

## תרגול Git בסיסי

מפגש 5/4 - חזרה על שנה א'

כותבות התרגיל: يولן לוי וטליה לכיש



### התקנות

1. אם עדין אין לכם - פתחו עכשוו משתמש ב-GitHub.

[יכנסו לקישור](#) ומלאו את הפרטים בשביול להורשים. הירשמו עם **המייל של ניצנים**.

2. [הורדת Git](#)

[יכנסו לקישור](#). לחצו על הטאב הראשון משמאל: **Windows**, ואז על: [Click here to download](#).

## Install

Latest version: 2.51.1 (Release Notes)

Windows    macOS    Linux    Build from Source

[Click here to download](#) the latest (2.51.1) x64 version of **Git for Windows**. This is the most recent maintained build. It was released **10 days ago**, on 2025-10-17.

3. קובץ ההתקנה **Git-2.51.1-64-bit.exe** יידל ממחשב שלכם (או הגרסה האחורונה עבור Windows).



4. לחזו על הקובץ כהורדות מסתimated, ועקבו אחר הוראות ההתקנה.

**שימוש ❤️** - המחשב שקיבלתם מ"ניצנים" מתאים להתקנת גרסה bit 64 Windows. אם אתם עובדים בנוסף בבית על מחשב אחר שמותקנת עליו מערכת הפעלה אחרת, למשל MacOS של Apple - בחרו במחשב זה בהורדת הגרסה המתאימה לו.

### תזכורת

- **Pull** – "מושך" (מוריד) את הגרסה המעודכנת ביותר מהRepository המקורי (שנמצא למשל ב-GitHub) אל המחשב המקומי.

- Add – מוסיף קבצים ל-area, Staging area, שהוא אזור-בנייה שבו בוחרים אילו שינויים ייכנסו ל-Commit הבא.
- Commit – שומר "צילום מצב" של הקוד ב-Repository המקומי, יחד עם הودעת תיאור קצרה (message) שמסבירת מה השתנה.
- Push – מעלה את כל ה-Commits שביצעו מהמחשב המקומי אל ה-Repository המרוחק בענן.

**קובץ .gitignore.** – הקובץ זהה מגדר ל-Git מיילו קבצים להטלים – כך שלא יהיו במעקב ולא יועלו ל-GitHub.

**למה צריך אותו?** יש קבצים שלא קשורים לשירות לפרויקט עצמו ולכן לא נרצה שייעלו ל-GitHub – למשל קבצי הגדרות אישיות של PyCharm (קובצי `idea`), קבצים זמינים Sh PyCharm יוצר בעצמו (`__pycache__` / `__pyc.*`), או קבצי מערכת. באמצעות `.gitignore`. אנו מונעים מקבצים מיותרים או אישיים להישמר בהיסטוריה השינויים בקוד או עלותRepository המשותף, כך שהוא נשאר נקי ומסדר ורക הקוד החשוב משותף לכל הוצאות.

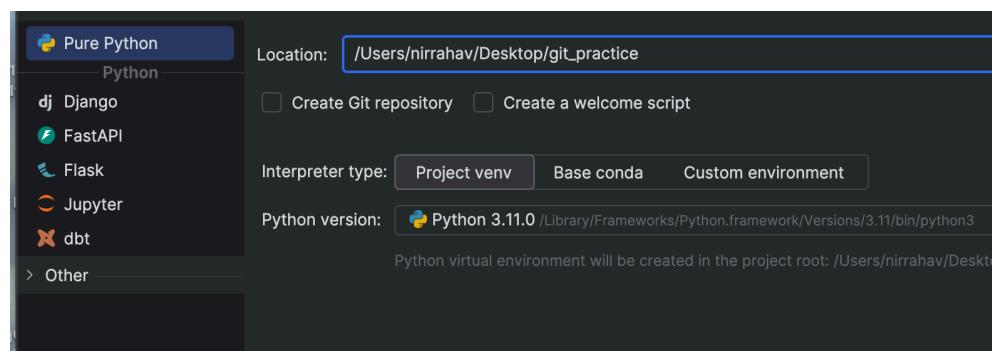
לרשומכם נספח **Git Cheat Sheet** ביחידה 00 | **תומכי למידה** בקורס.



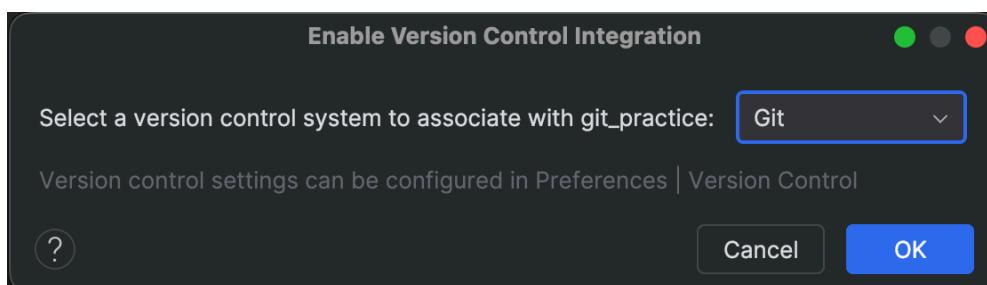
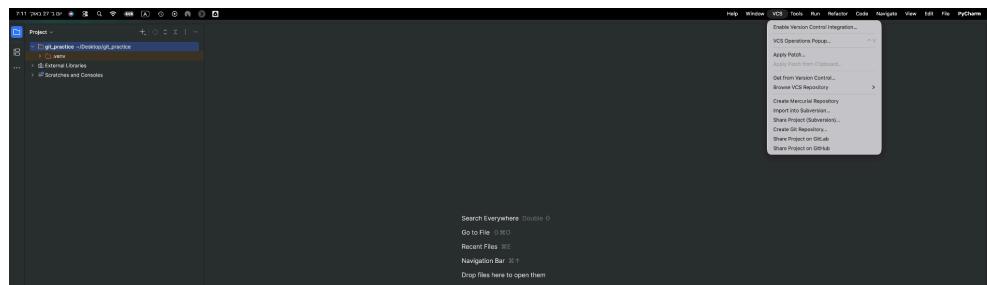
**שיםו ❤️** – כל השלבים מופיעים **במצגת המפגש**, אתם מוזמנים להיעזר בה.

### הפרויקט הראשון שלכם ב-GitHub (: GitHub)

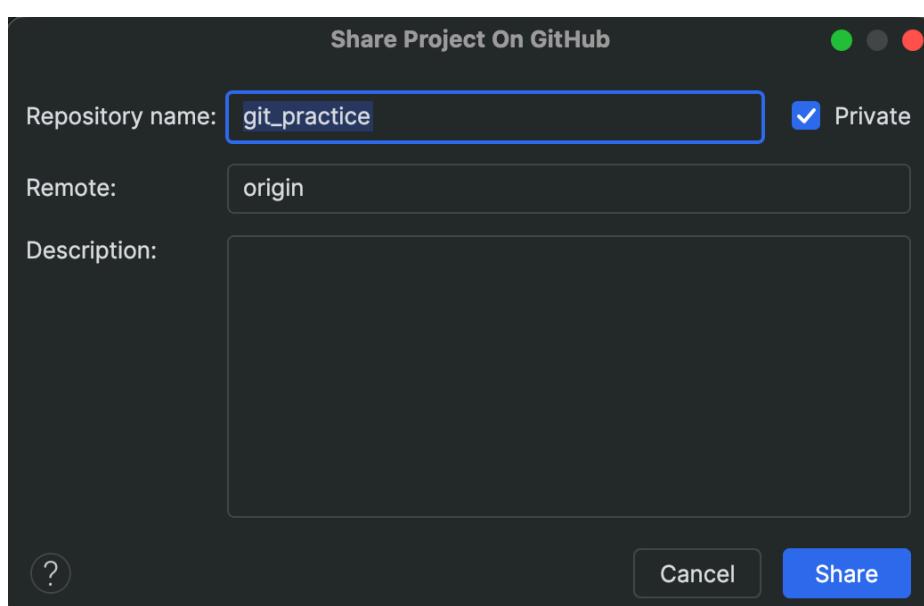
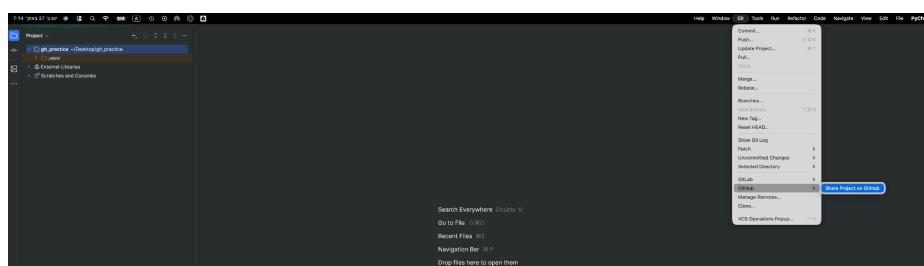
1. **פתחו** פרויקט חדש ב-PyCharm בשם `git_practice` בשם



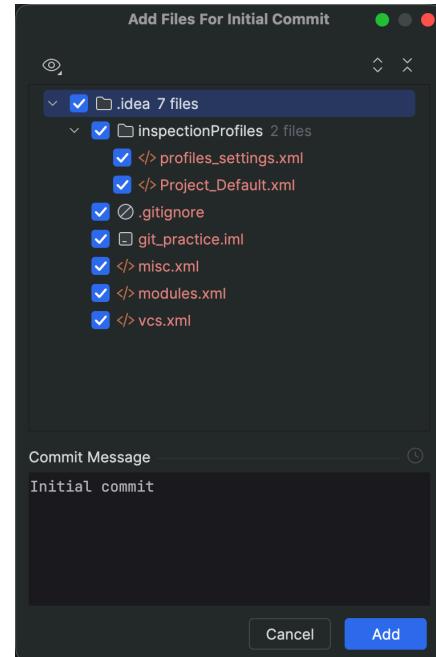
.(VCS → Enable Version Control Integration → Git) **אתחלו Git בפרויקט** .2



**חברו את ה-project המקומי ל-Repo המורוק שלכם ב-GitHub** .3



.4. בחלון שנספתח לחצו על **Cancel**.



.5. צור קובץ חדש בשם **main.py**, עם הפקודה:

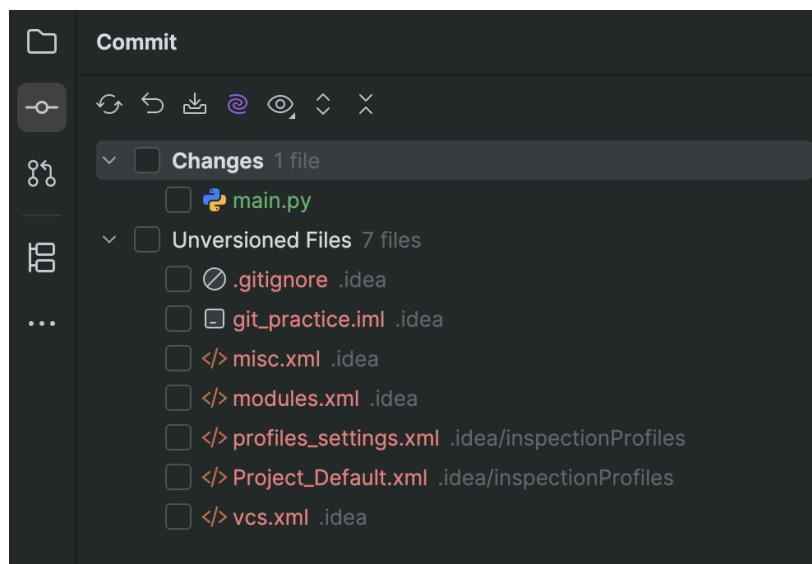
```
print("Hello Git")
```

.6. הריצו את **main.py**.

.7. פתחו את ה-**Commit tool window**. שם תראו את האזרחים:

**main.py** עם Changes .a

**.gitignore** Unversioned files .b



.8. **הוסיפו** את הקובץ `main.py` **בלבד** ל-Staging area **Add** (כלומר סמןו אותו ב-V)

שיםו ❤️ - אין באמת לחצן Add בממתק של PyCharm, אלא נסמן ב-V ב-**Commit tool window** את הקבצים אותם נרצה לעדכן ב-Commit הבא.

.9. בצעו **Commit and Push** ראשון ל-GitHub

שיםו ❤️ - **לפניהם** שאתם מבצעים Commit and Push - **כתבו** הודעה מתאימה!

(Add `main.py` with "hello" print)  
(למשל: `Commit and Push`)

.10. **יכנסו** לאתר GitHub וידאו שאכן נוצר Repo מרוחק.

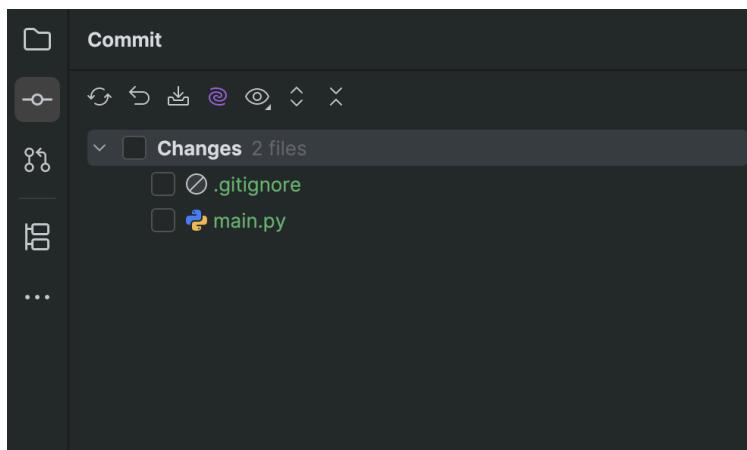
.11. שימו לב שב-Repo ב-GitHub נמצא עכשו ריק `main.py`.

.12. **לחצו על File**, צרו קובץ חדש בשם `.gitignore`.

.13. **כתבו** בו את השורות הבאות:

```
/__pycache__
pyc.*
/idea.
```

.14. **פתחו** שוב את ה-Changes. **Commit tool window** מופיע:



.15. **סמן** ב-V את `.gitignore`, ובצעו **Commit and Push** (למשל: **Added .ignore file**) **לפניהם** ביצוע Commit and Push (Changes)

זכירו לכתוב הודעה Commit (למשל: **Added .ignore file**) **לפניהם** ביצוע Commit and Push (Changes)

.16. **היכנסו** ל-GitHub ובדקו ב-Repo שモפייעים רק הקבצים: `Idea` ו-`.gitignore`

**?** **שאלת מחשבה:** למה חשוב להוסיף קובץ `.gitignore`? כבר בתחילת הפרויקט?

### עדכן הפרויקט - עבודה שוטפת

.1. **ערכו** את הקובץ כך שידפיס בנוסף להודעה הקודמת הודעה שמספרת על מהهو מגניב שעשיתם בחופש.

.2. **בצעו** **Add** → **Commit** → **Push** עם הודעה מתאימה.

.3. **בדקו** ב-GitHub שהשינויים מופיעים בהיסטוריה-commits.



- עשיתם Push ל-GitHub **ממחשב אחר?** כשתחזרו לעבוד על הפרויקט מהמחשב שלכם, טרם תמשיכו לעבוד - אל תשחחו לבצע **Pull** כדי "למשור" אליכם מחשב מה-Repository GitHub את גרסת הפרויקט الأخيرة היכי עדכנית שעבדתם עליה!
  - הקפידו על **Commit and Push** בסיום כל שינוי.
  - וודאו ב-GitHub שהשינויים מופיעים לאחר ה-Push.
  - בצעו Commit עם **הודעות** ברורות ותמציתיות (פירוט נספף במצגת ובלי"ע שקיבלוكم לקראות שיעור זה).
  - הימנעו מהעלאת קבצים מיותרים (כגון `idea/`, `__pycache__`, `.iml`) באמצעות קובץ `.gitignore`.
  - שימו לב לצבעי שמות הקבצים ב-**PyCharm** - צבע הקובץ נקבע לכם אינדיקציה לגבי הסטטוס שלו (**ירוק** – חדש, **כחול** – שונה, **אפור** – לא מנוהל ע"י גיט (ignored))).
- ? שאלה:** מה יהיה הצבע של קובץ שנמחק? מוזמנים לעיין [בערך צבעי הקטוצים בקישור](#).



### חקירה עצמאית - README.md file

לכל פרויקט טוב יש קובץ **README** שמסביר על הפרויקט ומתעד אותו.

הוא הדבר הראשון שمفatha אחר יראה כשייננס ל-Repository שלך ב-GitHub.  
המטרה שלו הוא:

- להסביר בקצרה מה הפרויקט עשו.
- לתת הוראות הרצה והתקנה.
- לשתף מידע חשוב (תלויות, קרדיט, דוגמאות שימוש).

**כל אזכור:** דמיינו שהחבר שלא היה איתכם בשיעור רוצה להריץ את הפרויקט שלכם – כל מה שהוא צריך לדעת צריך להיות כתוב ב-**README**.

#### דוגמה:

לפני שתכתבו README בעצמכם, היכנסו [לפרויקט Snake ב-GitHub](#) והסתכלו על קובץ README שלו.  
אל תקראו הכל לעומק, רק תציגו כדי להבין איך נראה README אמיתי.

### air כותבים קבצי README?

כדי ש-README יהיה קרייא וברור, אנחנו משתמשים בשפת עיצוב פשוטה שנקראה **Markdown**. Markdown מאפשרת להוסף כתורות, רישימות, קוד, קישורים ועוד (כפי שראיתם בדוגמה של Snake) בצורה מאוד קלה לכתיבה. לפניים טבלה עם חוקי העיצוב הבסיסיים ב-Markdown (כמובן שיש עוד).

What you want to do	Markdown syntax
Main heading	Heading #
Subheading	Heading ##
Bold text	**bold**
Italic text	*italic*
Numbered list	Item .1
Bulleted list	Item -
Inline code	`print("hello")`

מצורף קישור ל- [Markdown Cheat Sheet](#) (למידע נוספת :)

### כתבת קובי README לפרויקט שלכם

צרו קובי חדש בשם **README.md** בתחום הפרויקט, בעזרתו תבלת חוקי העיצוב.

[אבני דרך לכתיבה - על הקובי להכיל:](#)

- כתורת ראשית – שם הפרויקט שלכם.
- תיאור קצר – מה הפרויקט עושה, תוך שימוש בהדגשה (Bold) לפחות פעם אחת.
- דוגמת קוד – הוסיףו שורה קוד אמצעית מתוך הפרויקט שלכם.
- הוספה רשימה (list) Bulleted list – עם לפחות שני פריטים על מה אפשר למצוא בפרויקט.