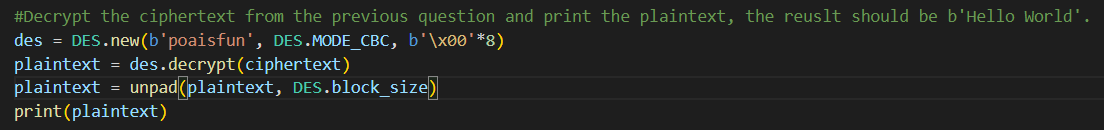
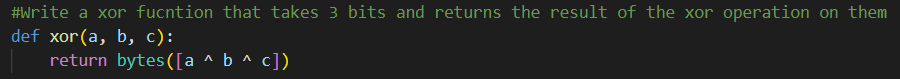
**אבטחת תקשורת – תרגיל 1**  
עידן סימאי 206821258.  
עידו אהרן 319024600.  
  
  
שלב 1:  
נתקין את הספריות הדרושות(בצורה הזו אנו לא מסתמכים על הגרסאות של הבודק, שכן בכל מקרה הוא יעבוד עם הגרסאות שכתבנו כאן).  
תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי  
  
שלב 2:  
נרפד את המחרוזת בצורה הבאה:  
תמונה שמכילה טקסט, גופן, צילום מסך

התיאור נוצר באופן אוטומטי  
ונראה שקיבלנו את הדרוש:  
  
וכמו שמצופה בDES, מס' הבתים שריפדנו הוא 5, ולכן 5 הבתים האחרונים הם 5.  
(Hello World זה 11 בתים, וצריך כפולה שלמה של 8).  
  
שלב 3:  
ניצור את המחרוזת המוצפנת בעזרת מופע של המחלקה DES, עם הפרמטרים המופיעים בקוד(המפתח 'poaisfun', אופן החלוקה לבלוקים, הפעלת ההצפנה, הפענוח – CBC ומחרוזת האתחול להצפנה הראשונית – IV היא מחרוזת שכולה 0).  
נזכור שגודל בלוק בDES הוא 8 בתים ולכן 8 פעמים x00.  
לאחר מכן הצפנו את הטקסט ממקודם(b'Hello World\x05\x05\x05\x05\x05'), והדפסנו בפורמט הקסה.  
תמונה שמכילה טקסט, צילום מסך, גופן, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי  
  
  
  
  
  
  
  
  
נראה שקיבלנו את הדרוש:  
תמונה שמכילה אביב, קפיץ סליל, טבע, כלי מתכת

התיאור נוצר באופן אוטומטי

שלב 4:  
נבטל את ההצפנה של המחרוזת מקודם.  
לאחר מכן נוריד את הריפוד ממקודם(של 5 הבתים האחרונים).   
  
נראה שקיבלנו את הדרוש:  
  
  
  
  
  
  
  
  
  
  
  
שלב 5:  
ניצור את פונקציית הXOR הבאה:  
נשים לב שאנחנו מחזירים את התוצאה בפורמט של בתים, כדי לקבל את הדרוש.  
  
נראה שקיבלנו את הדרוש:  
תמונה שמכילה טקסט, צילום מסך, גופן, טיפוגרפיה

התיאור נוצר באופן אוטומטי  
נקבל:  
תמונה שמכילה צילום מסך, גופן, טקסט, עיצוב

התיאור נוצר באופן אוטומטי  
  
שלב 6:  
נבנה את הפונק' הבאה:  
שזה בדיוק אותו דבר כמו הפענוח, רק שכאן לא בטוח שהוא יצליח כמו שצריך, ולכן אנחנו עוטפים בtry except.  
אם לא הצלחנו(כלומר במס' הבתים שרופדו בסוף לא קיבלנו בדיוק את אותו המס' בplaintext\_tag, לדוגמה במקרה שלנו לעיל ריפדנו ב5 – אז אם לאחר הפענוח אצלנו לא קיבלנו בplaintext\_tag בסוף 5 בתים של '\x05') האורקל יחזיר Fales.  
אם כן קיבלנו, נחזיר True ובעצם נוכל להתקדם לשלב הבא בהתקפה.  
תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי  
  
  
נראה שקיבלנו את הדרוש:  
תמונה שמכילה טקסט, צילום מסך, גופן, תוכנה

התיאור נוצר באופן אוטומטי  
ניסינו לשנות את הבית באינדקס ה5 בchipertext.  
ולאחר מכן הכנסנו את המקורית לOracle ונראה מה נקבל.  
תמונה שמכילה טקסט, גופן, צילום מסך, קו

התיאור נוצר באופן אוטומטי  
נראה מה קיבלנו:  
תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי  
השורה הראשונה היא הcipher\_text לאחר ששמנו את הבית x50 באינדקס ה5(כלומר בעצם modified\_bytes).  
ניתן לראות שלא כל 5 הבתים האחרונים הם \x05, ולכן קיבלנו False.  
לאחר מכן, כשהכנסנו לOracle(לא לtest\_oracle), קיבלנו True, כנדרש.  
**\*נשים לב, שיכול להיות שנשנה משהו בchipertext ובכל זאת נקבל True, אם רצף הבתים בסוף נשאר כרצף מרופד תקין, שזה בעצם כיצד ממשיכים את ההתקפה.**  
  
שלב 7:  
ניצור את המשתנה c בצורה הבאה:  
תמונה שמכילה טקסט, צילום מסך, גופן

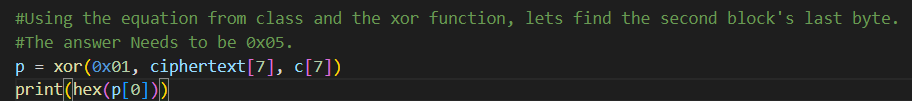
התיאור נוצר באופן אוטומטי  
כלומר שרשור של בית של \x00 ואז את הבלוק השני של ciphertext.  
נראה שקיבלנו את הדרוש:  
תמונה שמכילה אביב, קפיץ סליל, טבע, צילום מסך

התיאור נוצר באופן אוטומטי  
שלב 8:  
ניצור את הפונק' הבאה:  
תמונה שמכילה טקסט, צילום מסך, גופן, תצוגה

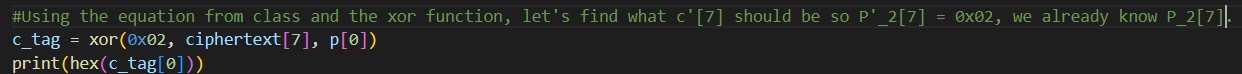
התיאור נוצר באופן אוטומטי  
נעבור בלולאה, כל עוד האורקל לא החזיר True על c.  
(נעזרנו בפונק' create\_c שהיא בעצם הפונק' מהסעיף הקודם, שיוצרת את c):  
תמונה שמכילה טקסט, גופן, צילום מסך, מספר

התיאור נוצר באופן אוטומטי  
הפונק' print\_c\_in\_hex:  
תמונה שמכילה טקסט, גופן, צילום מסך, מספר

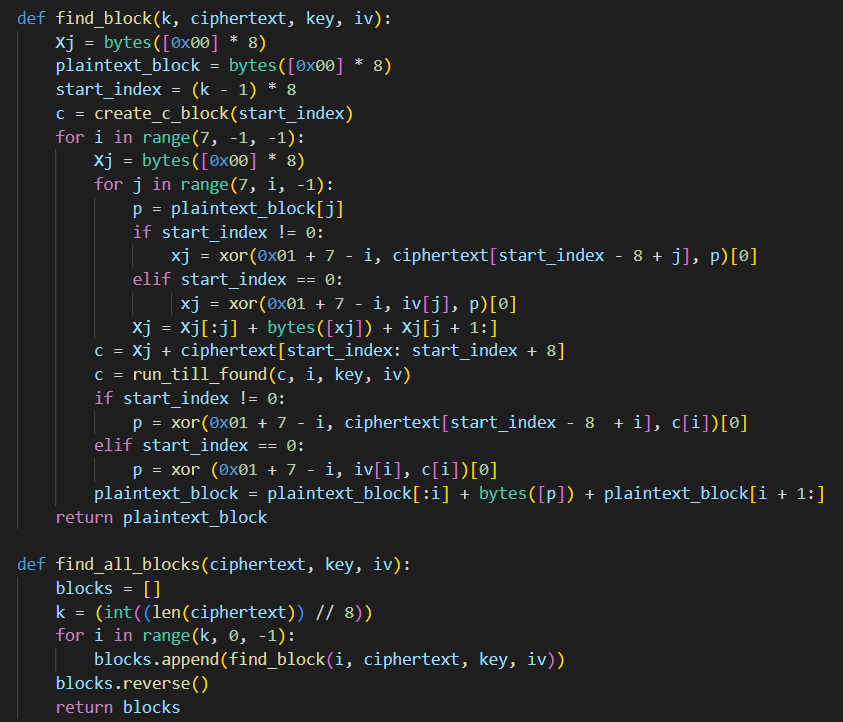
התיאור נוצר באופן אוטומטי  
לאחר שעצרנו בלולאה בrun\_till\_true, נבדוק מה קיבלנו:  
  
כלומר הבית ה8 השתנה ל0x7f.  
לעומת ההדפסה של c שהייתה לפני:  
  
  
  
שלב 9:  
כיוון שאנחנו רוצים לקבל את הבית האחרון בבלוק השני, לפי הנוסחה:  
תמונה שמכילה גופן, טקסט, טיפוגרפיה, קליגרפיה

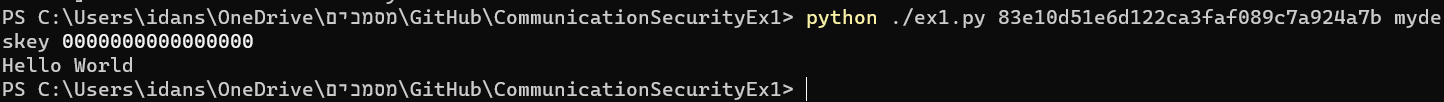
התיאור נוצר באופן אוטומטי  
אנחנו יודעים כי , כיוון שאנו רוצים שהאורקל יחזיר אמת(כלומר, שהשרשור בין ל בבית האחרון יחזיר אמת, ולפי DES אם יש בית אחד שרוצים לרפד שמים בו , אם שניים שמים פעמיים וכו').  
את אנחנו גם יודעים(זהו בעצם הבית באינדקס ה7 ב ciphertext המקורי), ועבור מצאנו אותו בסעיף הקודם, שכן זהו הבית שמחזיר true באורקל.  
  
כעת נראה את הערך שקיבלנו:  
   
כדרוש.

שלב 10:  
כיוון שאנחנו רוצים לקבל את הבית האחרון בבלוק השני, לפי הנוסחה:  
תמונה שמכילה גופן, טקסט, טיפוגרפיה, קליגרפיה

התיאור נוצר באופן אוטומטי  
אנחנו רוצים שיתקיים כי , את אנחנו גם יודעים(זהו בעצם הבית באינדקס ה7 ב ciphertext המקורי), וקיבלנו בסעיף הקודם ש.  
כעת, נחלץ את .  
  
נקבל:  
  
כלומר הבית באינדקס ה7 בc יהיה 0x7c.  
  
כדרוש.  
  
שלב 11:  
נשנה את הקוד בצורה הבאה:  
נחליף את הפונק' שרצה על האורקל שתהיה עם אינדקס כללי, במקום 7 קבוע שהיה מקודם.  
כעת, בפונק' הכללית למציאת הבלוק, אנו מאתחלים את לבלוק של 0, ונחזיר לבסוף את התוצאה בplain\_text\_second\_block.  
אנו מתחילים לרוץ על כל האינדקסים, על מנת לחשב מה הערכים של צריכים להיות, לפי הערכים בplaintext המקורי.  
(באיטרציה הראשונה איננו נכנסים לשם, כיוון שאנו יודעים שזה אמור להיות 0x01 בP', אז משיגים את התו האחרון בplaintext המקורי).  
לאחר מכן, אחרי שיש לנו את הערכים שאנו צריכים ב לפי הערכים הקודמים שיצאו לplaintext, אנו מייצרים את בלוק השרשור בין הבלוק השני המקורי של הciphertext לבין ה, ואת זה שולחים לבדיקה באורקל על מנת לדעת מה הבית הבא ב.  
לאחר שקיבלנו את השרשור המעודכן, נחשב את הבית הבא בטקסט המקורי.  
ככה נרוץ על כל האינדקסים(0 – 7).  
תמונה שמכילה טקסט, צילום מסך, תוכנה, תצוגה

התיאור נוצר באופן אוטומטי  
  
  
לבסוף קיבלנו:  
  
שזה בעצם:  
'rld\x05\x05\x05\x05\x05'  
  
שלב 12:  
נשנה את הפונק' שנעזרנו בהן מקודם לכלליות:  
תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי  
הראשונה היא כללית למציאת הבית שהאורקל מחזיר עליו אמת, השנייה היא ליצירת מחרוזת השרשור בין לCiphertext שאותה בעצם שולחים לאורקל.  
  
  
  
הפונק' find\_block היא זהה לפונק' מהסעיף הקודם, רק שכאן זה עבור בלוק כלשהו ולא בהכרח השני(וניתן לראות זאת לפי המשתנים), יש לשים לב שכאשר אנחנו מנסים לפענח את הבלוק הראשון אנחנו עושים זאת בעזרת ה ולא בעזרת הCiphertext (שכן לפי הנוסחה אנחנו צריכים את בלוק הCiphertext הקודם, אך אין קודם במקרה הזה, ולכן אנחנו משתמשים ב), וניתן לראות את התנאים בקוד שבודקים זאת.  
בפונק' השנייה אנחנו בעצם עוברים על כל הבלוקים(מהסוף להתחלה), ומכניסים את כולם למבנה הנתונים שהוא רשימה של בלוקים, ואז עושים reverse לרשימה.  
תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי  
כאן, כיוון שאנו מקבלים את הקלט ciphertext בהקסה, את המפתח בstring ואת הiv בהקסה, אנחנו ממירים, שולחים לפונק' לעיל, עושים לבסוף unpad אחרי ששרשרנו, ולבסוף ממירים חזקה לstring ומדפיסים למסך, כפי שנדרש בתרגיל.  
אנו קוראים לפונק' הזו עם הארגומנטים מהמשתמש.  
ניתן לראות שעבור הקלט מהמודל:  
83e10d51e6d122ca3faf089c7a924a7b mydeskey 0000000000000000  
נקבל:  
  
כדרוש.