SQLite3 & MongoDB – Python

היי תלמידים, בקובץ זה יהיו הסברים על מה הם מסדי נתונים, סוגים של מסדי נתונים ושימוש בשני סוגים של מסדי נתונים בפייתון. **עד סוף העמוד הראשון מוסבר על הגדרת מסד נתונים וסוגיו חלק זה אינו הכרחי להפעלת מסד נתונים אך ללמוד אותו רק יוסיף.**

**מאגרי מידע 🡨** כיום, אחד ממרכיבי הבסיס של העולם המודרני הוא מאגרי מידע. את השימוש בהם אנחנו מבצעים בלי שאנחנו שמים לב, למשל: בקנייה באינטרנט, משיכת כסף מכספומט, התחברות לאתר, רשתות חברתיות. לאור השימוש הרב במאגרי המידע היום כל ארגון ועסק משתמשים במערכות לניהול מסדי נתונים תפקידן של מערכות אלה הוא לתעד ולסדר את כל המידע שהן מכילות בכדי לאפשר לעבוד איתו בצורה נוחה.

**מסד נתונים 🡨** מסד נתונים (נקרא באנגלית Data Base ובקיצור הידוע DB) הוא אמצעי לאחסון של נתונים בצורה מסודרת למען אחזרום ועיבודם. מסד נתונים. מסד הנתונים בדרך כלל נשמר על גבי הדיסק הקשיח לשם גישה ישירה לנתונים. מסד הנתונים שימושי מאוד כשמדובר בחברות גדולות אשר רוצות מהנתונים השמורים להפיק דוחות, תרשימים וטבלאות לעומת מחשבים אישיים כאשר הם יכולים להסתפק ב-Excel. למשל גולש באינטרנט שרוצה לבדוק מהם המוצרים שיש לחברה מסוימת בקטגוריה ספציפית ומה המחיר שלהם מקבל זאת בקליק אבל מאחורי הקלעים מתבצעת שליפת נתונים מהמסד לפי הקטגוריה.

**מערכת לניהול מסדי נתונים (Database Management System) 🡨** מערכת זו הינה מערכת המסוגלת לעבד ולארגן מסדי נתונים. באמצעות תוכנה זו אפשר לגשת מהמחשב למסד נתונים ספציפי ולבצע עליו פעולות כגון: מחיקה, הוספה, אחסון, אחזור ושינוי נתונים. המערכת מתפקדת כממשק הקצה למסד נתונים ועוזרת לעיבוד של כמויות מידע גדולות.

**מסד נתונים רלציוני ((SQL 🡨** מסד נתונים רלציוני (נקרא גם יחסי או SQL Database) הינו סוג אחד של מסדי נתונים. סוג זה של מסד נתונים שומר את הנתונים בו בטבלאות כשכל טבלה מכילה מיידע על יישות כלשהי לכל שורה בטבלה יש מזהה ייחודי הנקרא מפתח וכל יחידת מידע נקראת שדה כאשר כל שדה מוגבל למספר תווים על מנת אימות אמינותו. למשל שדה המכיל תעודת זהות יוגבל ל-9 תווים וללא אותיות. במסד נתונים זה הקשר בין הטבלאות וסוג השדה נקרא סכמה. הקשרים בין הרשומות בטבלאות נעשים באמצעות שדה מפתח.

**SQL (Structure Query Language) 🡨** שפה לטיפול במסדי נתונים רלציונים המאפשרת ביצוע פעולות על מסד הנתונים כגון: קריאה, עדכון, מחיקה, הוספה ועוד... באמצעות שפה זו ניתן לבצע שאילתות, לעדכן ולארגן מחדש נתונים וגם יצירה ושינוי של המבנה (סכמה) של מסד הנתונים. SQL הינה שפה יחסית פשוטה עם מגוון קטן יחסית של פקודות המאפשרות ביצוע של כמעט כל פעולה על מסד הנתונים.

**מסד נתונים לא רלציוני (NoSQL) 🡨** בשונה ממסדי נתונים רלציונים מסדי נתונים שאינם רלציונים הם אינם טבלאיים ומאחסנים נתונים בפורמטים שונים. הייעוד של מסדי נתונים אלו הוא לטפל בכמויות גדולות של מידע שאינו מסודר בצורה טבלאית. הכוונה בכך היא שהמידע אינו מסודר לפי שיטה כלשהיא אלא מגוון מאוד בצורה שאי אפשר לסדר אותו בטבלאות. בעקבות הגמישות והגיוון בסוגי הנתונים אשר מאחסנים מסדים אלו הם הפכו לסוג מאוד פופולרי בקרב המון חברות כגון: גוגל, פייסבוק, אמזון וכו' שהמשותף לכל החברות אלו זה השימוש בכמויות עצומות של מידע. לסוג זה של מסדי נתונים קיימים ארבעה סוגים עיקריים: מסדי נתונים של מסמכים, מסדי נתונים של מפתח-ערך, מאגרי עמודות רחבים, ומסדי נתונים גרפיים. לכל אחד מהם ייעוד ומטרה משלו.

**מסדי נתונים של מסמכים ((Document Database 🡨** מסד נתונים זה הינו מסד נתונים עם מבנה מורכב של מידע הנקרא מסמך כשהמסמך מסתמן כמפתח ותוכן המסמך מסתמן כערך. כל מסמך מכיל זוגות של שדות וערכים. כאשר ערכים יכולים להיות: מחרוזות, מספרים, בוליאניים, מערכים ואובייקטים. מסדי נתונים כאלו בנויים בעיקר על מנת לבצע אחסון מידע של מסמכי JSON או XML.

**מסדי נתונים ערך-מפתח (Key-Value Database) 🡨** מסד נתונים זה דומה למילון בכך שהוא מאחסן בצורה של צמדי מפתוח-ערכים.

**מסדי נתונים גרפיים (Graph Database) 🡨** מסדי נתונים אלו משמשים להגדרת היחסים בין נקודות מאוחסנות. מסדי נתונים אלו מאחסנים נתונים בצמתים וקצוות. כאשר צמתים מאחסנים בדרך כלל מידע על אנשים, מקומות ודברים, והקצוות מאחסנות מידע על היחסים בין הצמתים. נשתמש בסוג זה של מסד נתונים כשנרצה להציג נתונים וניתחום באמצעות גרפים.

**מאגרי עמודות רחבות (Wide Column Database) 🡨** מסד נתונים זה מאחסן נתונים בצורת עמודות ולא שורות.

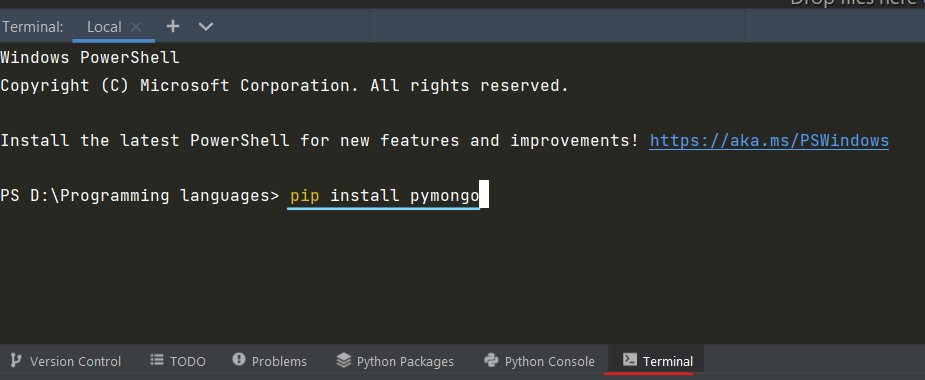
**חלק זה יסביר על עבודה של MongoDB בפייתון עד סוף עמוד 9.**

סרטון הסבר על התקנת מסד הנתונים במחשב [כאן](https://youtu.be/gB6WLkSrtJk?si=WACqYv0DadGFaeYy).

מסד נתונים זה הינו מסוג **NoSQL** בהמשך הקובץ יש הסבר גם על מסד נתונים **SQLite3 (עמוד 10).**

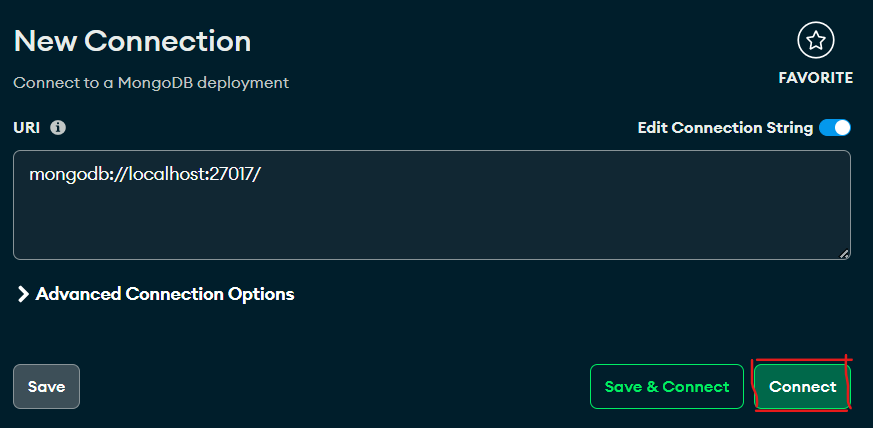
החלק הראשון בעבודה עם מסד נתונים זה הוא התקנת הספרייה באמצעותה ניתן לתקשר עם מסד הנתונים.

שם הספרייה הוא pymongo כך שההתקנה תיראה כך (כל צילומי המסך הם מתוך PyCharm)

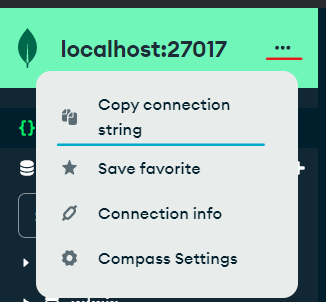


ראשית כל לחיצה על Terminal בתחתית המסך ולאחר מכן כתיבת הפקודה המסומנת בתכלת.

לאחר מכן נצטרך לדעת מהו המפתח שבאמצעותו נוכל להתחבר למסד. בשביל לקבל את זה נפתח את המסד וכשנפתח אותו יופיע לנו URI שהוא מייצג לנו חיבור חדש למסד נלחץ על Connect



לאחר מכן ייפתח לנו חיבור חדש ובצד שמאל למעלה יופיע לנו localhost: ואחרי הנקודתיים פורט כלשהו וליד הכתובת והפורט יהיו שלושה עיגולים צמודים, נלחץ עליהם ואז על Copy connection string



כעת, נתחבר למסד דרך הפייתון, ניכנס ל-PyCharm ונכתוב את הקוד הבא:

*from* pymongo *import* \* # Import the pymongo library  
  
DB\_connection = MongoClient("mongodb://localhost:27017") # Connect to the database

עכשיו אנחנו נמצאים בנקודה בה התחברנו, הדבר הבא אותו נצטרך לעשות הוא ליצור מסד נתונים. אחד היתרונות בשימוש עם MongoDB בפייתון הוא שאין צורך בהרבה מקרים להשתמש בפקודות מיוחדות על מנת ליצור דברים – אלא לנסות להתחבר לדבר מסוים ואם הוא קיים החיבור יתבצע ואם לא הדבר ייווצר ולאחר מכן יתבצע חיבור. למשל בדוגמא הבאה בה ניצור מסד נתונים שאיתו נעבוד בהסבר על MongoDB אנחנו לא נשתמש בפקודה מיוחדת ליצירת מסד נתונים אלא ננסה להתחבר למסד למרות שאינו קיים וכתוצאה מכך הוא ייווצר. (כל הדוגמאות על MongoDB יהיו על המסד שניצור עכשיו).

*from* pymongo *import* \* # Import the pymongo library  
  
DB\_connection = MongoClient("mongodb://localhost:27017") # Connect to the database  
  
clients\_DB = DB\_connection["Clients"] # Create database with the name "Clients"

כעת מה שעשינו זה כביכול לנסות לגשת לחיבור (DB\_connection) באינדקס Clients מה שמזכיר טיפה רשימות. מכיוון שזו פעם ראשונה שבה אנחנו מריצים את הקוד השורה שכתבנו עכשיו יוצרת **וגם** מתחברת למסד Clients ובפעמים הבאות בהן נתחבר למסד נכתוב בדיוק את אותו קוד ללא שינוי. עכשיו אם נלך לתוכנה הגרפית של ה-MongoDB נראה שהתווסף לנו מסד Clients (כמובן לאחר שנריץ את הקוד).

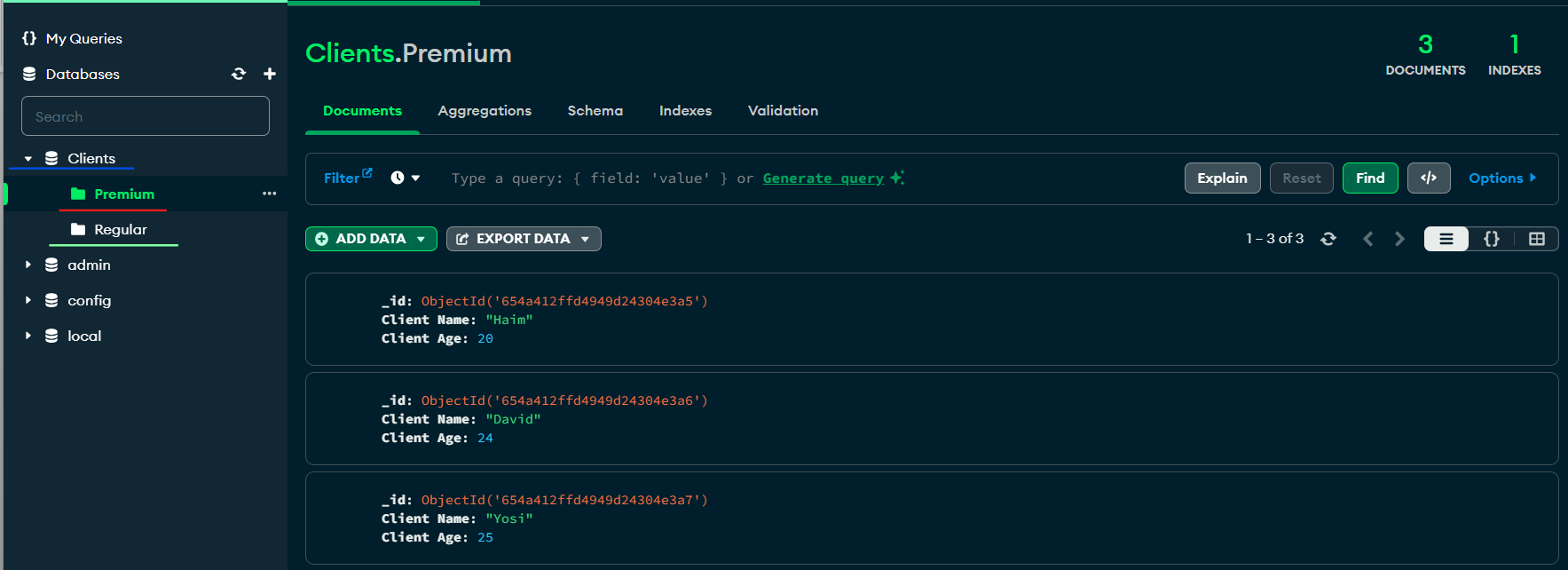
מי שינסה להריץ את הקוד יראה שלמרות שכתבנו יצירה של מסד נתונים המסד אינו נוצר בכלל, זאת מכיוון שמסדים אינם נוצרים ללא קולקציה ותוכן. לכן ניצור קולקציה בתוך המסד שהיא תכיל את כל הלקוחות שהם עם חשבון "Premium" והתוכן היה השמות והגיל שלהם. הקוד יראה כך:

*from* pymongo *import* \* # Import the pymongo library  
  
DB\_connection = MongoClient("mongodb://localhost:27017/") # Connect to the database  
  
clients\_DB = DB\_connection["Clients"] # Create database with the name "Clients"  
premiumClientsCollection = clients\_DB["Premium"] # Create collection with the name "Premium"  
  
premiumClientsList = [ # Create list of all the premium clients  
 {"Client Name": "Haim", "Client Age": 20}, # First client  
 {"Client Name": "David", "Client Age": 24}, # Second client  
 {"Client Name": "Yosi", "Client Age": 25} # Third client  
]  
  
premiumClientsCollection.insert\_many(premiumClientsList) # Insert all the client to the collection "Premium"

אפשר לראות שכמו שיצרנו מסד נתונים חדש כך יצרנו קולקציה חדשה עם המשתנה premiumClientsCollection בצורה שבה אנחנו כביכול ניגשים לאינדקס עם השם Premium וכך התכונה יודעת שאם הוא קיים היא תתחבר לקולקציה ואם לא היא תיצור **וגם** תתחבר לקולקציה. לאחר מכן יצרנו רשימה של כל הלקוחות עם המשתנה premiumClientsList שמשתנה זה הוא רשימה עם שלושה מילונים פשוטים כמו שנלמדים בקורס. בשורה האחרונה אפשר לראות שניגשו לקולקציה של הלקוחות שהם Premium והפעלנו על אותו משנה את הפונקציה insert\_many שכשמה כן היא – מכניסה לקולקציה כמה נתונים במכה ומכאן אפשר להבין שכמו שיש insert\_many שהיא מכניסה כמה במקביל יש גם פונקציה הופכית שהיא מכניסה משהו אחד במקביל ושמה הוא insert\_one שעושה את אותו תפקיד רק למשהו אחד. אפשר לראות [כאן](https://www.w3schools.com/python/python_mongodb_insert.asp) דוגמא (יש בקישור דוגמאות גם על insert\_many וגם על (insert\_one. כעת נסו ליצור בעצמכם קולקציה נוספת של לקוחות שהם "Regular" והוסיפו לקולקציה שלושה לקוחות חדשים שלכל אחד מהם שם וגיל כמו בדוגמא פה. פתרון בעמוד הבא.

*from* pymongo *import* \* # Import the pymongo library  
  
DB\_connection = MongoClient("mongodb://localhost:27017/") # Connect to the database  
  
clients\_DB = DB\_connection["Clients"] # Create database with the name "Clients"  
regularClientsCollection = clients\_DB["Regular"] # Create collection with the name "Regular"  
  
regularClientsList = [ # Create list of all the regular clients  
 {"Client Name": "Nitay", "Client Age": 17}, # First client  
 {"Client Name": "Yair", "Client Age": 20}, # Second client  
 {"Client Name": "Ilay", "Client Age": 30} # Third client  
]  
  
regularClientsCollection.insert\_many(regularClientsList) # Insert all the client to the collection "Regular"

כעת אם נפתח את התוכנה הגרפית של MongoDB נוכל לראות את כל מה שיצרנו עם כל הנתונים שהכנסנו.



כעת למדנו להתחבר וליצור, פעולה הכרחית נוספת היא חיפוש. פעולת החיפוש הכי בסיסית היא הפעולה שמביאה את האיבר הראשון בתוך קולקציה מסוימת ושמה הוא find\_one שכמו שמה אפשר להבין שהיא מוצאת משהו אחד. דוגמא:

*from* pymongo *import* \* # Import the pymongo library  
  
DB\_connection = MongoClient("mongodb://localhost:27017/") # Connect to the database  
  
clients\_DB = DB\_connection["Clients"] # Create database with the name "Clients"  
regularClientsCollection = clients\_DB["Regular"] # Create collection with the name "Regular"  
  
print(regularClientsCollection.find\_one()) # Print only the first index in the "Regular" collection

הקוד הזה יוציא רק את הלקוח הראשון שיש בקולקציה.

כמו שראינו, יש find\_one שמביאה לנו רק את הראשון, אבל מה אם נרצה את כולם? נשתמש בפונקציה ששמה הוא find ומכיוון שהיא מחזירה את כל הלקוחות היא מחזירה רשימה – מה שאומר שנפעיל את הפונקציה בתוך לולאת for . דוגמא:

*from* pymongo *import* \* # Import the pymongo library  
  
DB\_connection = MongoClient("mongodb://localhost:27017/") # Connect to the database  
  
clients\_DB = DB\_connection["Clients"] # Create database with the name "Clients"  
regularClientsCollection = clients\_DB["Regular"] # Create collection with the name "Regular"  
  
*for* regularClient *in* regularClientsCollection.find(): # Run on regular clients list  
 print(f"Client > {regularClient}") # Print the current client

כעת הצלחנו להדפיס, אבל מה אם נרצה להדפיס רק שדות ספציפיים? בשביל זה נוכל להוסיף תנאי בתוך הסוגריים של הפונקציה find התנאי יעבוד בצורה הבאה, הארגומנט הראשון שיהיה בתוך הסוגריים יהיה סוגריים מסולסלות ריקות (בהמשך נמלא גם שם ערכים) והארגומנט השני יהיה מילון שהתוכן בו יהיה: "שם השדה" : (1 או 0). מה שזה אומר שאם נרצה שדה מסוים נשים בערך המפתח את הערך 1 ואם לא נרצה נשים את הערך 0. למשל בדוגמא הבאה, ננסה להדפיס רק את השם והגיל וללא ה-id שבא כברירת מחדל עם כל אחד מהלקוחות:

*from* pymongo *import* \* # Import the pymongo library  
  
DB\_connection = MongoClient("mongodb://localhost:27017/") # Connect to the database  
  
clients\_DB = DB\_connection["Clients"] # Create database with the name "Clients"  
regularClientsCollection = clients\_DB["Regular"] # Create collection with the name "Regular"  
  
condition = {"\_id": 0, "Client Name": 1, "Client Age": 1} # The keys condition  
  
*for* regularClient *in* regularClientsCollection.find({}, condition): # Run on regular clients list  
 print(f"Client > {regularClient}") # Print the current client

בדוגמא הזאת הפרדתי את התנאי למשתנה נוסף בשם condition על מנת שיהיה מובן אך זה לא מחייב לכתוב בצורה הזאת ואפשר לכתוב את המילון ישירות בתוך הסוגריים.

דוגמא אחרונה לחיפוש תהיה חיפוש לפי ערך מסוים, למשל מי ששמו הוא יאיר. פעולה זו תיראה כך:

*from* pymongo *import* \* # Import the pymongo library  
  
DB\_connection = MongoClient("mongodb://localhost:27017/") # Connect to the database  
  
clients\_DB = DB\_connection["Clients"] # Create database with the name "Clients"  
regularClientsCollection = clients\_DB["Regular"] # Create collection with the name "Regular"  
  
findQuery = {"Client Name": "Yair"} # Query condition  
  
result = regularClientsCollection.find(findQuery) # Find all the clients by the condition  
  
*for* client *in* result: # Run on the list  
 print(f"Client > {client}") # Print the current client

אם נסתכל בקוד נראה שאת התוצאה הדפסתי בלולאה כאילו היא רשימה, השאלה שיכולה לעלות היא למה? התשובה לכך היא פשוטה, מכיוון שיכול להיות שכמה לקוחות יענו על תנאי מסוים אז אנחנו נקבל את **כל** מי שעונה על התנאי בתוך רשימה, במקרה הזה יש רק לקוח אחד שעונה על התנאי לכן הרשימה שנקבל מכילה ערך אחד אבל יכולים להיות מקרים בהם יהיו כמה לקוחות שייענו על התנאי. מומלץ לראות דוגמאות חיפוש מתקדמות יותר [כאן](https://www.w3schools.com/python/python_mongodb_query.asp).

כעת למדנו על התחברות, יצירה, הוספה, חיפוש ועכשיו נראה גם על מחיקה של דברים ומיון.

דבר ראשון נראה איך נוכל למיין נתונים מסוימים, למשל מיון השם לפי ה-ABC .

*from* pymongo *import* \* # Import the pymongo library  
  
DB\_connection = MongoClient("mongodb://localhost:27017/") # Connect to the database  
  
clients\_DB = DB\_connection["Clients"] # Create database with the name "Clients"  
regularClientsCollection = clients\_DB["Regular"] # Create collection with the name "Regular"  
  
sortedNames = regularClientsCollection.find().sort("Client Name") # Sort the names by the ABC  
  
*for* client *in* sortedNames: # Run on the list  
 print(f"Client > {client}") # Print the current client

אם נרצה לעשות את אותו דבר רק הפוך (כלומר מה שהכי קרוב ל-A יהיה אחרון ומה שהכי קרוב ל-Z יהיה ראשון) פשוט נוסיף לפונקציה sort לאחר שם השדה את הערך 1-.

*from* pymongo *import* \* # Import the pymongo library  
  
DB\_connection = MongoClient("mongodb://localhost:27017/") # Connect to the database  
  
clients\_DB = DB\_connection["Clients"] # Create database with the name "Clients"  
regularClientsCollection = clients\_DB["Regular"] # Create collection with the name "Regular"  
  
sortedNames = regularClientsCollection.find().sort("Client Name", -1) # Sort the names by the ABC (reversed)   
  
*for* client *in* sortedNames: # Run on the list  
 print(f"Client > {client}") # Print the current client

כעת נלמד איך למחוק לקוחות מסוימים לפי תנאי כלשהו, למשל מחיקה של מי שגילו הוא 20.

*from* pymongo *import* \* # Import the pymongo library  
  
DB\_connection = MongoClient("mongodb://localhost:27017/") # Connect to the database  
  
clients\_DB = DB\_connection["Clients"] # Create database with the name "Clients"  
regularClientsCollection = clients\_DB["Regular"] # Create collection with the name "Regular"  
  
queryCondition = {"Client Age": 20} # Condition to delete  
regularClientsCollection.delete\_one(queryCondition) # Delete by the condition

כמו שאפשר לראות פה מחקנו רק לקוח אחד אבל מה אם נרצה למחוק כמה? פשוט מאוד נשתמש בפונקציה בשם delete\_many . עוד דוגמאות של מחיקה מתקדמות יותר [כאן](https://www.w3schools.com/python/python_mongodb_delete.asp).

אם נרצה למחוק את הקולקציה עצמה נשתמש בקוד הבא:

*from* pymongo *import* \* # Import the pymongo library  
  
DB\_connection = MongoClient("mongodb://localhost:27017/") # Connect to the database  
  
clients\_DB = DB\_connection["Clients"] # Create database with the name "Clients"  
regularClientsCollection = clients\_DB["Regular"] # Create collection with the name "Regular"  
  
regularClientsCollection.drop() # Remove the collection

כעת למדנו גם את המחיקה והמיון, הפעולות האחרונות שנלמד הם: עדכון, הדפסה לפי הגבלה.

בשביל לעדכן נתונים של משהו ספציפי נשתמש בדוגמא הבאה, כשנרצה לעדכן גיל של לקוח מסוים:

*from* pymongo *import* \* # Import the pymongo library  
  
DB\_connection = MongoClient("mongodb://localhost:27017/") # Connect to the database  
  
clients\_DB = DB\_connection["Clients"] # Create database with the name "Clients"  
premiumClientsCollection = clients\_DB["Premium"] # Create collection with the name "Premium"  
  
currentQuery = {"Client Age": 20} # Current value of the client  
newData = {"$set": {"Client Age": 21}} # New data to update  
  
premiumClientsCollection.update\_one(currentQuery, newData) # Update the data

כמו שאפשר לראות ש-currentQuery הוא המשתנה שמכיל את הגיל הנוכחי של הלקוח והמשתנה newData הוא הערכים החדשים בצורה של מילון עם מפתח אחד ששמו הוא "$set" וערכו הוא מילון נוסף שהוא מייצג את הגיל החדש של הלקוח. פה עדכנו משהו אחד אפשר לראות דוגמאות לעדכון של כמה דברים [כאן](https://www.w3schools.com/python/python_mongodb_update.asp).

דבר אחרון שנלמד יהיה הדפסה לפי הגבלה, למשל אם קיבלנו רשימה של כמה לקוחות ואנחנו רוצים לקבל רק את השניים הראשונים נרשום את הקוד הבא:

*from* pymongo *import* \* # Import the pymongo library  
  
DB\_connection = MongoClient("mongodb://localhost:27017/") # Connect to the database  
  
clients\_DB = DB\_connection["Clients"] # Create database with the name "Clients"  
premiumClientsCollection = clients\_DB["Premium"] # Create collection with the name "Premium"  
  
allClients = premiumClientsCollection.find().limit(2) # Get only the two clients from the list  
  
*for* client *in* allClients: # Run on the list  
 print(f"Client > {client}") # Print the current client

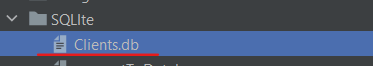
**חלק זה יסביר על עבודה של SQLite3 בפייתון.**

בחלק זה נלמד על SQLite3 שהיא ספרייה לעבודה עם שאילתות SQL בפייתון.

בעיקרון, היתרון בעבודה עם ספרייה זו הוא שחוץ מהתקנת SQLite3 לפייתון אין צורך להתקין תוכנה מיוחדת אבל אני כן ממליץ להוריד תוכנה שתוכל להציג את תוכן הקובץ שאיתו נעבוד בצורה נוחה.

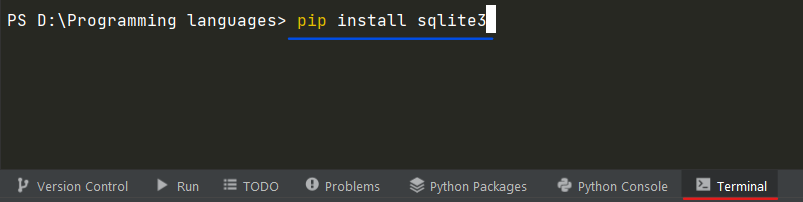
קישור להורדה [כאן](https://sqlitebrowser.org/blog/version-3-12-2-released/). למרות שדרך זו יותר נוחה כאן נשתמש אך ורק בפייתון.

בשלב הראשון בשביל לעבוד עם מסד נתונים כלשהו ניצור קובץ עם סיומת db.



אנחנו נעבוד כאן כל הזמן עם רק עם הקובץ הזה.

לפני שנתחיל לעבוד עם המסד בפייתון נצטרך להתקין את הספרייה sqlite3:



ראשית, נתחבר למסד על ידי הנתיב של הקובץ שיצרנו:

*from* sqlite3 *import* \* # Import the sqlite3 library  
  
connection = connect("./Clients.db") # Connect to the database by the path

לאחר שהתחברנו, נצטרך ליצור "סַמָן" שדרכו נריץ את הפקודות על המסד נתונים, דוגמא:

*from* sqlite3 *import* \* # Import the sqlite3 library  
  
connection = connect("./Clients.db") # Connect to the database by the path  
cursor = connection.cursor() # Create the cursor

כעת ניצור טבלה כלשהי, שהיא תכיל את כל הלקוחות שהם לקוחות "Premium":

*from* sqlite3 *import* \* # Import the sqlite3 library  
  
connection = connect("./Clients.db") # Connect to the database by the path  
cursor = connection.cursor() # Create the cursor  
  
# Create table command  
command = """  
CREATE TABLE PremiumClients (  
 name varchar(255) NOT NULL,  
 age int NOT NULL  
)  
"""  
  
cursor.execute(command) # Run the command  
connection.commit() # Commit the command

כמו שאפשר לראות התווסף משתנה בשם command המכיל את השאילתה שאותה אנחנו מבצעים על מנת ליצור טבלה. יוצרים טבלה ב-SQL בצורה הבאה: ראשית כל שימוש במילים CREATE TABLE שכשמו שאפשר להבין מהשם שלהם הם היוצרים את הטבלה, לאחר מכן יבוא השם במקרה שלנו PremiumClients ואחר כך סוגריים עגולות ובתוכן הערכים שהטבלה צריכה לקבל (אפשר להסתכל על זה בתור שבלונה שבאמצעותה ניצור בהמשך לקוחות), לטבלה שלנו יש שני מפתחות: name, age וכמו שאפשר לראות היצירה של מפתח נעשית באופן הבא: שם מפתח + סוג מפתח + הגדרות. למשל במפתח name השם של המפתח הוא name הסוג של המפתח הוא varchar(255) מה שאומר שהוא יכול לקבל תווים באורך 255 בתים מה שאומר שסוג המפתח הוא string וההגדרות של המפתח זה NOT NULL מה שאומר שמתי שנרצה ליצור לקוח חדש בטבלה הזאת נהיה חייבים להעביר ערך כלשהוא בתור המפתח הזה. ובסוף הקוד יש שימוש ב-" סַמָן" ובפונקציה שלו הנקראת execute שכשמה כן היא – מריצה את הפקודה ולאחר מכן הביצוע באמצעות commit.

עכשיו יש לנו טבלה המשמשת כשבלונה ללקוחות שהם Premium"" כעת ננסה ליצור כמה לקוחות לפי הטבלה הזאת:

*from* sqlite3 *import* \* # Import the sqlite3 library  
  
connection = connect("./Clients.db") # Connect to the database by the path  
cursor = connection.cursor() # Create the cursor  
  
# Insert into Premium Clients  
command = """  
INSERT INTO PremiumClients (name, age)  
VALUES ('Haim', 22), ('David', 20), ('Moshe', 30);  
"""  
  
res = cursor.execute(command) # Execute the command  
connection.commit() # Commit the command

כמו שאפשר לראות כאן השתמשנו בפקודה INSERT INTO שכמו שהיא אומרת תפקידה הוא להכניס ערכים לפי טבלה מסוימת (במקרה שלנו PremiumClients) ולאחר שם הטבלה יש את שמות המפתחות שאנחנו מעבירים וחשוב לשים לב לסדר שבו אנחנו כותבים את שמות המפתחות ולסדר של הערכים שאנחנו מעבירים.

למשל כמו שאפשר לראות פה כתבתי קודם כל name ואז age מה שאומר שמתי שאני אעביר ערכים באמצעות ה-VALUES אני אעביר לפי הסדר בו כתבתי לפני – קודם שם ואחר כך גיל. צריך לשים לב שלאחר המילה VALUES יבואו סוגריים עגולות שבתוך כל זוג יהיו ערכים שמייצגים לקוח מסוים (צבעתי בדוגמא למעלה כל זוג סוגריים שיהיה ברור). כעת נסו בעצמכם ליצור טבלה חדש של לקוחות רגילים ולהכניס לתוכה שלושה לקוחות חדשים כמו שעשינו כאן. פתרון בעמוד הבא.

יצירת הטבלה:

*from* sqlite3 *import* \* # Import the sqlite3 library  
  
connection = connect("./Clients.db") # Connect to the database by the path  
cursor = connection.cursor() # Create the cursor  
  
# Create table command  
command = """  
CREATE TABLE RegularClients (  
 name varchar(255) NOT NULL,  
 age int NOT NULL  
)  
"""  
  
cursor.execute(command) # Run the command  
connection.commit() # Commit the command

הכנסת הערכים:

*from* sqlite3 *import* \* # Import the sqlite3 library  
  
connection = connect("./Clients.db") # Connect to the database by the path  
cursor = connection.cursor() # Create the cursor  
  
# Insert values into Regular Clients  
command = """  
INSERT INTO RegularClients (name, age)  
VALUES ('Nitay', 17), ('Yosi', 22), ('Ilay', 20);  
"""  
  
res = cursor.execute(command) # Execute the command  
connection.commit() # Commit the command

בשביל לראות את התוכן של טבלה נשתמש בקוד הבא:

*from* sqlite3 *import* \* # Import the sqlite3 library  
  
connection = connect("./Clients.db") # Connect to the database by the path  
cursor = connection.cursor() # Create the cursor  
  
# View all the content from the table  
command = """  
SELECT \* FROM PremiumClients  
"""  
  
result = cursor.execute(command) # Run the command  
  
*for* line *in* result.fetchall(): # Run on the command result  
 print(f"Client > {line}") # Print the current client

בדוגמא פה נקבל את התוכן של לקוחות ה-Premium כמובן שעל מנת לראות את התוכן של הטבלה השנייה הפקודה תהיה זהה רק שם הפקודה אחרי המילה FROM תהיה RegularClients:

*from* sqlite3 *import* \* # Import the sqlite3 library  
  
connection = connect("./Clients.db") # Connect to the database by the path  
cursor = connection.cursor() # Create the cursor  
  
# View all the content from the table  
command = """  
SELECT \* FROM RegularClients  
"""  
  
result = cursor.execute(command) # Run the command  
  
*for* line *in* result.fetchall(): # Run on the command result  
 print(f"Client > {line}") # Print the current client

אם נרצה לקבל רק מפתח מסוים מהטבלה נצטרך לכתוב אחרי המילה SSELCT את שם המפתח למשל בדוגמא הבאה שתוציא רק את שמות הלקוחות:

*from* sqlite3 *import* \* # Import the sqlite3 library  
  
connection = connect("Clients.db") # Connect to the database by the path  
cursor = connection.cursor() # Create the cursor  
  
# Get specific key from table  
command = """  
SELECT name FROM PremiumClients  
"""  
  
result = cursor.execute(command) # Run the command  
  
*for* line *in* result.fetchall(): # Run on command result  
 print(f"Client > {line}") # Print current client

כמובן שאם נרצה להדפיס כמה מפתחות פשוט נוסיף פסיק אחרי שם המפתח הראשון ואחריו עוד מפתח:

*from* sqlite3 *import* \* # Import the sqlite3 library  
  
connection = connect("Clients.db") # Connect to the database by the path  
cursor = connection.cursor() # Create the cursor  
  
# Get some keys from table  
command = """  
SELECT name, age FROM PremiumClients  
"""  
  
result = cursor.execute(command) # Run the command  
  
*for* line *in* result.fetchall(): # Run on command result  
 print(f"Client > {line}") # Print current client

נוכל גם לחפש לקוח מסוים לפי תנאי כלשהו, למשל את הלקוחות שגילם גדול מ-20 . בשביל לעשות את זה נוסיף את הפקודה WHERE ולאחריה יבוא התנאי, למשל:

*from* sqlite3 *import* \* # Import the sqlite3 library  
  
connection = connect("./Clients.db") # Connect to the database by the path  
cursor = connection.cursor() # Create the cursor  
  
# View the content from the table by condition  
command = """  
SELECT \* FROM RegularClients WHERE age >= 20  
"""  
  
result = cursor.execute(command) # Run the command  
  
*for* line *in* result.fetchall(): # Run on the command result  
 print(f"Client > {line}") # Print the current client

כעת נסו בעצמכם להדפיס **רק** את השמות של הלקוחות שגילם גדול או שווה ל-20.

אם נרצה להדפיס לפי מיון נוסיף את המילים ORDER BY ולאחר מכן המפתח שאותו נמיין, למשל להדפיס את הלקוחות שהם Premium והדפסת השמות שלהם לפי ה-ABC:

*from* sqlite3 *import* \* # Import the sqlite3 library  
  
connection = connect("./Clients.db") # Connect to the database by the path  
cursor = connection.cursor() # Create the cursor  
  
# View the content from the table order by name  
command = """  
SELECT \* FROM PremiumClients ORDER BY name  
"""  
  
result = cursor.execute(command) # Run the command  
  
*for* line *in* result.fetchall(): # Run on the command result  
 print(f"Client > {line}") # Print the current client

כעת נסו להדפיס את הלקוחות שהם Regular ומיינו את ההדפסה לפי הגיל.

פתרון בעמוד הבא

*from* sqlite3 *import* \* # Import the sqlite3 library  
  
connection = connect("./Clients.db") # Connect to the database by the path  
cursor = connection.cursor() # Create the cursor  
  
# View the content from the table order by age  
command = """  
SELECT \* FROM RegularClients ORDER BY age  
"""  
  
result = cursor.execute(command) # Run the command  
  
*for* line *in* result.fetchall(): # Run on the command result  
 print(f"Client > {line}") # Print the current client

עכשיו נראה איך נמחק לקוחות לפי תנאי כלשהו, נשתמש במילה DELETE ואז את שם הטבלה והתנאי, למשל מחיקה של לקוחות Regular שגילם קטן מ-20:

*from* sqlite3 *import* \* # Import the sqlite3 library  
  
connection = connect("Clients.db") # Connect to the database by the path  
cursor = connection.cursor() # Create the cursor  
  
# Delete by condition  
command = """  
DELETE FROM RegularClients WHERE age < 20  
"""  
  
cursor.execute(command) # Run command  
connection.commit() # Commit command

עכשיו ננסה לעדכן נתונים, לפי תנאי מסוים, למשל מי שהוא לקוח Premium ושמו הוא דוד גילו יתעדכן מ-20 ל-21, באמצעות הפקודות UPDATE שאחריה יבוא שם הטבלה ו-SET שאחריה יבואו הערכים החדשים ולבסוף תנאי:

*from* sqlite3 *import* \* # Import the sqlite3 library  
  
connection = connect("Clients.db") # Connect to the database by the path  
cursor = connection.cursor() # Create the cursor  
  
# Update data  
command = """  
UPDATE PremiumClients  
SET age=21  
WHERE name='David'  
"""  
  
cursor.execute(command) # Run command  
connection.commit() # Commit command

כמובן שאפשר לבצע יותר מפעולה אחת אחרי ה-SET פשוט לשים פסיק ואחר כך להוסיף שינויים.