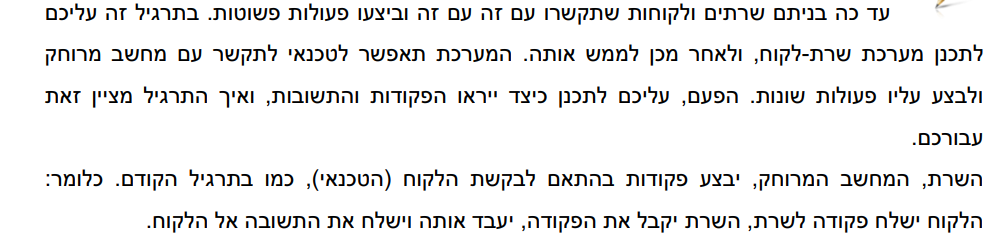
תרגיל 2.7 – טכנאי מרוחק

תרגיל זה מבוסס על תרגיל 2.7 בספר הלימוד - יש לקרוא אותו .

בכל מקום שיש התנגשות בין הדרישות שמופיעות בספר הלימוד ובמסמך זה, מה שקובע הוא **המסמך הזה**.



עליכם להשתמש בעקרונות של משימת החקר, תרגיל 2.6. מומלץ ורצוי להשתמש בקטעי הקוד שסופקו.

השרת יהיה מסוגל לקבל פקודות (הוראות) מרחוק (דרך התקשורת) ולבצע אותן.

בנוסף לפעולות שהי ב 2.6 (שם, מספר רנדומלי, date ו exit )   
השרת יממש לפחות את המשימות הבאות:

1. אפשרות הרצת תכנית על המחשב המרוחק. EXECUTE (כל exe הקיים בשרת)
2. הצגת תוכן ספרייה במחשב המרוחק (שרת) DIR
3. אפשרות למחוק קובץ במחשב המרוחק DELETE
4. ביצוע העתקת קובץ על המחשב המרוחק COPY
5. שליחת תוכן קובץ מהמחשב המרוחק (שרת) למחשב הטכנאי (לקוח) (הבאת קובץ ללקוח) (יש להביא קובץ ב Chunks (לא לשמור את כל תוכנו במבנה נתונים בזיכרון) השיטה קריאת מקטע מהצד השולח שליחת chunk והצד המקבל מייד שם את זה בקובץ היעד)   
   ניתן לבצע את זה עם acknowledge לאחר כל מקטע או בלי.
6. צילום מסך ושמירתו כקובץ בצד המרוחק (בצד השרת). והבאתו והצגתו בלקוח Screen (יש להשתמש ב 5 לשם הבאת הקובץ אך יש להפעילו אוטומטי ולא לחכות לקלט מהמשתמש)

רשות –   
 צפייה ברצף מהיר של תמונות ממסך השרת (וידאו) ,   
 העברת עכבר לשרת (תזוזה ולחיצה) ,   
 העברת קובץ מהלקוח לשרת   
 העברת המצלמה מהשרת ללקוח (כמו צפייה של מסך השרת אבל המקור הוא המצלמה)

יש לתכנן פרוטוקול לפני תחילת הקוד.   
  
כדאי להכניס את tcp\_by\_size לקוד הנוכחי של 2.6 למחוק את recv ו send של 2.6 וה log ולבדוק ש4 הפעולות עובדות תקין . רק לאחר מכן להתחיל עם הפקודות החדשות של 2.7 .

יש להפריד בין הפרוטוקול (ההודעות שעוברות בתקשורת) לבין ממשק המשתמש.

מומלץ להשיג הפרדה זו בעזרת תפריט ותפריטי משנה בצד הלקוח.

לדוגמא במסך ראשי יהיה תפריט הכולל את כל 10 הפעולות שהוזכרו לעיל. לאחר בחירה של אחד האפשרויות יוצג למשתמש תפריט משני. לדוגמא עבור אפשרות 1 (צילום מסך) יוצג למשתמש בקשה להקלדת שם הקובץ שאליו ירצה לשמור את צילום המשך. ועבור אופציה 2 (שליחת קובץ) יוצג בתפריט המשנה שתי בקשות, אחת לספק מסלול מלא של הקובץ שרוצים להעביר מהמחשב המרוחק והשנייה שם הקובץ שאליו ישמר הקובץ המועבר במחשב המקומי (מחשב הטכנאי, לקוח)

עבור כל פעולה יש להגדיר קוד פעולה שהלקוח ישלח לשרת וקוד פעולה שהשרת יחזיר תשובה ללקוח. גם אם אין לשרת להחזיר מידע עבור הפעולה, עדין השרת יכול להחזיר ללקוח תשובה בסגנון "הצלחה" או כישלון או לדוגמא את שם הקובץ שנשמר בצד המרוחק.

## 1. צילום מסך

לצורך ביצוע צילום מסך השתמשו במודול pyautigui

מודול זה לא מגיע כחלק מהתקנה הסטנדרטית של פייתון וכנראה תצטרכו לבצע התקנה

pip install pyautogui

בקוד תצטרכו להוסיף את השורה

Import pyautogui

הכדי לצלם את המסך ולשמור את התמונה לקובץ

image = pyautogui.screenshot()  
image.save(r'C:\screen.jpg')

את שם הקובץ לשמירה השרת יקבל מהלקוח כפרמטר כחלק מהפקודה

## 2. שליחת קובץ

הלקוח ישלח לשרת פרמטר שיציין את שם הקובץ שעליו לשלוח חזרה. שם הקובץ יכול להיות עם מסלול מלא (full path) או מסלול יחסי (relative path). הלקוח יקבל את רצף הבתים (תוכן) הקובץ וישמור אותו מקומית (במחשב בטכנאי) בשם שהמשתמש הזין בממשק המשתמש קודם לכן.

שימו לב שהקובץ לא חייב להיות מסוג טקסט (לכן כנראה שתצטרכו לפתוח אותו עם הדגל ‘rb’)

שימו את הדעת לכך שגודל הקובץ יכול להיות גדול מאוד ולא יהיה ניתן לשלוח את כולו בפעם אחת וכמו כן בצד הלקוח יכול להיות שגם הוא לא יתקבל בפעולת recv אחת. כנראה תצטרכו לבצע לולאה של שליחות.

חישבו איך הצד מקבל יידע שכל תוכן הקובץ הגיע. כנראה שתרצו להשתמש בשדה אורך. וכנראה ששדה זה יהיה שימושי לכם ברוב הפקודות, אז מומלץ להשתמש בו עבור כל הפקודות כחלק מהפרוטוקול.

## 3. הצגת תוכן הספרייה (בדומה לביצוע הפקודה dir)

הלקוח ישלח לשרת את שם הספרייה שהוא מעוניין לקבל את המידע עבורה והשרת יחזיר לא מחרוזת שתתאר את כל הקבצים. הלקוח לא ידפיס למסך את המחרוזת כמו שקיבל אותה מהשרת. הלקוח יבצע עיבוד כלשהו על המידע, לדוגמא יפריד כל שם לשורה נפרדת ללא תווים נוספים כמו פסיקים או סוגריים כלשהם. בקובץ יש להדפיס את גודלו ושם עם סיומת. בתיקייה לציין שזו תיקייה

לצורך קבלת תוכן הספרייה בצד השרת תוכלו להשתמש לדוגמא במודול glob

import glob

files\_list = glob.glob(r’c:\cyber\\*.\*’)

חיקרו והבינו איך להשתמש בפעולה זו. הבינו מה הפרמטר שהיא מקבלת ואיך היא מחזירה את התוצאה.

אח"כ עליכם לבנות מתוצאה זו תשובה על פי הפרוטוקול שבניתם.

שימו לב: תוכן ספרייה יכול להיות מאוד מאוד גדול. בידקו אם הפקודה עובדת אצלכם לדוגמא על הספרייה

C:\windows\system32

## 4. מחיקת קובץ

הלקוח ישלח לשרת פרמטר שיציין את שם הקובץ שעליו למחוק בצד השרת. שם הקובץ יכול להיות ללא ציון מסלול ואז הוא ממוקם בספרייה הנוכחית של השרת, אם עם מסלול מלא (full path) או בעל מסלול יחסי.

יש להחזיר ללקוח האם הפעולה הצליחה ואם לא אז סוג שגיאה (כמו קובץ לא קיים, או קובץ תפוס ולא ניתן למחיקה ועוד)

לצורך מחיקת קובץ אתם יכולים להעזר במודול os

import os

os.remove(r’c:\cyber\blabla.txt’)

## 5. העתקת קובץ

הלקוח ישלח לשרת פרמטר שיציין את שם הקובץ שעליו להעתיק וגם את שם ספריית (ואולי גם שם הקובץ) היעד שאליו להעתיק. פעולת העתקה מתבצעת בצד המרוחק.

להעתקה של קבצים ניתן להשתמש במודול shutil

import shutil

shutil.copy()

## 6. הרצת תוכנית

פעולה זו תאפשר לטכנאי להריץ תוכניות כמו calc, word, notepad++ ועוד על המחשב המרוחק. (חישבו איזו יכולת זה יכול לתת לטכנאי)

על מנת להריץ תוכנות נוכל להשתמש במודול **subprocess**

Import subprocess

subprocess.call(‘notepad’)

לפעמים לא מספיק לתת רק שם של תוכנית, אלא יש לספק את המסלול המלא את קובץ ההרצה

subprocess.call(r‘c:\windows\system32\notepad.exe’)

## 7. יציאה/התנתקות

הלקוח ישלח פקודת התנתקות. השרת יסגור את התקשורת עם הלקוח

במידה והפרוטוקול שלכם שולח תשובת הצלחה לפעולה זו ללקוח ומיד סוגר את הסוקט, חישבו מה עלול לקרות להודעה שהשרת שלח). טפלו במקרים חריגים.

# הנחיות ומחוון לתרגיל 2.7

* יש לצרף מסמך תיכנון פרוטוקול כחלק מקבצי הקוד.
  + את הפרוטוקול יש לבנות לפי העקרונות שלמדנו.
  + יש לספק שדה "קוד פעולה" ל-2 הכיוונים (מלקוח לשרת וגם בחזרה)
  + כל הפעולות יכולות להכשל ויש להתמודד עם שגיאות ולהחזיר ללקוח תשובות מתאימות
  + כל פעולת ביצוע מהלקוח לשרת תיהיה מורכבת מהודעה אחת עם פרמטרים ולא שיחה שכל פעם הלקוח שולח לשרת פרמטר. (לדוגמא אם לקוח מבקש לבצע פעולת העתקת קובץ אז כל הפרמטרים יועברו בהודעה אחת, שם הקובץ ויעד העתקה)
* כל הקוד עטוף בפונקציות
* יש להשתמש ב- if \_\_name\_\_ == “\_\_main\_\_”
* יש להגן בפני חריגות
* לקוח שמתנתק בפתאומיות לא מקריס את השרת
* אחרי שלקוח מתנתק מהשרת, השרת ממשיך לעבוד ולחכות ללקוח חדש שיתחבר
* שרת שמתנתק בפתאומיות לא מקריס את הלקוח. הלקוח יזהה זאת בשלב כלשהו ויצא מסודר מהתוכנית, אחרי שהדפיס הודעה מתאימה למסך (לדוגמא: לא נמצא שרת זמין)
* שרת ולקוח מטפלים בקבלת מידע באורך 0
* יש לדאוג ולטפל במצבי שגיאה.
* לדוגמא מצב שגיאה: לקוח שולח פעולה שהשרת לא יודע לטפל
* יש לתמוך בהעברת מידע גדול, כמו קבצים וזאת באמצעות לולאה שקוראת מהסוקט כל פעם כמות מסויימת (נניח 16KB ) עד אשר כל המידע הועבר
* יש להגדיר קבועים בתחילת הסקריפט ולהשתמש בקבועים האלו בתוכנית
* במידה ויש שימוש במספרים או מחרוזות במהלך הסקריפט ואלו לא הוגדרו כקבועים בתחילת הסקריפט, צריכה להיות סיבה טובה (והסבר לכך) מדוע זה בוצע כך
* כאשר הלקוח מביא קובץ מהשרת ושומר אותו בספרייה אצלו, יש להשתדל להשתמש במסלול יחסי (relative path), יחסי למיקום ריצת הסקריפט, ולהימנע בשימוש במסלול שמתאים למחשב ספציפי (כמו המחשב שלכם) ולא מתאים למחשב אחר (כמו של הבודק של התרגיל שלכם)
* יש להדפיס למסך או לקובץ מידע שעוזר בדיבאג של התוכנית
* חייבים לתמוך במצב של פיצול הודעות וחיבור הודעות לכן יש להצמיד את מודול tcp\_by\_size.py כקובץ לתיקייה וכל התקשורת תעשה ע"י שתי הפעולות שבתוכו. הפעולות מבצעות גם log וגם מוסיפות ומנתקות את שדה האורך בתחילת כל הודעה. (יש להבין כל שורה גם בקובץ זה)  
  כדי להשתמש בפעולות יש לכתוב את הבא גם בשרת וגם בלקוח

from tcp\_by\_size import send\_with\_size ,recv\_by\_size

* אסור שתהיה יותר מקריאה אחת ל- send\_with\_size ו-recv\_by\_size והקריאות חייבות להופיע באותה הפעולה (באותה הרמה). כלומר הפעולה ששולחת היא גם זו שמקבלת. פעולות אחרות יכולות להכין את המידע שצריך לשלוח.
* חריג (לסעיף הקודם): אם שולחים קובץ בחלקים מותר לשלוח בלולאה הרבה send (עדיין חייבים לשלוח קוד הודעה באורך תיקני) אבל כדאי שהצד השני ידע בדיוק כמה recv לעשות .
* אין להניח שכל התעבורה תהיה מחרוזתית (בהחלט יכול להיות שישלחו גם קבצים בינאריים יש לעבוד עם encode/decode בחוכמה.
* יש לבדוק את כל הפעולות גם על מחשבים שונים לשרת וללקוח.
* יש להפריד בין ממשק המשתמש לפרוטוקול.
  + כלומר הלקוח לא מציג את המידע שהשרת שולח לו כמו שהוא, אלא מבצע עיבוד כלשהו למידע ואז מציג אותו
* **בונוס: הוספה של פעולות/פקודות בין הלקוח לשרת מעניינות נוספות מעבר למה שמפורט בתרגיל.**
  + **לדוגמא העברת סרטון live של מה שקורה בצד השרת.**
* **בונוס: שימוש במודולים שלא נלמדו בכיתה, לדוגמא logging**