תרגיל שרת טכנאי
תרגיל 2.7 מהספר

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 יראו הפקודות והתשובות, ואין 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.

להלן הפקודות שעליכם לממש:

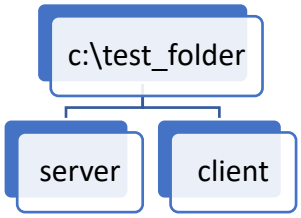
- **TAKE_SCREENSHOT** – צילום מסך השרת
- **SEND_FILE <file path>** - שליחת קובץ מהשת ללקוח
- **DIR <folder name>** - שליחת שמות הקבצים שנמצאים בתיקייה
- **DELETE <file path>** - שליחת שמות
- **COPY <file path> <folder name>** - העתקת קובץ cשרת לתיקייה בשרת
- **EXECUTE <program>** - ביצוע תוכנית על השרת
- **RELOAD** – import של קובץ הפעולות מחדש
- **EXIT –** יציאת השרת והלקוח
- **QUIT** - יציאת הלקוח

נשתמש בשרת ובלקוח אותם כתבנו בתרגיל הקודם ועל גבם נבנה את השרת והלקוח החדשים

זיכרו בכל שלב בתוכנית שלכם לבדוק שכל מה שעשיתם עד כה לא התקלקל.

לפתיחה

נשים לב שעל מנת שה testים יש להקפיד על שמות הקבצים ועל הפלט המדויקים.



- בתרגיל פעולות רבות על קבצים ותיקיות. כדי לנהל אותן צרו ב c:\ תיקיה בשם tets_folder ובתוכה צרו 2 תיקיות server ו client
- הקפידו בבקשה על שמות ומיקומים מדויקים למען ה test האוטומטי
- בתיקייה אחרת, שמרו עותק חדש של השרת והלקוח מתרגיל הקודם ועל גביו עבדו. בדקו שכשהלקוח שלכם יורד השרת סוגר את ה socket שלו בצורה מסודרת וממתין ללקוח חדש (עם accept) וכשהשרת יורד הלקוח יוצא בצורה מסודרת (בפקודה הראשונה לאחר ירידת השרת)
- קראו לקובץ השרת **server.py** ולקובץ הלקוח **client.py**
- לשרת וללקוח יש קבועים משותפים כגון port, אורך גודל ההודעה. העבירו אותם לקובץ נפרד בשם **constants.py** ויבאו את תוכן הקובץ האמצעות הפקודה:

  from constants import *

בדקו שוב שהכל עובד

נרצה להכין את השרת והלקוח שלנו לפקודות החדשות

שרת

- receive_client_request – לפקודות שלנו יש גם פרמטרים. לכן כשנקבל את הבקשה נחלק אותה לפקודה ולרשימת פרמטרים והפעולה תחזיר את שניהם

```
def receive_client_request(client_socket):
    # read from socket
    request = protocol.recv(client_socket)
    if request == ''
        return None, None
# split to request and parameters
    req_and_prms = request.split()
    if len(req_and_prms) > 1:
        return req_and_prms[0].upper(), req_and_prms[1:]
    else:
        return req_and_prms[0].upper(), None  # no parameters
```

פקודה

פרמטרים

פקודה

- handle_client_request – הפעם הפעולה מקבלת פרמטרים ומעבירה אותם לפעולות שב methods.py.

```
def handle_client_request(request, params):
    try:
        return getattr(methods, request)(params)
    except Exception as e:
…
```

- לא נשכח כמובן לקבל את הפרמטרים מהזמון של receive_client_request ולשלוח אותם ל handle_client_request

לקוח

השאירו ב valid_request רק את הבדיקה של exit וquit תכף נבדוק את חוקיות הפקודות החדשות. בדקו גם שאין להן פרמטרים

Methods

נמחק את הפעולות TIME RAND ו NAME ואת הקבועים ופקודות ה-import הקשורים אליהן ונשאיר את EXIT ו QUIT.

לא נשכח להוסיף את הפרמטרים לפעולות ב methods

בדקו שבפקודות QUIT ו EXIT עובדות

נוסיף את הפקודות אחת אחת...

TAKE_SCREENSHOT

כדי להבין מה מתרחש במחשב המרוחק, הטכנאי שלנו רוצה לקבל תצלום מסך של המחשב המרוחק.

עליכם לתמוך בכך בשרת ובלקוח.

תימכו בפקודה TAKE_SCREENSHOT, שתגרום לכך שהשרת יבצע צילום מסך וישמור את הקובץ במחשב השרת, במיקום לפי בחירתכם.

הדרכה...

לקוח

הוסיפו ל valid_request בדיקה שמאשרת את הבקשה TAKE_SCREENSHOT אם אין לה פרמטרים

METHODS

- שמרו את מיקום ושם קובץ צילום המסך כקבוע בראש הקובץ (methods.py). קיבעו את אותו להיות **c:\test_folder\server\screen.jpg**
- השתמשו במודול **PIL** לעבודה עם תמונות
אם אינו מותקן על המחשב שלכם ניתן להתקינו – פתחו חלון command line וכתבו:

Pip install Pillow

Import בראש הקובץ

```
from PIL import ImageGrab

im = ImageGrab.grab()
im.save('C:\\test_folder\\server\\screen.jpg')
```

- כתבו בקובץ methods.py את הפעולה TAKE_SCREENSHOT השתמשו בקוד הבא בכדי לצלם את המסך ולשמור את התמונה לקובץ (הקוד שומר את התמונה למיקום: c:\test_folder\server\screen.jpg), שנו אותו לפי הצורך:

שימו לב שגם שהפעולה הזאת כמו הפעולות שכתבתם קודם תחזיר את תוצאת הפעולה.
במקרה שלנו **"screenshot taken".**

הוסיפו main בקובץ methods ובדקו את הפעולה. אם היא עובדת בדקו את השרת והלקוח

בדקו שוב שהכל עובד

DIR

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 תיקיה מסויימת במחשב המרוחק. לדוגמה, הצגת רשימת הקבצים שבתיקייה C:\Cyber. תימכו בפקודת DIR- השרת ישלח ללקוח את התוכן של תיקיה 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\*.*')
```

methods

- נוסיף את הפעולה DIR שתחזיר את רשימת הקבצים בתיקייה - נשים לב:
- הפעולות שלנו ב methods מקבלות את רשימת הפרמטרים. יש לשלוף את שם התיקייה מהרשימה שלנו שעבור הפקודה DIR מכילה פרמטר אחד
- כי הפעולה glob מחזירה רשימה בעוד שאנחנו משתמשים במחרוזות (byte arrays) בין השרת ללקוח. לכן את נמיר את file_list שהחזירה הפעולה globe למחרוזת ונחזיר str(file_list). את המחרוזת הזאת נחזיר ללקוח ושם היא תודפס

לקוח

- נוסיף בדיקה ל valid_request שבודקת את חוקיות הפקודה dir שימו לב שהפעם עש לפעולה פרמטר

בדקו שוב שהכל עובד

טפלו בפקודות הבאות כפי שטיפלתם בקודמות. עבור כל אחת מהן הוסיפו בדיקה בלקוח ופעולה עבור כל אחת מהן ב methods.py שתזומן מ handle_client_request

DELETE

הטכנאי הגיע למסקנה שאחד הקבצים אינו צריך להיות בספריה 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')
```

במקרה של הצלחה נחזיר מהשרת ללקוח **file deleted**

בדקו שוב שהכל עובד

COPY

כעת הטכנאי שלנו רוצה להעתיק קובץ כלשהו לספריה עליה הוא עובד. הקובץ המבוקש נמצא בספריה אחרת במחשב אליו מתבצעת ההעתקה. הוסיפו תמיכה בפקודת COPY (לדוגמה: העתק את הקובץ C:\Cyber\file.txt לתיקייה C:\Cyber\folder). 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\folder

שם הקובץ יהיה מורכב משם התיקייה החדשה ושם הקובץ המקורי

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

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

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

במקרה של הצלחה נחזיר מהשרת ללקוח: **file copied**

בדקו שוב שהכל עובד

EXECUTE

הטכנאי שלנו רוצה לבדוק שהתוכנה עובדת עכשיו היטב. תימכו 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, בכדי להריץ את notepad, נוכל לבצע:

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

נשים לב שלעתים נצטרך לתת את ה-path המלא של קובץ ההרצה שאנו רוצים להריץ[1], למשל כך:

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

במקרה של הצלחה נחזיר מהשרת ללקוח **program executed**

בדקו שוב שהכל עובד

FILE_SEND

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 (chunks) ושילחו אותם ללקוח. המציאו פרוטוקול שיאפשר ללקוח לדעת שהשרת סיים לשלוח אליו את כל התמונה.

ניתן בסוף שליחת הקובץ לשלוח סימון שהיגענו לסוף. הגדירו קבוע מסוג byte array ותנו לו ערך מוסכם כלשהו לדוגמא b'-1' . בסוף שליחת הקובץ שלחו את הקבוע (כמובן הוסיפו את אורכו לפניו) אם בצד הלקוח קיבלתם b'-1' סימן שזהו סוף הקובץ ויש להפסיק לקבל chunks מהשרת.

את הקובץ שקיבלנו בלקוח נשמור בתיקייה שהוגדרה מראש אם השם שיש לו בשרת כלומר אם נגדיר מראש את התיקייה c:\clnt\files בצד הלקוח לאחר הפקודה send_file c:\screenshots\scrn.jpg יווצר בצד הלקוח הקובץ c:\client\scrn.jpg

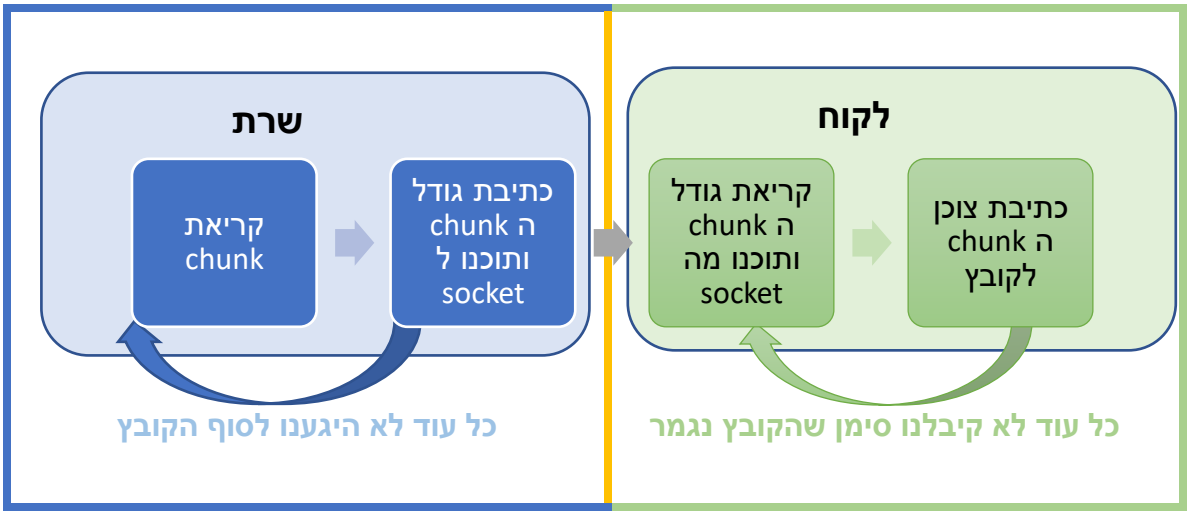
נעבוד עם הקובץ כמו שעבדנו בתרגיל copy-paste.

בשרת:

- נפתח את הקובץ
- בלולאה – כל עוד לא היגענו לסוף הקובץ:
  - נקרא 1024 בתים - chunk
  - נכתוב אותם ל socket
- נשלח סימן שהגענו לסוף הקובץ

בלקוח

- ניצור את הנתיב המלא לקובץ
- נפתח את הקובץ
- בלולאה – כל עוד לא קיבלנו סימון לסוף הקובץ
  - נקרא chunk מה socket
  - נכתוב אותו לקובץ



Protocol

נשים לב כי כשאנחנו קוראים וכותבים לקובץ אנחנו עובדים עם chunkים בינאריים ולא עם מחרוזות. לכן אין צורך ב encode ו decode כמו שעשינו עד עכשיו. הוסיפו לקובץ protocol את הפעולות send_bin ו recv_bin ששולחות ומקבלות data בינארית ולא מחרוזות ולכן לא עושות encode או decode

לקוח

- נעדכן את הפרוצדורה valid_request כך שתאשר גם את הפקודה SEND_FILE אם יש לה פרמטר אחד.

```
elif req_and_params[0] == 'SEND_FILE' and len(req_and_params) == 2
    return True
```

- נעביר לפרוצדורה handle_server_response גם את הבקשה וגם את ה socket.

def handle_server_response(my_socket, request):

בפרוצדורה handle_server_response נבדוק אם הפקודה שנשלחה מתחילה ב SEND_FILE.

נשים לב שהבקשה כולה מורכבת מפקודה ופרמטרים וצריך לפרק אותה לפני הבדיקה.

אם אכן הפקודה היא SEND_FILE, נזמן פעולה receive_file_request שתכתבו בקובץ methods

```
if request == 'SEND_FILE:'
    methods.receive_file_request(request, my_socket)
```

Methods - receive_file_request

נוסיף את הפעולה receive_file_request ל methods למרות שאינה מזומנת מהשרת. בהמשך גם השרת ישתמש בה

הפעולה מקבלת בלולאה הודעות מהשרת וכותבת אותן לקובץ. לפעולה הזאת נעביר את ה socket ממנו נקרא את הקובץ.

העזרו בתרגיל copy-paste שעשינו. הגדירו קבוע המכיל את שם התיקייה בה ישמרו הקבצים המתקבלים: RECEIVED_FILE_LOCATION = **"c:\\test_folder\\client"**

הכניסו למשתנה answer_file את שם הנתיב המלא בו תשמרו את הקובץ:

שם הקובץ המקורי + "\\" + RECEIVED_FILE_LOCATION

שימו לב לדייק בשם הקובץ בשביל ה test האוטומטי

```
def receive_file_request(request, my_socket):
    # here you generate answer_file name from request
    done = False
    with open(answer_file, "wb") as f:
        while not done:
            data = protocol.recv_bin(my_socket)
            if data == EOF:
                done = True
            else: # we write to file received data as is since we are in binary mode
                f.write(data)
```

Methods – SEND_FILE

- נוסיף פרוצדורה SEND_FILE שתקבל כפרמטר שם של קובץ. בפרוצדורה הזאת בלולאה נקרא קוד ונשלח אותו מהשרת ללקוח בחלקים בגודל 1024.
- נשים לב שהפרוצדורה צריכה לקבל שם של קובץ ואת ה socket אליו נרצה לכתוב. היות וכל הפרוצדורות המזומנות בעזרת getattr מקבלות את אותם הפרמטרים הוסיפו לכולן את הפרמטר sock ללא שימוש בו
- העזרו בתרגיל copy-paste לקריאת הקובץ בחלקים
- את הקובץ נפתח כקובץ בינארי ונקרא ממנו. היות ומקובץ בינארי אנחנו קוראים byte array ניתן לשלוח את הנתונים שקראנו מהקובץ ללא encode לכן נשתמש ב send_bin שהוספנו לפרוטוקול
- במקרה של הצלחה נחזיר מהשרת ללקוח לאחר העבזת הקובץ: **file sent**

שרת

- handle_client_request – אם קיבלנו פקוד SEND_FILE נזמן את הפעולה FILE_SEND ששולחת את הקובץ בחלקים בעזרת getattr. נשים לב ש SEND_FILE מקבלת כפרמטרים את שם הקובץ ואת ה socket ולכן שלחנו את ה socket לכל הפעולות. לכן גם handle_client_request שמזמנת אותן צריכה לקבל את ה socket

  def handle_client_request(request, params, sock):
- הפעולה SEND_FILE תחזיר לנו מחרוזת שאומרת אם הקובץ נשלח בהצלחה או לא. כמו בשאר הפעולות שעושה השרת נחזיר את המחרוזת הזאת מ handle_client_request
- אם ההודעה אינה תקינה ו get_attr נכשל במקרה של send_file עלינו לשלוח את הסימון לסוף קובץ שכן הלקוח מצידו מחכה לקבלת הקובץ

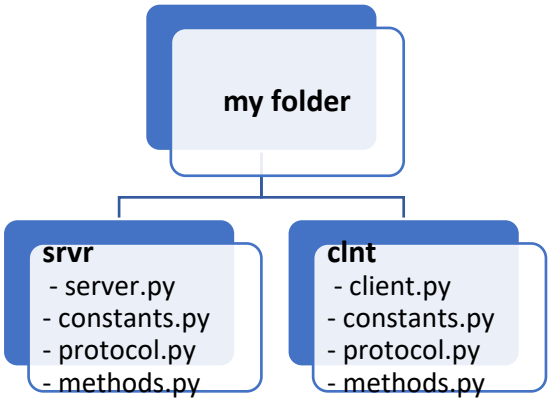
לכן נוסיף ל exception ב handle_client_reques שליחה של EOF במקרה ש SEND_FILE נכשל

```
def handle_client_request(request, params, client_socket):
    try:
        return getattr(methods, request)(params, client_socket)
    except Exception as e:
        print("handle_client_request", e)
        if request == 'SEND_FILE':
            send_response_to_client(EOF.decode(), client_socket)
        return "illegal command"
```

בדקו שוב שהכל עובד

Reload

נרצה לתקו בעיה שגילינו בקובץ methods ולטעון אותו מחדש. לשם כך ניצור פרויקט חדש ובו נחלק את השרת והלקוח ל 2 תיקיות, אחת בשם srvr עבור השרת ואחת בשם clnt עבור הלקוח.



בדקו שהשרת והלקוח בתיקיות החדשות עובדים

הלקוח

בצד הלקוח במידה והמשתמש ביקש RELOAD נשלח לשרת את הבקשה ואחריה את תוכן הקובץ methods.py. כדי להעביר את הקובץ נשתמש בפעולה SEND_FILE אותה נתאים לבקשה שלנו

Methods

שימו לב

הקובץ methods הוא משותף אבל שמנו אותו ב 2 תיקיות. כל שינוי יש לכתוב באחד הקבצים ולהעתיק לשני. **הפעולה reload עצמה דורסת את הקובץ שבתיקיית השרת** (מעתיקה את זה שבתיקית הלקוח לתיקיית השרת

ניצור פעולות send_file ו receive_file שיתאימו גם ל SEND_FILE וגם ל RELOAD

SEND_FILE – הפרידו את הפעולה ל-2 פעולות

- send_file – שמקבלת שם של קובץ ו socket ושולחת את הקובץ על ה socket
- SEND_FILE שמקבלת params ו socket שולפת את שם הקובץ מ params ומזמנת את send_file

receive_file_request – הפרידו את הפעולה ל 2 פעולות

- receive_file שמקבלת שם של קובץ ו socket קוראת מה socket וכותבת לקובץ
- receive_file_request – שמקבלת את הבקשה של SEND_FILE יוצרת בעזרתה את שם הקובץ שאליו נכתוב ומזמנת את receive_file

---

בדקו ש send_file עדיין עובד

RELOAD – הפעולה RELOAD תקבל את הקובץ בעזרת receive_file, תשמור אותו על methods.py ואז תטען אותו מחדש

טעינה מחדש של הקובץ בו אנחנו עובדים:

```
import importlib
import sys
importlib.reload(sys.modules[__name__])
```

שימו לב גם פה אם שליחת הקובץ מהלקוח נכשלת יש לשלוח EOF שכן השרת מחכה לו

על מנת לבדוק את ה RELOAD נכניס שינוי קטן לקובץ methods.py שבתיקיית הלקוח. החזירו למשל "screenshot taken**n**" במקום "screensot taken" מ SCREENSHOT_TAKE. הריצו RELOAD ואחריו הריצו TAKE_SCREENSHOT. מה קיבלתם? 😉

בדקו שוב שהכל עובד

שרת טכנאי – בדיקות

להלן רשימת הפקודות של תרגיל "שרת טכנאי" כשלכל אחת מצורף תיאור ורשימת בדיקות שצריכות להתבצע על ידכם לפני ההגשה

פקודה	תיאור	בדיקות
TAKE_SCREENSHOT	מצלמת תמונת מסך ושומרת בתיקייה ששמה נקבע מראש בקבוע. אם התיקייה לא קיימת או צילום המסך נכשל חוזרת הודעת שגיאה שמודפסת בצד הלקוח אם הכל בסדר חוזר screenshot taken ומודפס בצד הלקוח	• האם נשמרה תמונה ומודפס שם הקובץ • האם חזרה הודעת שגיאה והודפסה במידה שהתיקייה לא קיימת או במידה שהפקודה נכשלה

SEND_FILE	מקבלת שם של קובץ ושולחת אותו בחלקים ללקוח בצד הלקוח הקובץ יישמר בקובץ שמיקומו שמור בקבוע אם אחד הקבצים לא קיים או אם ההעתקה נכשלת מסיבה כלשהיא תצא הודעה למסך בסיום ההעברה תצא הודעה למסך	• האם השליחה הצליחה עבור קובץ קטן (chunk אחד) • האם השליחה הצליחה עבור קובץ גדול • האם כל הקובץ נשמר • האם יוצאת הודעה מתאימה כשהקובץ לא קיים • האם יוצאת הודעה מתאימה כשהתיקייה בצד הלקוח אינה קיימת
DIR	מקבלת שם של תיקייה בצד השרת ומעבירה לצד הלקוח את רשימת הקבצים שבה אם התיקייה לא קיימת או פעולת ה dir נכשלת תצא למסך הודעת שגיאה בצד הלקוח יודפסו שמות הקבצים	• האם פקודת ה dir הצליחה ומופיעים שמות כל הקבצים • האם כשהתיקייה אינה קיימת יצא הודעה מתאימה
DELETE	מקבל שם של קובץ בצד השרת ומוחק אותו. אם הקובץ לא קיים או אם הפעולה נכשלת יוצאת הודעת שגיאה אם הפעולה הצליחה יוצאת הודעה מתאימה	• האם פקודת המחיקה הצליחה • אם הקובץ לא קיים או הפעולה נכשלה האם יוצאת הודעה מתאימה
COPY	מקבל שם של קבץ ותיקייה ומעתיק את הקובץ לתיקייה אם הקובץ או התיקייה לא קיימים השרת מחזיר שגיאה שמודפסת בצד הלקוח אם השרת מצליח חוזרת הודעת הצלחה שמודפסת בצד הלקוח	• האם פקודת המחיקה הצליחה • אם הקובץ או התיקייה לא קיימים או הפעולה נכשלה האם יוצאת הודעה מתאימה
EXECUTE	מקבל שם של תוכנית או נתיב מלא אליה ומבצע אותה בצד השרת אם אינו עובד בצד השרת יוצאת הודעה מתאימה אם השרת מצליח חוזרת הודעת הצלחה שמודפסת בצד הלקוח	• האם מתבצעת התוכנית • האם כשניתן שם שלא קיים יוצאת הודעה מסודרת
EXIT	מתבצע ניתוק מסודר של השרת מהלקוח והשרת יורד	• האם מתבצעת יציאה מסודרת
QUIT	מתבצע ניתוק מסודר של השרת מהלקוח והשרת יורד	• האם מתבצעת יציאה מסודרת
RELOAD	מעבירה וטוענת את ה methods.py module אם העברת הקובץ או הטעינה נכשלת יוצאת הודעה למסך	• האם המודול עבר ונטען מחדש • אם שם המודול בקוד שגוי בשרת או בלקוח יוצאת הודעת שגיאה

הערות כלליות

- הפקודות ייכתבו **בדיוק** כמו במסמך על מנת להקל את הבדיקה ויבצעו בדיוק את המתואר לעייל
- בשום מצב אסור לשרת או ללקוח לעוף (exception)
- עבור כל פקודה שאינה חוקית צריכה לצאת הדפסה מתאימה
- בדיקה ראשונית תתבצע בצד הלקוח:
  - האם הפקודה קיימת
  - האם מספר הפרמטרים שלה נכון

אם הבדיקה נכשלה הלקוח ידפיס **illegal request**
בשרת נבצע exception handling מסביב לזימון של getattr. במידה ויהיה exception ישלח השרת ללקוח **illegal command** והלקוח ידפיס את התשובה

- אם השרת נופל/יוצא הלקוח אינו קורס כי אם יוצא בצורה מסודרת
- אם הלקוח נופל/ יוצא השרת אינו קורס
- הפרוטוקול יהיה מתועד בצורה בה הוא יובן מקריאת התיעוד
- הקוד יהיה מחולק לפרוצדורות שלכל אחת מהן יהיה תיעוד
- ואל תשכחו.... קבועים ו pep8

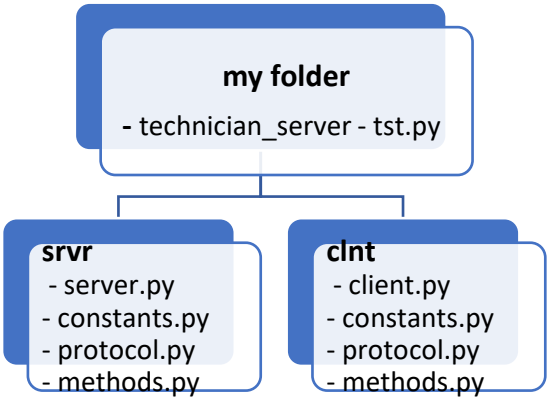
ה test ים האוטומטים

הרצת ה test

- הורידו את קובץ ה test מה moodle

  והעתיקו אותו לתיקיה שמעל התיקיות srvr ו clnt.
- הריצו את הטסט מתוך ה pycharm
- שימו לב הטסט מוחק את התיקייה test_folder

  בסיום הריצה. כדאי לשמור גיבוי שלה



על מנת שה test ים יעברו, עליכם להקפיד על התנאים הבאים:

	פקודה	test
1.	כללי	אם הפקודה לא קיימת או מספר הפרמטרים שלה שגוית הלקוח יגלה זאת וידפיס illegal request
2.	כללי	אם הפקודה חוקית ומספר הפרמטרים שלה נכון אבל היא לא יכולה להצליח (לדוגמא: קובץ/תיקייה לא קיימים) השרת יחזיר illegal command והלקוח ידפיס זאת

.3	כללי	בקשת ה input של הלקוח: Input(**'please enter a request '**)
.4	take_screeshot	את קובץ התמונה שימרו ב **"c:\test_folder\server\screen.jpg"**
.5	take_screeshot	החזירו מהשרת ללקוח והדפיסו בלקוח: **screenshot taken**
.6	dir	את רשימת הקבצים שחוזרים מהשרת ללקוח החזירו בפורמט רשימה שהומרה למחרוזת. לדוגמא: ['c:\\\\server\\\\f1.txt', 'c:\\\\ server\\\\f2.txt', 'c:\\\\ server\\\\f3.txt']
.7	copy	אם הפקודה הצליחה השרת יחזיר והלקוח ידפיס **file copied**
.8	delete	אם הפקודה הצליחה השרת יחזיר והלקוח ידפיס **file deleted**
.9	execute	אם הפקודה הצליחה השרת יחזיר והלקוח ידפיס **program executed**
.10	send_file	את הקובץ הנשלח ישמור הלקוי ב **c:\test_folder\client**
.11	send_file	אם הפקודה הצליחה לאחר העברת הקובץ השרת יחזיר והלקוח ידפיס **file sent**
.12	reload	אם הפקודה הצליחה לאחר העברת וטעינת הקובץ השרת יחזיר והלקוח ידפיס **module reloaded** בקובץ methods בתיקיית הלקוח הפעולה TAKE_SCREENSHOT תחזיר screenshot taken**n**

הגשה

צרו במודל 2 תיקיות srvr ו clnt והעלו לכל אחת את הקבצים שלכם. בקובץ ה methods בתיקיית הלקוח החזירו screenshot taken**n** במקום screenshot taken

בהצלחה