**הסבר קוד סקרייפינג**

הסבר כללי:

TMDB API (The Movie Database API) הוא ממשק נתונים פתוח המאפשר גישה למידע עשיר על סרטים, סדרות, שחקנים וטריילרים.

ה־API מספק מידע מובנה ועדכני, כולל תקצירים, דירוגים, תקציבים, הכנסות, פלטפורמות צפייה ועוד — לשימוש בפרויקטים של ניתוח מידע, אפליקציות ואתרים.

<https://developer.themoviedb.org/reference/movie-details>

הסבר כללי על API:

API (Application Programming Interface) הוא ממשק תקשורת שמאפשר לתוכנות לדבר אחת עם השנייה בצורה מסודרת ומבוקרת. אפשר לדמיין אותו כמו תפריט במסעדה: התפריט מציג לנו אילו מנות אפשר להזמין (כלומר, אילו שירותים זמינים), ואנחנו פשוט שולחים בקשה למלצר (ה־API), והוא מחזיר לנו את המנה (הנתונים).

במונחים טכניים, API הוא אוסף של כתובות (URLs) וכללים שמאפשרים לשלוח בקשות לשרת ולקבל ממנו תגובות — בדרך כלל בפורמט JSON. לדוגמה, אם אנחנו רוצים מידע על סרט מסוים, אנחנו שולחים בקשה לכתובת ספציפית עם מזהה הסרט, והשרת מחזיר לנו את כל המידע עליו.

בפרויקט שלנו השתמשנו ב־TMDB API כדי למשוך מידע עדכני על אלפי סרטים: כמו שם הסרט, תקציב, הכנסות, שפה, דירוגים, פלטפורמות צפייה ועוד. במקום להזין את המידע ידנית, שלחנו אלפי בקשות אוטומטיות ל־API וקיבלנו את המידע ישירות לקוד — מה שאפשר לנו לבנות קובץ נתונים מוכן לניתוח בצורה מהירה, מדויקת ואפקטיבית.

כדי להשתמש ב־API, בדרך כלל צריך לקבל מפתח גישה (API Key) — זהו קוד מזהה ייחודי שניתן למשתמש כדי לוודא שהשימוש בשירות נעשה בצורה מבוקרת, ושלא יישלחו יותר מדי בקשות בבת אחת. בפרויקט שלנו השתמשנו במפתח כזה כדי לגשת למידע מתוך TMDB בצורה חוקית ומוסדרת.

הסבר על הקוד:

1. יבוא ספריות 📚

* requests – לשליחת בקשות ל-TMDB API ולקבל תשובות
* pandas – כדי לאחסן ולעבד את המידע בטבלה (DataFrame)
* time – מאפשר לשים השהיה (sleep) כדי לא להעמיס על השרת

1. הגדרת פרטי API 🔑

api\_key - מפתח הגישה האישי ל־TMDB API. כמו מעין סיסמה לזיהוי מול השרת — כל פעם שנשלח בקשה ל־TMDB, המפתח מצורף כדי להראות שיש לנו גישה לשירות.

בעזרת המפתח הם יכולים לעקוב אחרי כמות הבקשות שלנו.

base\_url - זו הכתובת הראשית של ה־API של TMDB.

1. פונקציה שמביאה את כל פרטי הסרט 🎬

* הגדרת פונקציה לקבל פרטי הסרטים שנראה לי בהמשך בלולאה הראשית (while).
* הפונקציה מקבלת movie\_id (מזהה של סרט) ויוצרת כתובת URL בצירוף המפתח שלנו על מנת לשלוח אותה לשרת. בנוסף לבקשת פרטי הסרט נוספת בכתובת בקשה לקבלת טריילרים (videos) ופלטפורמות צפייה (watch/providers).
* כאשר אנחנו עובדים עם API, אנחנו שולחים לשרת בקשות (requests) כדי לקבל ממנו מידע. לכל בקשה כזו יש שיטת פעולה — לדוגמה: בקשת GET היא בקשה שנועדה "למשוך" מידע מהשרת, בלי לשנות כלום. אנחנו פשוט אומרים לשרת: "תן לי את כל הנתונים על הסרט הזה". השרת מחזיר תשובה, ובתוכה יש קוד סטטוס (status code) – מספר שמסביר מה קרה עם הבקשה.
* 200 - הבקשה הצליחה והתקבלה תשובה תקינה.
* 404 - מידע לא נמצא
* 500 - שגיאה בשרת.

בקוד שלנו אנחנו שולחים בקשות GET ל־TMDB API, ואם התשובה שקיבלנו תקינה (status 200), אנחנו שומרים את המידע שהתקבל בפורמט JSON — שהוא מבנה נתונים שמאפשר לנו לגשת בקלות לפרטי הסרט — וממשיכים הלאה.

אם הייתה תקלה כלשהי/ שלא התקבלה תשובה מהAPI, מוחזר NONE כדי שהקוד יוכל להתקדם מבלי להיתקע.

1. הגדרת משתנים התחלתיים 🧾

* max\_movies – הגדרת כמות הסרטים שנאסוף
* movies\_list – הגדרת רשימה שבה יישמרו כל הסרטים, כל אחד כמילון עם הפרטים שלו.
* movie\_ids – זוהי קבוצת מזהים (set), בה נשמור את ה־IDs של הסרטים שכבר נאספו. המטרה היא לוודא שלא נוסיף אותו סרט פעמיים (אם הוא מופיע ביותר מקטגוריה).
* page – מכיוון שה־API מחזיר את התוצאות בעמודים, נגדיר שמתחילים מהעמוד הראשון.
* movie\_fields – הגדרת הפרטים שנאסוף על כל סרט. חלק מהשדות הם מחרוזות רגילות, חלק הם פונקציות (lambda) שדורשות עיבוד מיוחד (למשל רשימות של שמות).

1. הורדת הנתונים מה-API (לולאת איסוף)

פונקציה 1: מעבר על קטגוריות ואיסוף הסרטים

הפונקציה אחראית על איסוף רשימת סרטים מתוך ארבע קטגוריות ב־TMDB API:

* פופולריים (popular)
* מדורגים גבוה (top\_rated)
* מוקרנים עכשיו (now\_playing)
* בקרוב (upcoming)

הפונקציה רצה בתוך לולאה עד שייאספו מספיק סרטים (לפי מה שהוגדר ב־max\_movies). בכל סבב, היא עוברת על ארבע הקטגוריות, ולכל קטגוריה היא בונה כתובת URL לפי מספר העמוד הנוכחי (page) ושולחת בקשת API מסוג GET.

אם הבקשה מצליחה (קוד תשובה 200), היא מקבלת אובייקט תגובה (response) וממירה אותו לפורמט JSON באמצעות response.json(), שמחזיר מילון. המידע החשוב שנמצא שם הוא רשימת הסרטים שנשלפה מתוך המפתח 'results'. עבור כל סרט ברשימה הזו, הפונקציה מיד קוראת לפונקציה add\_movie\_by\_id(), שמעבירה את מזהה הסרט (ID) ומחזירה את כל הפרטים המורחבים עליו.

הדבר החשוב הוא שאיסוף רשימת הסרטים והבאת הפרטים המורחבים מתבצעים יחד — מיד לאחר כל שלב של שליפת סרטים, ולא בהפרדה לשני שלבים נפרדים.

פונקציה 2: שליפת פרטי סרט לפי ID והוספה לרשימה

הפונקציה add\_movie\_by\_id() אחראית לאסוף פרטים של סרט אחד בכל פעם. היא מקבלת את מזהה הסרט (ID), בודקת אם הסרט הזה כבר נאסף, ואם לא — שומרת את ה־ID כדי למנוע כפילויות. לאחר מכן, היא שולחת בקשת API נפרדת כדי לקבל את כל הפרטים המורחבים על הסרט.

את השדות שאנחנו רוצים לשמור דרך המידע שמתקבל מה־API הגדרנו בmovie\_fields .

אם השדה דורש עיבוד (למשל: רשימת ז'אנרים שצריך לחבר למחרוזת אחת), מופעלת פונקציה מתאימה.

ואם זה שדה פשוט כמו כותרת של הסרט, הערך נלקח ישירות מהמילון.

בסיום, הפונקציה בונה מילון מסודר עם כל הנתונים של הסרט ומוסיפה אותו לרשימת הסרטים הכללית.

הערה: המשמעות של הפרמטר end='\r' בהדפסה הוא בכל פעם שנאסף סרט חדש, הפלט של ההדפסה נכתב מחדש באותה שורה במסך (כלומר דורס את ההדפסה הקודמת) כדי לקבל עידכון בלייב על כמות הרשומות שנאספו בצורה מצומצמת.

הסדר של שתי הפונקציות בקוד עצמו הפוך

1. שמירת הנתונים לקובץ CSV 📥

בסיום תהליך האיסוף, ממירים את רשימת הסרטים (movies\_list) לטבלת DataFrame בעזרת pandas — כך שכל סרט הופך לשורה, וכל פריט מידע (כמו שם הסרט, דירוג, ז'אנר וכו') הופך לעמודה.

לאחר מכן שומרים את הטבלה כקובץ CSV בשם all\_movies.csv, מבלי לכלול עמודת אינדקס מיותרת (index=False).