ספריות PYPI- מבנה החבילה

ועכשיו לנושא המרכזי של חלק התכנות של הקורס:
לאחר שעבדנו קשה על בניית הפרויקט שלנו, נרצה שעוד מתכנתים יוכלו ליהנות מפרי עמלנו, כלומר שגם מתכנתים אחרים יוכלו לעשות pip install לספרייה שבנינו.
דבר שראוי לציין, ל-python package index אין איזשהו סנן לספריות שמועלות לאתר, מתכנת יכול לעלות מודולים ברמה של numpy או scipy בדיוק כמו שהוא יכול לעלות מודול שמכיל רק הדפסה של ''hello world" למסך.
כמובן שכמתכנתים נשאף לעלות רק ספריות איכות, אבל לצורך הדוגמא במסמך הבא נעלה ספרייה פשוטה שלא תורמת יותר מידי לקהילה.

מבנה סטנדרטי של ספריית PYPI -

המבנה הסטנדרטי של הספרייה מכיל כמה קבצים שחייבים להיות בספרייה כדי שיהיה ניתן להוריד אותה מהאתר:
- קובץ readme עם סיומת 'md.' שיסביר על הספרייה בכמה מילים עבור מתכנתים אחרים שנתקלו בספרייה באתר.
- קובץ setup.py, נסביר עליו בהמשך.
- קובץ LICENSE גם עליו נסביר בהמשך.
- הספרייה עצמה בתיקייה נפרדת. מומלץ שהספרייה תכיל קובץ __init__.py כדי להכריז עבור הפרויקט שהספרייה היא לא מרחב שם גרידא אלא תיקייה של הפרויקט ,להסבר מורחב יותר מומלץ לחזור למסמך מג'אווה לפייתון- מבנה של מסמך פייתון, שם תמצאו הסבר מפורט מה בדיוק ההבדל בין תיקייה עם הקובץ ובלעדיה. ברמה העיקרון הקובץ __init__.py לא אמור להכיל איזשהו תוכן, אבל נהוג להכניס בתוכו משתנים קבועים וגרסת הספרייה.

__MAIN__.PY:

משהו שלא יצא לנו לדבר עליו הוא על הרצת ספריית פייתון ולא סקריפט בודד.
לפעמים נרצה להריץ ספריית פייתון ולא קובץ סקריפט בודד, למשל אם בנינו מנוע גרפי או תוכנה אחרת, ונרצה להריץ אותה על המחשב. אפשרות אחת היא להריץ את הסקריפט הראשי של הספרייה מה שמצריך להיכנס פנימה ולמצוא אותו אפשרות נוחה יותר תהיה להריץ את כל הספרייה משורת הפקודה, נניח לספרייה קוראים my_package:

```
python -m my_packge
```

כדי להשתמש בדגל '-m ' (על ספריות) נצטרך להגדיר קובץ שממנו רצה התוכנית הראשית (__main__==__name__) , והקובץ צריך להיקרא __main__.py , ואז כאשר מריצים את הפקודה היא מחפשת את הקובץ ומריצה אותו.
כלומר היא חוסכת כניסה לספרייה והרצה של הקובץ הראשי ידנית.
אין חובה להגדיר קובץ זה, למעשה מרבית הספריות שהשתמשנו בהן לא מכילות בכלל קובץ __main__.py , אבל חשוב שנכיר אותו.
הערה: הדגל -m יכול לשמש גם כדי להריץ סקריפטים ולא רק ספריות, ולא צריך להוסיף סיומת 'py.' בהרצה של סקריפטים.

PIP REQUIREMENT –

עוד נושא שלא יצא לנו לדבר עליו במהלך הקורס הוא על דגלים שונים שיש למערכת ניהול החבילות של פייתון.
ברמת העיקרון לא היינו צריכים איזשהו דגל מיוחד במהלך הקורס כדי להתקין חבילה ספציפית, וגם אין לנו צורך ממש להכיר את כל האפשרויות להתקנה.
חשוב להכיר אבל את '-U' שמעדכן גם את החבילות שנצרכות בהתקנת החבילה לגירסה העדכנית שלהם.
ועוד דגל שחשוב להכיר הוא '-r' או '--requirement'. הדגל הזה מאפשר לנו להתקין יותר מספרייה אחת ע"י שימוש בקובץ טקסט שמכיל רשימה של ספריות שאותן נרצה להתקין.
למשל אם אנחנו רוצים להתקין את החבילות flask ו-networkx, במקום להתקין כל ספרייה בנפרד, נכין קובץ טקסט שבתוכו מופיעים הקבצים שאותם אנחנו רוצים להתקין (כל חבילה בשורה נפרדת), נניח קוראים לקובץ requirement.txt אז נוכל להקין את הקבצים בבאת אחת כך:

```
pip install -U -r requirement.txt
```

איך זה קשור לנושא שלנו? כשאנחנו יוצרים ספרייה נרצה להוסיף קובץ requirement כזה כדי שמתכנתים אחרים שמשתמשים בספרייה יוכלו להתקין אותה עם כל ה-dependencies שלה, כלומר מתי שהם יתקינו את הספרייה של הפרויקט עם pip install כל הספריות הנלוות לפרויקט שלנו יותקנו יחד איתה.
למעשה אם אנחנו משתמשים בסביבת עבודה וירטואלית, כפי שראינו בשיעור בנושא flask, אנחנו לא צריכים לעבוד קשה ,ויכולים להגדיר את קובץ requirement בקלות עם הפקודה הבאה:

```
pip freeze > requirement.txt
```

מה שעושה הפקודה זה לקחת את כל הספריות שמותקנות דרך pip ולהכניס אותן לקובץ טקסט שקוראים לו requirement.
אמרנו שצריך להשתמש בסביבת עבודה וירטואלית ,אך האמת שגם אם לא היינו משתמשים בסביבה וירטואלית היינו מצליחים לעשות את זה, הבעיה שהיינו מכניסים הרבה מאוד ספריות שלא נצרכות בכלל עבור המתכנתים האחרים.

תיקיית טסטים-

לרוב החבילות ,אם לא כולן, מוסיפים תיקייה של טסטים (unitests) לקוד של החבילה בשם tests.
עבור כל פונקציה או מחלקה נהוג לבנות סקריפט נפרד. עם שם המילה test_ ושם הפונקציה או המחלקה שאותה הוא בוחן.
דוגמא לתיקייה מהחבילה num2words שממירה מספרים למילים או מילים למספרים בכמה שפות: num2words.