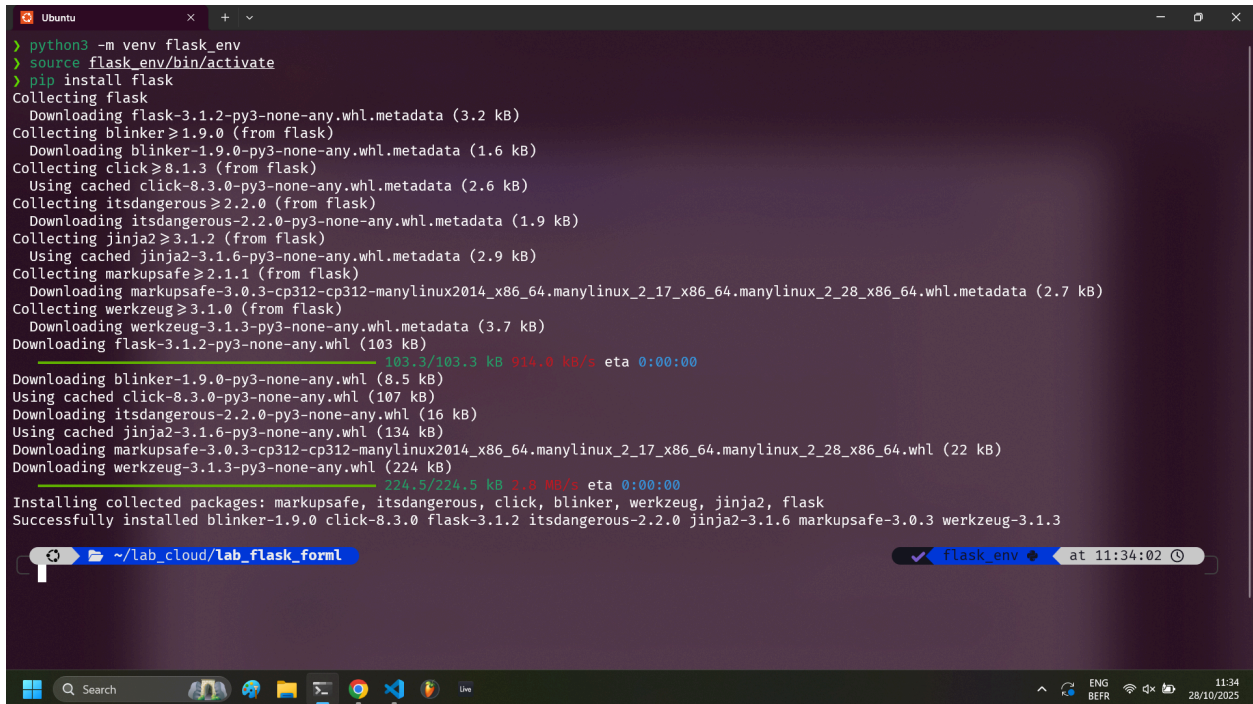


Installing Flask



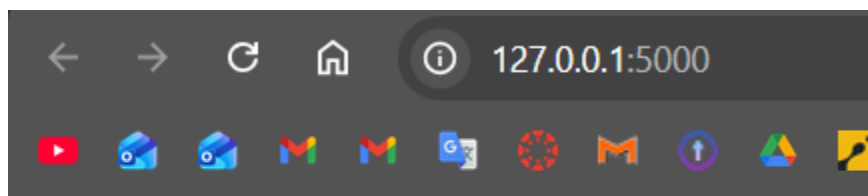
```
> python3 -m venv flask_env
> source flask_env/bin/activate
> pip install flask
Collecting flask
  Downloading flask-3.1.2-py3-none-any.whl.metadata (3.2 kB)
Collecting blinker>=1.9.0 (from flask)
  Downloading blinker-1.9.0-py3-none-any.whl.metadata (1.6 kB)
Collecting click>=8.1.3 (from flask)
  Using cached click-8.3.0-py3-none-any.whl.metadata (2.6 kB)
Collecting itsdangerous>=2.2.0 (from flask)
  Downloading itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting jinja2>=3.1.2 (from flask)
  Using cached jinja2-3.1.6-py3-none-any.whl.metadata (2.9 kB)
Collecting markupsafe>=2.1.1 (from flask)
  Downloading markupsafe-3.0.3-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl.metadata (2.7 kB)
Collecting werkzeug>=3.1.0 (from flask)
  Downloading werkzeug-3.1.3-py3-none-any.whl.metadata (3.7 kB)
Download flask-3.1.2-py3-none-any.whl (103 kB)
103.3/103.3 kB 914.0 kB/s eta 0:00:00
Download blinker-1.9.0-py3-none-any.whl (8.5 kB)
Using cached click-8.3.0-py3-none-any.whl (107 kB)
Download itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Using cached jinja2-3.1.6-py3-none-any.whl (134 kB)
Download markupsafe-3.0.3-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl (22 kB)
Download werkzeug-3.1.3-py3-none-any.whl (224 kB)
224.5/224.5 kB 3.0 MB/s eta 0:00:00
Installing collected packages: markupsafe, itsdangerous, click, blinker, werkzeug, jinja2, flask
Successfully installed blinker-1.9.0 click-8.3.0 flask-3.1.2 itsdangerous-2.2.0 jinja2-3.1.6 markupsafe-3.0.3 werkzeug-3.1.3
```

The screenshot shows a terminal window with the following elements:

- Terminal Title Bar:** Ubuntu
- Command Prompt:** A series of commands to create a virtual environment, activate it, and install Flask.
- Output:** Detailed output from pip showing the collection and downloading of Flask and its dependencies (blinker, click, itsdangerous, jinja2, markupsafe, werkzeug).
- Progress Bars:** Visual progress indicators for the downloads of flask-3.1.2-py3-none-any.whl and werkzeug-3.1.3-py3-none-any.whl.
- Final Status:** A message indicating that all packages were successfully installed.
- Terminal UI:** Includes a file explorer bar at the bottom showing the current directory as ~/lab_cloud/lab_flask_form1 and a status bar at the bottom right showing the time as 11:34 and date as 28/10/2025.

Example 1:

```
hello.py ×  
hello.py > ...  
1  from flask import Flask  
2  
3  app = Flask(__name__)  
4  
5  @app.route('/')  
6  def hello_world():  
7      return 'Hello, World!'  
8  
9  if __name__ == '__main__':  
10     app.run()  
11
```

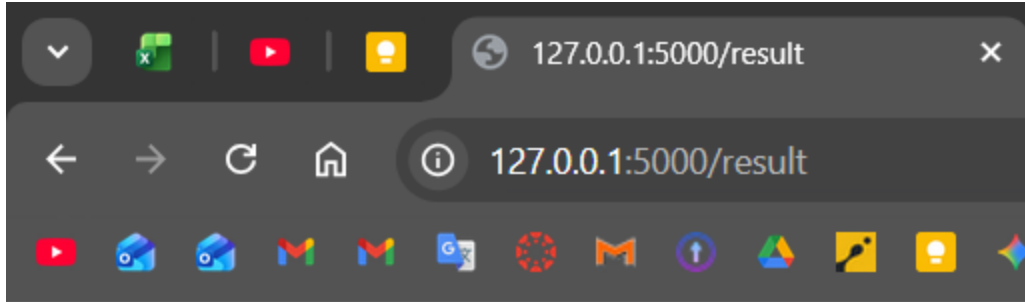


Hello, World!

Example 2:

```
app.py x
app.py > ...
1  from flask import Flask, render_template
2
3  app = Flask(__name__)
4
5  @app.route('/result')
6  def result():
7      dict = {'phy':50, 'che':60, 'maths':70}
8      return render_template('result.html', result=dict)
9
10 if __name__ == '__main__':
11     app.run(debug=True)
12
```

```
app.py  results.html x
templates > results.html > ...
1  <!doctype html>
2  <html>
3      <body>
4          <table border = 1>
5              {% for key, value in result.items() %}
6                  <tr>
7                      <th>{{ key }}</th>
8                      <td>{{ value }}</td>
9                  </tr>
10             {% endfor %}
11         </table>
12     </body>
13 </html>
14
```

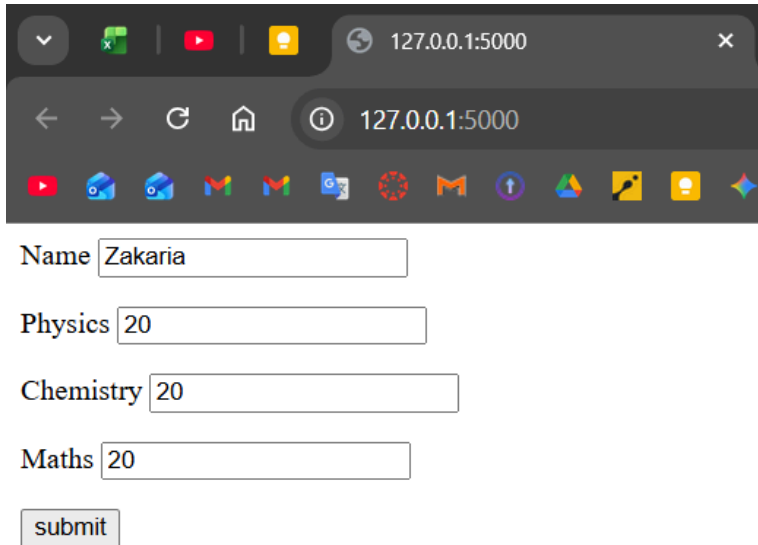


phy	50
che	60
maths	70

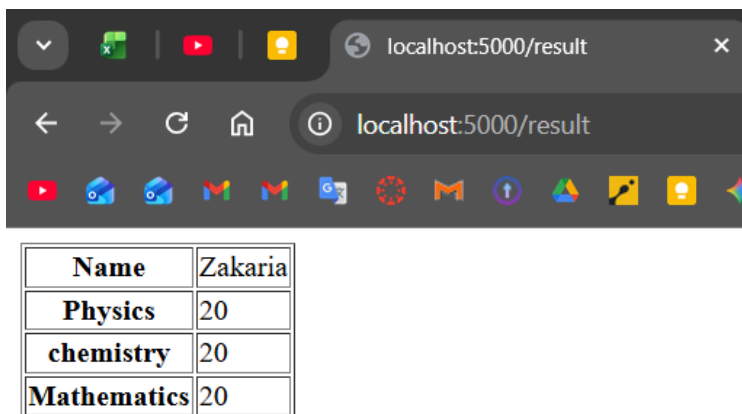
Example 3:

```
app.py
app.py > ...
1  from flask import Flask, render_template, request
2
3  app = Flask(__name__)
4
5  @app.route('/')
6  def student():
7      ... return render_template('student.html')
8
9  @app.route('/result', methods=['POST', 'GET'])
10 def result():
11     ... if request.method == 'POST':
12         ... result = request.form
13         ... return render_template('result.html', result=result)
14
15 if __name__ == '__main__':
16     ... app.run(debug=True)
17
```

Zakaria CHOUKRI



A screenshot of a web browser window. The address bar shows '127.0.0.1:5000'. The page contains a form with four input fields: 'Name' (containing 'Zakaria'), 'Physics' (containing '20'), 'Chemistry' (containing '20'), and 'Maths' (containing '20'). Below these fields is a 'submit' button.



A screenshot of a web browser window. The address bar shows 'localhost:5000/result'. The page displays a table with the following data:

Name	Zakaria
Physics	20
chemistry	20
Mathematics	20

Heroku part:

I couldn't do the Heroku part because it needs a credit card:

heroku create furniture-prediction-app --region eu

Assignment:

1. Choose a dataset, containing tabular data, of your choice
I chose the Iris dataset

```
1 import pandas as pd
2 import numpy as np
3 from sklearn.datasets import load_iris
4 from sklearn.model_selection import train_test_split
5 from sklearn.preprocessing import StandardScaler
6 from sklearn.linear_model import LogisticRegression
7 from sklearn.ensemble import RandomForestClassifier
8 from sklearn.svm import SVC
9 from sklearn.metrics import accuracy_score
10 import joblib
11
12 iris = load_iris()
13 X = pd.DataFrame(iris.data, columns=iris.feature_names)
14 y = pd.Series(iris.target, name='target')
15
16 print(f"Dataset shape: {X.shape}")
17 print(f"Missing values: {X.isnull().sum().sum()}")
18 print(f"Target distribution:\n{y.value_counts()}")
```

2. Prepare your data (treatment of missing values, etc.)

There is nothing to be done

```
Dataset shape: (150, 4)
Missing values: 0
Target distribution:
target
0      50
1      50
2      50
Name: count, dtype: int64
```

4. Use these characteristics to create the prediction page (like in the example above)

Iris Flower Classification

Sepal Length (cm)

Sepal Width (cm)

Petal Length (cm)

Petal Width (cm)

5. Compare the performance of multiple ML models to keep the best one at the end.

```
1 joblib.dump(best_model, 'model.pkl')
2 joblib.dump(scaler, 'scaler.pkl')
3 joblib.dump(iris.target_names, 'target_names.pkl')
4 print("Model, scaler, and target names saved!")
```

✓ 0.0s

Model, scaler, and target names saved!

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
2
3 scaler = StandardScaler()
4 X_train_scaled = scaler.fit_transform(X_train)
5 X_test_scaled = scaler.transform(X_test)
6
7 models = {
8     'LogisticRegression': LogisticRegression(max_iter=200, random_state=0),
9     'RandomForest': RandomForestClassifier(n_estimators=100, random_state=0),
10    'SVM': SVC(kernel='rbf', random_state=0)
11 }
12
13 scores = {}
14 for name, model in models.items():
15     model.fit(X_train_scaled, y_train)
16     pred = model.predict(X_test_scaled)
17     acc = accuracy_score(y_test, pred)
18     scores[name] = acc
19     print(f"{name}: {acc:.4f}")
20
21 best_model_name = max(scores, key=scores.get)
22 best_model = models[best_model_name]
23 print(f"\nBest model: {best_model_name} with accuracy {scores[best_model_name]:.4f}")
```

[2]

```
... LogisticRegression: 1.0000
RandomForest: 1.0000
SVM: 1.0000

Best model: LogisticRegression with accuracy 1.0000
```

Python

6. Create an application (i.e., a form) that predicts the target (value or class) of a record.

Iris Flower Classification

Sepal Length (cm)

Sepal Width (cm)

Petal Length (cm)

Petal Width (cm)

Predict

Predicted Iris Class: virginica