פרויקט סטגנוגרפיה

הדר תפארת, 205492507

הערה כלליות להרצת קבצי הקוד

שני קבצי הקוד עושים שימוש בספריות סטנדרטיות של פיתון (כדוגמת sys, math, secrets בעבור תכונות אופציונליות מסוימות) אך בנוסף נעשה שימוש בשתי ספריות חיצוניות:

- PIL -
בסביבת העבודה שלי (virtual environment של התוכנה pycharm – השתמשתי בwrapper של הספרייה PIL בשם Pillow. קישור לספרייה: https://pillow.readthedocs.io/en/stable/.
השימוש בספרייה נעשה בכדי לקרוא את קבציי התמונה, לקבל את מטריצת ערכי הפיקסלים של תמונה נתונה וכדי לשמור תמונה חדשה מתוך מטריצת ערכי פיקסלים נתונה.
- Numpy -
קישור לספרייה: https://numpy.org/.
השימוש בספרייה נעשה בכדי להתנהל עם מערך ערכי הפיקסלים של התמונות ובכדי להקל על ביצוע פעולות על מערך נתון (שינוי מבנה המערך או פעולות חישוביות על ערכי המערך).

בנוסף, שני קבצי הקוד יכולים לרוץ ללא ארגומנטים - עם ערכים נתונים מראש או עם ארגומנטים של זמן ריצה, כמפורט לכל קובץ בסעיף המתאים במסמך זה.

קבצי הדוגמאות המפורטים במסמך זה מצורפים לקבצי המטלה, ומופיעים בתיקייה examples.

חלק א' – הסתרת טקסט בתמונה

תקציר

בהינתן קובץ תמונה וטקסט, התכנית מסתירה את הטקסט הנתון בסיבית הנמוכה (LSB) של ערכים עוקבים במערך ערכי הRGB של הפיקסלים בתמונה ושומרת את התוצאה בקובץ חדש עם שם התמונה עם תוספת "_hidden".

הוראות הרצה

לתכנית יש 3 אופני הרצה, כתלות במספר וסוג הארגומנטים הנשלחים כcommand line argument בעת הקריאה לתכנית.

1. ללא ארגומנטים.

```
> python steg_hide.py
```

בהרצה כזו, לא מצויינים קובץ התמונה בתוכה נרצה להחביא את התוכן וגם לא התוכן אותו ברצוננו להחביא בתמונה. לכן, התכנית תנסה לרוץ עם ערכים דיפולטיביים של photo = "hidden.png" ו message = "this is a default message".

2. עם ארגומנט אחד. למשל:

```
> python steg_hide.py imagefile.png
```

בהרצה מסוג זה, התכנית תרוץ על התמונה הנתונה כארגומנט היחיד ועם הערך הדיפולטיבי של ההודעה.

3. עם שני ארגומנטים או יותר. למשל:

```
> python steg_hide.py imagefile.png please hide this message
```

בהרצה כזו, התכנית תרוץ על התמונה הנתונה כארגומנט הראשון וההודעה הנתונה כצירוף הארגומנט השני ואילך.

ההודעה שיש להחביא יכולה להיות נתונה כמילה יחידה כמו בדוגמה לעיל, או כקובץ טקסט כגון:

```
> python steg_hide.py imagefile.png message.txt
```

אם ההודעה (הארגומנט השני) מצויין כקובץ טקסט – התכנית תחביא בתמונה את תוכן הקובץ הנתון.

הערה:

ניתן להוסיף את הסימון האופציונלי $r כארגומנט ההרצה האחרון בכדי לבקש מהתכנית להתחיל את החבאת ההודאה במקום אפשרי רדנומלי ולאו דווקא בתחילת מערך הפיקסלים, למשל:

```
> python steg_hide.py imagefile.png please hide this message $r
> python steg_hide.py imagefile.png message.txt $r
```

פירוט מהלך התכנית

תחילה, התכנית מפעילה את המתודה extract_message, אשר בוחנת את ההודעה הנתונה ומחזירה את ההודעה עצמה או את תוכן הקובץ אותו ההודעה מציינת, כתלות בארגומנט שצורף בזמן ריצה.

לאחר מכן, משתמשים במתודה image_to_array, אשר מחלצת מתוך קובץ התמונה הנתון, בעזרת הספרייה PIL, את מערך ערכי הRGB של הפיקסלים בתמונה ומחזירה מערך זה כnumpy.array.

כעת, מבוצע שימוש במתודה hide, על המערך שחולץ.

המתודה hide מקבלת כפרמטרים את מערך ערכי הRGB של הפיקסלים של התמונה, ההודעה אותה יש להסתיר ופרמטר בוליאני המציין האם להסתיר את ההודעה החל מאינדקס חוקי רנדומלי במערך (True) או בתחילת המערך (False).

ראשית, אנו שומרים את מידות המערך כדי שנוכל לשחזר אותו לצורתו המקורית לאחר ביצוע השינויים.

כעת, משטחים את המערך, ממירים את ההודעה למערך של ביטים כאשר כל תו מופיע כרצף ביטים בייצוג big endian (בעזרת פונקציות של המחלקה numpy).

מבוצע חישוב של גודל מערך ערכי הפיקסלים לעומת גודל מערך ההודעה בייצוג בינארי, בכדי לבחון האם מערך הפיקסלים גדול דיו להכיל את ההודעה, ובמידה ולא ניתן – התכנית מסתיימת עם הודעת שגיאה.

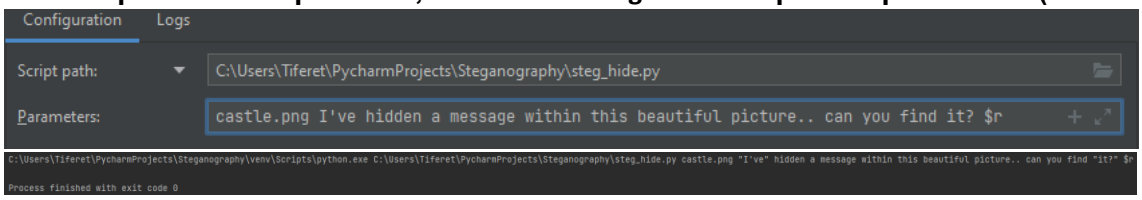
אחרת, בוחרים את אינדקס ההתחלה כ0, או כערך רנדומלי אשר מכיל מספיק מיקומים במערך ערכי הפיקסלים בכדי להכיל את ההודעה במידה וצוין בזמן הרצת התכנית להתחיל את ההסתרה ממקום רנדומלי.

ואז כל ערך במערך הפיקסלים החל מהאינדקס שנבחר ועד לסיום מערך הייצוג הבינארי של ההודעה משתנה בהתאם למתודה modify_bit, אשר משנה את ערכו של הביט הנמוך ביותר בערך להיות זהה לערך הביט התואם במערך ההודעה.

המתודה hide מחזירה את מערך ערכי הפיקסלים בצורתו המקורית לאחר ביצוע החבאת ההודעה ולבסוף, באמצעות המתודה array_to_image אנו מייצרים תמונה חדשה עם מערך זה בשם התמונה המקורי בתוספת _hidden.

דוגמאות הרצה

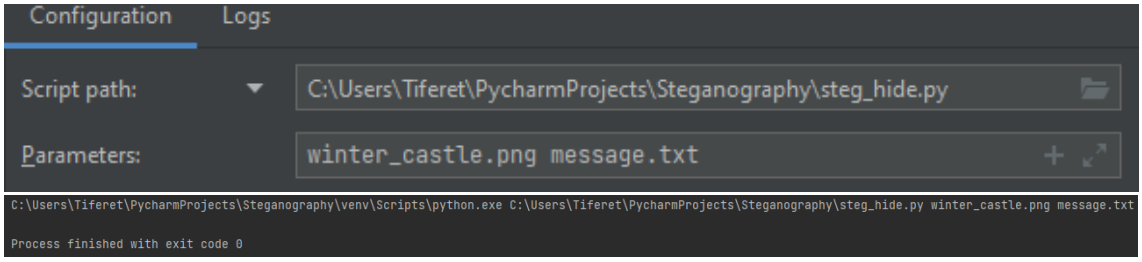
1) הסתרת טקסט המוקלד כcommand line argument, החל ממקום רנדומלי חוקי:



castle.png:

castle_hidden.png:

2) הסתרת טקסט מתוך קובץ txt. נתון, החל מתחילת מערך הפיקסלים:



winter_castle.png:

winter_castle_hidden.png:

חלק ב' – פענוח טקסט שהוסתר בתמונה

תקציר

בהינתן תמונה, התכנית מנסה לפענח את ההודעה שהוסתרה בתמונה כרצף של מילים עם רווחים ביניהן כך שכל מילה יכולה להופיע באחד משלושת הסיביות התחתונות בייצוג הבינארי של מספרים ממערך ערכי הRGB של הפיקסלים בתמונה הנתונה.

הפענוח משתמש במילון מילים נפוצות באנגלית אשר מסופק לתכנית כקובץ טקסט חיצוני.

הוראות הרצה

לתכנית יש 4 אופני הרצה, כתלות במספר הארגומנטים הנשלחים כcommand line arguments בעת הקריאה לתכנית.

1. <u>ללא ארגומנטים</u>.

```
> python steg_decode.py
```

בהרצה כזו, לא צויינו קובץ התמונה, קובץ הטקסט אליו יש לשמור את התוצאה וקובץ המילון. לכן התכנית תנסה לרוץ עם ערכים דיפולטיביים – תמונה לפענוח: hidden.png, קובץ שמירת תוצאה: קובץ טקסט בשם מספר תעודת הזהות שלי (205492507.txt, כנדרש בהגדרות המטלה), מילון מילים נפוצות באנגלית: word_dictionary.txt, ואופציית הדפסת התוצאות הכי סבירות כבויה.

2. <u>עם ארגומנט אחד</u>, למשל:

```
> python steg_decode.py imagefile.png
```

בהרצה כזו התכנית תנסה לפענח את הטקסט החבוי בתוך התמונה המצויינת כארגומנט היחיד, עם קובץ שמירת התוצאה, מילון המילים הנפוצות באנגלית עם הערכים הדיפולטוביים ואופציית הדפסת התוצאות הכי סבירות כבויה.

3. <u>עם שני ארגומנטים</u>, למשל:

```
> python steg_decode.py imagefile.png output.txt
```

בהרצה כזו מסופקים לתכנית קובץ התמונה כארגומנט הראשון וקובץ הטקסט אליו יש להזין את הטקסט שפוענח מתוך התמונה כארגומנט השני. קובץ מילון המילים הנפוצות ילקח כערך הדיפולטיבי.

4. <u>עם שלושה ארגומנטים</u>, למשל:

```
> python steg_decode.py imagefile.png output.txt dictionary.txt
```

בהרצה כזו מסופקים כל שלושת הקבצים אשר התכנית צורכת. הארגומנט הראשון הוא התמונה עליה נבקש לעבוד, הארגומנט השני הוא קובץ הטקסט אליו יש לכתוב את תוצאת הפענוח והארגומנט השלישי מגדיר את מילון המילים הנפוצות לערך אחר מהערך הדיפולטיבי (קובץ המילון הדיפולטיבי מצורף לקבצי הקוד ושמו word_dictionary.txt)

הערה:

ניתן להוסיף את הסימון האופציונלי $p כארגומנט ההרצה האחרון בכדי לבקש מהתכנית להדפיס למסך בנוסף רשימה של תוצאות הפענוח הסבירות ביותר, בצירוף ניקוד שהן קיבלו במהלך השוואת התוצאות, למשל:

```
> python steg_decode.py imagefile.png output.txt dictionary.txt $p
> python steg_decode.py imagefile.png output.txt $p
> python steg_decode.py imagefile.png $p
```

מילון מילים נפוצות באנגלית

בניתי את התכנית כך שתעזר במילון מילים נפוצות חיצוני לתכנית. המילון צריך להכיל שורות של מילים לצד ציון בעבור כל מילה. הקובץ איתו הרצתי את התכנית נלקח מהלינק: https://norvig.com/ngrams/, המילון שבחרתי הוא הקובץ המופיע בלינק בשם count_1w.txt. בפרוייקט הקובץ הינו בשם word_dictionary.txt, קובץ זה מכיל 333333 מילים, לצד ציון המייצג כמה המילה נפוצה.

בנוסף מצורף קובץ מילון אלטרנטיבי המכיל 100 אלף מילים, אשר נלקח גם הוא מהלינק הנ"ל, הקובץ המופיע בשם count_1w100k.txt. בפרויקט הקובץ הינו בשם alternative_word_dictionary.txt.

ניתן להחליף את קובץ המילון בקובץ אחר, בהתאם להוראות ההרצה מהסעיף הקודם, אך יש לשים לב לכך שמידת ההצלחה של התכנית בפענוח נכון של טקסטים מוסתרים תלויה בנכונות המילון, שלמותו וקרבתו למציאות.

פירוט מהלך התכנית

תחילה, התכנית מפעילה את המתודה build_word_dictionary, אשר קוראת את הקובץ המצויין כקובץ מילון המילים הנפוצות ומרכיבה ממנו מילון של מילים וציונים למילים. בניתי את התכנית כך שתצפה לקבל שורות המכילות מילה וערך (המציין את פופולריות המילה).

המתודה עוברת על כל המילים והערכים התואמים להן וכך מזהה את הערך הגבוה ביותר והערך הנמוך ביותר ברשימה. עם מידע זה, מתבצעת קריאה למתודה normalize_score אשר מנרמלת כל ערך לסקאלה מוגדרת מראש. הציון הסופי שכל מילה תקבל במילון יחושב על ידי המתודה calculate_word_score והוא מהווה שילוב של ציון המילה כפי שמצויין בקובץ המילון הנתון ביחד עם אורך המילה, כך שמילים ארוכות יותר יקבלו עדיפות (שכן פחות סביר שמילה ארוכה תיווצר באופן רנדומלי במערך, מאשר שהיא הוסתרה שם במתכוון).

לאחר בניית המילון בזיכרון, מבוצעת קריאה לimage_to_array, אשר קוראת את קובץ התמונה הנתון ומחלצת ממנו את מערך ערכי הRGB בעבור הפיקסלים של התמונה, בדומה לפונקציה בעלת אותו השם בחלק א'.

ואז מתבצעת קריאה עם מערך ערכי הפיקסלים של התמונה, מילון המילים וסטטוס הדפסת התוצאות למתודה decode.

המתודה decode מנסה לחלץ מתוך מערך ערכי הפיקסלים את הטקסט הכי סביר שהוסתר בתמונה. היא עושה זאת על ידי שימוש בפונקציות עזר. המתודה משתמשת בתכנות מקבילי בכדי לחסוך בזמן בעבור חישוב אשר חוזר על עצמו על מערכים שונים זה מזה. הצורך נובע מכך שאין אנו יודעים החל מאיזה ערך הוחבא הטקסט בתמונה. למשל, בתמונה הבאה:

[222, 155, 112], [222, 155, 112], [223, 156, 113],

ייתכן שכל אחד מהערכים הנתונים מהווה כמיקום ממנו התחיל הפירוט של תו כלשהו בייצוג בינארי באחת משלושת הסיביות הנמוכות של הערך. מכאן, שישנם 8 אופנים (כמספר הסיביות לייצוג תו) לפרש את רצף הסיביות הנוצרות מבידוד כל אחד משלושת הסיביות הנמוכות של ערכי המערך המקורי, לקבלת כיסוי מלא של כל אפשרויות החבאת התווים. לכן, באמצעות המתודה check_subarray נבחן בכל פעם תת מערך אשר מתחיל בערך אחד אחרי תת המערך הקודם לו (8 תת מערכים בסך הכל) ונתייחס בכל פעם לתת מערך זה כאל המערך האופציונלי לכך שבו הוסתר הטקסט. לבסוף נקבל את תוצאות כל המתודות הללו ובעזרת הציון שכל טקסט אופציונלי שזוהה יקבל – נבחר את הטקסט הסביר ביותר אותו נחזיר כתוצאה ובמידה והמשתמש בחר (על ידי קריאה לתכנית עם הארגומנט $p כארגומנט זמן ריצה) להדפיס מספר חלופות – נדפיס למסך כמה מהאפשרויות לטקסט המוסתר אשר קיבלו את ציון הסבירות הגבוה ביותר.

לבסוף, המתודה write_result תכתוב לקובץ התוצאה שצויין את הטקסט הנבחר כטקסט הסביר ביותר.

הסבר לאופן חיפוש הטקסט המוסתר

המתודה check_subarray מקבלת תת מערך של ערכי הפיקסלים של התמונה, את מילון המילים הנפוצות ומספר הביטים החל מהביט הנמוך ביותר בבית בהם יכולות להיות מוסתרות מילים מהטקסט.

לכל אחד מהביטים בהם עלולות להיות מוסתרות מילים (מצויין במטלה כ3) המתודה משתמשת בפונקציה decode_bit אשר עוברת על מערך הערכים הבינאריים אשר מתקבל מקריאת ערכי הביט שצויין בלבד מכל ערך בתת מערך ערכי הפיקסלים הנתון, מפרשת מערך זה למערך ערכים על ידי פירוש הייצוג הבינארי במערך זה כbig endian, בעזרת פונקציה של ספריית numpy, ולאחר מכן עוברת על מערך ערכי הפירושים (כלומר תווים אופציונליים במידה והביט המצויין הינו הביט שנבחר לייצוג מילה מסויימת) ובונה מערך של מילים אשר מוכרות על ידי מילון המילים הנפוצות, ביחד עם ציון המילה במילון – מסודרות לפי האינדקס ממנו מתחילה המילה בתת המערך. בנוסף, ממופים גם מערך של תוצאות, אשר מציין היכן קיימת מילה אופציונלית על גבי תת המערך המקורי וגם מילון של רווחים, אשר מציין איזה סימן רווח נמצא באיזה אינדקס על התת המערך המקורי.

כל תוצאות אלו נשלחות חזרה למתודה check_subarray ושם מורכבים יחדיו רשימה של מילוני המילים שנמצאו בעבור כל ביט חיפוש, מסודרים לפי אינדקס, מילון של סימני רווח, לפי האינדקס בו הופיעו – על כל אחד מהביטים עליהם חיפשנו ועל ידי ביצוע פעולת or על ערכי מערך התוצאות – אנו מגלים איזורים בעלי סבירות גבוהה להימצאות טקסט, שכן ניתן להצליב את מערכי התוצאות של כל אחד מביטי החיפוש ולבנות רשימה של אינדקסים המכילים מילים אפשריות על אחד או יותר ממערכי ביט חיפוש המחוברות ביניהן על ידי אינדקס אחד או יותר המכיל סימן רווח באחד או יותר ממערכי ביט החיפוש.

כעת, כאשר יש בידינו מילונים המציינים מילים שנמצאו באחד ממערכי סיביות החיפוש (3 לפי הגדרת המטלה), מילון של סימניי רווח לפי האינדקס בהם הופיעו ורשימה של איזורים בעלי סבירות גבוהה להמצאות טקסט – נשמתש במתודה find_best_sequences למציאת קטעי הטקסט הסבירים ביותר.

- הערה:
  ניסיתי לממש מתודה למציאת קטעי הטקסט בהינתן מילוני מילים שנמצאו בכל המערכים, לפי האינדקסים בהם נמצאו המילים במספר אופנים. ראשית בחנתי אופציה לחיפוש רקורסיבי, אשר עובר על כל אינדקס במערך המקורי ומנסה למצוא בכל פעם מילה אשר מתחילה באינדקס הנוכחי, לאחר קיום רווח בין מילה למילה, ולבסוף – להחזיר את רצף המילים המופרדות על ידי רווחים אשר קיבל את הציון הקולקטיבי של מילים (בהתאם למילון) הגדול ביותר. שיטה זו עבדה די במהירות על טקסט החבוי בתמונה הנתונה כדוגמה למטלה (hidden.png), אך נכשלה כאשר ניסיתי לפענח טקסטים ארוכים יותר שהחבאתי בתמונות באמצעות התכנית שכתבתי לחלק א' של המטלה. לאחר ניסיון של מספר שיטות נוספות – החלטתי לבחור בשיטה המופיעה בתכנית זו, אשר מתרחשת במתודה find_best_sequences.

המתודה find_best_sequences בונה מילון של רצפיי מילים אשר נמצאים באיזור אשר סומן כבעל סיכוי גבוה להמצאות טקסט קודם לכן. השיטה הינה להתחיל ממילון של כל המילים המצויות באינדקס אשר מצויין לפי האיזור, לאחר מכן לנסות לבנות מילון של כל הרצפים המכילים 2 מילים באיזור, אחר כך מילון של רצפים המכילים 4 מילים באיזור וכך הלאה.

לאחר שלא ניתן למצוא כלל רצפים גדולים יותר מאשר אלו המצויים במילון בשלב קודם – מבצעים את התהליך בכיוון ההפוך, כלומר – מנסים למצוא במילון הרצפים העדכני (נניח רצפים בעלי n מילים) אופציות להוספת (n/2 מילים) ממילון קודם וחוזרים על תהליך זה עד שנשארים עם כל הרצפים העדכניים להם ניתן להוסיף מילה בודדת.

כעת בוחרים ברצפים אשר קיבלו את הציון הגבוה ביותר ושולחים אותם חזרה למתודה check_subarray.

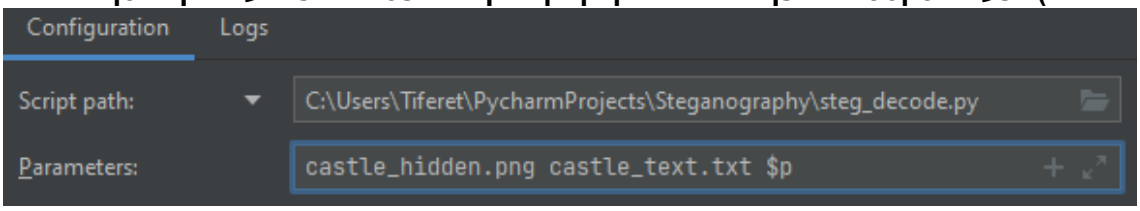
בשלב זה ניתן להשוות בין כל התוצאות שהתקבלו מכל איזור אופציונלי ולבחור בתוצאות הטובות ביותר לשלוח חזרה כתשובה אופציונלית לטקסט שהוסתר בתמונה.

מצאתי ששיטה זו, של ניסיון למצוא בכל פעם רצפים של n + n מילים, תוך כדי סינון רצפים שלא ניתן להוסיף להם רצף אחר – מפחיתה משמעותית את מספר האופציות לסידור המילים האופציונליות וכך גורמת לתכנית לעבוד מהר יותר. לאחר מכן, קל יותר לבחון את הרצפים שנותרו ולבחון בעבור כל אחד מהם תוספות של n/2 מילים בכל פעם עד להגעה להוספת מילה בודדה.

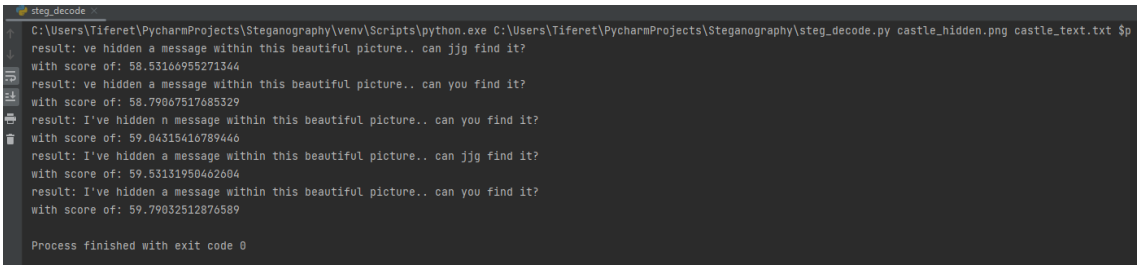
כך אנו מגיעים לבסוף רק לרצפים אשר מכילים כמות מקסימלית של מילים שזוהו ברצף וביניהם בוחרים את הרצף הסביר ביותר לפי הניקוד שקיבלו.

דוגמאות הרצה

1) פענוח טקסט המוצפן בתמונה לתוך קובץ נתון + הדפסת חלופות עם ציון למסך:

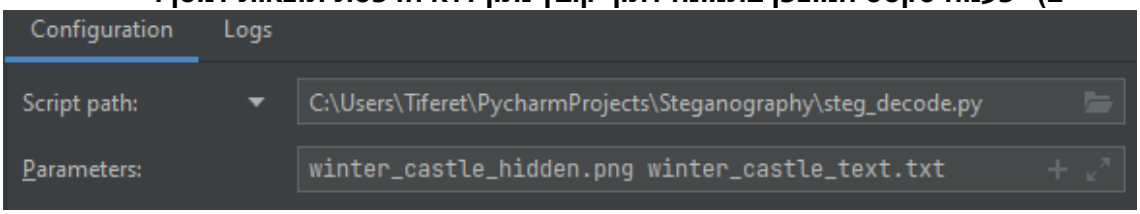


Console:



castle_text.txt:

2) פענוח טקסט המוצפן בתמונה לתוך קובץ נתון ללא הדפסת תוצאות למסך:



Console:

```
C:\Users\Tiferet\PycharmProjects\Steganography\venv\Scripts\python.exe C:\Users\Tiferet\PycharmProjects\Steganography\steg_decode.py winter_castle_hidden.png winter_castle_text.txt
```

winter_castle_text.txt:

winter_castle_text.txt - Notepad

File Edit View

```
Steganography is the practice of representing information within another message or physical object, in such a manner that the presence of the information is not evident to human inspection.
In computing or electronic contexts, a computer file, message, image, or video is concealed within another file, message, image, or video.

Generally, the hidden messages appear to be or to be part of something else: images, articles, shopping lists, or some other cover text.
For example, the hidden message may be in invisible ink between the visible lines of a private letter.
Some implementations of steganography that lack a shared secret are forms of security through obscurity.

The advantage of steganography over cryptography alone is that the intended secret message does not attract attention to itself as an object of scrutiny.
Plainly visible encrypted messages, no matter how unbreakable they are, arouse interest and may in themselves be incriminating in countries in which encryption is illegal.

Whereas cryptography is the practice of protecting the contents of a message alone, steganography is concerned with concealing the fact that a secret message is being sent and its contents.

Steganography includes the concealment of information within computer files.
In digital steganography, electronic communications may include coding inside of a transport layer, such as a document file, image file, program, or protocol.
Media files are ideal for transmission because of their large size.
For example, a sender might start with an innocuous image file and adjust the color of every hundredth pixel to correspond to a letter in the alphabet.
The change is so subtle that someone who is not specifically looking for it is unlikely to notice the change.
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

3) פענוח הטקסט המוצפן בתמונה הנתונה במטלה (hidden.png) לקובץ הדיפולטיבי:

Configuration | Logs

Script path: C:\Users\Tiferet\PycharmProjects\Steganography\steg_decode.py

Parameters: hidden.png

console:

steg_decode

```
C:\Users\Tiferet\PycharmProjects\Steganography\venv\Scripts\python.exe C:\Users\Tiferet\PycharmProjects\Steganography\steg_decode.py hidden.png

Process finished with exit code 0
```

205492507.txt:

205492507.txt - Notepad

File Edit View

```
What would you do if you had one hour to solve this task? Brute force? Random search? Well I would choose...
```

Ln 1, Col 109 | 100% | Windows (CRLF) | UTF-8