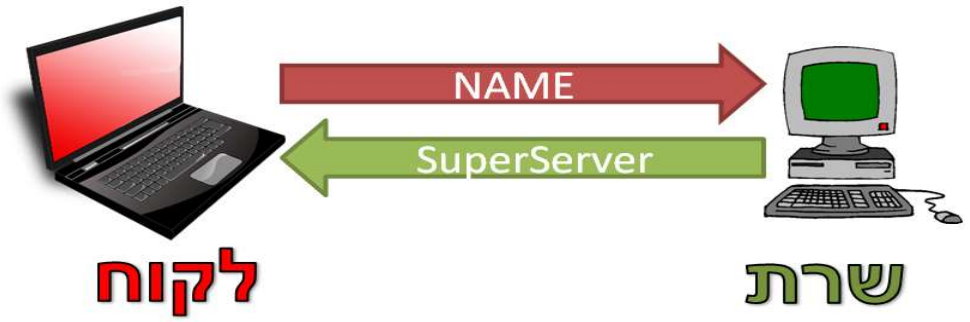

שלב 2:

תרגול כתיבת פרוטוקול – האם בשלב 1 הלקוח שלכם ביצע socket.recv(1024)? כעת, צרו פרוטוקול שמאפשר לשלוח מהשרת ללקוח הודעות באורך שונה. יכולת זו תשמש אתכם בתרגילים בהם לא תוכלו להניח שאורך ההודעה מהשרת ללקוח הוא 1024 בתים או מספר קבוע כלשהו – לדוגמה בהעברת קובץ.

אפשרויות לדוגמה:

- השרת ישלח ללקוח הודעה שכתוב בה מה אורך המידע שעליו לקבל בתור תשובה.
- השרת ישלח ללקוח הודעה מיוחדת שמשמעותה "שליחת המידע הסתיימה".

שלב 3:

כיוון שאנו עוסקים בהגנת סייבר, עלינו לכתוב שרת יציב, כלומר גם אם הלקוח שולח "זבל", לשרת שלנו אסור לקרוס. בדקו את יציבות השרת באמצעות הודעות מסוגים שונים.

הנחיות לתרגיל 2.6

לפני שאתם ניגשים לכתוב את הקוד, בצעו תכנון ראשוני. חשבו מה בדיוק תממשו בצד הלקוח ומה בצד השרת, ונסו לצפות מראש בעיות בהן תתקלו.

בצד הלקוח – ראשית, עליכם לבקש מהמשתמש לבחור באחת מהפקודות שצוינו לעיל (רמז: היעזרו בפונקציה **raw_input**).

לאחר מכן, שלחו את הבקשה לשרת, קבלו את התשובה והציגו אותה למשתמש.

בצד השרת – עליכם לקבל חיבור מהלקוח, להבין את הבקשה שלו ולהגיב אליה בהתאם. לאחר מכן, נתקו את החיבור ותוכלו לספק שירות ללקוח חדש.

בהצלחה!

תרגיל 2.7 – שרת פקודות מתקדם- טכנאי מרוחק (אתגר)

עד כה בניתם שרתים ולקוחות שתקשרו עם זה עם זה וביצעו פעולות פשוטות. בתרגיל זה עליכם לתכנן מערכת שרת-לקוח, ולאחר מכן לממש אותה. המערכת תאפשר לטכנאי לתקשר עם מחשב

מרוחק ולבצע עליו פעולות שונות. הפעם, עליכם לתכנן כיצד ייראו הפקודות והתשובות, ואיך התרגיל מציין זאת עבורכם.

השרת, המחשב המרוחק, יבצע פקודות בהתאם לבקשת הלקוח (הטכנאי), כמו בתרגיל הקודם. כלומר: הלקוח ישלח פקודה לשרת, השרת יקבל את הפקודה, יעבד אותה וישלח את התשובה אל הלקוח.

שימו לב, לרשותכם קבצים המכילים את שלד התרגיל הן בשרת והן בלקוח. עליכם למלא בתוכן את הפונקציות, לפי התיעוד שנמצא בפונקציות ולפי ההדרכה בהמשך.

שלד השרת: data.cyber.org.il/networks/server_template.py

שלד הלקוח: data.cyber.org.il/networks/client_template.py

להלן הפקודות שעליכם לממש:

2.7.1

כדי להבין מה מתרחש במחשב המרוחק, הטכנאי שלנו רוצה לקבל תצלום מסך של המחשב המרוחק. עליכם לתמוך בכך בשרת ובלקוח.

- המשתמש בלקוח יוכל להקליד פקודות, שהלקוח יעביר לשרת.
- תמכו בפקודה TAKE_SCREENSHOT, שתגרום לכך שהשרת יבצע צילום מסך וישמור את הקובץ במחשב השרת, במיקום לפי בחירתכם
- מודול פייתון **PIL** לעבודה עם תמונות – מותקן יחד עם סביבת גבהים (להתקנה עצמאית- http://www.pythonware.com/products/pil).
- השתמשו בקוד הבא כדי לצלם את המסך ולשמור את התמונה לקובץ (הקוד שומר את התמונה למיקום: C:\screen.jpg), שנו אותו לפי הצורך:

```
from PIL import ImageGrab
im = ImageGrab.grab()
im.save(r'C:\screen.jpg')
```

2.7.2

צילום המסך נוצר בשרת, אולם כדי שהטכנאי יוכל להשתמש בו, עליכם לשלוח אותו ללקוח. פקודת SEND_FILE יחד עם שם הקובץ הרצוי צריכה לגרום לשרת לשלוח את הקובץ המבוקש ללקוח. שימו לב – צילום המסך גדול למדי, חלקו אותו לחלקים ושילחו אותם ללקוח. המציאו פרוטוקול שיאפשר ללקוח לדעת שהשרת סיים לשלוח אליו את כל התמונה.

2.7.3

לאחר שצפה בתצלום המסך של המחשב המרוחק, הטכנאי שלנו חושד שהקבצים של תוכנה כלשהי לא נמצאים במקום או שלא כל הקבצים נמצאים. הטכנאי מעוניין להציג תוכן של תיקייה מסוימת במחשב

המרוחק. לדוגמה, הצגת רשימת הקבצים שבתיקייה C:\Cyber. תמכו בפקודת DIR – השרת ישלח ללקוח את התוכן של תיקייה מבוקשת. לדוגמה:

```
DIR C:\Cyber
```

לחיפוש על פי שמות קבצים, ניתן להשתמש במודול **glob**.

לדוגמה, להצגת כל הקבצים בתיקייה C:\Cyber, ניתן להריץ:

```
import glob
files_list = glob.glob(r'C:\Cyber\*.*')
```

2.7.4

הטכנאי הגיע למסקנה שאחד הקבצים אינו צריך להיות בתיקייה ויש למחוק אותו. צרו בשרת ובלקוח אפשרות להורות על מחיקת קובץ כלשהו, באמצעות פקודת DELETE, לדוגמה:

```
DELETE C:\Cyber\blabla.txt
```

למחיקת קובץ ניתן להשתמש במודול OS, לדוגמה:

```
import os
os.remove(r'C:\Cyber\blabla.txt')
```

2.7.5

כעת הטכנאי שלנו רוצה להעתיק קובץ כלשהו לתיקייה עליה הוא עובד. הקובץ המבוקש נמצא בתיקייה אחרת במחשב אליו מתבצעת ההעתקה. הוסיפו תמיכה בפקודת COPY (לדוגמה: העתק את הקובץ C:\Cyber\1.txt אל C:\Cyber\2.txt). אין הכוונה לשליחת הקובץ אל הלקוח, אלא לביצוע הפעולה על השרת בלבד. במקרה זה, השרת יחזיר ללקוח האם הפעולה הצליחה או לא. לדוגמה:

```
COPY C:\Cyber\1.txt C:\Cyber\2.txt
```

להעתקה של קבצים, ניתן להשתמש במודול **shutil**.

לדוגמה, להעתקת הקובץ C:\1.txt אל C:\2.txt, ניתן להריץ:

```
import shutil
shutil.copy(r'C:\1.txt', r'C:\2.txt')
```

2.7.6

הטכנאי שלנו רוצה לבדוק שהתוכנה עובדת עכשיו היטב. תמכו בפקודת EXECUTE אשר תגרום להפעלת תוכנה אצל השרת (לדוגמה – הרצה של תוכנת Word). במקרה כזה, על השרת להגיב ללקוח האם הפעולה הצליחה או נכשלה.

```
EXECUTE notepad.exe
```

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

לדוגמה, כדי להריץ את notepad, נוכל לבצע:

```
import subprocess
subprocess.call('notepad')
```

נשים לב שלעיתים נצטרך לתת את ה-path המלא של קובץ ההרצה שאנו רוצים להריץ[10], למשל כך:

```
subprocess.call(r'C:\Windows\notepad.exe')
```

2.7.7

לבסוף הטכנאי שלנו רוצה להודיע לשרת לסגור את החיבור. תמכו בפקודת EXIT, שתגרום לשרת לסגור את הסוקט מול הלקוח

טיפים לפתרון התרגיל

מספר דברים שכדאי לשים לב אליהם:

1. שימו לב, שכאשר מבצעים בפייתון socket.send(message) עבור message ריק, לא מתבצעת שליחה כלל. כלומר לא ניתן להסתמך על קבלת מסר ריק בלקוח כדי לדעת שהשרת סיים את שליחת הקובץ.
2. שימו לב שבפייתון, על מנת לייצג מחרוזת שכוללת את התו backslash ("\"), נצטרך לכתוב את התו פעמיים – כלומר "\\". זאת היות שהתו backslash יכול להיות חלק מתווים מיוחדים, כגון "\n" שמסמל התחלה של שורה חדשה. לחלופין, ניתן להשתמש באות r לפני הגדרת המחרוזת. שימו לב לדוגמה הבאה:

```
>>> not_what_we_mean = "C:\file.txt"
>>> not_what_we_mean
'C:\x0cile.txt'
>>> what_we_mean_1 = "C:\\file.txt"
>>> what_we_mean_1
'C:\\file.txt'
>>> what_we_mean_2 = r"C:\file.txt"
>>> what_we_mean_2
'C:\\file.txt'
>>> what_we_mean_1 == what_we_mean_2
True
```

3. אם אתם מכירים את השימוש ב-Exceptions בפייתון, מומלץ להשתמש בהם.

[10] דבר זה נכון כאשר קובץ ההרצה אינו נמצא במשתנה הסביבה Path של מערכת ההפעלה. תוכלו לראות את הסרטון הבא כדי להוסיף תוכניות ל-Path במידה ותרצו:

data.cyber.org.il/networks/videos/adding-python-to-path.html