

## סביבת עבודה וירטואלית

### - VIRTUAL ENV

שפות-תיכנות בימינו בנויות בצורה מודולרית: על-גבי השפה הבסיסית, יש הרבה חבילות המרחיבות אותה בדרכים שונות. יש חבילות בסיסיות ועוד חבילות מתקדמות יותר המסתמכות עליהן.

אחד האתגרים בעבודה עם חבילות רבות הוא, שכל חבילה מסתמכת על חבילות אחרות, ולפעמים יש אי-התאמה בין גרסאות של חבילות דרושות. לדוגמה, ייתכן שעבור פרוייקט מסוים, אתם צריכים את חבילה א, המשתמשת בגרסה 1 של חבילה ג; אבל עבור פרוייקט אחר, אתם צריכים את חבילה ב, והיא משתמשת דווקא בגרסה 2 של חבילה ג. איזו גרסה של חבילה ג תתקינו? אתם יכולים בכל פעם להחליף גרסאות בעזרת pip, אבל זו טירחה מיותרת. במקום זה, אפשר ליצור שתי **סביבות וירטואליות** שונות, ולעבור מסביבה לסביבה לפי הצורך. ישנן הרבה שפות תיכנות מודרניות התומכות במנגנון של סביבה וירטואלית; אחת מהן היא פייתון.

כדי להסביר איך עובד המנגנון של סביבה וירטואלית, נבין קודם איך עובד המנגנון הבסיסי יותר של התקנת ספריות. רוב הספריות מאוחסנות בתת תיקייה ב-sys.prefix, או בהקשר שלנו, ספריות צד שלישי של פייתון שמתקנים עם pip בדרך ממוקמות באחת התיקיות ש-site.getsitepackages מצביע עליהן.

```
>>> import site
```

```
>>> site.getsitepackages()
```

על חלונות, התוצאה יכולה להיראות למשל כך:

```
['C:\\Users\\user\\AppData\\Local\\Programs\\Python\\Python310', 'C:\\Users\\user\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages']
```

אלה שתי תיקיות שבהן נמצאות ספריות שמתקנים בעזרת pip. חשוב לדעת את זה כי כברירת מחדל, כל פרויקט במערכת ישתמש באותן התיקיות כדי לאחסן ולקבל את הספריות האלה. במבט ראשוני זה לא נראה רציני כל-כך, וזה באמת לא עבור system packages (ספריות שהן חלק מהספרייה הסטנדרטית של פייתון), אבל עבור site packages זה כבר משהו אחר. למשל נניח יש לנו שני פרוייקטים, פרויקט א' ופרוייקט ב', לשניהם יש תלות (dependency) באותה הספרייה, נניח פרוייקט ג'. הבעיה היא כאשר אנחנו צריכים גרסאות שונות לאותה ספרייה. יכול להיות שפרוייקט א' צריך גרסה ישנה יותר כי הוא בנוי על פיצ'רים ישנים שלה. וזאת בעיה רצינית לפייתון היות והשפה לא מצליחה להבדיל בין הגרסאות השונות. אז אם הגרסה החדשה יותר היא שהותקנה אחרונה, היא תישמר באותו מקום של הספרייה הישנה ותישא את אותו שם.

וכאן בדיוק נכנס הרעיון של סביבת עבודה וירטואלית.

סביבת עבודה וירטואלית או virtual environment מאפשרת ליצור סביבת עבודה מבודדת עבור פרויקטים בפייתון.

זה אומר שכל פרוייקט יכול להכיל את ה-dependencies שלו, מבלי להתייחס ל-dependencies שפרויקטים אחרים צריכים.

בדוגמא שהצגנו לעיל כל מה שנצטרך זה ליצור סביבת עבודה וירטואלית נפרדת עבור כל פרוייקט.

הדבר הנהדר בסביבות עבודה וירטואליות הוא שאין הגבלה למספר הסביבות האפשריות. בנוסף, פשוט מאוד ליצור אותן עם virtualenv או pypenv. ברמת העיקרון המודול virtualenv אמור להגיע עם הספרייה הסטנדרטית של פייתון בגרסה שלוש ומעלה, במידה ואין לנו את הגרסה, או שנרצה לעדכן לגרסה המעודכנת ביותר נוכל להתקין את הספרייה בקלות עם pip:

```
pip install virtualenv
```



הדבר הראשון שעושים, כדי להשתמש במודול כמו שצריך, זה ליצור תיקייה שבה נגדיר את סביבת העבודה החדשה.

כדי ליצור את סביבת העבודה עצמה ניכנס לתיקייה ושורת הפקודה (בווינדוס זה cmd לינוקס טרמינל וכו') נקליד virtualenv ואחריו את שם הפרוייקט, למשל:

```
$ virtualenv .venv
```

עכשיו בשביל להיכנס לסביבה הווירטואלית נקליד את הפקודה הבאה:

```
$ source .venv/bin/activate
```

ואם אתם משתמשים במערכות של ווינדוס צריך להקליד:

```
> .venv\Scripts\activate
```

איך נדע שנכנסו לסביבה הווירטואלית? עכשיו לפני כל פקודה יופיעו סוגריים עם שם התיקייה, למשל במקרה שלנו נראה (venv.) לפני כל שורת פקודה.

אם נרצה לראות באמת שאנחנו משתמשים בגרסת פייתון ו-pip של הסביבה הווירטואלית, נוכל בלינוקס להשתמש בפקודה `which pip` או `which python`, שאמורה להחזיר לנו את ה-path למקום בו מותקן ה-pip והפייתון, ה-path אמור להיות לתיקייה של הסביבה. ועם הפקודה `pip list` ניתן לראות את כל המודולים המותקנים בסביבה. עבור סביבה וירטואלית חדשה אנחנו אמורים לראות רק את הספרייה הסטנדרטית של פייתון. כל קובץ פייתון שנרצה להוסיף יהיה בתיקייה שיצרנו עבור הסביבה הווירטואלית ומחוץ לתיקייה שהסביבה הווירטואלית יצרה כשיצרנו אחת חדשה.

## סביבה וירטואלית עם גרסאות שונות של פייתון

בעזרת סביבות וירטואליות, אפשר לעבוד גם עם גרסאות שונות של שפת פייתון עצמה. לשם כך יש להוריד את הגרסאות הרצויות מכאן: <https://www.python.org/downloads> ולזכור באיזו תיקיה כל גרסה הותקנה. אצלי, במערכת חלונות, כל הגרסאות מותקנות כאן:

```
C:\Users\user\AppData\Local\Programs\Python\PythonNN
```

כאשר NN הוא מספר הגרסה, למשל 37 (עבור גרסה 3.7).

כדי להתקין סביבה וירטואלית המסתמכת על גרסה מסוימת של פייתון, צריך פשוט לקרוא ל venv עם הפייתון המתאים. למשל, בסביבת חלונות:

```
$ virtualenv
--python= C:\Users\user\AppData\Local\Programs\Python\Python38\python.exe
.venv38
```

```
$ virtualenv
--python= C:\Users\user\AppData\Local\Programs\Python\Python39\python.exe
.venv39
```

יוצר שתי סביבות חדשות, עבור פייתון 3.8 ו-3.9. כדי לוודא, נפעיל את הסביבות:

```
$ .venv38\Scripts\activate
```

```
$ python -version
```



```
Python 3.8.10
```

```
$ .venv39\Scripts\activate
```

```
$ python -version
```

```
Python 3.9.1
```

בכל אחת מהסביבות אפשר להתקין את החבילות שרוצים לעבוד בהן עם אותה גרסה של פייתון.

### איך VIRTUALENV בעצם עובדת?

מה זה בעצם סביבה ווירטואלית פעילה? כפי שהזכרנו קודם, כשמבצעים את הפקודה `which python` או `which` `pip` אנחנו רואים את ה-`path` לקובץ ההתקנה של פייתון, כלומר לאיזה פייתון ו-`pip` אנחנו מחוברים. כשהתקנו את השפה בפעם הראשונה במערכת, לפחות בווינדוס, הוספנו את הכתובת של התיקייה בה התקנו את פייתון במחשב למשתנה `PATH`, זהו אותו מקשר לקובץ פייתון. בסביבה ווירטואלית לעומת זאת אנחנו מקשרים את הקוד לקובץ פייתון אחר שממוקם איפה שיצרנו את הסביבה.

