FLASK INTRODUCTION

הערה חשובה- השיעורים הבאים יהיו בנושא flask. את כל השיעורים לקחתי מערוץ היוטיוב של Corey Schafer.
הערוץ מכיל כמות אדירה של חומר בנושא פייתון (ועוד ...) בכל רמה אפשרית.
לצערנו כמות החומר שאפשר להעביר בשיעור מוגבלת מאוד, ולא הספקנו לעבור על כל החומר של סדרת השיעורים של קורי. נמליץ בחום על פלייליסט השיעורים בנושא flask , ועל הערוץ בכלל.

מה זה FLASK-

WSGI (Web Server Gateway Interface) הוא ממשק רשת שמחבר בין צד קוד שרת וצד קוד לקוח.
הוא מפרט המתאר את דרכי ההתקשרות בין השרת לאפליקציות רשת, ואיך אפליקציות רשת יכולות להתחבר יחד כדי לבצע בקשה כלשהי.
flask הוא framework פשוט המבוסס על wsgi. הוא עוצב כך שיהיה קל ומהר להתחיל לעבוד איתו, עם יכולת להרחבה בהמשך לאפליקציות מתקדמות.
flask הוא מודול של פייתון שמאפר לפתח אפליקציות רשת במהירות. המודול הוא microframework שאינו כולל ORM (Object Rational Manager).
אבל flask כן מכיל הרבה מאוד פיצרים מגניבים כמו ניתוב כתובות url, בניית שלד לאפליקציה ועוד.
אז מה זה web framework?
Web framework הוא אוסף של ספריות ומודולים שמאפשרים פיתוח של אפליקציות רשת מבלי לדאוג לפרטים ברמת ה-low level כמו פרוטוקולים, ניהול תרדים וכו'.
אז מה זה micro framework?
"מיקרו" אין הכוונה שכל האפליקציה אמורה להיכנס בתוך מסמך פייתון אחד(אם כי זה בהחלט אפשרי), זה גם לא אומר שחסר פונקציונאליות ל-flask. הכוונה ב"מיקרו" היא שהליבה של המערכת פשוטה אך ניתנת להרחבה.
המודול flask אינו מקבל הרבה החלטות בשביל המתכנת, כמו באיזה מסד נתונים להשתמש. אבל הוא כן מקבל החלטות שניתן לשנות בקלות, כך ש-flask יכול להיות כל מה שאנחנו צריכים.
אז למה כדאי לבחור ב-flask ולא באלטרנטיבות אחרות כמו nodejs או Django?
שלא כמו Django, flask הוא מאוד פייתוני, כלומר הוא כלי שמאוד קל להיכנס אליו עבור מתכנתים בפייתון, ואין לו עקומת למידה גדולה מידי. יותר מזה המודול מאוד מפורש, מה שמשפר את רמת הקריאות של הקוד.
למשל כדי ליצור אפילקציית רשת פשוטה שכל מה שהיא מכילה זה רק את הכותרת "hellow world" נצטרך רק כמה שורות בודדות של קוד:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```

flask היא אחת הספריות הפופולריות ביותר (ולא רק עבור מתכנתי פייתון), ונחשבת לפי github ככלי השני שמקבל הכי הרבה כוכבים.

לפני שמתחילים רק *הערה*: אנחנו יוצאים מנקודת הנחה שיש לכם ידע בסיסי במשגים הקשורים בעולם ה-web וב-html ו-css. אם אין לכם שום ידע בנושא נמליץ מראש לעיין באתר w3schools בנושא html, אם כי אני מאמין שגם בלעדיו תוכלו להסתדר.

משום שלרוב העבודות בתחום הווב מורכבות מצוותים ולא עבודה של בודדים, מומלץ לבנות סביבה ווירטואלית שתהווה

מוסכמה של הצוות בנוגע לאילו גרסאות של מודולים המתכנתים ישתמשו במהלך העבודה.

התקנה של flask- ההתקנה של flask בדיוק כמו התקנה של שאר המודולים בפייתון, תעשה בשורת הפקודה:

```
pip install flask
```

אם רוצים לבדוק שאכן ההתקנה התבצעה כמו שצריך, אפשר לפתוח את המצב האינטראקטיבי ולהריץ:

```
>>> import flask
```

אם לא קיבלנו שגיאה, אזי ההתקנה התבצעה כפי שרצינו.

עבודה בסיסית עם FLASK:

בואו נתחיל בליצור תיקייה חדשה עבור הפרוייקט, ונוסיף לה סקריפט פייתון חדש (במסך זה נקרא לו flask_intro.py).
במסמך נכניס את הקוד שהצגנו קודם:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```

בהתחלה אנחנו מייבאים את המחלקה Flask מהמודול.
מאתחלים את המחלקה בזה שאנחנו נותנים לה את המשתנה __name__ כארגומנט.
תזכורת: המשתנה __name__ הוא אובייקט שמחזיק את שם המודול, ובמידה ואנחנו מריצים את הסקריפט ישירות מפייתון המודול מקבל את השם __main__. המטרה של הארגומנט הוא כדי שמחלקה תדע אילו dependencies יש למודול הספציפי שעליו flask פועל.
מתחת יש פונקציה hello_world() שאנחנו שמים את הקשטן app.route. עוד מעט נבין בהרחבה מה עושה בדיוק הקשטן ,כרגע רק נתייחס אליו כאילו הוא מחזיר לרשת את התוכן של האתר בכתובת ספציפית.
ובסוף הקוד מורים לתוכנית להפעיל את app עם הפונקציה run() בפונקציה המרכזית של הסקריפט.

נריץ את הסקריפט משורת הפקודה ונקבל משהו כזה:

```
python flask_intro.py
 * Serving Flask app "flask_intro" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

אז מה אנחנו רואים פה בעצם? Flask יצר לנו כמין שרת על הכתובת http://127.0.0.1 בפורט 5000.
אנחנו יוצאים מנקודת הנחה שכולם מבינים מה הכתובות האלה אומרות, לפחות בצורה בסיסית, אם לא אז על קצה המזלג:
הכתובת הזאת מוכרת גם כ-local host אומרת שהשרת עובד כרגע ברשת המקומית של המחשב ולא על כתובת שניתן להגיע אליה מבחוץ. הפורט הוא פתח ההתכתבות של שרת עם המחשב.
שימו לב שהכתובת הזאת נוצרה אוטומטית כאשר הפעלנו את הסקריפט.
אם ניכנס לכתובת הזאת נראה שהיא פותחת לנו את האתר הבא:

כל מה שיש באתר הוא בסה"כ כותרת עם המשפט !Hello World , כי זה באמת מה שהחזרנו מהפונקציה hello_world.
למעשה יכולנו לשלוח ממש מחרוזת של קוד html ולהציג אותה באתר. למשל במקום להחזיר "Hello World" יכולנו להחזיר אובייקט "<h1>Hello World!</h1>" שהיה הופך את הטקסט לכותרת.

טוב אם נרצה כעת לבצע שינוי בקוד, נצטרך לסגור את השרת ולהפעיל אותו מחדש, אחרת השינוי לא יהיה ניכר.
זה יכול להיות מאוד מעייף, שכן יכול להיות שנבצע הרבה מאוד שינויים ונרצה לראות אותם בזמן אמת ,וכל פעם נצטרך לסגור את השרת ולהקים מחדש, בשביל זה אנחנו צריכים להפעיל את ה- debug mode.
כדי להפעיל את ה-debug mode נצטרך בסה"כ להוסיף עוד ארגומנט debug לפונקציה run שמקבל את הערך True:

```
if __name__ == '__main__':
    app.run(debug = True)
```

לפני שנריץ שוב, ישנה דרך נוספת להפעיל שרת flask משורת הפקודה (לא בווינדוס)-
נקליד:

```
export FLASK_DEBUG=1
export FLASK_ENV = development
flask run
```

עכשיו נריץ את הקוד שוב ,וכדי לבדוק שאכן השינוי קורה בזמן אמת נשנה את הפונקציה hello_world שתחזיר משהו אחר:

```
@app.route('/')
def hello_world():
    return '<h1>Hello World!</h1>'
```

נעשה ריענון לעמוד ונראה שהאתר השתנה למשהו כזה:

נחזור לקשטן route . מה שהקשטן בעצם עושה הוא "מרנדר" תוכן מסוים לפי כתובת.
כרגע אנחנו נמצאים בכתובת שקיבלנו כברירת מחדל, אבל בהרבה אתרים יש כמה דפים ולכל דף יש ניתוב אחר, למשל הדף הראשי יהיה בכתובת האתר עם סיומת '/home' או בלי סיומת, דף בסגנון 'about' יהיה בכתובת האתר עם סיומת '/about' . זהו פחות או יותר הרעיון של הקשטן, הוא מנתב תוכן לפי סיומות שונות לכתובת האתר.
נראה דוגמא יותר מוחשית.
נבנה פונקציה חדשה שמרנדרת את הכותרת about בכתובת localhost:5000/about :

```
@app.route('/about')
def about():
    return '<h1>about</h1>'
```

ועכשיו אם נכנס לכתובת http://localhost:5000/about או http://127.0.0.1:5000/about נראה את הדף הבא:


127.0.0.1
http://127.0.0.1:5000/about
About