לפני כן, נסביר מעט יותר על המימוש של **root directory**. כפי שציינו קודם, הפנייה למשאב מתבצעת כמו במערכת קבצים – פנייה ל-"/folder_a/folder_b/file_name.txt" למעשה פונה לקובץ file_name.txt בתיקייה folder_b שבתיקייה folder_a. אך היכן נמצאת תיקייה folder_a? היא נמצאת בתיקיית השורש, root directory, של האתר. בפועל, התיקייה הזו היא פשוט תיקייה כלשהי במחשב שהמתכנת הגדיר אותה בתור root directory. בחירה נפוצה היא להגדיר את C:\wwwroot בתור ה-root directory.

כעת, כאשר הלקוח פונה ושולח בקשה מסוג:

GET /folder_a/folder_b/file_name.txt HTTP/1.1

הבקשה תתבצע למעשה לקובץ הבא אצל השרת[23]:

C:\wwwroot\folder_a\folder_b\file_name.txt

תרגיל 4.4 – כתיבת שרת HTTP

לאחר כל אחד מהשלבים הבאים, הקפידו לבדוק את השרת שלכם על ידי הרצה של התוכנית, ושימוש בדפדפן כלקוח; היזכרו במשמעות של הכתובת 127.0.0.1 אותה הזכרנו בפרק תכנות ב- Sockets – הכתובת שאליה נתחבר באמצעות הדפדפן תהיה http://127.0.0.1:80 (כאשר 80 הוא הפורט בו נשתמש). על מנת לבדוק את הפתרון שלכם, אנו ממליצים להוריד אתר לדוגמה מהכתובת: data.cyber.org.il/networks/webroot.zip. העתיקו את תוכן קובץ ה-ZIP אל תיקייה כלשהי (כמובן שיש לפתוח את הקובץ) והשתמשו בה בתור ה-root directory שלכם. המטרה היא שהשרת ישלח ללקוח את index.html ויתמוך באפשרויות השונות שיש בעמוד אינטרנט זה.

1. כתבו שרת המחכה לתקשורת מהלקוח בפרוטוקול TCP בפורט 80. לאחר סגירת החיבור על ידי הלקוח, התוכנית נסגרת.
2. הוסיפו תמיכה בחיבורים עוקבים של לקוחות. כלומר, לאחר שהחיבור מול לקוח נסגר, השרת יוכל לקבל חיבור חדש מלקוח.
3. גרמו לשרת לוודא כי הפקטה שהוא מקבל היא HTTP GET, כלומר – ההודעה שהתקבלה היא מחרוזת מהצורה שראינו עד כה: מתחילה במילה GET, רווח, URL כלשהו, רווח, גרסת הפרוטוקול (HTTP/1.1), ולבסוף התווים \r ו- \n.
   - אם הפקטה שהתקבלה אינה HTTP GET – סגרו את החיבור.
4. בהנחה שהשרת מקבל בקשת HTTP GET תקינה ובה שם קובץ, החזירו את שם הקובץ המבוקש אל הלקוח. בשלב זה, החזירו את שם הקובץ בלבד, ולא את התוכן שלו.

[23] זאת בהנחה שהשרת מריץ מערכת הפעלה Windows. כאמור, במערכות הפעלה שונות ה-path עשוי להיראות בצורה שונה.

- שימו לב שב-Windows משתמשים ב-"\" כמפריד בציון מיקום קובץ, בעוד שבאינטרנט וגם בלינוקס משתמשים ב-"/".
- הערה: את שם הקובץ יש להעביר בתור שם משאב מבוקש, ולא ב-Header נפרד.
- הערה נוספת: בשלב זה, אל תעבירו Headerים של HTTP כגון הגירסה או קוד התגובה.

5. כעת החזירו את הקובץ עצמו (כלומר, את התוכן שלו).
   - אם מתבקש קובץ שלא קיים – פשוט סגרו את החיבור (היעזרו ב- os.path.isfile).
   - הערה: בניגוד לכמה מהתרגילים הקודמים, כאן יש לשלוח את כל הקובץ מייד, ולא לחלק אותו למקטעים בגודל קבוע (כפי שעשינו, למשל, בתרגיל 2.7).
6. הוסיפו את שורת התגובה ו-Headerים של HTTP:
   - גרסה HTTP 1.0.
   - קוד תגובה: 200 (OK).
   - השורה Content-Length: (מלאו בה את גודל הקובץ שמוחזר).
7. במקרה שבו לא קיים קובץ בשם שהתקבל בבקשה, החזירו קוד תגובה 404 (Not Found).
8. אם השרת מקבל בקשת GET ל-root (כלומר למיקום "/") – החזירו את הקובץ index.html (כמובן, וודאו שקיים קובץ כזה; תוכלו ליצור קובץ בשם index.html שמכיל מחרוזת קצרה, רק לשם הבדיקה).
9. אם השרת מקבל בקשות לקבצים מסוגים שונים, הוסיפו ל-Header של התשובה את השדה Content Type, בהתאם לסוג הקובץ שהתבקש. תוכלו להעזר בנתונים הבאים:
   - קבצים בסיומת txt או html:

Content-Type: text/html; charset=utf-8

   - קבצים בסיומת jpg:

Content-Type: image/jpeg

   - קבצים בסיומת js:

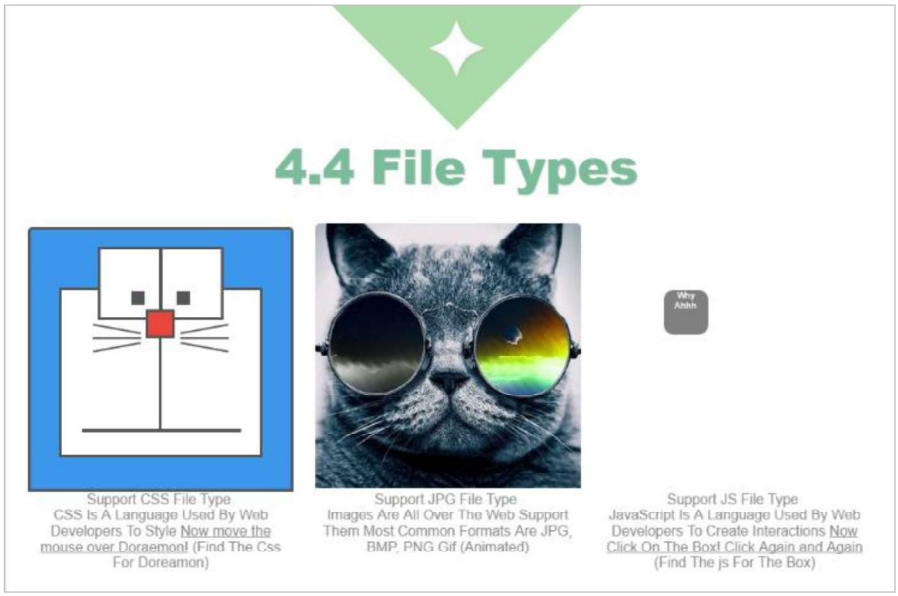
Content-Type: text/javascript; charset=UTF-8

   - קבצים בסיומת css:

Content-Type: text/css

10. כעת הוסיפו תמיכה במספר Status Codes נוספים (היעזרו בוויקיפדיה):
    1) 403 Forbidden – הוסיפו מספר קבצים שאליהם למשתמש אין הרשאה לגשת.
    2) 302 Moved Temporarily – הוסיפו מספר קבצים שהמיקום שלהם "זז". כך למשל, משתמש שיבקש את המשאב page1.html, יקבל תשובת 302 שתגרום לו לפנות אל המשאב page2.html. לאחר מכן, הלקוח יבקש את המשאב page2.html, ועבורו יקבל תשובת 200 OK.
    3) 500 Internal Server Error – במקרה שהשרת קיבל בקשה שהוא לא מבין, במקום לסגור את החיבור, החזירו קוד תגובה 500.

נסו את השרת שלכם באמצעות הדפדפן- גרמו לשרת לשלוח את המידע שנמצא ב-webroot ותוכלו לצפות באתר הבא:



*אתר בדיקת תרגיל שרת HTTP – קרדיט **תומר טלגם***

מדריך לכתיבה ודיבוג של תרגיל כתיבת שרת HTTP

קוד של שרת HTTP הוא קוד מורכב יחסית לתרגילים קודמים ולכן מומלץ לתכנן אותו מראש ולחלק אותו לפונקציות. יש דרכים רבות לכתוב את קוד השרת, תוכלו להתבסס על שלד התוכנית הבא:

http://data.cyber.org.il/networks/HTTP_server_shell.py

המקומות שעליכם להשלים בעצמכם נמצאים תחת הערה "TO DO". שימו לב שכדי שהתוכנית תעבוד תצטרכו להוסיף לה קבועים ופונקציות נוספות, השלד אמור רק לסייע לכם להתמקד.

טיפים לכתיבה

1. שימו לב שהתשובות שאתם מחזירים הם לפי כל השדות של HTTP. אל תפספסו אף סימן רווח או ירידת שורה...
2. לעיתים עלול להיות מצב שבו הן השרת והן הדפדפן מצפים לקבל מידע. כדי לצאת ממצב זה, ניתן להגדיר SOCKET_TIMEOUT. שימו לב שאם הזמן שהגדרתם עבר, תקבלו exception. עליכם לטפל בו כדי שהשרת לא יסיים את הריצה.