FLASK- HTML TEMPLATES

אז בנתיים הצלחנו לרנדר סוג של עמוד html פרימיטיבי, אבל זה ממש לא עמוד html טיפוסי עם תגיות וסקירפטים.
למעשה אנחנו יכולים ממש להחזיר מפונקציית route מחרוזת שמייצגת עמוד html עם כל השתמע מכך, אבל זה לא הדבר הכי יעיל, בטח שעובדים על אפליקציות רשת גדולות עם הרבה מפתחים.
פתרון יעיל לבעיה הוא להשתמש בפונקציה render_template שיכולה לרנדר עמודי html שלמים ולא רק מחרוזות שלהם.
כדי להשתמש בפונקציה נצטרך א. לייבא אותה מ-flask ב. לשמור את קבצי ה-html בתיקייה שנקראת templates .
לאחר שעשינו את זה נוכל להחזיר מפונקציית ה-route את render_template(page_name.html):

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

@app.route('/')
def hello_world():
    return render_template('home.html')

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

ועכשיו אוטומטית ירונדר העמוד home.html כאשר ניכנס לכתובת של השרת.

דפי HTML דינאמיים-

פרמטר שיש ל-render_template הוא kwargs** כלומר הפונקציה מייצרת משתנים בעצמה. למה משמשים המשתנים?
כדי ליצור דפי html דינאמיים.
בהרבה דפים נרצה לרנדר תוכן משתנה ולא אחד ססטי, למשל בפורומים נרצה שהפורום יעלה כל פעם את ההודעה החדשה ביותר שהתקבלה, או באתר חדשותי נרצה לעלות כל פעם תוכן חדש.
עם flask ניתן להשתמש בדפי html בצורה דינאמית ולהחדיר להם קוד פייתון עם סוגרים מסולסלים והסימון מיוחד '%'-{% ... %}.
ניקח למשל את הדף html הבא (בשביל להפריד אותו מהדוגמא הקודמת נניח קוראים לו dynamic.html):

```
<!DOCTYPE html>
<html>
    <head>
        <title>Page Title</title>
    </head>
    <body>
        <h1>Body</h1>
    </body>
</html>
```

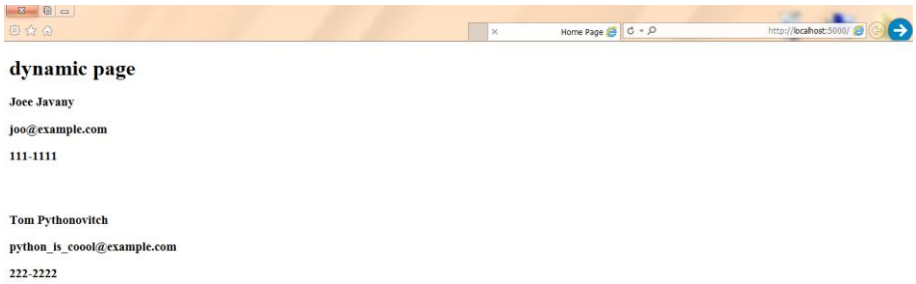
אנחנו רוצים להשתמש בו כדי ליצור תבנית דינאמית עבור משתנים שנקבל מהפונקציה render_temlate().
נניח הפונקציה שולחת רשימה של אנשים עם המייל שלהם ומספר טלפון. הפונקציה תשלח את רשימת האנשים כרשימה של מילונים כך:

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

users = [
    {'name': 'Joee Javany',
    'email': 'joo@example.com',
    'phone': '111-1111'},
    {'name': 'Tom Pythonovitch',
    'email': 'python_is_coool@example.com',
    'phone': '222-2222'},
]

@app.route('/')
def hello_world():
    return render_template('dynamic.html' , users = users)

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

עכשיו הפרמטר עובר לדף ה-html וניתן לבצע עליו קוד פייתון.
כל פקודת בקרת זרימה- למשל if או for צריך להיות מסומן גם בסופו, היות ודף html לא רגיש להזחות.
את הפקודה בהתחלה אנחנו כותבים כמו כל פקודת פייתון, רק בלי נקודתיים ובתוך סוגריים מסולסלים עם % בהתחלה ובסוף.
סוף הפקודה תסומן גם בסוגריים מסולסלים עם שני סימני '%', רק שבפנים צריך לכתוב end ובלי רווחים לכתוב את שם הפקודה למשל אם סיימנו פקודת if נכתוב {% endif %}
בתוך הפקודה נוכל להשתמש במשתנים שהעברנו עם סוגרים מסולסלים כפולים {{ ... }}:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Page Title</title>
    </head>
    <body>
        <h1>dynamic page</h1>
        {% for user in users %}
            <h3>{{user.name}}</h3>

            <h3>{{user.email}}</h3>
            <h3>{{user.phone}}</h3>
        {% endfor %}
    </body>
</html>
```

ועכשיו אם נפעיל את השרת נראה באותה כתובת את שהתוכן ששלחנו אכן הגיע:



אם נסתכל על מקור הדף (צורת ה-html שלו) נוכל לראות כיצד מתבטא השינוי מאחורי הקלעים:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Home Page</title>
    </head>
    <body>
        <h1>dynamic page</h1>

            <h3>Joee Javany</h3>
            <h3>joo@example.com</h3>
            <h3>111-1111</h3>
            <br><br>

            <h3>Tom Pythonovitch</h3>
            <h3>python_is_coool@example.com</h3>
            <h3>222-2222</h3>
            <br><br>

    </body>
</html>
```

-LAYOUT

למרבית האתרים יש מבנה זהה לכל עמוד, כמין פיגום לכל דף באתר- התפריט הראשי למעלה כל מיני אייקונים למטה וכו' ובפנים התוכן שונה בין עמוד לעמוד.
ניתן להגדיר עם flask איזשהו layout שישמור את העיצוב שלו בין הדפים.
היות ואנחנו לא קורס fronend רק נציג דוגמא מבלי יותר מידי להתעכב עליה. לקחנו איזשהו פיגום של דף אינטרנט מהאתר w3schools עם סטייל שהם מספקים והפכנו אותו ל-layout של האתר.
את ה-layout המלא אפשר לראות בתיקייה '0.code' <-'2.templates' <- 'templates' <- layout.html.
אבל לצורך הדוגמא הנוכחית נסתכל רק על החלק בתוך ה-body:

```
<!DOCTYPE html>
<html>
   ...
    <body>
      ...

    </body>
</html>
```

ניצור איזשהו בלוק תוכן שישתנה בין הדפים של האתר. הבלוק צריך להיות מוגדר גם בסופו:

```
<!DOCTYPE html>
<html>
   ...
    <body>
      ...
       {% block content %}
       {% endblock %}
    </body>
</html>
```

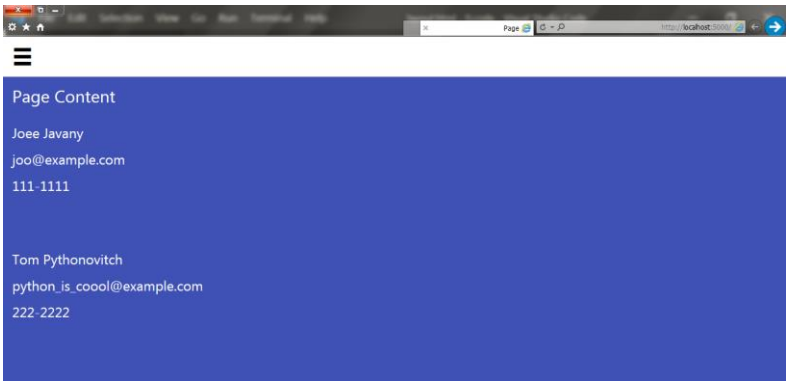
ניקח למשל את הדף html שעשינו קודם ונשנה אותו כך שיהווה דף תוכן במקום דף html , כדי להפריד אותו מהקוד הקודם נקרא לו dynaic_layout.html.
מה שחשוב לנו בדף הוא שיכיל רק את חלק ההדפסה של המשתמשים עם השמות , המיילים, והמספרי טלפון שלהם, ולא מעניין אותנו כל המעטפת:

```
{% for user in users %}
    <h3>{{user.name}}</h3>

    <h3>{{user.email}}</h3>
    <h3>{{user.phone}}</h3>
{% endfor %}
```

בשביל להשתמש ב-layout שהגדרנו קודם נצטרך "להרחיב" אותו ולהגדיר היכן מתחיל הבלוק תוכן והיכן הוא מסתיים.

```
{extends "layout.html"}
    {% block content %}
        {% for user in users %}
            <h3>{{user.name}}</h3>
            <h3>{{user.email}}</h3>
            <h3>{{user.phone}}</h3>
            <br><br>
        {% endfor %}
    {% endblock %}
```

ועכשיו אם נעביר את הפרמטרים דרך השרת שלנו הוא אמור לרנדר את האתר עם ה-layout שיצרנו:



-ERROR SCREEN

נושא אחרון לפרק זה הוא מצב debugger.
נניח בקוד עשינו איזושהי שגיאה, למשל במקום לרנדר את הדף dynamic-layout.html מרנדרים את הדף dynamic-layout.html, דף שלא הגדרנו, ולכן הפעולה לא אמורה לעבוד, היינו מגיעים לדף הזה במקום:

```
jinja2.exceptions.TemplateNotFound
jinja2.exceptions.TemplateNotFound: dynamic-layou.html

Traceback (most recent call last)

File "C:\Users\user\AppData\Local\Programs\Python\Python38\Lib\site-packages\flask\app.py", line 2464, in __call__
  return self.wsgi_app(environ, start_response)
File "C:\Users\user\AppData\Local\Programs\Python\Python38\Lib\site-packages\flask\app.py", line 2450, in wsgi_app
  response = self.handle_exception(e)
File "C:\Users\user\AppData\Local\Programs\Python\Python38\Lib\site-packages\flask\app.py", line 1867, in handle_exception
  reraise(exc_type, exc_value, tb)
File "C:\Users\user\AppData\Local\Programs\Python\Python38\Lib\site-packages\flask\_compat.py", line 39, in reraise
  raise value
File "C:\Users\user\AppData\Local\Programs\Python\Python38\Lib\site-packages\flask\app.py", line 2447, in wsgi_app
  response = self.full_dispatch_request()
File "C:\Users\user\AppData\Local\Programs\Python\Python38\Lib\site-packages\flask\app.py", line 1952, in full_dispatch_request
  rv = self.handle_user_exception(e)
File "C:\Users\user\AppData\Local\Programs\Python\Python38\Lib\site-packages\flask\app.py", line 1821, in handle_user_exception
  reraise(exc_type, exc_value, tb)
File "C:\Users\user\AppData\Local\Programs\Python\Python38\Lib\site-packages\flask\_compat.py", line 39, in reraise
  raise value
```

דף זה נקרא error screen והוא מגיע כאשר נזרקת איזושהי חריגה מהקוד שלא נתפסה.
המצב הזה הוא מאוד יעיל, אבל הוא גם הסיבה שבגללה לא נרצה שהפרויקט יהיה במצב debug כאשר אנחנו נפרסם אותו לשרת חיצוני, שכן הוא מציג יותר מידי אינפורמציה עבור אנשים אחרים.
ה-error screen מראה לנו איזו סוג של שגיאה נזרקה, באיזו שורה, ואפילו מאפשרת לנו להריץ חלקי קוד כדי לבדוק כיצד נראית השגיאה בזמן אמת.
למשל אצלנו הוא מציג לנו ממש שבשורה 15 בפונקציה hello() נתפסה השגיאה:

```
  return self.view_functions[rule.endpoint](**req.view_args)
File "C:\Users\user\Desktop2\מעת\Github\Python_Ariel\5.flask\0.code\3.templates\4.debug.py", line 15, in hello
  return render_template('dynamic-layou.html' , users = users)
File "C:\Users\user\AppData\Local\Programs\Python\Python38\Lib\site-packages\flask\templating.py", line 138, in render_template
  ctx.app.jinja_env.get_or_select_template(template_name_or_list),
File "C:\Users\user\AppData\Local\Programs\Python\Python38\Lib\site-packages\jinja2\environment.py", line 930, in get_or_select_template
```

אם נעבור על השורה עם העכבר אז מצד ימין יופיע לנו כמין icon של cmd או טרמינל, הוא מאפשר לנו להריץ את הקוד בזמן אמת, אבל אם נלחץ עליו הוא יבקש איזשהו pin כדי להפעיל אותו:

```
Console Locked
The console is locked and needs to be unlocked
by entering the PIN. You can find the PIN
printed out on the standard output of your
shell that runs the server.
PIN: [          ] Confirm Pin
```

ה-PIN מופיע בשורת הפקודה בה הרצנו את השרת, תחת השם debugger PIN, זהו איזשהו מנגנון הגנה נוסף של הספרייה, נכניס אותו ונוכל להריץ את הקוד ב-error screen:

```
File "C:\Users\user\Desktop2\מעת\Github\Python_Ariel\5.flask\0.code\3.templates\4.debug.py", line 15, in hello
  return render_template('dynamic-layou.html' , users = users)
[console ready]
>>>
```