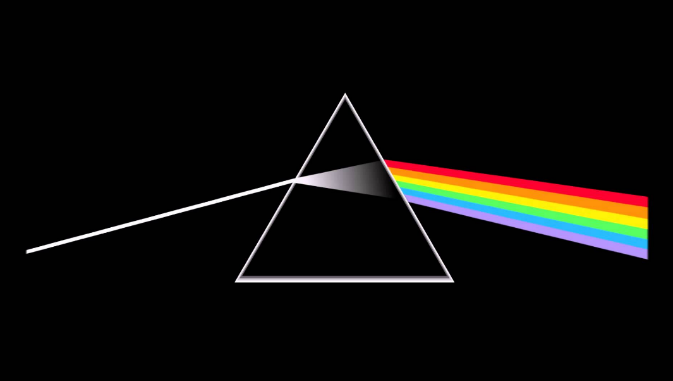
שיעור 4+5 – פרוייקט סוקטים

**פרוייקט סוקטים: הפינק פלוידר**

בפרוייקט זה שיתפרש על פני שני שיעורים נשלב את כל היכולות שרכשנו עד עתה בפייתון: מבני נתונים, עבודה עם קבצים וסוקטים, לפרוייקט אחד! במסגרת הפרוייקט תבנו שרת המספק מידע על הלהקה פינק פלויד ולקוח אשר מתשאל אותו.



1. **רגע, פינק מה?**

Pink Floyd הייתה להקת רוק בריטית שפעלה מסוף שנות השישים עד אמצע שנות התשעים של המאה ה-20. היא נחשבת לאחת הלהקות המשפיעות והמצליחות ביותר בהיסטוריה של המוזיקה הפופולרית, ואחת היחידות מז'אנר הרוק המתקדם שזכתה לפופולריות גם בקרב הקהל הרחב. האלבום המפורסם ביותר של הלהקה, "הצד האפל של הירח" (בתמונה: העטיפה האלמותית שלו, המציגה קרן אור העוברת דרך [מנסרה](https://he.wikipedia.org/wiki/%D7%9E%D7%A0%D7%A1%D7%A8%D7%94_(%D7%90%D7%95%D7%A4%D7%98%D7%99%D7%A7%D7%94))), הוא אלבום קונספט פילוסופי שעוסק בצדדיו האפלים של העידן המודרני ומקיף את מחזור החיים של האדם, מלידה ועד מוות. למרות שהוא מורכב משירים (הידוע מביניהם הוא [Money](https://www.youtube.com/watch?v=cpbbuaIA3Ds)), ניתן לשמוע את האלבום כרצועה אחת מתמשכת. [מומלץ להאזין בחושך :)](https://www.youtube.com/watch?v=mke0SNqWVCQ)

1. **תיאור צד השרת - server.py**

* מטרת השרת היא לספק מידע על להקת פינק פלויד. המידע כולל את כל האלבומים, השירים, אורך כל שיר, ומילות כל השירים.
* השרת עובד מול מאגר נתונים השמור בקובץ Pink\_Floyd\_DB.txt (אסור לשנות את הקובץ!)
* לאחר שמתחברים אליו, השרת ישלח הודעת Welcome קצרה ללקוח ויחכה לבחירת הפקודה של הלקוח.
* השרת יענה ללקוח לפי בקשתו ולאחר מכן יחכה לפקודה נוספת, עד אשר תשלח הודעת התנתקות מצד הלקוח.

1. **תיאור צד הלקוח - client.py**

* הלקוח יתחבר לשרת ע"פ כתובת ופורט שיהיו שמורים בקבועים בקוד.
* לאחר התחברות מוצלחת, הלקוח יציג בפני המשתמש תפריט של כל הפקודות הקיימות.
* לאחר בחירת הפקודה והקשת המידע, הלקוח יצור את ההודעה המתאימה וישלח לשרת.
* לאחר שהשרת יענה, הלקוח יציג למשתמש את התשובה שחזרה מהשרת.
* לאחר מכן יציג שוב את תפריט הפקודות ויחכה לפקודה נוספת מהמשתמש.

1. **רשימת הפקודות**

|  |  |  |
| --- | --- | --- |
| 1 | **רשימת אלבומים** | המשתמש מקבל את רשימת האלבומים. |
| 2 | **רשימת שירים באלבום** | המשתמש מקיש שם של אלבום ומקבל את רשימת כל השירים באלבום זה. |
| 3 | **קבלת אורך של שיר** | המשתמש מקיש שם של שיר ומקבל את אורכו. |
| 4 | **קבלת מילים של שיר** | המשתמש מקיש שם של שיר ומקבל את כל המילים. |
| 5 | **באיזה אלבום נמצא השיר?** | המשתמש מקיש שם של שיר ומקבל את האלבום שלו |
| 6 | **חיפוש שיר לפי שם** | המשתמש מקיש מילה ומקבל את שמות כל השירים שכוללים את המילה הזאת (\* אין חשיבות לגודל אותיות) |
| 7 | **חיפוש שיר לפי מילים בשיר** | המשתמש מקיש מילה ומקבל את שמות כל השירים שכוללים את המילה הזאת ב-Lyrics (\* אין חשיבות לגודל אותיות) |
| 8 | **יציאה** |  |

**\***  בפקודה בה אין חשיבות לגודל אותיות, גם אם המשתמש חיפש time ובשרת שמורה המילה כ-Time, השרת ימצא את התשובה.

**\*** בכל פקודות החיפוש (6,7) ימצאו גם חצאי מילים, למשל החיפוש “ti” ימצא את השיר time.

**\*** בפקודות 3-4 יש לכתוב את שם השיר במדויק.

1. **דגשים כלליים:**
   * השרת לא צריך לשלוח את התפריט ללקוח (הלקוח פשוט מדפיס אותו).
   * נסו לכתוב את הקוד כך שאין בו תנאי ארוך שבודק מהו סוג ההודעה בכל פעם. חשבו איך אתם יכולים להשתמש במילונים כדי לפתור בעיה זו.
   * כדי לבדוק את התוכנה, נריץ את השרת והלקוח על המחשב שלנו ונתחבר ל-127.0.0.1. הפורט – לבחירתכם.
   * תזכורת – כמה שפחות "מספרי קסם" בקוד – השתמשו בקבועים.
   * אם משהו לא עובד בתוכנה, נסו לבצע הדפסות של מה שאתם שולחים ומקבלים (רק זכרו להסיר את ההדפסות לפני ההגשה :).

**שלבי העבודה**

1. **שלב א' – מודול תקשורת**

**בזוגות, בכיתה עם דף ועט:**

* יש להגדיר את הפרוטוקול בו השרת והלקוח יתקשרו. יש לבנות פרוטוקול טקסטואלי. למעשה עליכם להגדיר כיצד נראית כל הודעה אשר השרת או הלקוח יכולים לשלוח. יש לשים לב שלכל הודעה יש קוד מסויים (כל הקודים יהיו באורך זהה) שמזהה אותה. ניתן להשתמש בקוד מספרי או טקסטואלי.
* יש לקבל את אישור המדריכ/ה לפני שעוברים לשלב הבא.

**🞀 מכאן והלאה כל אחד עובד על קוד משלו. אפשר להתייעץ ולהיעזר אחד בשני.**

**ביחידים, בכיתה ובבית:**

* + **צד השרת:**
    - יש לכתוב "קוד שלד" שמטפל בכל החלק התקשורתי, ללא מידע אמיתי. יש לכתוב שרת המקבל התחברות מלקוח, שולח הודעת Welcome ולאחר מכן מקבל הודעת בקשה. על השרת להבין איזו הודעה זאת (ע"פ קוד ההודעה), ולהחזיר תשובה דיפולטית לא אמיתית (שתוגדר בקבוע) עבור כל הודעה. בשלב זה השרת לא צריך להחזיר מידע אמיתי, רק להראות שהוא מזהה את סוג ההודעה.
    - על השרת לעבוד בלולאה כך שיוכל לקבל עוד בקשות מהלקוח מבלי לסגור את הסוקט, עד שהוא מקבל הודעה של Quit.
    - יש לוודא שהשרת לא קורס במקרה שלקוח מתנתק בצורה פתאומית. במקרה כזה הוא צריך לחזור ולהאזין. (כדי לבדוק סעיף זה, הריצו את הלקוח וסגרו אותו בצורה ברוטאלית עם Ctrl+C. אם השרת קרס – הרי שיש שגיאה שלא נתפסה).
    - השאירו את פיתוח האקספשנים לסוף, כדי שזה לא יפריע לכם בדיבוג הקוד שלכם.
  + **צד הלקוח:**
    - יש לכתוב לקוח אשר מתחבר אל השרת, מציג את הודעת ה-Welcome שקיבל, מציג למשתמש תפריט ושולח הודעה בהתאם לבחירת המשתמש.
    - יש לכתוב לולאה אשר מסוגלת לשלוח בקשות בצורה חוזרת, עד אשר המשתמש בוחר לצאת.
    - יש לוודא שהלקוח לא קורס במקרה שהשרת מתנתק בצורה פתאומית. במקרה כזה הוא צריך להציג הודעת שגיאה ולהיסגר.

**🞀 בסיום שלב זה כל חניך מגיש את השרת והלקוח שלו ב-ZIP לבדיקת המדריך ב-NEO. על השרת והלקוח לתפקד לגמרי מלבד העובדה שהשרת ישלח תשובות עם מידע לא אמיתי.**

1. **שלב ב' – מודול המידע**

**בזוגות, בכיתה עם דף ועט:**

* + יש להגדיר את מבני הנתונים הרצויים בהם יאוחסן כל המידע. המבנה יכול להיות רשימה, מילון, טאפל וכל שילוב שלהם (יכול להיות מבנה אחד או יותר, לעיתים נוח יותר לעבוד עם מספר מבנים). חשבו על השליפות שתצטרכו לבצע (בהתאם לרשימת הפקודות) ובנו מבנים שיהיה לכם נוח לשלוף מהם וידרשו כמה שפחות תהליכים של חיפוש. עם זאת, נסו למזער כמה שאפשר את שכפול המידע בין המבנים השונים.
  + יש לבדוק את השרת והלקוח של חברי הזוג זה מול זה (כלומר, לקוח של חניך א' מול שרת של חניך ב', ולהיפך). במידה ויש בעיה בתקשורת – על בני הזוג למצוא אותה ולתקן אותה.

**ביחידים, בכיתה ובבית:**

* + **צד השרת:**
    - בצד השרת יש לבנות מודול (ספריה) חדשה בשם אשר מטפל בכל הנושא של שאילתות המידע. המודול יכתב בקובץ נפרד בשם data.py וייובא בעזרת import לתוכנת השרת הראשית. על המודול לעסוק במידע בלבד; אין עליו לעסוק בתקשורת או בפלט/קלט למסך **בכלל.**
    - יש לבנות פונקציה אשר מפרסרת את קובץ המידע, כלומר קוראת אותו ומכניסה את כל המידע למבני הנתונים הפיתוניים (אחד או יותר, לפי התכנון שלכם). הפונקציה תחזיר את מבני הנתונים.
    - יש למקם את קובץ המידע בתיקיית התוכנה, כך שה-Path של הקובץ יהיה לא מלא   
      (כלומר FILE\_PATH = “Pink\_Floyd\_DB.txt” ולא FILE\_PATH=”C:\\Users……”).
    - שימו לב, אסור לכם לערוך את הקובץ עצמו!
    - יש לבנות פונקציה עבור **כל אחת** מסוגי השאילתות הקיימות. כל פונקציה תקבל מבני הנתונים ואת מילת החיפוש, ותחזיר את התשובה/תשובות המתאימות.
  + **צד הלקוח:**
    - על הלקוח להציג את התשובות החוזרות מהשרת בצורה הבאה:
      * אם יש תשובה אחת היא תיכתב כך: “Time”.
      * אם יש מספר תשובות, הן יוצגו עם פסיק ורווח ביניהם, בשורה אחת. למשל, אם ביצענו חיפוש לפי מילים, התוצאות יוצגו כך:  
        “Time, Wish you were here, Money” (ללא המרכאות).

**🞀 כדי לסיים שלב זה יש לבדוק את השרת והלקוח של חברי הזוג זה מול זה ולוודא שהם מתואמים ועובדים.**

1. **שלב ג' – אופציות מתקדמות**עליכם לבחור לפחות אופציה אחת מהרשימה הבאה של אופציות מתקדמות. תלמיד שיעשה יותר מאופציה אחת – יזכה לנק' בונוס בציון הפרוייקט. בתור בונוס ניתן גם לממש אופציות פרי דמיונכם – בתנאי שתתקבלו אישור על כך מהמדריכ/ה.
   * **Multi User**הפכו את השרת ל-Multi User – שרת אשר יכול לשרת בו זמנית מספר משתמשים. כלומר, גם כאשר נוצר סוקט שיחה חדש, השרת עדיין יכול להאזין לחיבורים חדשים ע"י סוקט ההאזנה ולטפל בהם גם (רמז: קראו על הפונקציה select. אין צורך להשתמש ב-Threads).
   * **לוגאין**הוסיפו מנגנון לוגאין במהלכו המשתמש מעביר ססמא בלבד. לאחר ההתחברות, השרת ישלח את ההודעה: “Please enter the password”. המשתמש יצטרך לכתוב את הססמא בלבד (בצורה לא מוצפנת). כדי לא לשמור ססמאות plain text על גבי השרת, הססמא תישמר בקבוע בקוד, בצורה Hashed בעזרת אחד מאלגוריתמי ה-hash הקיימים (MD5 / SHA). כאשר השרת יקבל את הססמא, הוא יבצע את פעולת ה-Hash וישווה את התוצאה לזו ששמורה אצלו בקבוע. במידה ויש התאמה – סימן שהמשתמש שלח את הססמא הנכונה והוא יכול להיכנס לשרת.
   * **סטטיסטיקות**
     + הוסיפו פקודה אשר מחזירה את רשימת 50 המילים הנפוצות ביותר מתוך כל שיריהם של פינק פלויד, וכמה פעמים כל אחת מהן מופיעה.
     + הוסיפו פקודה אשר מחזירה את רשימת האלבומים של פינק פלויד, מהארוך ביותר ועד לקצר ביותר (אורך כולל של שירים) ומדפיסה את אורכם.

🞀 **הגשה סופית של הפרוייקט, ביחידים, דרך ה-NEO.   
יש להגיש את 3 הקבצים ב-ZIP (client.py, server.py, data.py)**