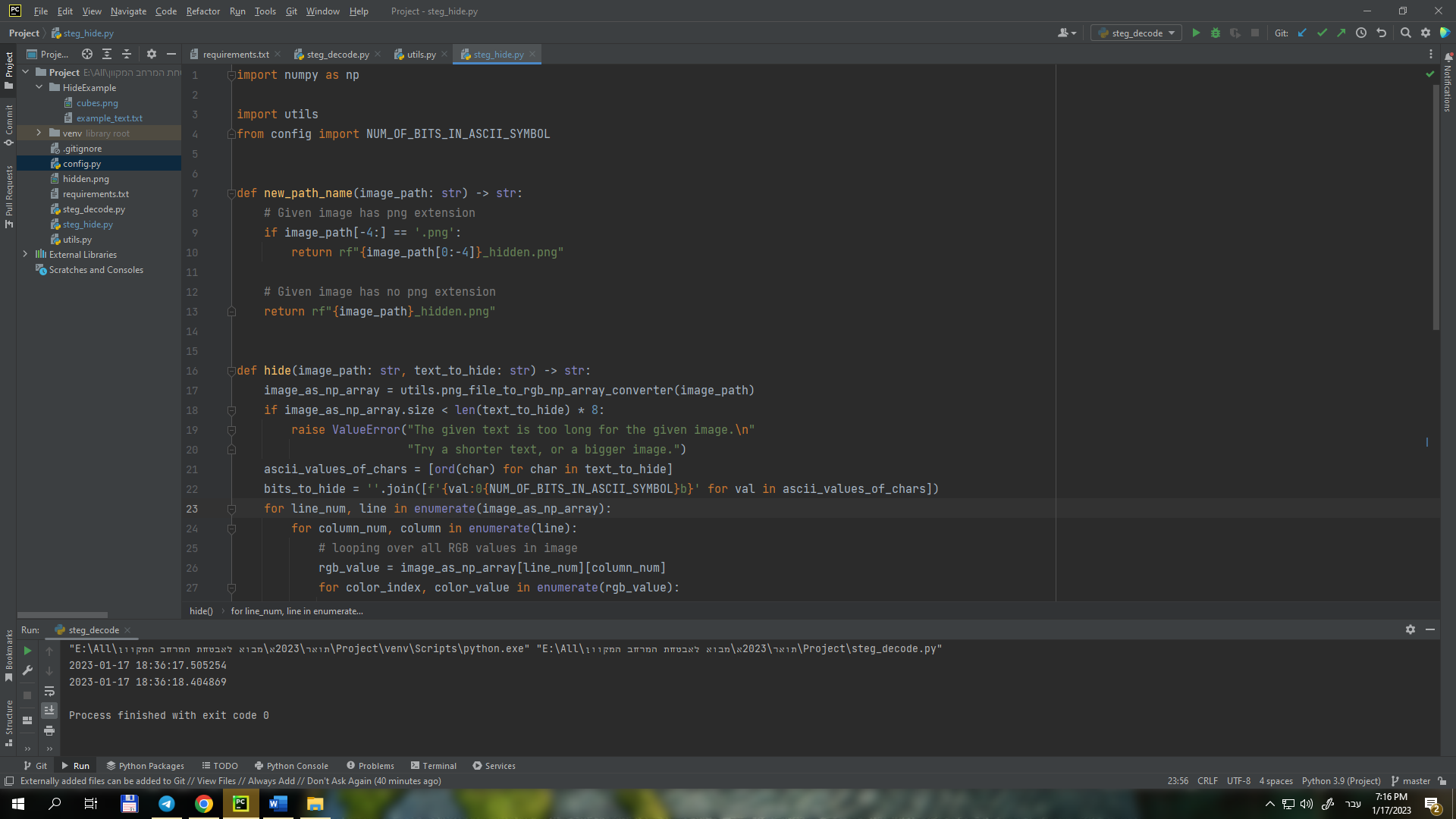
כאשר בכל פעם נשים את הביט שהגענו אליו בתור ה-LSB של הבית הבא שבתמונה.

כך נעשה עד שיסתיימו כל התווים בטקסט שאנו רוצים להסתיר.

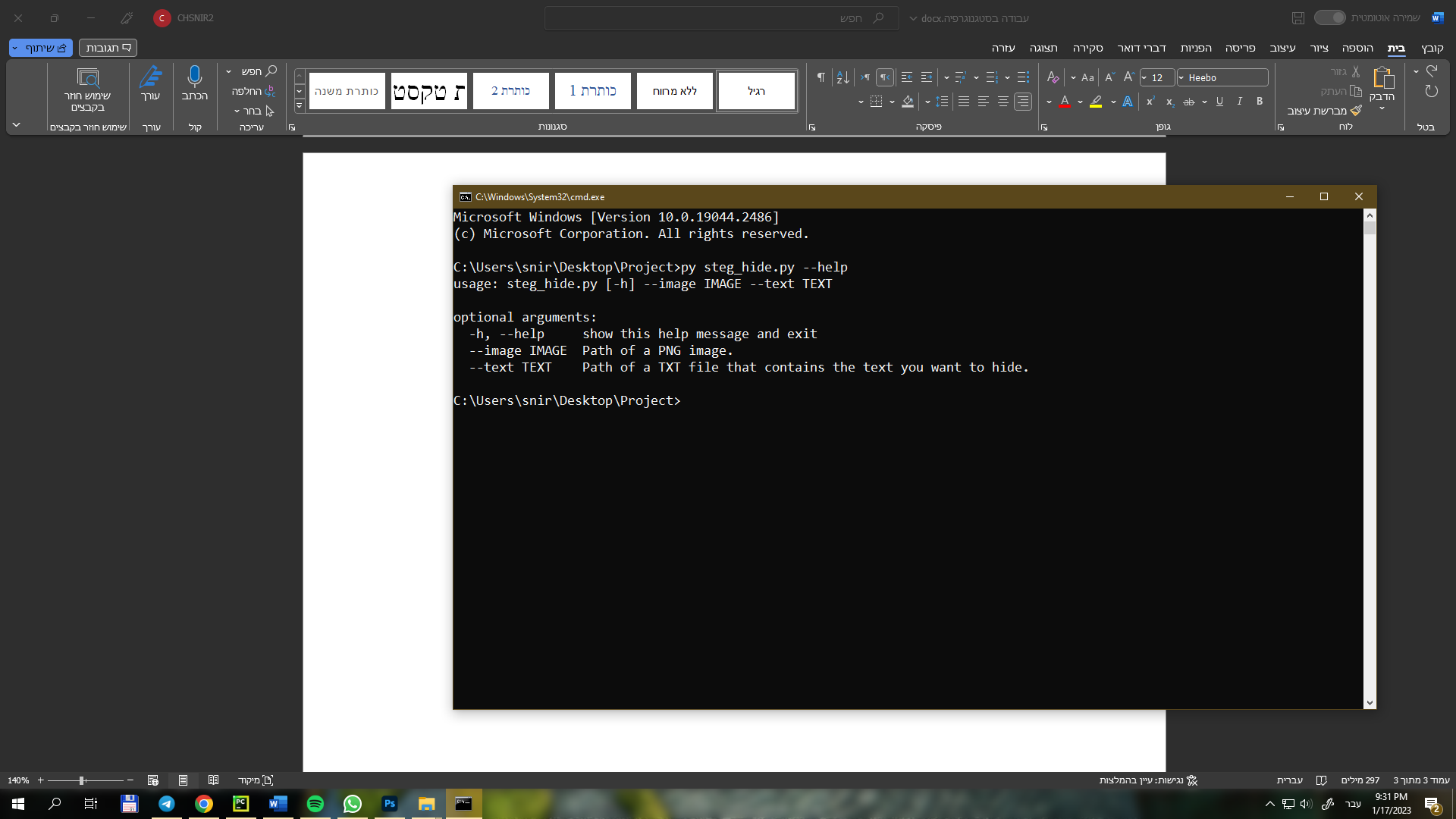
המימוש שלי בא לידי ביטוי בפונקציה שתחילה בודקת את תקינות הקלט, ולאחר מכן ממשיכה עם לולאה מקוננת שמחליפה את הביטים המתאימים. הפונקציה נראית כך:



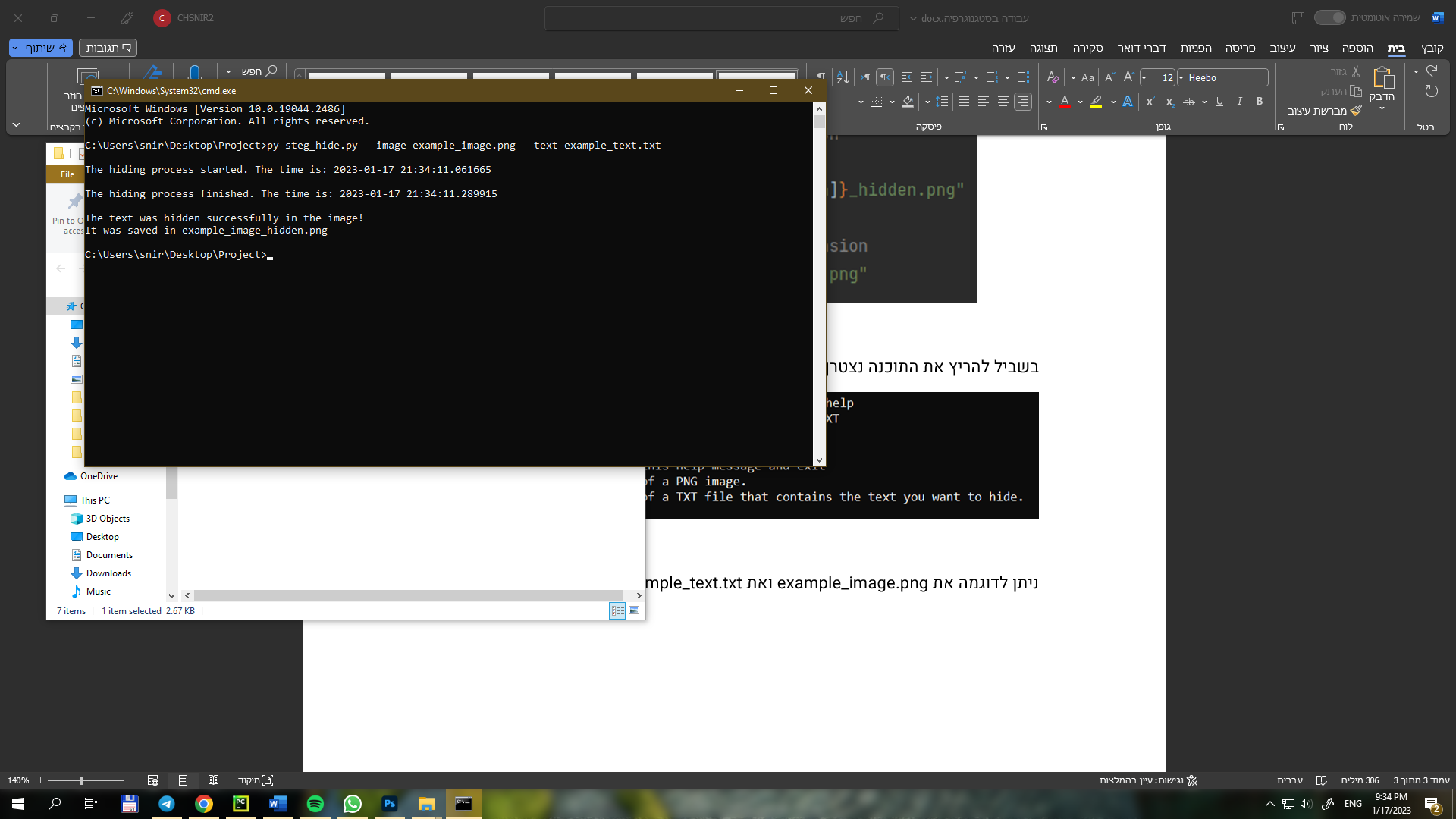
ולבסוף, לאחר שסיימנו להסתיר את כל הביטים, והתמונה כעת עם הטקסט מוסתר בתוכה, היא נשמרת בנתיב זהה, ועם אותו שם בתוספת \_hidden ועם סיומת PNG:

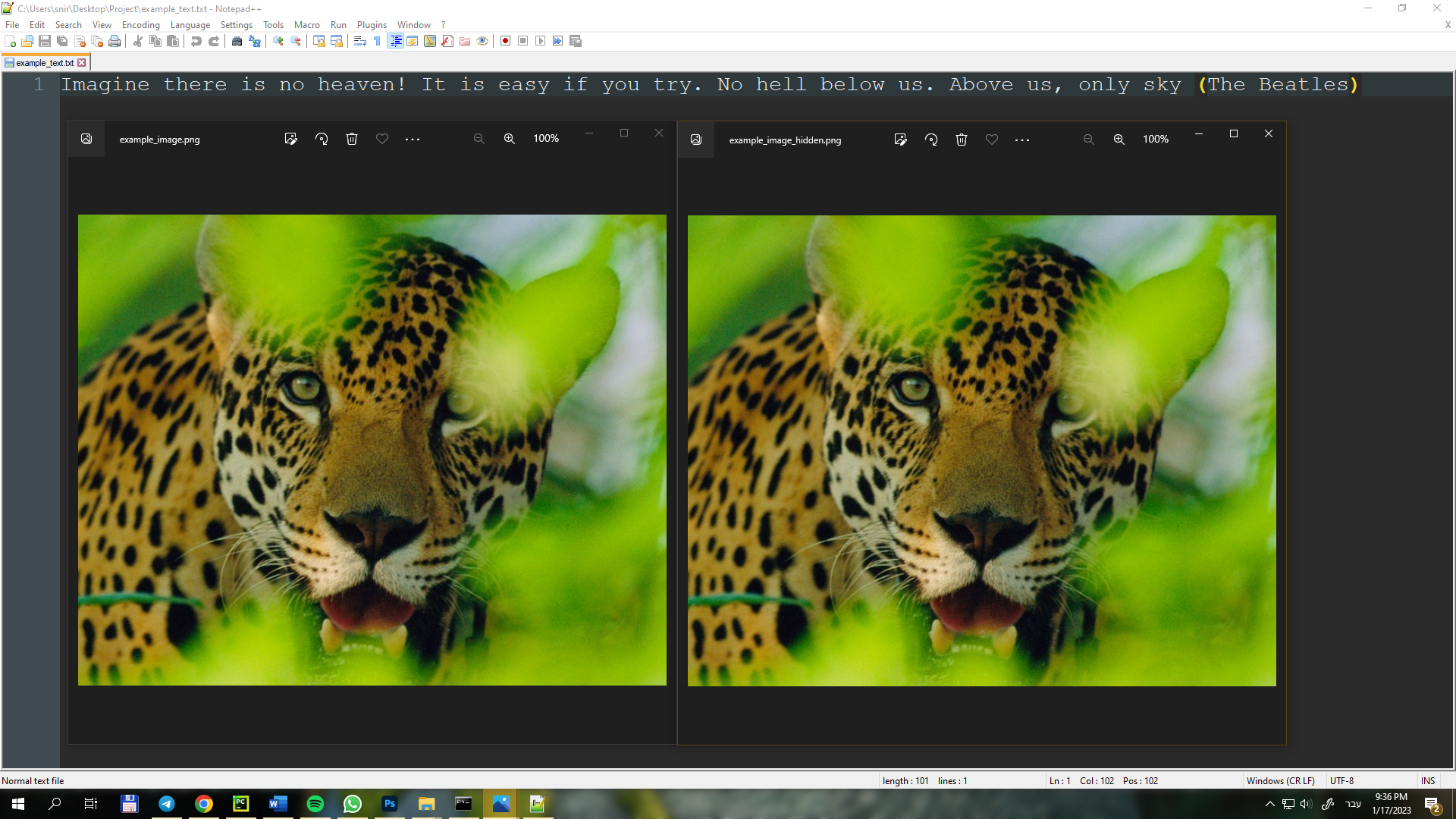


בשביל להריץ את התוכנה נצטרך לספק לה פרמטרים לקובץ תמונה וקובץ טקסט:



ניתן לדוגמה את example\_image.png ואת example\_text.txt ונריץ את התוכנה:



הריצה הצליחה. להלן שתי התמונות, וקובץ הטקסט:

משמאל התמונה הרגילה, ומימין התמונה עם הטקסט המוסתר –

"Imagine there is no heaven! It is easy if you try. No hell below us. Above us, only sky (The Beatles)"

השינוי אינו נראה לעין. נראה שהוא באמת התבצע בחלק ב כשנפענח את התמונה שמימין.

**חלק ב**

בחלק זה התבקשנו לממש פענוח של טקסט מתוך תמונה עם טקסט חבוי – רק שהפעם יש קאטצ':

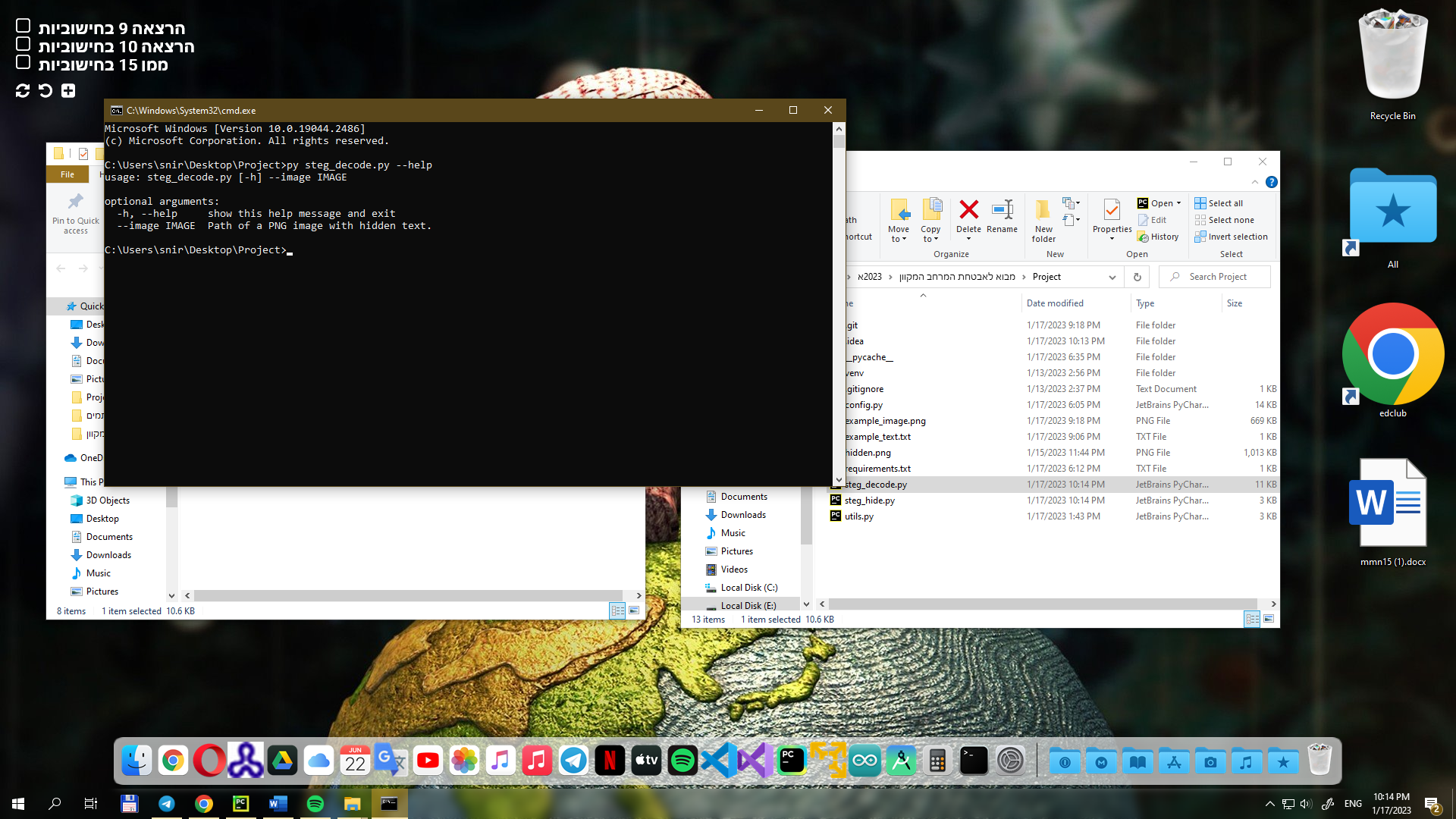
הטקסט לאו דווקא חבוי בביטים הכי פחות משמעותיים שבכל בית, אלא יכול להיות חבוי בכל אחד משלושת הביטים הכי פחות משמעותיים (מעתה ואילך אתייחס אליהם בתור ערוצי LSB\_0 LSB\_1 LSB\_2) – בצורה כזו שכל מילה במשפט יכולה להיות ב"ערוץ LSB" אחר. עם זאת, כל מילה מוסתרת מתחילתה ועד סופה באותו ערוץ LSB.

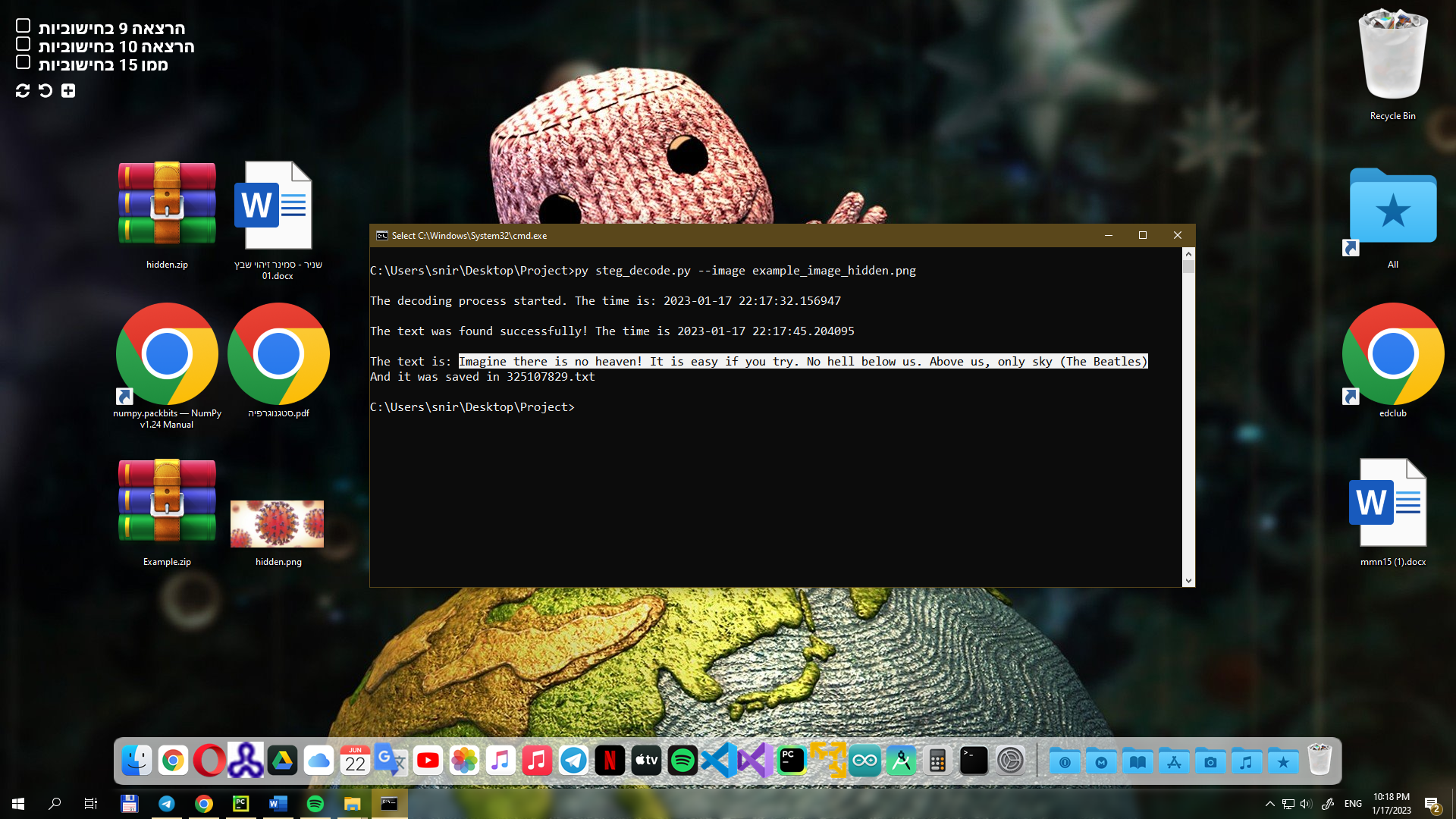
(אך מילים שונות במשפט יכולות להיות באחד משלושת ערוצי ה-LSB השונים)

ואם זה לא מספיק מסובך, אז לאו דווקא מתחילים להסתיר את המשפט בתחילת התמונה, אלא יכולים להתחיל את ההסתרה מכל בית בתמונה.

אז איך מימשתי את זה?

מיד אסביר, אך לפני כן, נראה שהמימוש אכן עובד, תוך פענוח של הטקסט שהסתרנו קודם לכן, בחלק א:



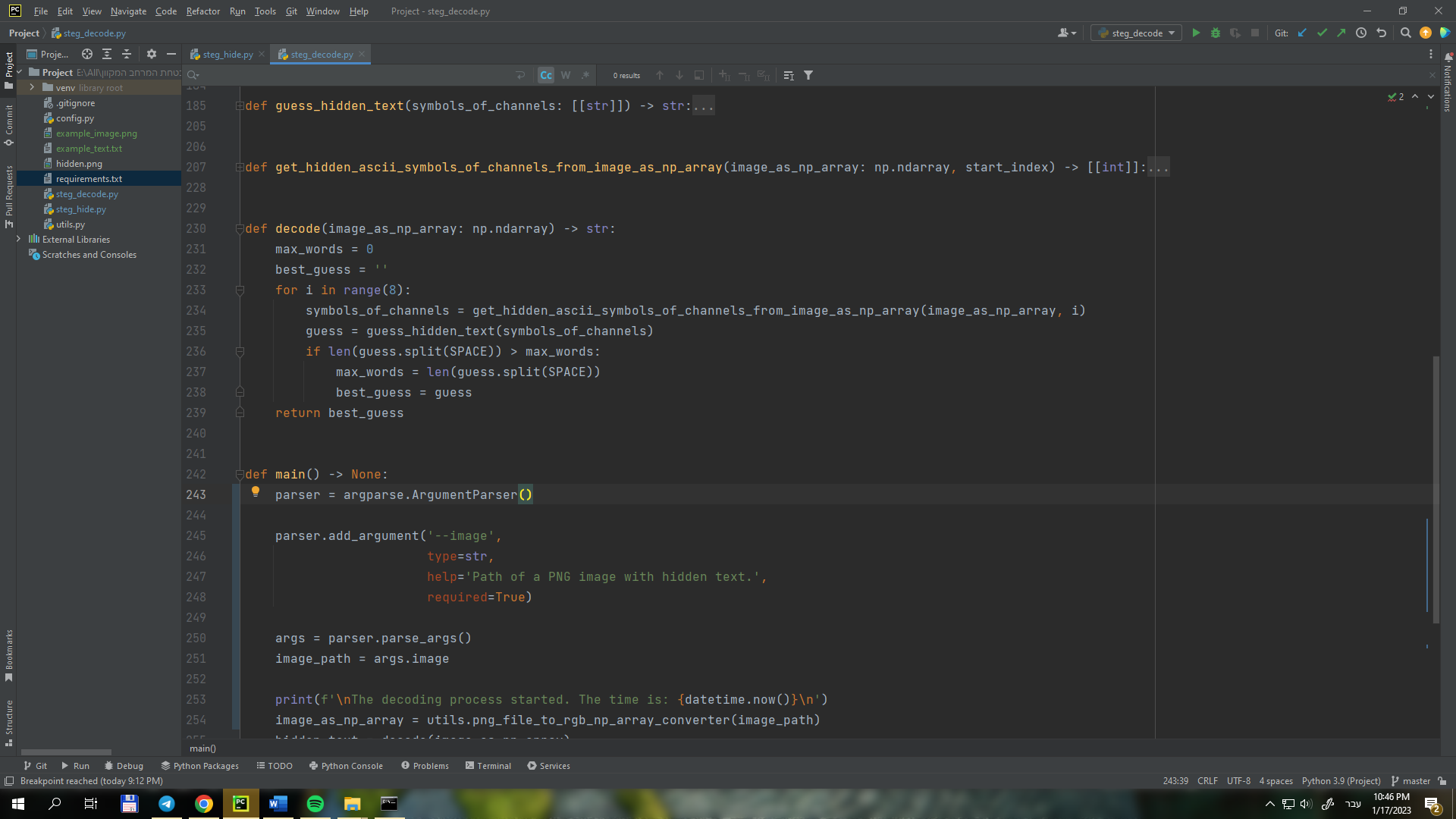
נספק את הנתיב המתאים ונריץ את התוכנה המפענחת:

ואכן קיבלנו את הטקסט המתאים!

עכשיו כשהשתכנענו שההסתרה והפענוח עובדים, נעבור לחלק הכיף – המימוש.

המימוש היה די ארוך, אעבור כאן על הקטעים המרכזיים והחשובים.

הכל מתחיל בפונקציה הראשית decode:

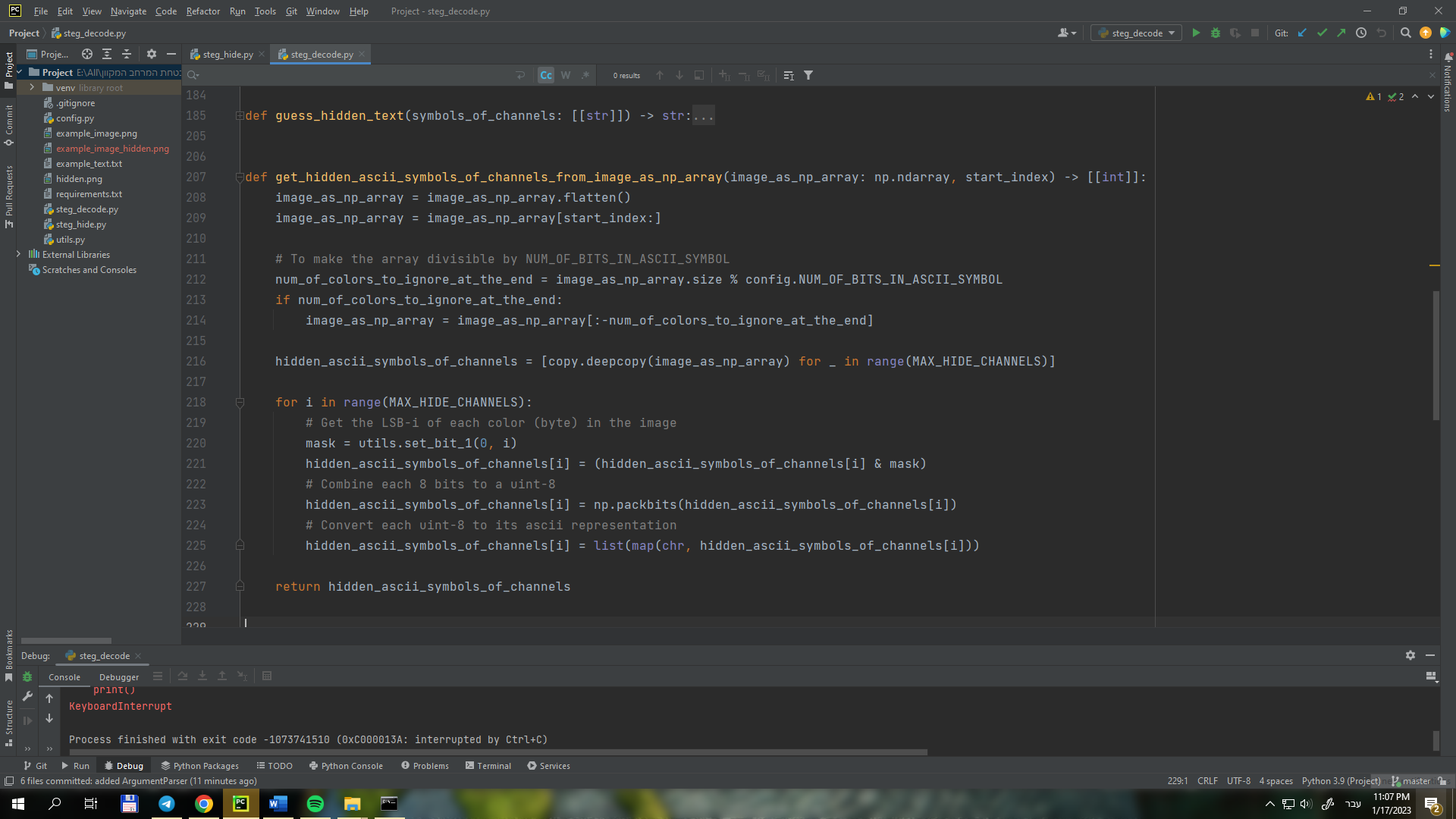


הפונקציה הזו מייצרת בכל פעם רשימה של סימבולים (תווי ascii) משלושת ערוצי ה-LSB.

לאחר מכן, היא שולחת את הסימבולים משלושת הערוצים שמצאנו לפונקציה guess\_hidden\_text והיא בתורה מספקת את הניחוש הטוב ביותר שלה לטקסט חבוי.

בסופו של דבר, המפשט הארוך ביותר שנמצא, הוא זה שנחזיר בתור הניחוש הטוב ביותר שלנו, ואותו נדפיס ונשמור לקובץ.

יש כאן לולאה של 8 איטרציות כדי לבדוק את כל המיקומים מהם התחילו להצפין את המשפט. למעשה בכל פעם אנו מספקים לפונקציה get\_hidden\_ascii\_symbols\_of\_channels\_from\_image\_as\_np\_array

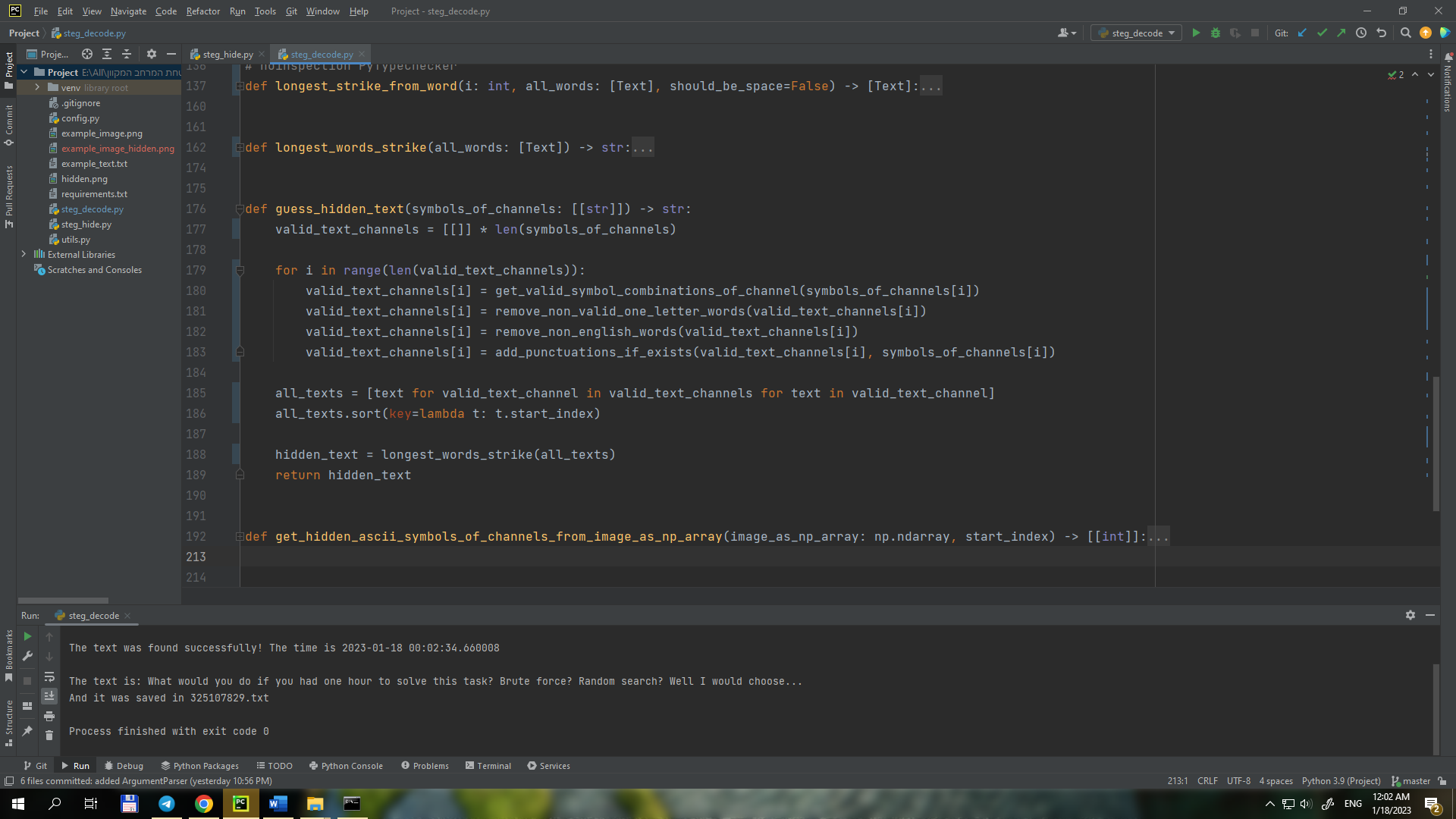
אופסט מסוים, בין 0 ל-7 וממנו אנו מתחילים לבצע את תהליך חילוץ הסימבולים:

תהליך חילוץ הסימבולים משתמש בספרייה בשם numpy שעוזרת לנו בפעולות שמצריכות מטריצות, כמו כאן.

תכלס מה שקורה כאן זה שאני מייצר mask-ים שונים לכל ערוץ LSB, ומחלץ בעזרת פעולות bitwise את הביטים המתאימים. לאחר מכן כל 8 ביטים אני הופך לבית בייצוג מספרי, ולבסוף כל ייצוג מספרי אני הופך לתו ה-ascii המתאים.

הידד!

עכשיו כשבידינו כל הסימבולים שבפוטנציאל חלקם הם אלו שהוסתרו, אנו ממשיכים לשלב הבא – הלוא הוא ניחוש הטקסט החבוי:



אנחנו מתחילים עם שלושה ערוצי LSB שמלאים בסימבולים.

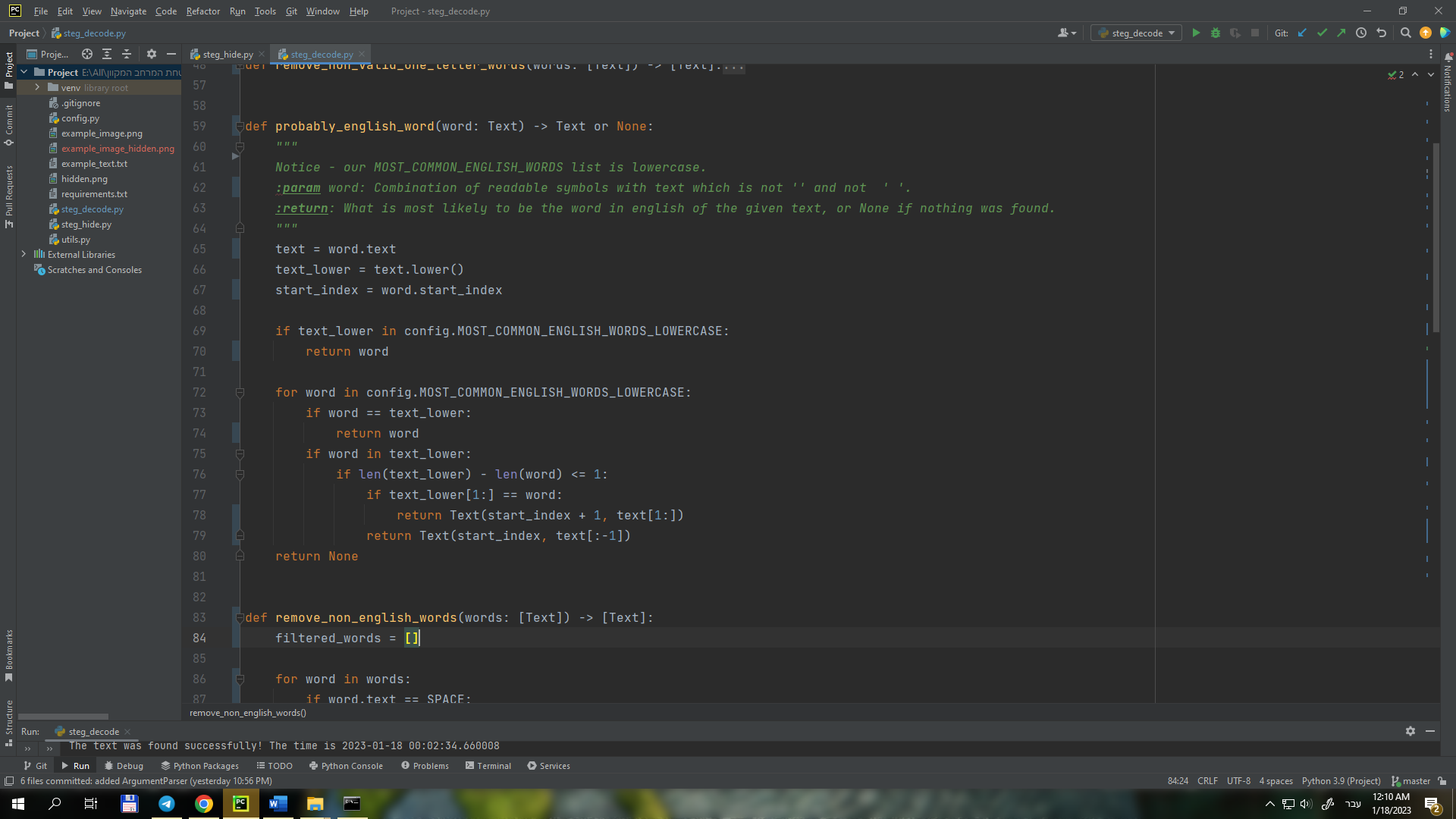
עובדים בשיטת המסננת – בעבור כל ערוץ, אנחנו מסננים את כל מה שלא קשור, עד כמה שאפשר. השלבים הם:

1. סינון התווים שהם לא ascii – בסיום שלב זה אנחנו נשארים עם מערך של רצפי תווים קריאים (אותיות abc, רווחים וסימני פיסוק). בעבור כל רצף כזה אני שומר גם start\_index המסמל את האינדקס של התו בו התחיל הרצף מבין התווים בערוץ ה-LSB המתאים.
2. סינון המילים שהן רק אות אחת, והן לא רווח, i או a (אלו הם התווים היחידים שעומדים בפני עצמם)
3. סינון מילים שהן לא מילים באנגלית – ארחיב בהמשך.
4. הוספת סימני פיסוק למילים, אם היו סימני פיסוק מוסתרים לצד המילה.

לבסוף, כשיש רצפי מילים הגיוניות עם פיסוק ורווחים בשלושה ערוצים שונים, אנו מאחדים את כל הערוצים לרשימה אחת, וממיינים אותה לפי הstart\_index- של כל מילה.

כשיש לנו את הרשימה המאוחדת הזו, אנו רצים עליה ומחפשים את הרצף הכי ארוך (משפט). התהליך הזה נעשה בפונקציה longest\_words\_strike עליה ארחיב בהמשך.

תמונה שמכילה טקסט, צג, אלקטרוניקה, צילום מסך

התיאור נוצר באופן אוטומטיאבל קודם, אסביר איך סיננתי את המילים שלא נמצאות בשפה האנגלית:

לפונקציות האלו אנו מגיעים מצוידים מראש, עם רשימה ב-lowercase של המילים הנפוצות ביותר באנגלית. (מעתה ואילך אתייחס לרשימה הזו כאל המילון שלנו)

תחילה אנחנו בודקים אם יש התאמה מושלמת של המילה שלנו למילה במילון. אם כן – מצוין, אנחנו נשמור את המילה.

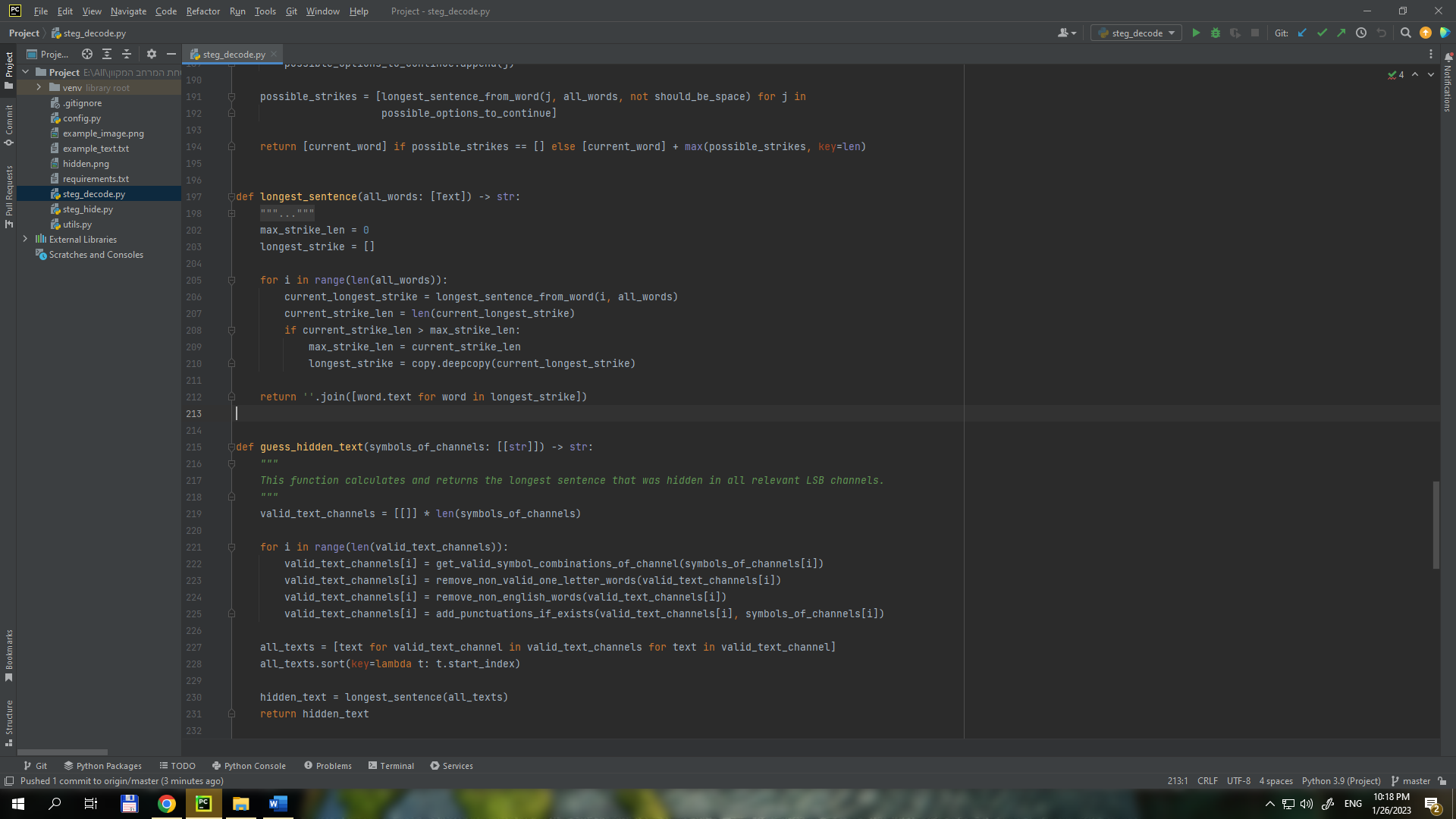
אם אין התאמה, יכול להיות שעדיין מדובר במילה שנרצה להשאיר. לדוגמה hellox או xhello זו לא מילה באנגלית, אך יכול להיות שבמקרה הופיעה האות x או כל אות אחרת לפני או אחרי מילה אמיתית באנגלית.

אני בודק זאת, ואם אכן מדובר במילה כזו – מעולה, אני משאיר אותה, ומעדכן את הטקסט שלה ואת ה-start\_index במידת הצורך.

בדיקה אחרונה לפני שאני מחזיר את רשימת המילים באנגלית שמצאתי, היא בדיקה של אותיות קטנות/גדולות. מילים באנגלית יכולות להתחיל באות גדולה, אך לא יכול להיות שבאמצע מילה תהיה אות גדולה – ולכן אני מסיר את האופציות האלו.

כל המילים באנגלית בידינו!

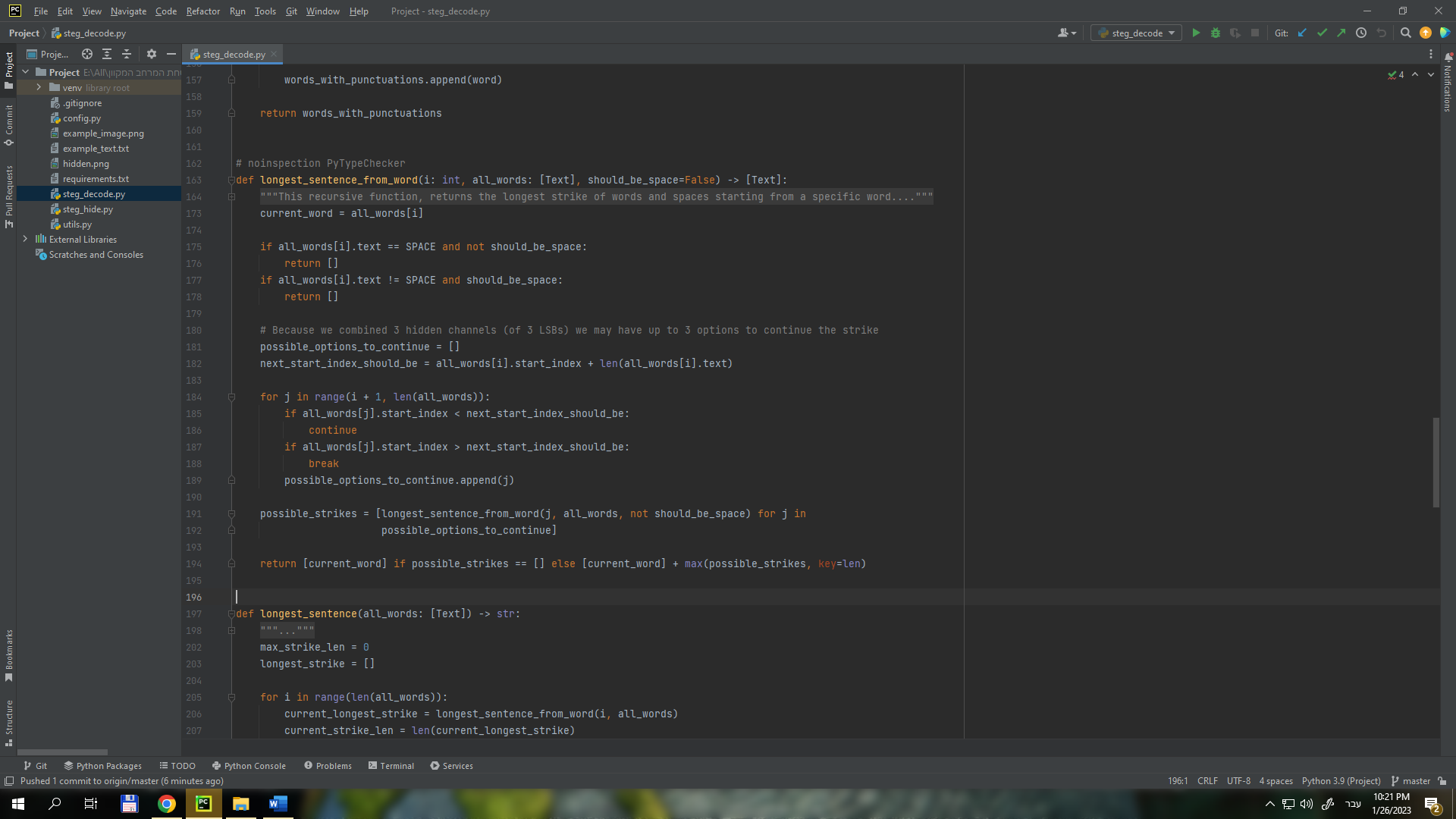
עשינו זאת בעבור כל אחד משלושת ערוצי ה-LSB, ועכשיו אחרי שאיחדנו את שלושת הערוצים לרשימה אחת ארוכה, הגיע הזמן לבדוק מה המשפט הארוך ביותר, ולהחזיר אותו.



אנו עוברים על כל המילים.

כל מילה אנחנו שולחים לפונקציה שמתחילה לבדוק רצף הגיוני ארוך ביותר החל מהמילה הנוכחית.

לבסוף, אנחנו מכריזים על המשפט הארוך ביותר כעל הרצף הארוך ביותר שהגענו אליו מכל בדיקות הרצפים האפשריים, ומחזירים אותו.

כדי לבדוק מהו הרצף ההגיוני הארוך ביותר החל ממילה מסוימת, אנו משתמשים בפונקציה רקורסיבית לא מאוד מסובכת:

הפונקציה הזו מסתכלת על ה-start\_index של המילים, ולפי גודל המילה, מחשבת מה האינדקס בו אמורה להיות המילה הבאה בתור.

כמו כן, מתבצעת בדיקה שלאחר כל מילה באנגלית יש רווח, ולאחר כל רווח יש מילה באנגלית (כדי שנקבל משפט תקין)

סיימנו לעבור על כל הלוגיקה של התוכנה!

הגענו לרגע המיוחל - להריץ את תוכנת הפענוח שלנו על התמונה הנתונה hidden.png:

hidden.png


אז לסיכום – מה הייתי עושה אילו הייתה לי רק שעה לפתור את המשימה הזו?

ככל הנראה הייתי כותב קוד שמפענח את כל ערוצי ה-LSB האפשריים, מדביק את הכל בnotepad++ ומחפש עם Ctrl+F מילים רנדומליות עד שהיה נגמר לי הזמן 😊