

## לי"ע Git מתקדם

לקראת פרויקט Arcade

כותב: ניר רהב



שלום לכולם!

במהלך הסדרה הקודמת הכרתם את הפקודות הבסיסיות של Git - יצרת שינויים (Add), שמירה שלהם (Commit), ושלילה לענן (Pull / Push).

עד עכשיו עבדתם בעיקר לבד על פרויקטים אישיים, אבל בעולם האמיתי - **רוב העבודה נעשית בצוותים.** בדיק בשביל זה נועדה הסדרה הזאת.

Git הוא לא רק כלי לשמירה על גרסאות - הוא כלי המאפשר תיעוד של הפרויקט, **שיתוף פעולה** חכם ועובדת במקביל של כמה מפתחים, **מעקב אחר שינויים** בפרויקט ובקרה על הקוד שלו, **גיבוי** לפרויקט וחזרה לגרסאות קודמות **במידת הצורך.**

במהלך הסדרה הקרויה תלמדו **פקודות מתקדמות** שיעזרו לכם לעבוד בצורה מקצועית יותר, להבין איך להתמודד עם טעויות, ולשלוט טוב יותר בפרויקטים שלכם.

### Clone

תהליך של ייצור עותק מקומי במחשב של repository אחר שנמצא ב-GitHub. כלומר, אנו יוצרים **עותק** מקומי **מלא** של הפרויקט שנמצא באותו repo כולל כל הקבצים, ההיסטוריה והענפים (נלמד זאת בהמשך) של אותו repository. לאחר שעשינו את העותק המקומי, ניתן לעבוד על הפרויקט במחשב האישי, לבצע שינויים ולשמור אותם באמצעות Commits. כיצד נבצע Clone לター המחשב שלנו?

נפתח את האתר של GitHub וניגש ל-[repository](#) שעליו נרצה לבצע את הפעולה. כעת קיימות שתי אופציות:

1. מדובר ב-[repository](#), חדש ولكن אין קבצים.
  2. קיימים קבצים ב-[repository](#).
- נתחיל עם אופציה מס' 2.

בחולנית של Quick setup נעתיק את הקישור המופיע - על ידי לחיצה על **כפתור העתקה**.

Quick setup — if you've done this kind of thing before

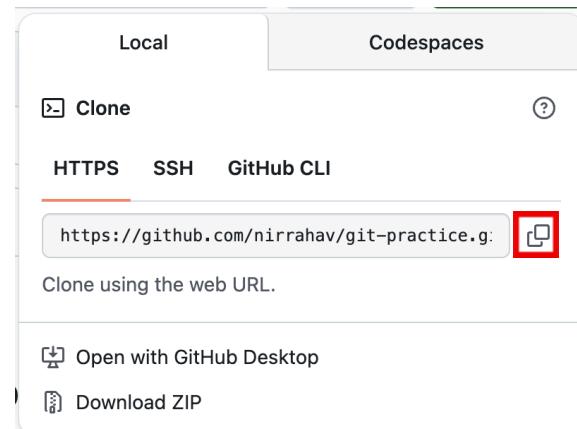
[ Set up in Desktop] or [[HTTPS](#)] [[SSH](#)] <https://github.com/nirrahav/git-practice.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

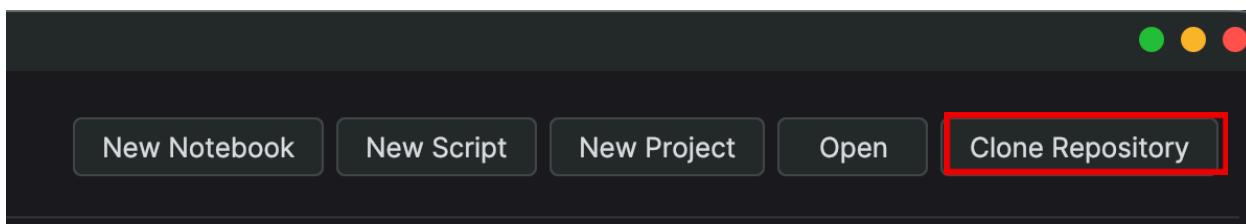
Code

במידה וקיים קבצים ב-repository, ניתן לנתקו – נלחץ על

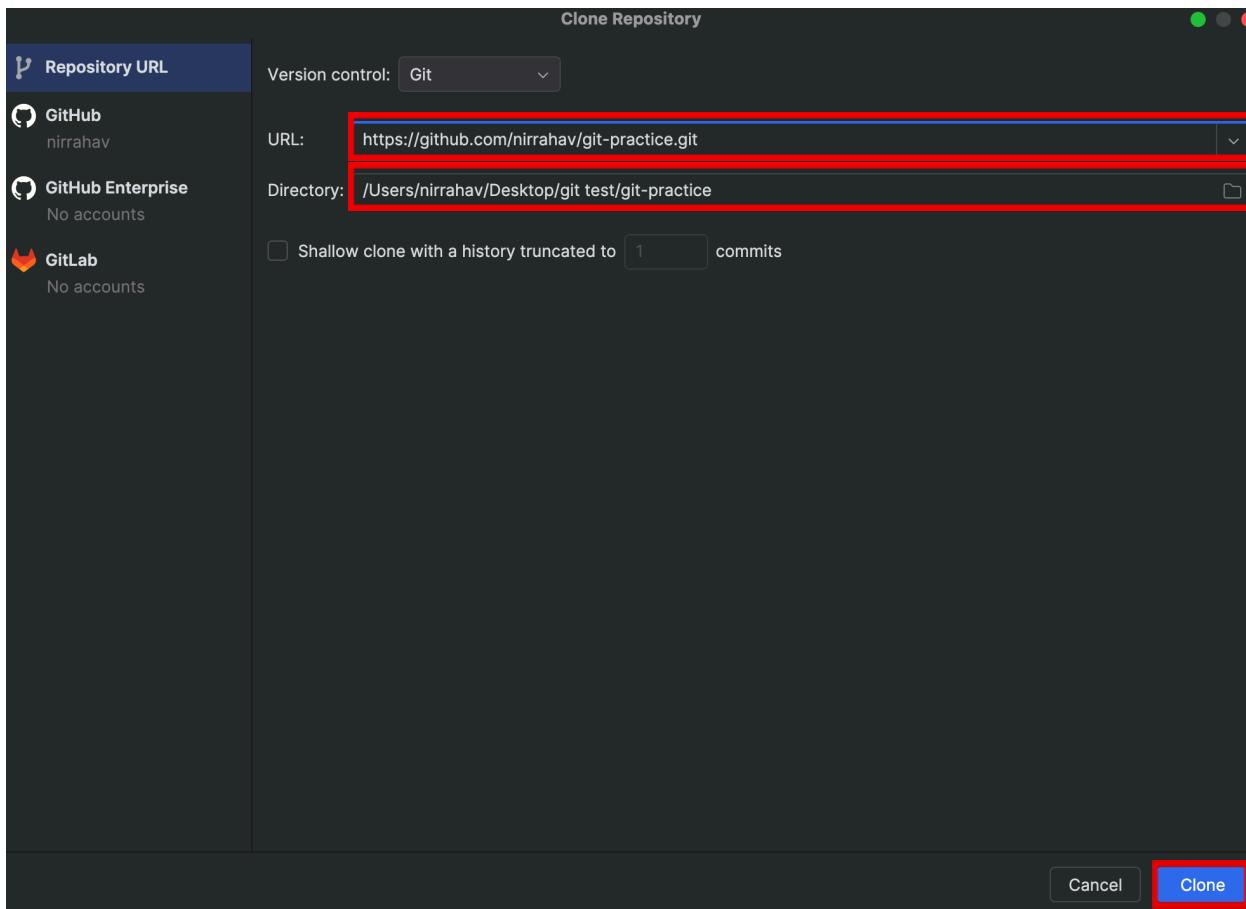
נעתיק את הקישור שקיים שם על ידי לחיצה על **כפתור העתקה**.



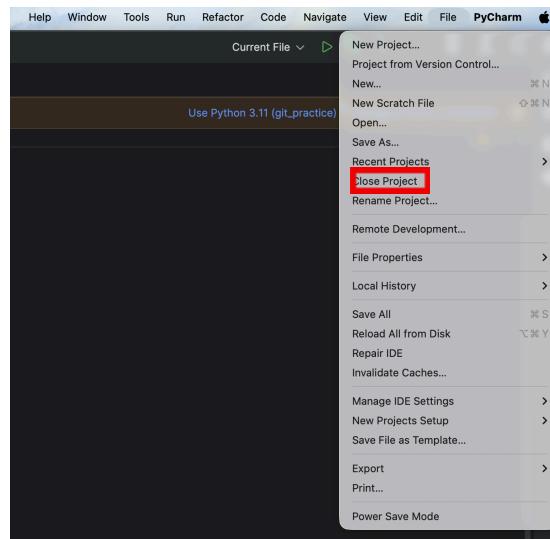
נפתח את PyCharm במחשב, ונלחץ על **Clone Repository** בעמוד פתיחה של PyCharm



בחולון שיפתח - הדיביקו את הקישור שקיבלתם, ובחרו את מקום התיקייה שבה תרצו לשמר את הפרויקט, ולאחר מכן לחצו על **Clone** בתחתית המסך מצד ימין.



לאחר שלוחצים על **Clone**, סביבת הפיתוח PyCharm תוריד למחשב שלכם את הקבצים הרלוונטיים, תיצור סדר מקומי?!, ופתח לכם את הפרויקט במיקום שבחרתם.  
שימוש ❤️ - בהנחה ונפתח פרויקט PyCharm אין מה להיכנס לחץ, לחצו **.File -> Close Project**



## קובץ requirements.txt

מה זה requirements.txt, ולמה בכלל צריך את זה?

כשעובדים על פרויקט python, כמעט תמיד משתמשים בספריות חיצונית למשל (flask, pandas, numpy).  
לכל מחשב יכולות להיות גרסאות שונות של אותן ספריות. ללא תיאום - הקוד יעבד אצלך, אבל ייפול אצל חבר/ה לצווית או במחשב בית-הספר.

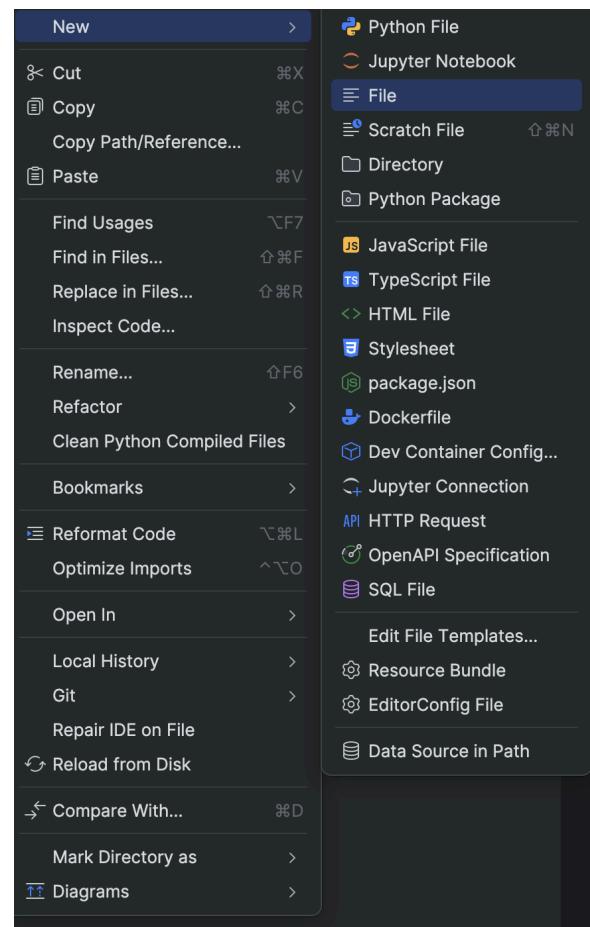
requirements.txt הוא מעין "רשימת קניות" של הפרויקט: הוא אומר למחשב לבדוק אילו ספריות צריך, ואילו גרסאות.

### זה פותר 3 בעיות אמיתיות:

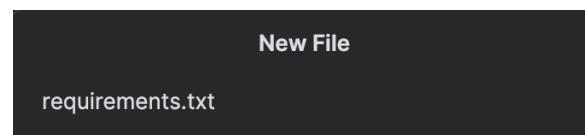
1. שחזוריות: כל אחד שמריד את הפרויקט יוכל להתקן אותו סע חבילות, וזה הקוד יתנהג אותו דבר.
2. שיתוף צוותי: ככל עובדים עם אותה סביבת עבודה ← פחות "אצלி זה לא עובד".
3. Deploy ובדיקות: בעזרת GitHub אפשר להציג תהליכי אוטומטיים שבודקים את הקוד, ומעלים אותו לשרת אמיתי באינטראנט לשימוש רחב.

### از איך עושים את זה?

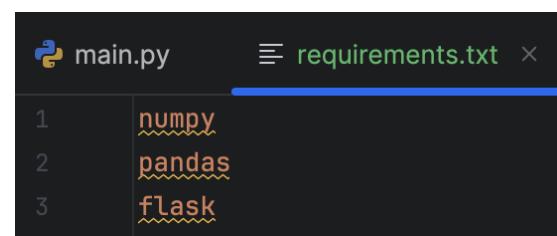
פתחים את הפרויקט ב-**PyCharm**, לוחצים מקש ימני בעכבר **File** -> **New** -> **File**



כעת נפתח קובץ חדש בשם requirements.txt (פתיחת הקובץ זהה לפתיחת קובץ python חדש).



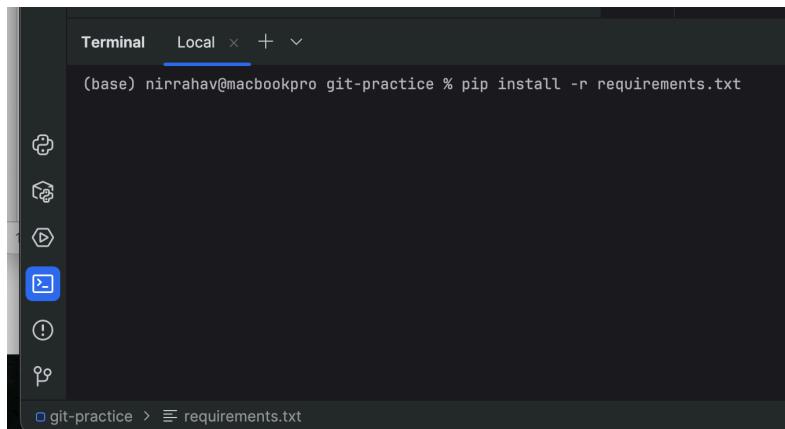
כעת לקובץ החדש נוסיף את כל הספריות אשר נutowד איתם בפרויקט שלנו.



**? חשוב:** איז איר חבר הוצאות שלנו מתקן את כל הספריות הנדרשות?

פשוט מאוד, נפתח את הטרמינל ב-**PyCharm** ונקתוב את הפקודה הבאה:

`pip install -r requirements.txt`



וכעת PyCharm יתקן לנו את כל הספריות הרלוונטיות!



הקוד הנadan בקישור לgist, משתמש בספרייה requests כדי לשלוח בקשה לאתר שמחזיר בדיחה רנדומלית.  
אם הכל עובד כמו שצריך – תראו בדיחה שモרכיבת משורה ראשונה (שאלת או פתיח) ושורת פאנץ' אחרונה.

**טיפ: ספריית requests** - ספרייה ששולחת בקשות לאינטרנט ומקבלת מידע מאתרים.

מטרת הקוד היא לא הלוגיקה עצמה, אלא לתרגל הורדה של פרויקט אמיתי, התקנת תלויות, והרצה בסביבה מקומית.

1. בצעו `Clone` ל-`.repository`.
2. התקינו את החבילות הנדרשות עם `pip install -r requirements.txt`.
3. הריצו את הקוד ובדקו שהוא מציג בדיחה מהאינטרנט.

לחצטו כאן על [הקישור לgist](#)



### ≠ הורדה רגילה Clone

- פעולה של clone לא רק מורידה קבצים, אלא **מחברת** את המחשב שלכם אל הפרויקט שבגן (`repository`).
- מהרגע שעשיתם `Clone` – אתם עובדים על עותק חי שמסyncר עם GitHub.

- Clone הוא חיבור מתמך המאפשר עדכנים דואים ברגע להורדת קז שזהו צילום רגעי של הקוד.

### גרסה מקומית מול גרסה בענן (GitHub)

- לאחר Clone, נוצר לכם **עותק מקומי** שלuproject על המחשב.
- אתם יכולים לשנות קבצים, לשומר (Commit) ולבודק את השינויים – אבל הם נשאים אצלכם מקומית.
- רק לאחר Push השינויים נשלחים ל-GitHub, ורק לאחר Pull תקבלו את העדכנים מהוצאות שתעבדו אותו.



### עבודה מסונכרנת עם Git

המטרה שלנו בעבודה עם Git היא **להישאר מסונכרנים** – כלומר שכל הוצאות יעבדו על אותו קוד מעודכן. כדי שזה יקרה, אנחנו משתמשים בשתי פקודות עיקריות:

1. `git pull` – אותה כבר למדנו בבדיקה Git הבסיסית.
2. `git fetch` – אותה נלמד עכשו.

### מה בעצם קורה כשהאנחנו מסתנכרנים?

יש לנו שני מאגרים:

- **Repository מקומי (Clone)** – זה הקוד שעליו אנחנו עובדים במחשב שלנו.
- **Repository מרוחק (GitHub)** – זה המאגר הראשי של הוצאות בענן.

אנחנו רוצים שני המאגרים האלה יהיו **באותו מצב**, ושנדע בכל רגע מה השתנה אצל אחרים.

### Pull – תזכורת

פקודה זו מביאה את כל השינויים מה-repository המרוחק וממזג אותם אוטומטית לתוך הקוד המקומי שלנו. במלחים אחרות – גם מעדכנת, וגם משלבת את השינויים ישיר לתוך הקבצים עצמם.

### Fetch

פקודה גם מביאה את השינויים מה-repository המרוחק, אבל, לא ממזג אותם מיד אל תוך הקוד שלנו. היא רק מאפשרת לנו לראות מה השתנה אצל אחרים לפני שאנחנו מחליטים לשלב את זה בעצמנו. זה נותן למפתח שליטה מלאה על תהליך העדכון – מתי ואיך למזג את הקוד החדש.

לסיכום:

- `git pull` – עשה שתי פעולות במקה אחת (`fetch + merge`).

- רק מביא את השינויים, בלי לגעת בקוד המקומי.

**לקריאה נוספת:**

אפשר להעמיק ולקראן עוד על הפקודה באתר הרשמי של Git

 <https://git-scm.com/docs/git-fetch>