תאריך:12.3.25

שם תלמיד: רוני אלפסי

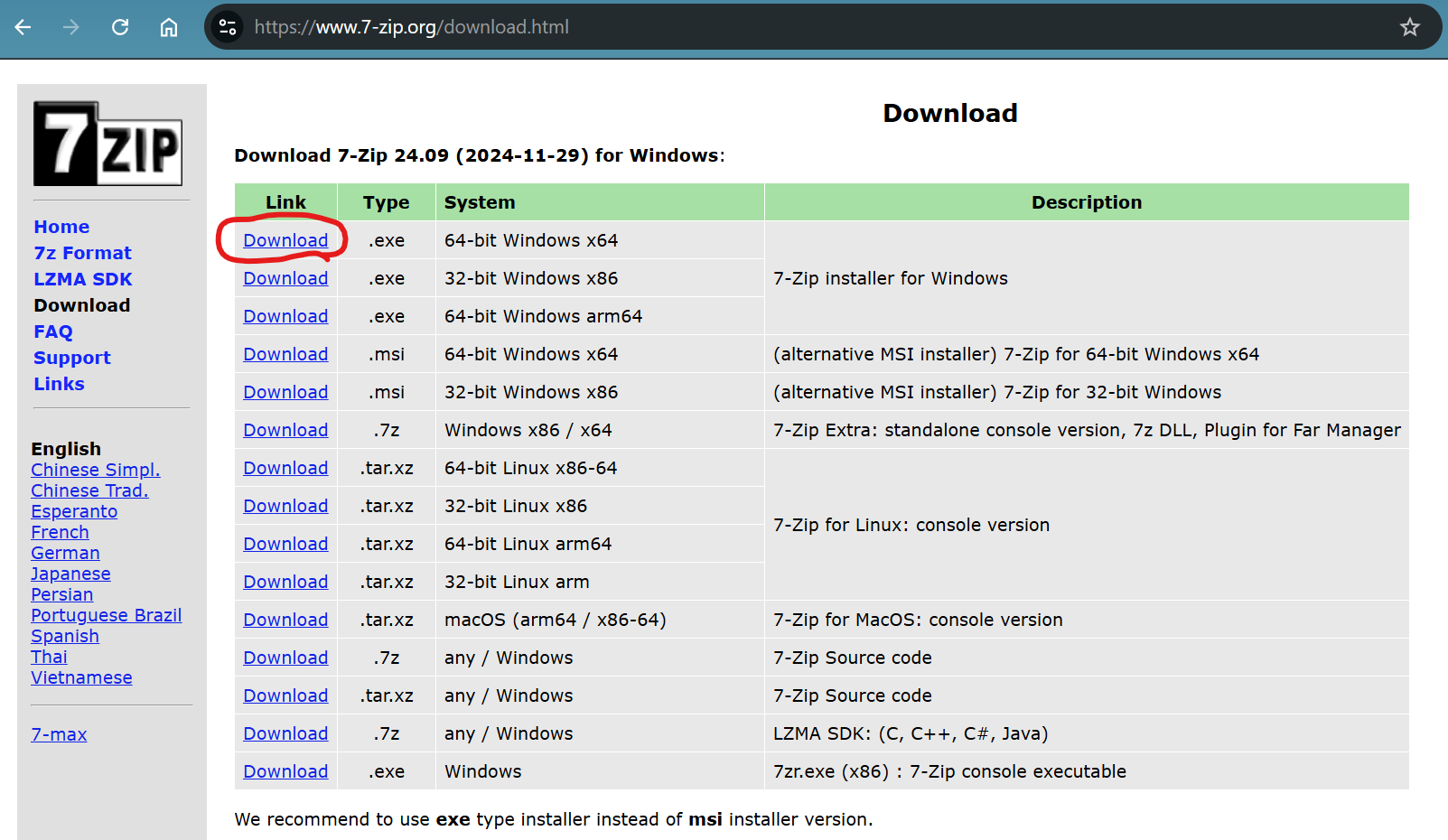
**מטלת סוף מודול פייתון**

למטלת סוף המודול בחרתי בתרגיל מספר 5.

הסבר קצר על הסקריפט: על המשתמש לספק קובץ טקסט (בעזרת מיקום הקובץ המלא לדוגמה: "C:\Users\Roni\Desktop\PossiblePasswords.txt" ) שבו כל המילים יהפכו לרשימה שהסקריפט ישתמש בהם וינסה לחלץ את התכנים של הקובץ הדחוס והמוגן על ידי סיסמה (על המשתמש לספק גם את מיקום הקובץ הדחוס לפי אותה הדוגמה שניתנה מקודם. אך לפני כל הבקשות האלו המשתמש מתבקש לבחור בסוג הקובץ, zip או rar. בנוסף לכך, כפי שהתבקש בתרגיל התהליך משתמש במלוא משאבי המחשב על ידי חילוק פעולת בדיקת הסיסמאות לכל thread פנוי במעבד.

לפני הרצת הסקריפט יש להוריד את התוכנה 7-Zip בגלל שאין ספריות בפייתון שיכולות לחלץ קבצי rar עם סיסמה מבלי להשתמש בתוכנה שיודעת לעבוד עם קבצי rar כמו WinRAR או 7-Zip והחלטתי להשתמש בתוכנה 7-Zip בכדי לבצע את הפעולות הללו. לאחר מכן יש להוסיף את הpath של התוכנה לsystem variables כדי שהסקריפט יוכל להשתמש בפקודות מערכת המשתמשות בתוכנה.

**שלב 1:** התקן את התוכנה **7-Zip** (יש לזכור את מיקום ההתקנה): <https://www.7-zip.org/download.html>

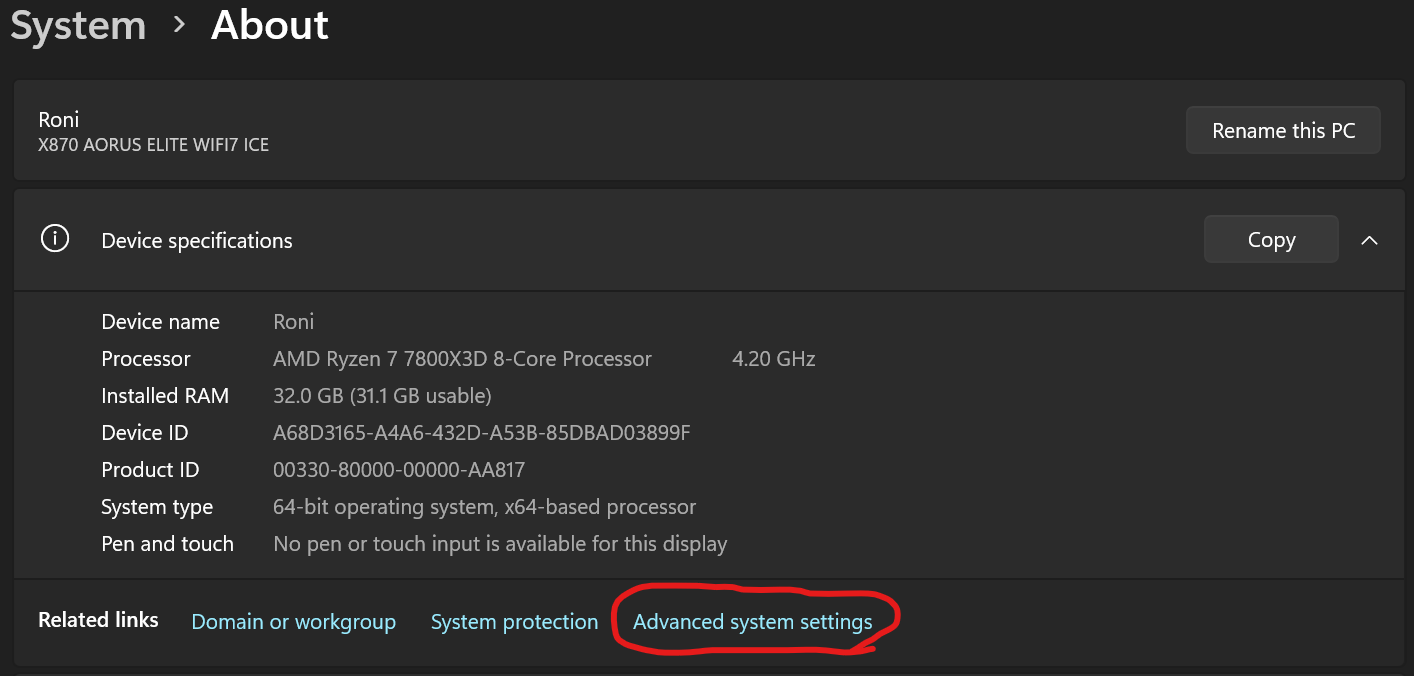


**שלב 2:** ללחוץ קליק ימני על כפתור הווינדוס במחשב ואז להכנס ל**System**:

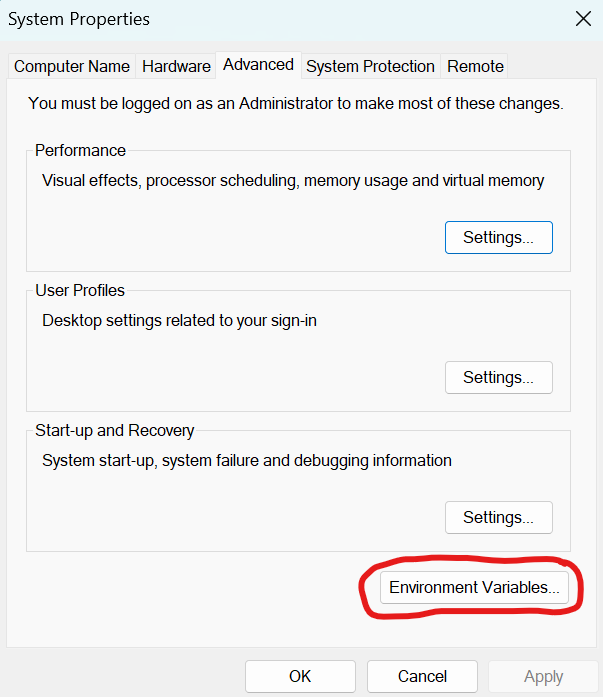
תמונה שמכילה טקסט, צילום מסך, מולטימדיה, תוכנה

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.

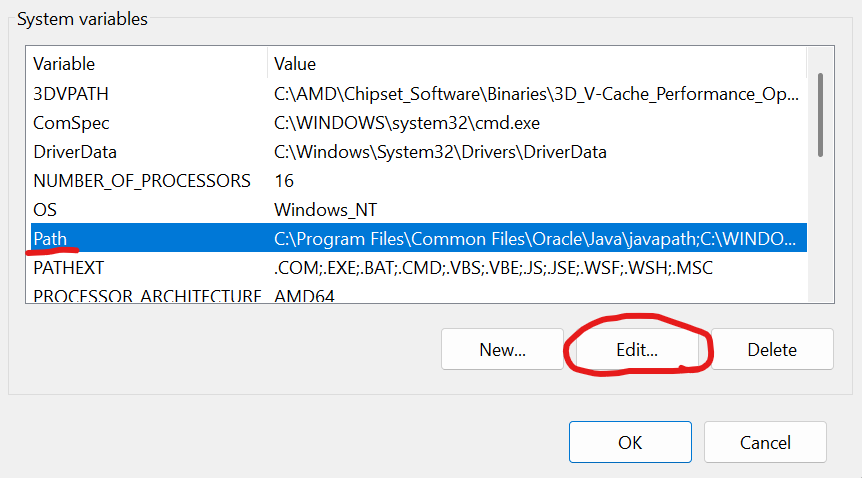
**שלב 3:** להכנס ל**Advanced system settings**:



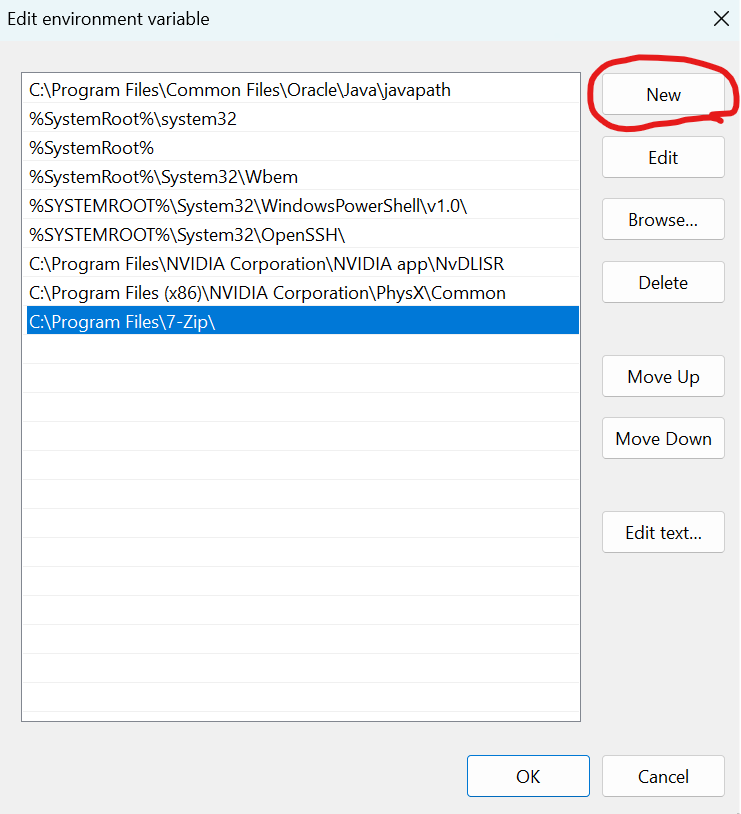
**שלב 4:** להכנס ל**Environment Variables**:

****

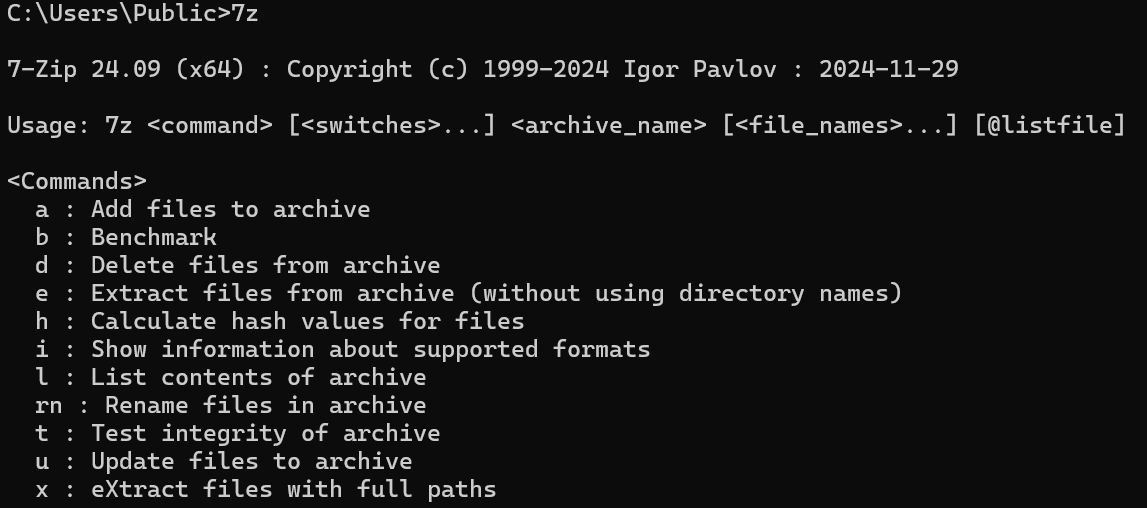
**שלב 5:** סמן את **Path** ולאחר מכן לחץ על **Edit**:

****

**שלב 6:** לחץ על **New** ולאחר מכן הדבק את המיקום שבו התקנת את התוכנה **7-Zip** (במקרה שלי התוכנה מותקנת ב**”C:\Program Files\7-Zip”**) ולבסוף יש ללחוץ על **OK** בכל החלונות שנפתחו.

****

**שלב 7:** לבדיקה שהתוכנה של **7-Zip** עובדת יש לרשום את הפקודה **7z** ב**cmd** ואם הכל עבד תופיע גרסת התוכנה וכל האפשרויות שלה.



**הסבר מפורט על הסקריפט:**

# ייבוא של ספריות שאיתן אנו מסוגלים לבצע פעולות כמו לחלץ קובץ דחוס מסוג zip עם pyzipper, לחלץ קובץ דחוס מסוג

# rar עם פקודות מערכת המשתמשות בתוכנה 7-Zip בעזרת subprocess ולחלק פעולות שהסקריפט עושה לthreads כך

# שנוכל להשתמש בכל הכוח של המעבד שלנו ולהריץ את הבדיקות סיסמה בכל הthreads הפנויים בו זמנית

# בעזרת concurrent.futures

import pyzipper  
import subprocess  
import concurrent.futures

# יצירת פונקציה של פורץ הקובץ הדחוס מסוג zip כדי שאחר כך יהיה קל לקרוא לפונקציה כשצריך

def zipcracker():

# בקשה מהמשתמש להזין את מיקום קובץ הסיסמאות ומיקום הקובץ הדחוס

passwordListPath = input(r"enter password list path: ")  
 compressedFilePath = input(r"enter compressed file path: ")

# הפיכת קובץ הסיסמאות לרשימה בתוך הסקריפט תוך כדי מחיקת רווחים

passwordList = [passwords.strip() for passwords in open(passwordListPath)]

# יצירת משתנה לקובץ הדחוס וסימונו כקובץ zip על ידי ספריית pyzipper שנוכל לבצע עליו פונקציות בהמשך

compressedFile = pyzipper.AESZipFile(compressedFilePath)

# יצירת משתנה לסיסמה שנמצא בהמשך. בינתיים שמתי "" בתור ממלא מקום

foundPassword = ""

# יצירת פונקציה ספציפית למציאת הסיסמה כדי שאחר כך נוכל לחלק את הפעולות שמתבצעות בפונקציה הזאת לthreads

# עם אופציה לסיסמה כך שאחר כך כשנחלק את הפונקציה לthreads, ונשים את הפעולה בלופ, נוכל לשים שם את

# הארגומנט “password” ואז הוא ישים את הסיסמאות מהרשימה

def tryzippassword(password):

# הפיכת המשתנה ל"לא מקומי" כדי שאחרי זה אם יהיה שינוי אליו (במידה והסיסמה נמצאת) נוכל להקריא את הסיסמה

# למשתמש

nonlocal foundPassword

# תנאי הבודק אם הסיסמה נמצאה כדי שהסקריפט יפסיק לרוץ ולחפש סיסמה

if foundPassword:  
 return

# שמתי כאן “try block” המנסה לחלץ את הקובץ הדחוס עם סיסמה (הספרייה של pyzipper בודקת סיסמאות עם

# סטרינגים בינארים ולכן יש לקודד את הסיסמה בנוסף) בשורה לאחר מכן המשתנה “foundPassword” הופך לסיסמה

# רק במידה והסיסמה אכן נכונה, לכן אני בודק שרק אם התנאי “foundPassword” השתנה שיקריא למשתמש את הסיסמה

# שנמצאה. במקרה והסיסמה לא הייתה נכונה תופיע תקלה ולכן עשיתי שהסקריפט ימשיך כאילו לא קרה כלום

try:  
 compressedFile.extractall(pwd=password.encode())  
 foundPassword=password  
 if foundPassword != "":  
 print(f"Password found!: {foundPassword}")  
 except Exception:  
 pass

# שימוש בספריית concurrent.futures המחלקת את הפעולה של הפונקציה tryzippassword() עם סיסמאות מהרשימה

# לthreads כך שהפעולה תתבצע בכל הthreads הפנויים במעבד

with concurrent.futures.ThreadPoolExecutor() as executor:  
 futures = [executor.submit(tryzippassword, password) for password in passwordList]

# במקרה והסיסמה לא נמצאה ברשימה, הנתון foundPassword ישאר "" ולכן שמתי תנאי שיראה שהסיסמה לא נמצאה

# ברשימה במקרה הזה

if foundPassword == "":  
 print("Password not found in the wordlist!")

# יצירת פונקצייה הפורצת סיסמאות של קבצים דחוסים אבל של קבצים מסוג rar. הפונקציה ברובה דומה לפונקצייה

# הקודמת ולכן ארשום הסבר רק איפה שיש שינוי

def rarcracker():  
 passwordListPath = input(r"enter password list path: ")  
 compressedFilePath = input(r"enter compressed file path: ")  
 passwordList = [passwords.strip() for passwords in open(passwordListPath)]  
 foundPassword = ""  
 def tryrarpassword(password):  
 nonlocal foundPassword  
 if foundPassword:  
 return

# ב”try block” הזה השינוי היחידי הוא דרך החילוץ של הקובץ מסוג rar. ה”7z” משתמש בתוכנה 7-Zip, ה”x” מבצע את

# פעולת החילוץ, “compressedFilePath” הוא מיקום הקובץ שברצוננו לחלץ, f”-p{password}” קובע את הסיסמה

# ש7-Zip ינסה לחלץ את הקובץ איתה ו”-y” נותן yes כתשובה במידה ואנו מתבקשים לוודא את הבקשה

try:  
 result = subprocess.run(["7z", "x", compressedFilePath, f"-p{password}", "-y"], stdout=subprocess.PIPE, stderr=subprocess.PIPE)  
 if result.returncode == 0:  
 foundPassword = password  
 print(f"Password found!: {foundPassword}")  
 except Exception:

pass  
 with concurrent.futures.ThreadPoolExecutor() as executor:  
 futures = [executor.submit(tryrarpassword, password) for password in passwordList]  
 if foundPassword == "":  
 print("Password not found in the wordlist!")

# זאת ההודעה הראשונה שתופיע למשתמש המברכת את המשתמש עם תיאור קטן על הסקריפט

print("Welcome to my zip/rar file password cracker!")

# לולאת “while True” שתמיד תרוץ עד שהסיסמה תימצא או שיתגלה שהיא לא ברשימה. השורה הראשונה בה שואלת

# את המשתמש איזה סוג קובץ דחוס ברצונו לפרוץ כדי שבהמשך בתנאי “if” נוכל לקרוא לכל פונקצייה בהתאם (בשיטה זו

# יש אפשרות גם להוסיף סוגי קבצים נוספים שהסקריפט יתמוך בפריצה שלהם. לבסוף, אם והמשתמש כותב סוג קובץ

# דחוס שלא נתמך או שהוא עושה שגיאת כתיב בטעות, תינתן לו הודעה שהקובץ לא נתמך ויש לבחור אך ורק בין zip או

# rar ואז הלולאת while תרוץ מחדש והמשתמש יתבקש להזין את סוג הקובץ שוב

while True:  
 fileType = input("Choose a file type between zip/rar: ".lower())  
 if fileType == "zip":  
 zipcracker()  
 break  
 elif fileType == "rar":  
 rarcracker()  
 break  
 else:  
 print(fileType + " is not a supported file type. You may only choose between zip and rar!")