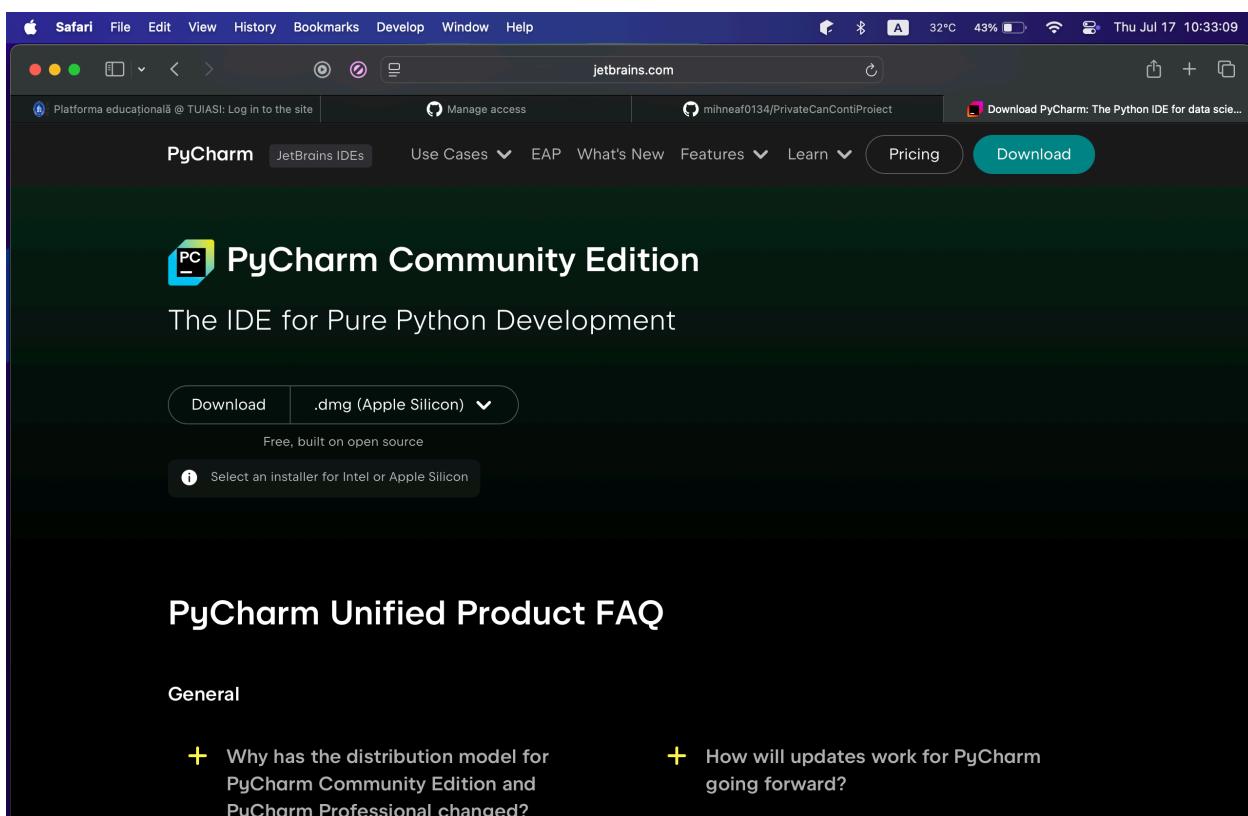


Aici aveți instructiuni cu inceperea mediului de lucru. M-am gândit ca toată lumea să aibă la început propriul WebServer (Flask) local de pe care să va faceți parte.

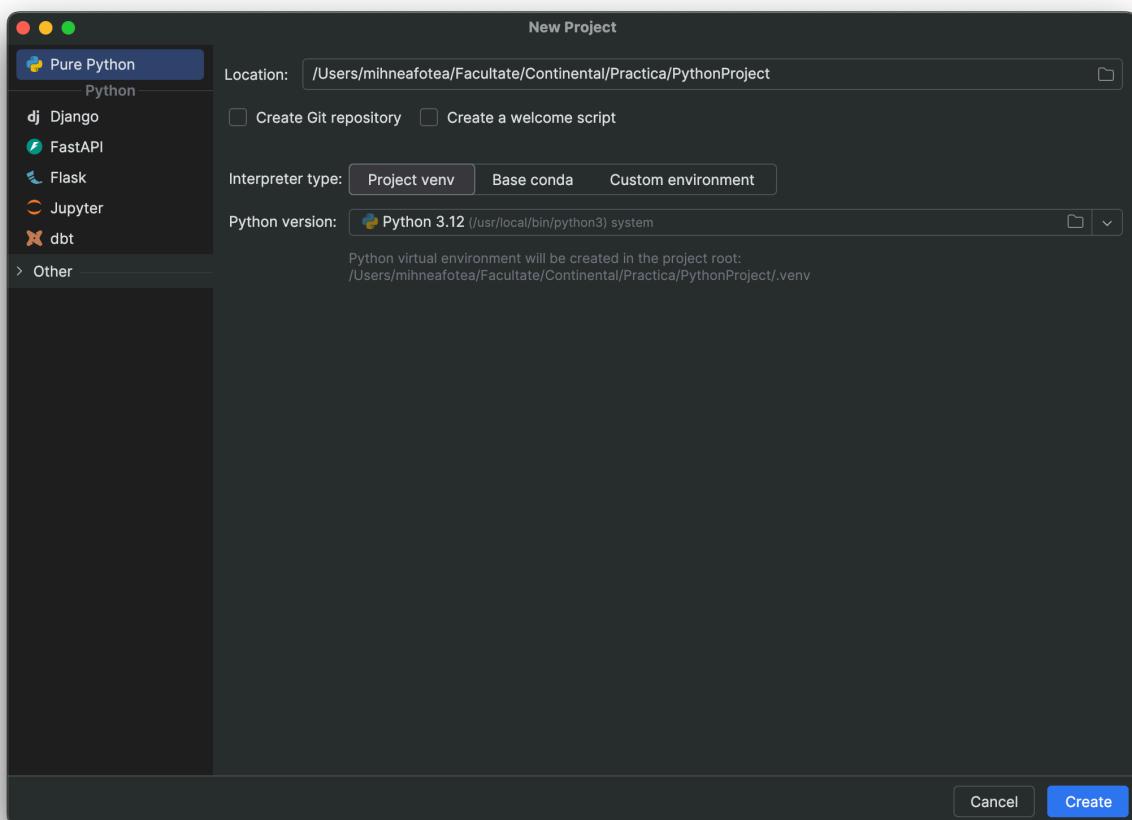
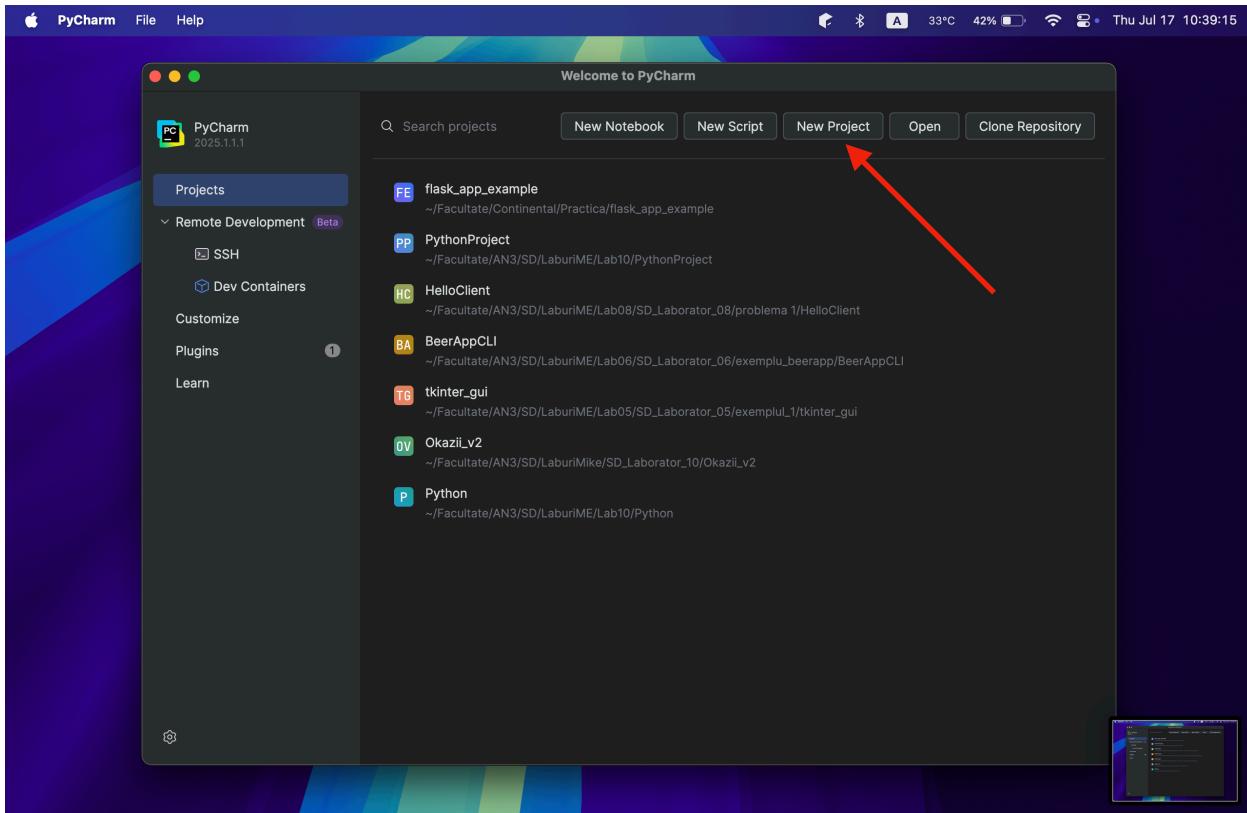
Dacă nu aveți python deloc instalat intrati pe
<https://www.python.org/downloads/release/python-3120/>
Dati scroll în jos și alegeți-va versiunea voastră pentru sistemul de operare.

1. Pentru început, instalati-va Pycharm **Community Edition**
<https://www.jetbrains.com/pycharm/download/>

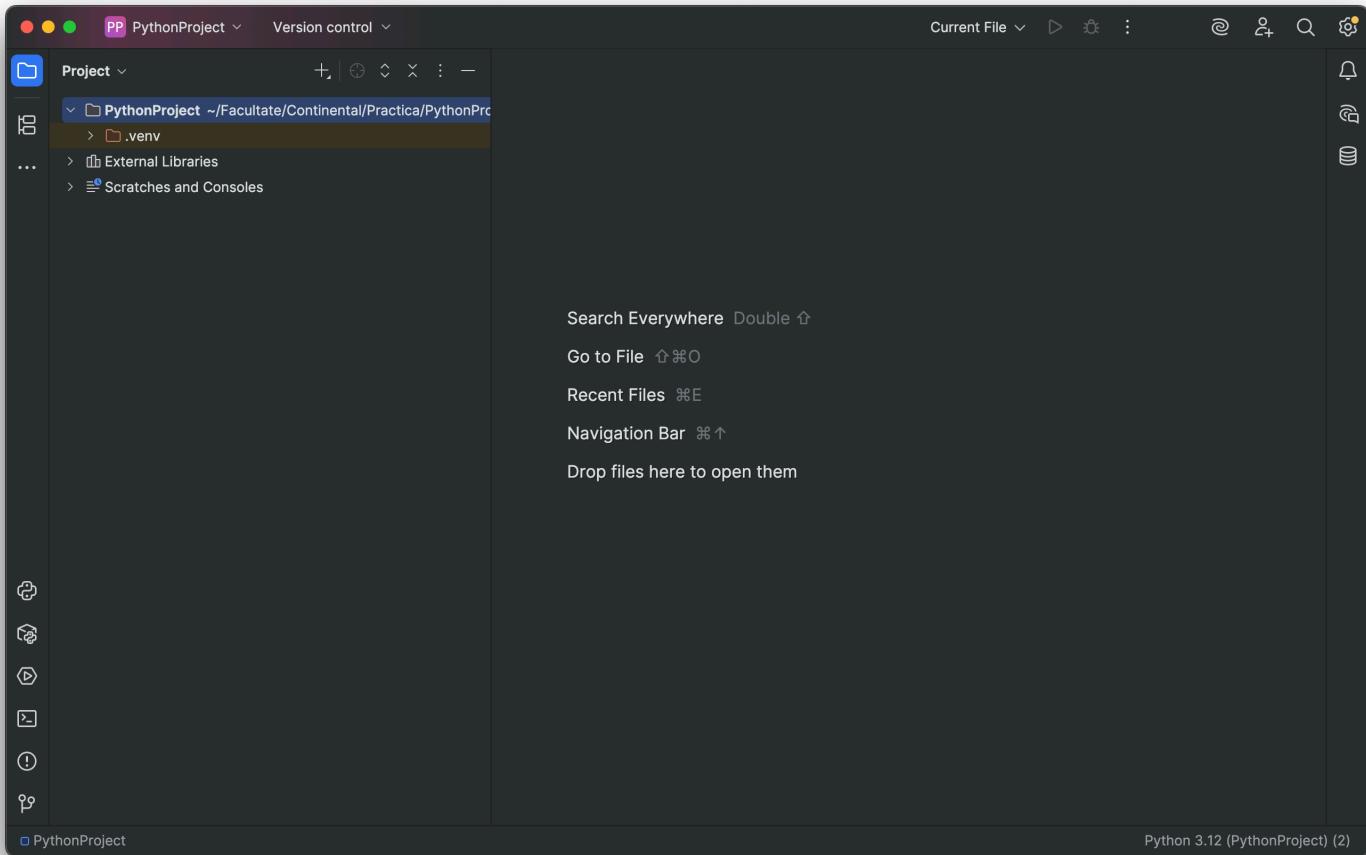
(Nu primul de sus numit PyCharm Unified, acela trebuie plătit, dati scroll în jos până vedeti community edition și apăsați pe download pentru sistemul vostru de operare (Windows,Linux,Mac))



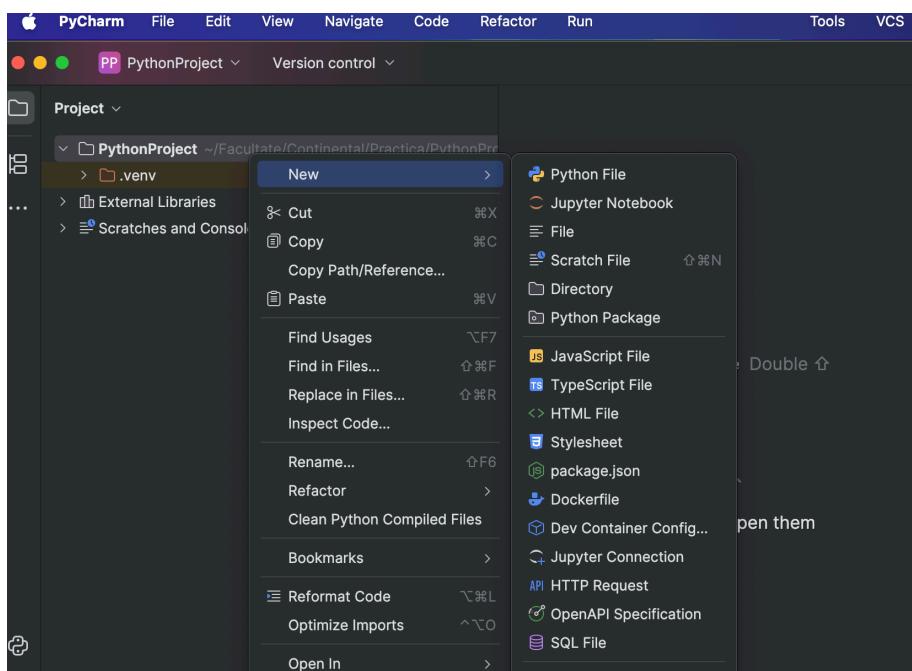
2. Dupa instalare creati un proiect nou si dati click pe create dupa ce ati selectat calea pentru proiect.



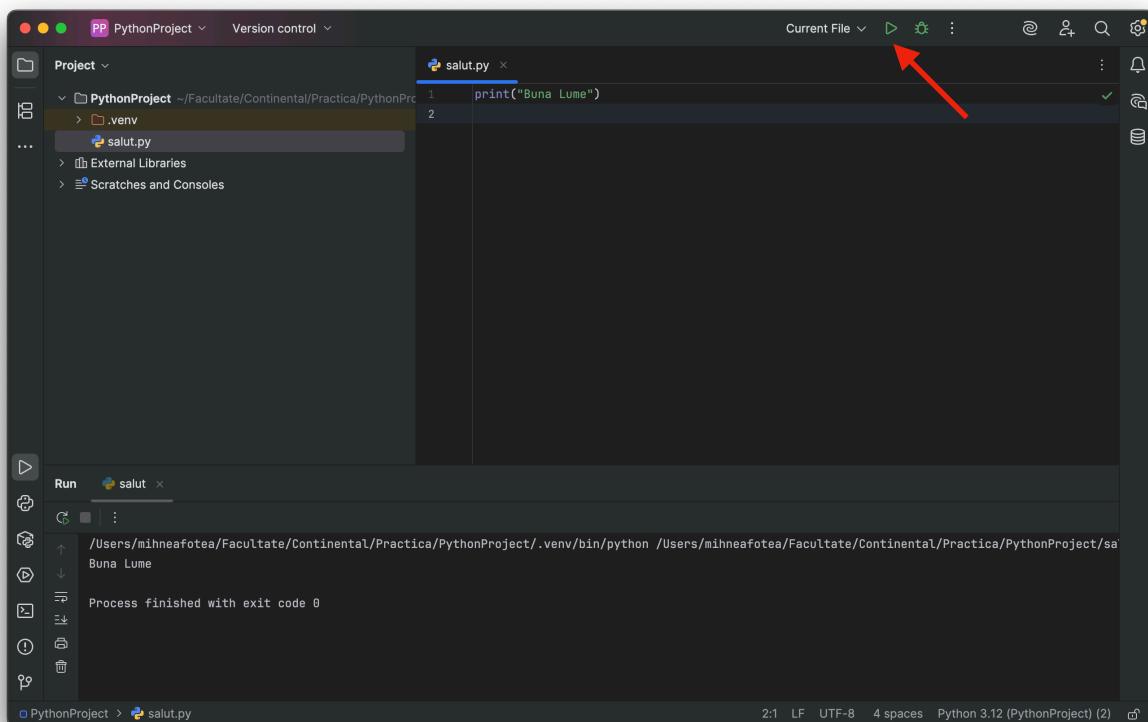
3. Ar trebui sa arate asa pana acum.



Dupa pentru a putea crea un nou fisier python pe care sa l rulati, apasati right-click pe numele Proiectului si la New - - > Python File.



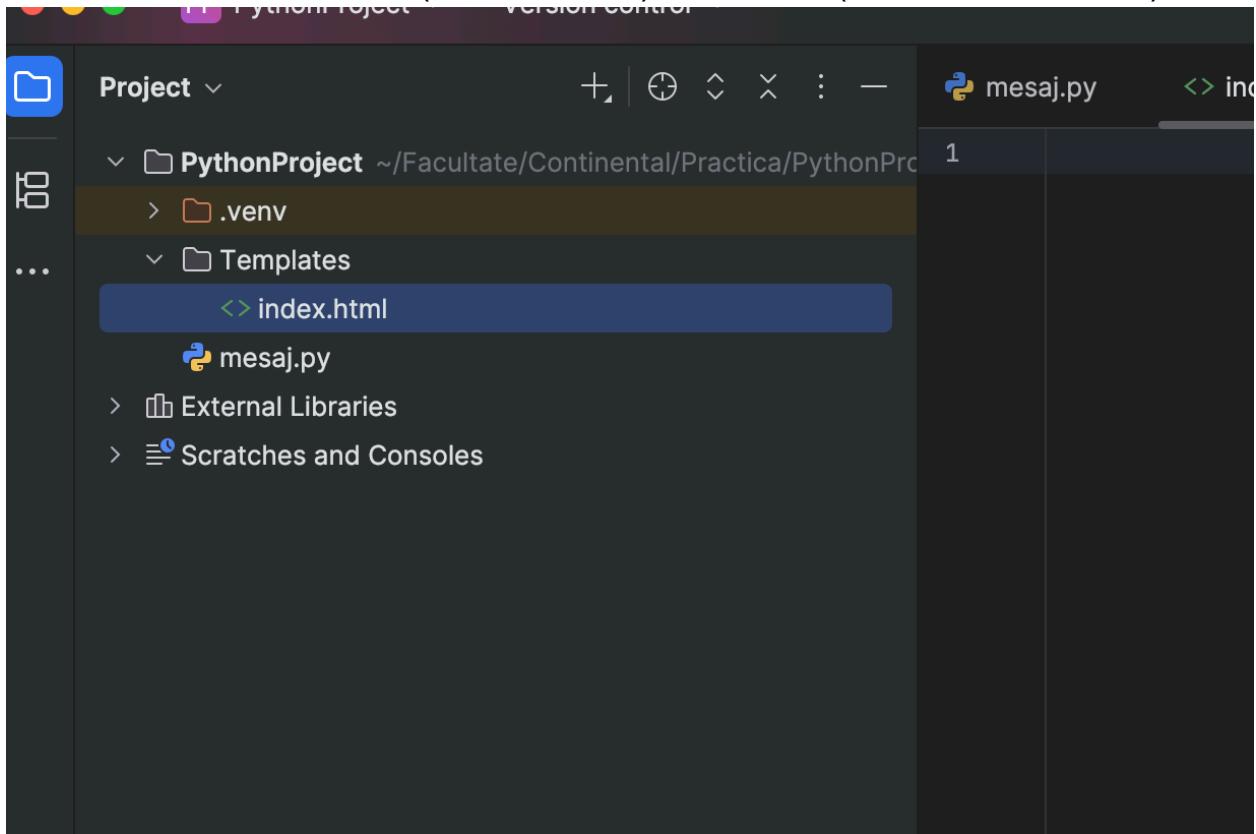
4. Dupa ar trebui sa vi sa deschida o fereastra cu fisierul respectiv. Scripti un simplu cod de afisare iar cand vreti sa executati apasati pe butonul verde de Play (Atentie! Executia are loc la nivel de ce fereastra este selectata. Daca aveti mai multe fisiere .py atunci butonul va executa numai fisierul la care aveti fereastra deschisa)



Dupa jos aveti consola cu output-ul.

5. Acum ca ne am familiarizat cu aplicatia, vom trece la cum se proiecteaza un WebAPI in Pycharm folosind Flask.

Prima data creati un mesaj.py si un folder (New → Directory) numit Templates (NEAPARAT TEMPLATES ca asa este conventia FLASK) in care sa se afle un fisier numit (New → File) index.html(NEAPARAT INDEX).

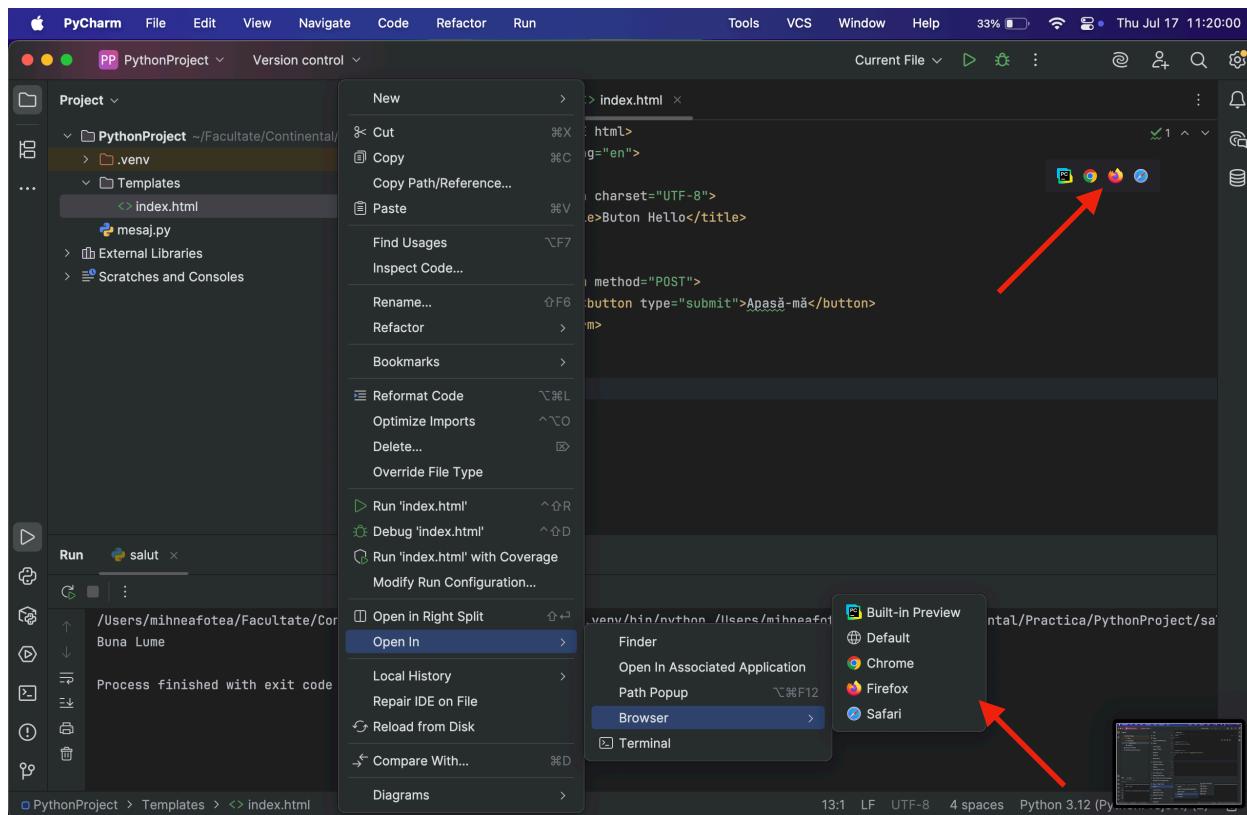


* index.html va fi site-ul nostru cu butoanele noastre care vor comunica cu mesaj.py.

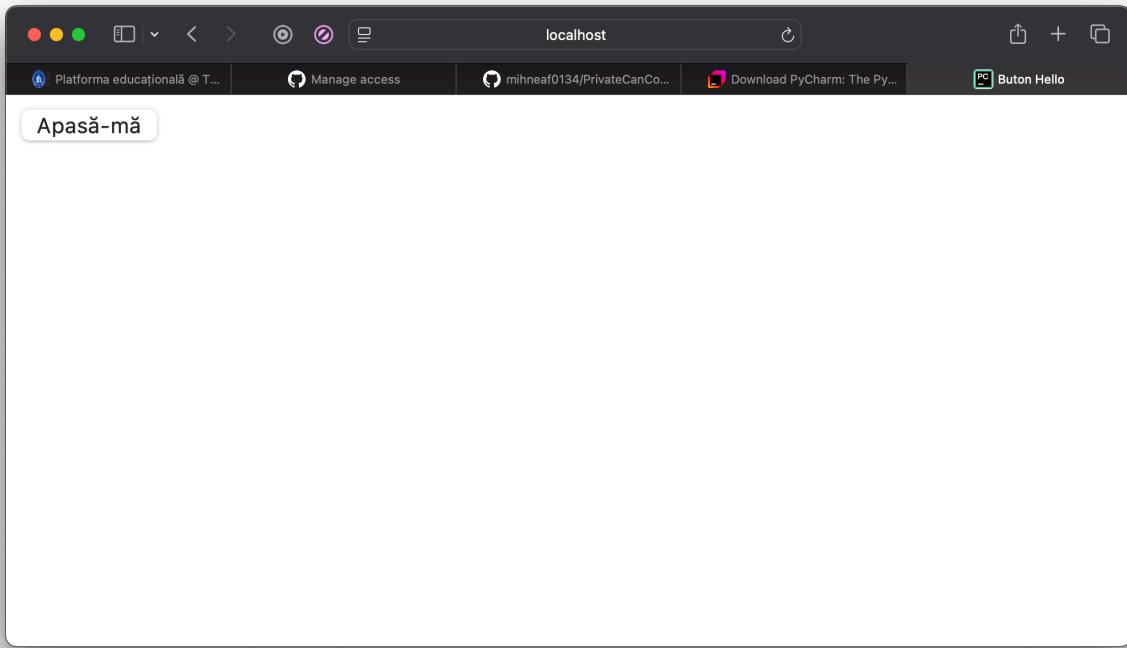
6. Scrieti urmatorul cod in index.html (Intrebati chatgpt-ul linie cu linie sa va explice daca nu stiti ce se intampla in cod)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Button Hello</title>
</head>
<body>
    <form method="POST">
        <button type="submit">Apasă-mă</button>
    </form>
</body>
</html>
```

Pentru a deschide site-ul din Pycharm aveti 2 optiuni. Ori cu right-click pe fisier ori va aparea pe dreapta un prompt cu browser ul dorit.



Site-ul nostru 😊



Insa butonul nostru nu face nimic momentan pentru ca nu l-am configurat. in codul index.html este scris doar aspectul paginii (Cred ca va trebui sa fie un rol de Web Design). **ATENTIE ACESTA NU E SERVER ESTE DOAR O METODA DE A VIZUALIZA OUTPUT-UL UNUI FISIER .HTML. SERVER-UL ESTE DAT DE CODUL PYTHON MESAJ.PY CARE VA COMUNICA CU INDEX.HTML.**

7. Acum trecem la partea mai grea, si anume server-ul cu aplicatia. Scrieti in mesaj.py codul urmator. (Din nou puneti pe chat ce nu intelegeti si spuneti i sa va explice linie cu linie, se pricepe la asta)

```
from flask import Flask, render_template, request
```

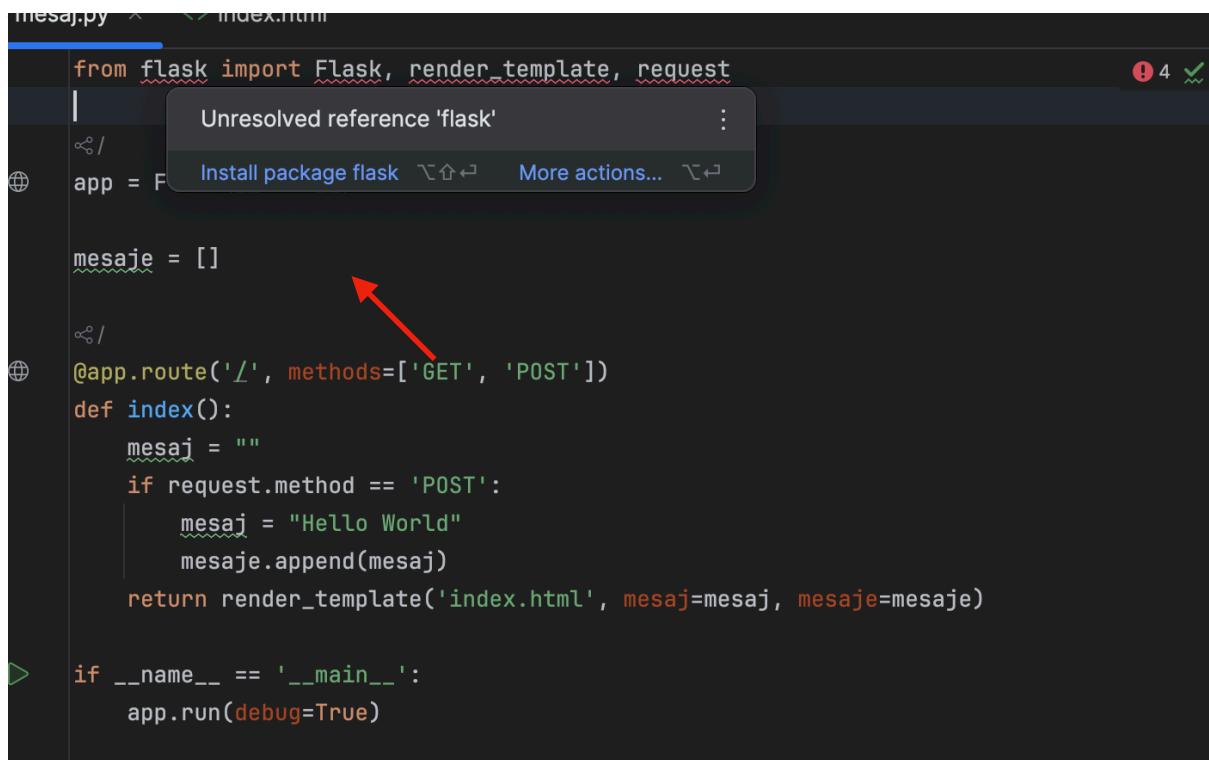
```
app = Flask(__name__)
```

```
mesaje = []
```

```
@app.route('/', methods=['GET', 'POST'])
def index():
    mesaj = ""
    if request.method == 'POST':
        mesaj = "Hello World"
        mesaje.append(mesaj)
    return render_template('index.html', mesaj=mesaj,
mesaje=mesaje)
```

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

Va veti lovi de o eroare in continuare.



```
mesaj.py  index.html

from flask import Flask, render_template, request
Unresolved reference 'flask' ...
Install package flask  More actions...
app = F

mesaje = []

@app.route('/', methods=['GET', 'POST'])
def index():
    mesaj = ""
    if request.method == 'POST':
        mesaj = "Hello World"
        mesaje.append(mesaj)
    return render_template('index.html', mesaj=mesaj, mesaje=mesaje)

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

Eroare se refera la faptul ca codul mesaj.py nu stie de libraria Flask fiind externa. Din fericire, Pycharm ul poate instala automat libraria in proiect. Dati hover cu mouse-ul pe eroarea flask si apasati pe butonul albastru Install Package.

Dupa rulati mesaj.py. Ar trebui sa aveti urmatorul output.



```
Run mesaj
Up   * Serving Flask app 'mesaj'
Down * Debug mode: on
≡   WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
≡   * Running on http://127.0.0.1:5000
≡   Press CTRL+C to quit
≡   * Restarting with stat
≡   * Debugger is active!
≡   * Debugger PIN: 477-598-602
```

8. Tocmai ati creat un server care ruleaza pe adresa <http://127.0.0.1:5000> (Adresa locala pentru lucru a calculatorului, toata lumea o are la fel, se mai numeste loopback)

Acum daca scrieti pe internet acea adresa veti vedea acel index.html (Nu va ganditi ca o sa mearga daca va puneti prietenii sa intre, este vorba de o adresa locala la nivel de calculator nici macar retea.)

Totusi insa butonul tot nu face nimic. In spate el face ceva. La fiecare apasare vectorul 'mesaje' este populat cu string-ul "Hello World".

```
mesaje = []
.
.
.
if request.method == 'POST':
    mesaj = "Hello World"
    mesaje.append(mesaj)
```

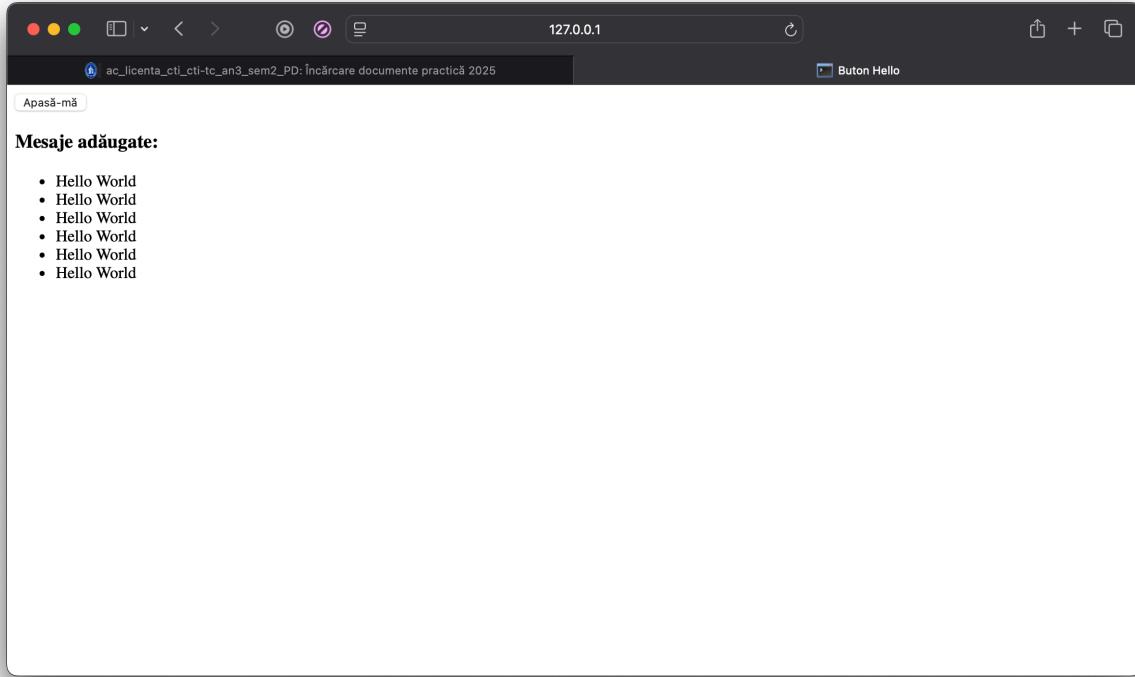
Adica daca avem mesaje = ["Hello World"] dupa mesaje.append(mesaj) vom avea ["Hello World", "Hello World"].

Acum sa punem in index.html feedback pentru buton. Adaugati in index.html urmatoare chestie.



```
4      <meta charset="utf-8" >
5      <title>Buton Hello</title>
6  </head>
7  <body>
8      <form method="POST">
9          <button type="submit">Apasă-mă</button>
10     </form>
11
12     <h3>Mesaje adăugate:</h3>
13     <ul>
14         {% for m in mesaje %}
15             <li>{{ m }}</li>
16         {% endfor %}
17     </ul>
18
19 </body>
</html>
```

Rulati din nou mesaj.py si accesati <http://127.0.0.1:5000>



Butonul ar trebui sa mearga si sa adauge mesajele. De aici puteti face o mica corelatie cu ce ar trebui noi sa facem. Spre exemplu pana acum am facut o functie de adaugat mesaje si de a le tine intr o variabila numita mesaje. Aceeasi maniera poate fi pentru semnale s.a.m.d.

That's it.