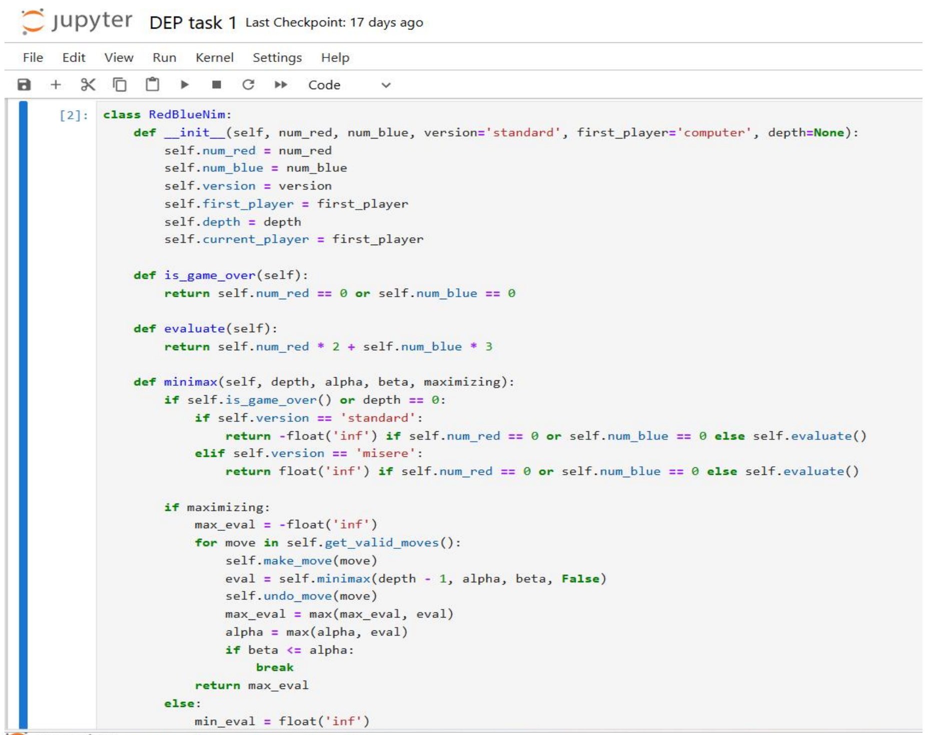
❖ DEP VIRTUAL INTERNSHIP PROGRAM

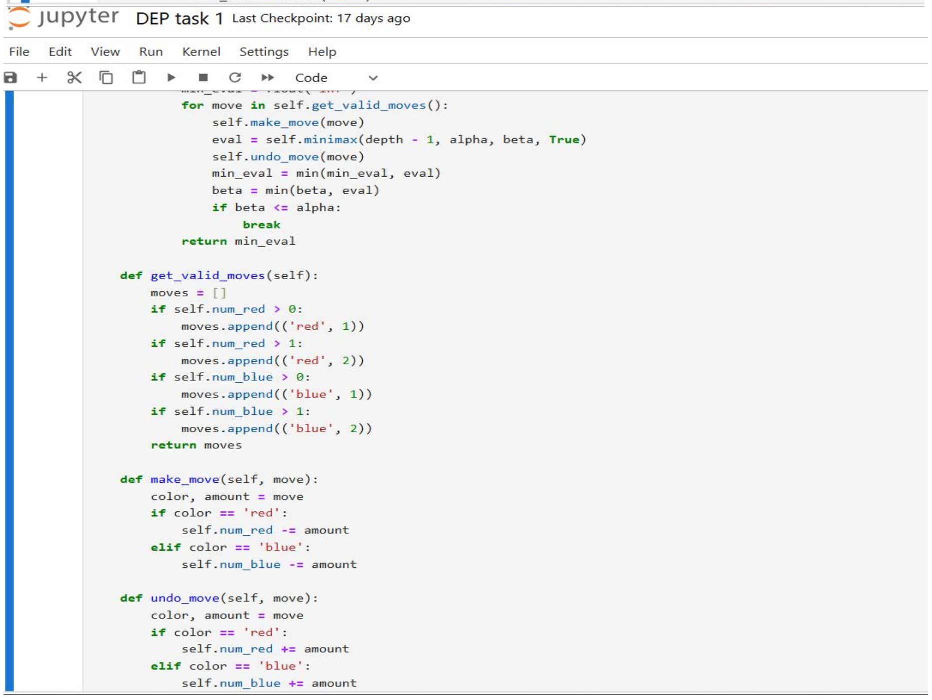
➢ Task 1 :

✧ Provide an overview of the Red-Blue Nim Game and explain the two game versions (Standard and Misère).Highlight the objectives and goals of implementing this game in Python.

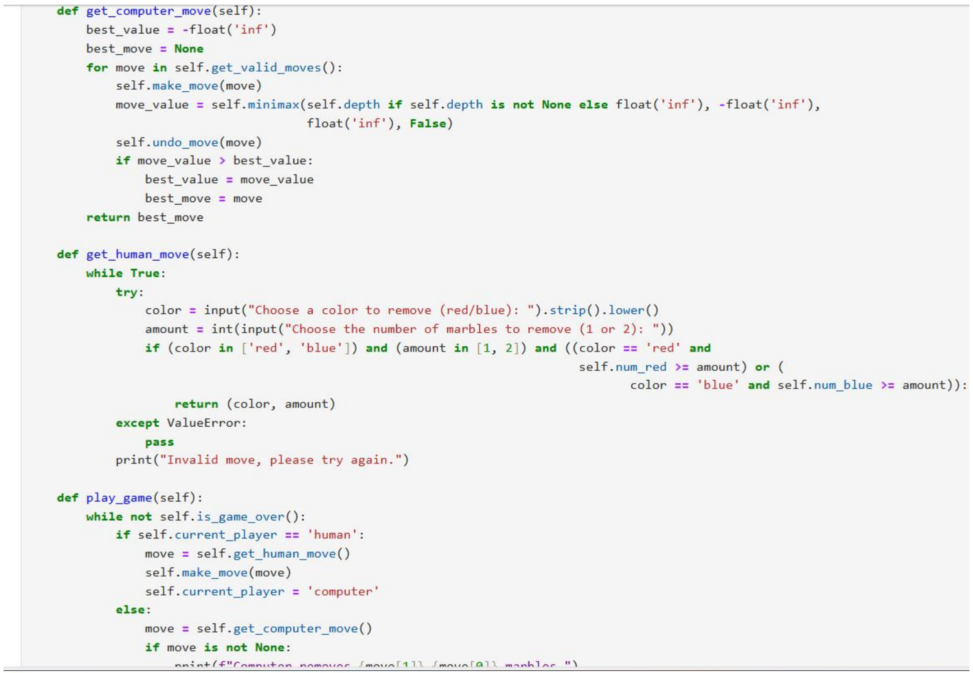
➢ Code :


jupyter DEP task 1 Last Checkpoint: 17 days ago
File Edit View Run Kernel Settings Help
Code

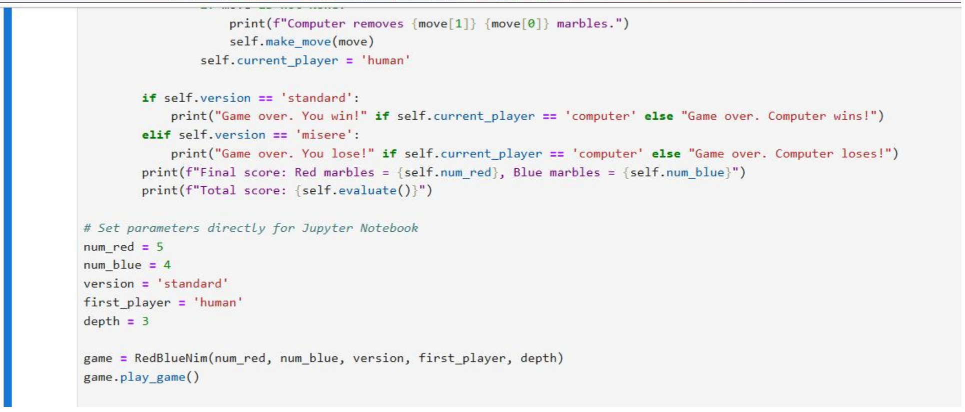
```
[2]: class RedBlueNim:
    def __init__(self, num_red, num_blue, version='standard', first_player='computer', depth=None):
        self.num_red = num_red
        self.num_blue = num_blue
        self.version = version
        self.first_player = first_player
        self.depth = depth
        self.current_player = first_player

    def is_game_over(self):
        return self.num_red == 0 or self.num_blue == 0

    def evaluate(self):
        return self.num_red * 2 + self.num_blue * 3

    def minimax(self, depth, alpha, beta, maximizing):
        if self.is_game_over() or depth == 0:
            if self.version == 'standard':
                return -float('inf') if self.num_red == 0 or self.num_blue == 0 else self.evaluate()
            elif self.version == 'misere':
                return float('inf') if self.num_red == 0 or self.num_blue == 0 else self.evaluate()

        if maximizing:
            max_eval = -float('inf')
            for move in self.get_valid_moves():
                self.make_move(move)
                eval = self.minimax(depth - 1, alpha, beta, False)
                self.undo_move(move)
                max_eval = max(max_eval, eval)
                alpha = max(alpha, eval)
                if beta <= alpha:
                    break
            return max_eval
        else:
            min_eval = float('inf')
```


jupyter DEP task 1 Last Checkpoint: 17 days ago
File Edit View Run Kernel Settings Help
Code

```
            for move in self.get_valid_moves():
                self.make_move(move)
                eval = self.minimax(depth - 1, alpha, beta, True)
                self.undo_move(move)
                min_eval = min(min_eval, eval)
                beta = min(beta, eval)
                if beta <= alpha:
                    break
            return min_eval

    def get_valid_moves(self):
        moves = []
        if self.num_red > 0:
            moves.append(('red', 1))
        if self.num_red > 1:
            moves.append(('red', 2))
        if self.num_blue > 0:
            moves.append(('blue', 1))
        if self.num_blue > 1:
            moves.append(('blue', 2))
        return moves

    def make_move(self, move):
        color, amount = move
        if color == 'red':
            self.num_red -= amount
        elif color == 'blue':
            self.num_blue -= amount

    def undo_move(self, move):
        color, amount = move
        if color == 'red':
            self.num_red += amount
        elif color == 'blue':
            self.num_blue += amount
```

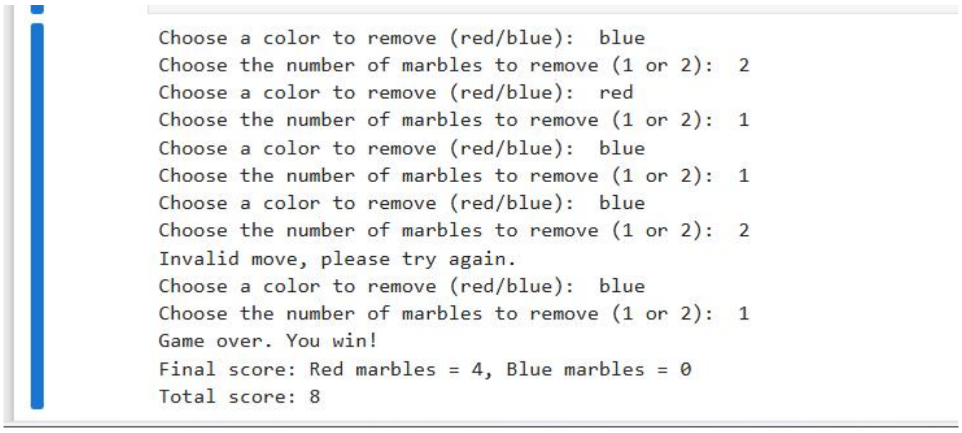


```
    def get_computer_move(self):
        best_value = -float('inf')
        best_move = None
        for move in self.get_valid_moves():
            self.make_move(move)
            move_value = self.minimax(self.depth if self.depth is not None else float('inf'), -float('inf'),
                                      float('inf'), False)
            self.undo_move(move)
            if move_value > best_value:
                best_value = move_value
                best_move = move
        return best_move

    def get_human_move(self):
        while True:
            try:
                color = input("Choose a color to remove (red/blue): ").strip().lower()
                amount = int(input("Choose the number of marbles to remove (1 or 2): "))
                if (color in ['red', 'blue']) and (amount in [1, 2]) and ((color == 'red' and
                                                                          self.num_red >= amount) or (
                                                                              color == 'blue' and self.num_blue >= amount)):
                    return (color, amount)
            except ValueError:
                pass
            print("Invalid move, please try again.")

    def play_game(self):
        while not self.is_game_over():
            if self.current_player == 'human':
                move = self.get_human_move()
                self.make_move(move)
                self.current_player = 'computer'
            else:
                move = self.get_computer_move()
                if move is not None:
```



```
                    print(f"Computer removes {move[1]} {move[0]} marbles.")
                    self.make_move(move)
                self.current_player = 'human'

        if self.version == 'standard':
            print("Game over. You win!" if self.current_player == 'computer' else "Game over. Computer wins!")
        elif self.version == 'misere':
            print("Game over. You lose!" if self.current_player == 'computer' else "Game over. Computer loses!")
        print(f"Final score: Red marbles = {self.num_red}, Blue marbles = {self.num_blue}")
        print(f"Total score: {self.evaluate()}")

# Set parameters directly for Jupyter Notebook
num_red = 5
num_blue = 4
version = 'standard'
first_player = 'human'
depth = 3

game = RedBlueNim(num_red, num_blue, version, first_player, depth)
game.play_game()
```

➢ Output :

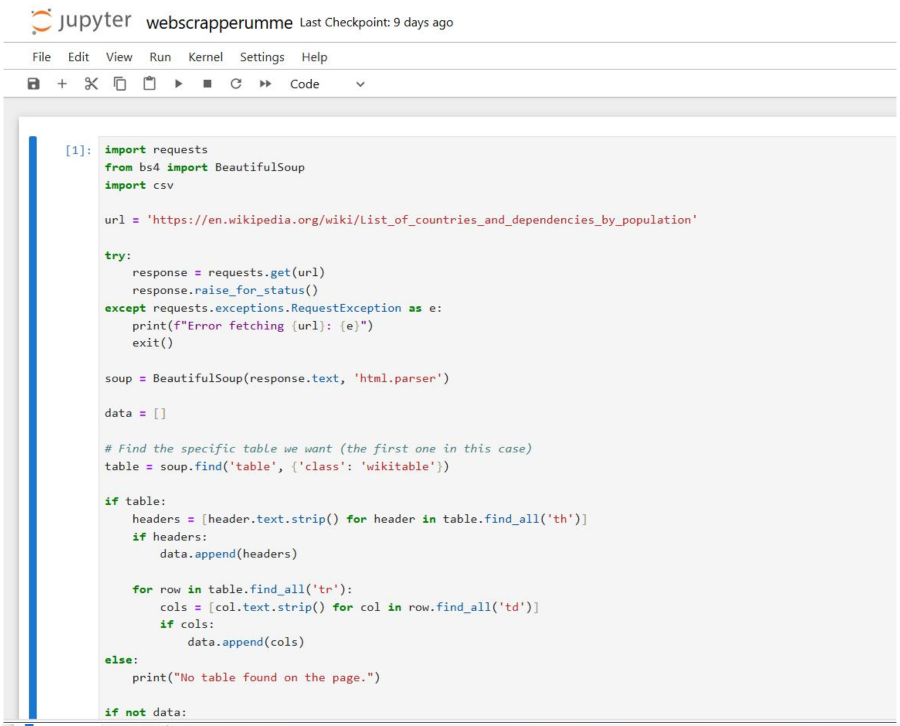


```
Choose a color to remove (red/blue):  blue
Choose the number of marbles to remove (1 or 2):  2
Choose a color to remove (red/blue):  red
Choose the number of marbles to remove (1 or 2):  1
Choose a color to remove (red/blue):  blue
Choose the number of marbles to remove (1 or 2):  1
Choose a color to remove (red/blue):  blue
Choose the number of marbles to remove (1 or 2):  2
Invalid move, please try again.
Choose a color to remove (red/blue):  blue
Choose the number of marbles to remove (1 or 2):  1
Game over. You win!
Final score: Red marbles = 4, Blue marbles = 0
Total score: 8
```

➢ Task 2:

Create a web scraper using libraries like Beautiful Soup and requests to extract data from a website and store it in a CSV file.

➢ Code :

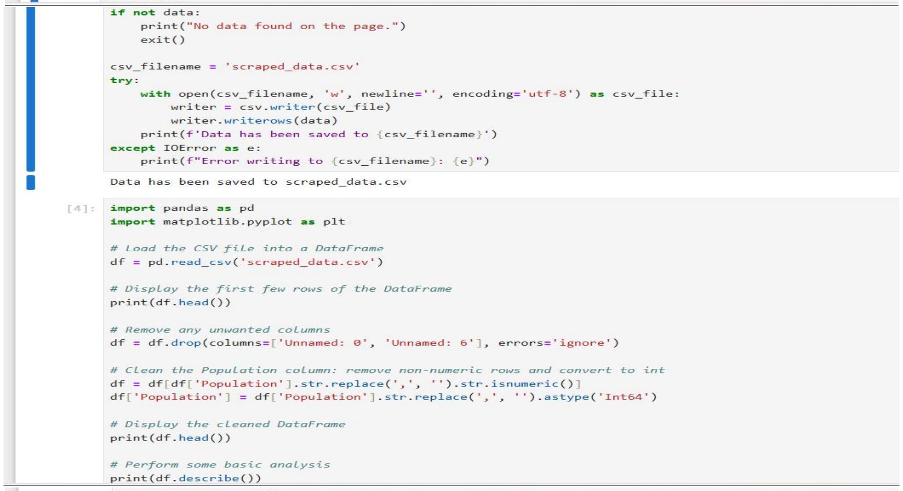


jupyter webscrapperumme Last Checkpoint: 9 days ago

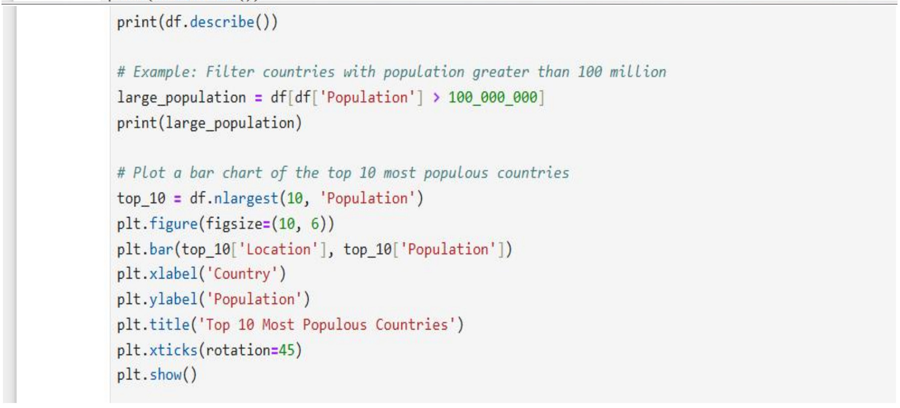
File Edit View Run Kernel Settings Help

Code

```
[1]: import requests
     from bs4 import BeautifulSoup
     import csv

     url = 'https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population'

     try:
         response = requests.get(url)
         response.raise_for_status()
     except requests.exceptions.RequestException as e:
         print(f"Error fetching {url}: {e}")
         exit()

     soup = BeautifulSoup(response.text, 'html.parser')

     data = []

     # Find the specific table we want (the first one in this case)
     table = soup.find('table', {'class': 'wikitable'})

     if table:
         headers = [header.text.strip() for header in table.find_all('th')]
         if headers:
             data.append(headers)

         for row in table.find_all('tr'):
             cols = [col.text.strip() for col in row.find_all('td')]
             if cols:
                 data.append(cols)
     else:
         print("No table found on the page.")

     if not data:
```

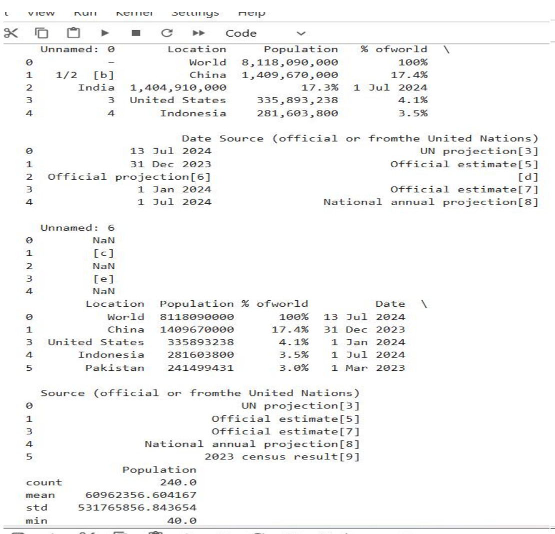


```
     if not data:
         print("No data found on the page.")
         exit()

     csv_filename = 'scraped_data.csv'
     try:
         with open(csv_filename, 'w', newline='', encoding='utf-8') as csv_file:
             writer = csv.writer(csv_file)
             writer.writerows(data)
         print(f'Data has been saved to {csv_filename}')
     except IOError as e:
         print(f"Error writing to {csv_filename}: {e}")

     Data has been saved to scraped_data.csv

[4]: import pandas as pd
     import matplotlib.pyplot as plt

     # Load the CSV file into a DataFrame
     df = pd.read_csv('scraped_data.csv')

     # Display the first few rows of the DataFrame
     print(df.head())

     # Remove any unwanted columns
     df = df.drop(columns=['Unnamed: 0', 'Unnamed: 6'], errors='ignore')

     # Clean the Population column: remove non-numeric rows and convert to int
     df = df[df['Population'].str.replace(',', '').str.isnumeric()]
     df['Population'] = df['Population'].str.replace(',', '').astype('Int64')

     # Display the cleaned DataFrame
     print(df.head())

     # Perform some basic analysis
     print(df.describe())
```

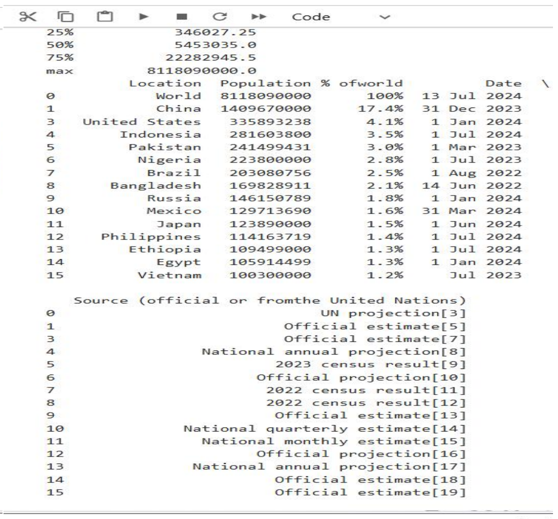


```
     print(df.describe())

     # Example: Filter countries with population greater than 100 million
     large_population = df[df['Population'] > 100_000_000]
     print(large_population)

     # Plot a bar chart of the top 10 most populous countries
     top_10 = df.nlargest(10, 'Population')
     plt.figure(figsize=(10, 6))
     plt.bar(top_10['Location'], top_10['Population'])
     plt.xlabel('Country')
     plt.ylabel('Population')
     plt.title('Top 10 Most Populous Countries')
     plt.xticks(rotation=45)
     plt.show()
```

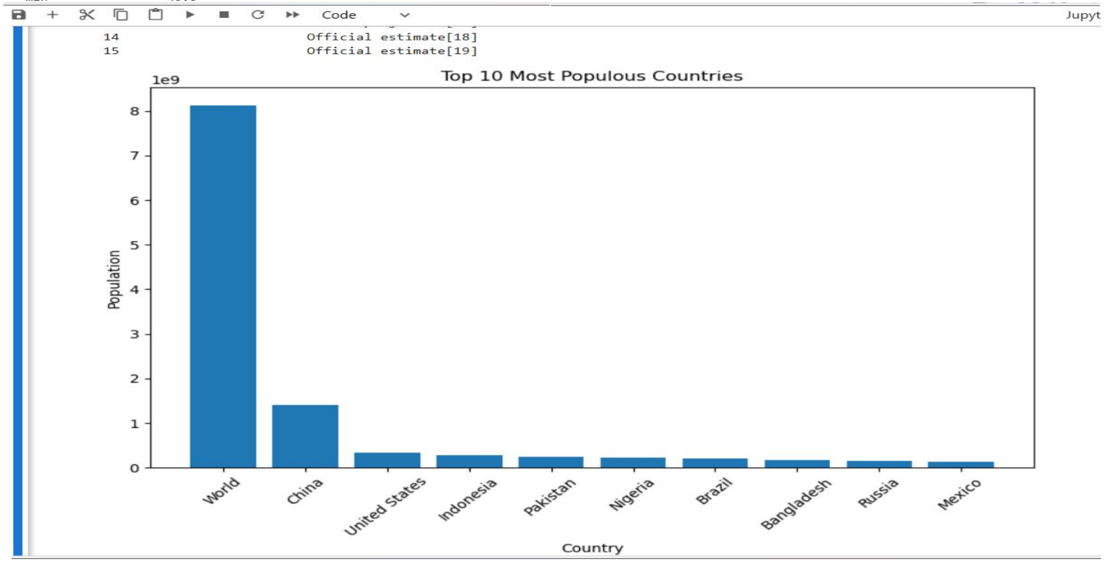
➢ Output :



```
  Unnamed: 0        Location      Population   % ofworld  \
0          -           World   8,118,090,000        100%
1   1/2  [b]           China   1,409,670,000       17.4%
2      India   1,404,910,000           17.3%  1 Jul 2024
3          3   United States     335,893,238        4.1%
4          4       Indonesia     281,603,800        3.5%

                Date Source (official or fromthe United Nations)
0          13 Jul 2024                              UN projection[3]
1          31 Dec 2023                          Official estimate[5]
2  Official projection[6]                                        [d]
3           1 Jan 2024                          Official estimate[7]
4           1 Jul 2024                National annual projection[8]

  Unnamed: 6
0        NaN
1        [c]
2        NaN
3        [e]
4        NaN
        Location  Population % ofworld         Date  \
0          World  8118090000      100%  13 Jul 2024
1          China  1409670000     17.4%  31 Dec 2023
3  United States   335893238      4.1%   1 Jan 2024
4      Indonesia   281603800      3.5%   1 Jul 2024
5       Pakistan   241499431      3.0%   1 Mar 2023

  Source (official or fromthe United Nations)
0                            UN projection[3]
1                        Official estimate[5]
3                        Official estimate[7]
4               National annual projection[8]
5                       2023 census result[9]
           Population
count           240.0
mean    60962356.604167
std    531765856.843654
min             40.0
```



```
25%         346027.25
50%         5453035.0
75%        22282945.5
max      8118090000.0
         Location  Population % ofworld         Date  \
0           World  8118090000      100%  13 Jul 2024
1           China  1409670000     17.4%  31 Dec 2023
3   United States   335893238      4.1%   1 Jan 2024
4       Indonesia   281603800      3.5%   1 Jul 2024
5        Pakistan   241499431      3.0%   1 Mar 2023
6         Nigeria   223800000      2.8%   1 Jul 2023
7          Brazil   203080756      2.5%   1 Aug 2022
8      Bangladesh   169828911      2.1%  14 Jun 2022
9          Russia   146150789      1.8%   1 Jan 2024
10         Mexico   129713690      1.6%  31 Mar 2024
11          Japan   123890000      1.5%   1 Jun 2024
12    Philippines   114163719      1.4%   1 Jul 2024
13       Ethiopia   109499000      1.3%   1 Jul 2024
14          Egypt   105914499      1.3%   1 Jan 2024
15        Vietnam   100300000      1.2%     Jul 2023

   Source (official or fromthe United Nations)
0                             UN projection[3]
1                         Official estimate[5]
3                         Official estimate[7]
4                National annual projection[8]
5                        2023 census result[9]
6                      Official projection[10]
7                       2022 census result[11]
8                       2022 census result[12]
9                        Official estimate[13]
10            National quarterly estimate[14]
11              National monthly estimate[15]
12                     Official projection[16]
13             National annual projection[17]
14                       Official estimate[18]
15                       Official estimate[19]
```

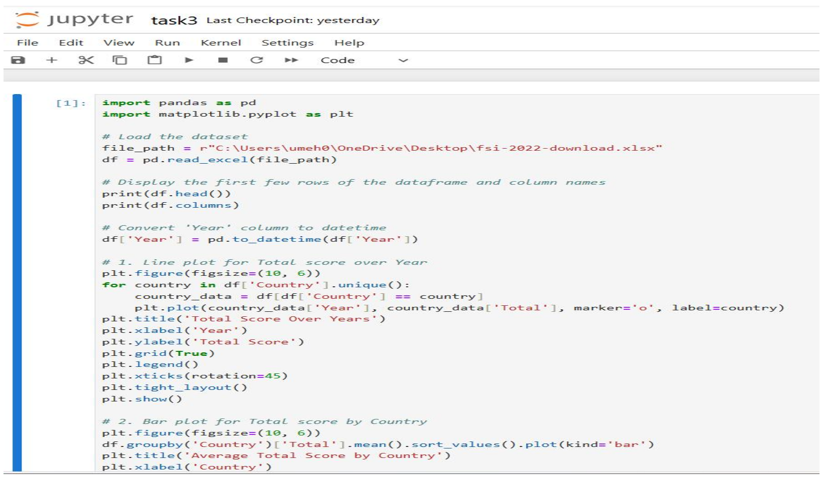


```
     14                       Official estimate[18]
     15                       Official estimate[19]
```

➢ Task 3:

Implement a data analysis project using pandas and matplotlib to explore and visualize a dataset of your choice.

➢ Code :

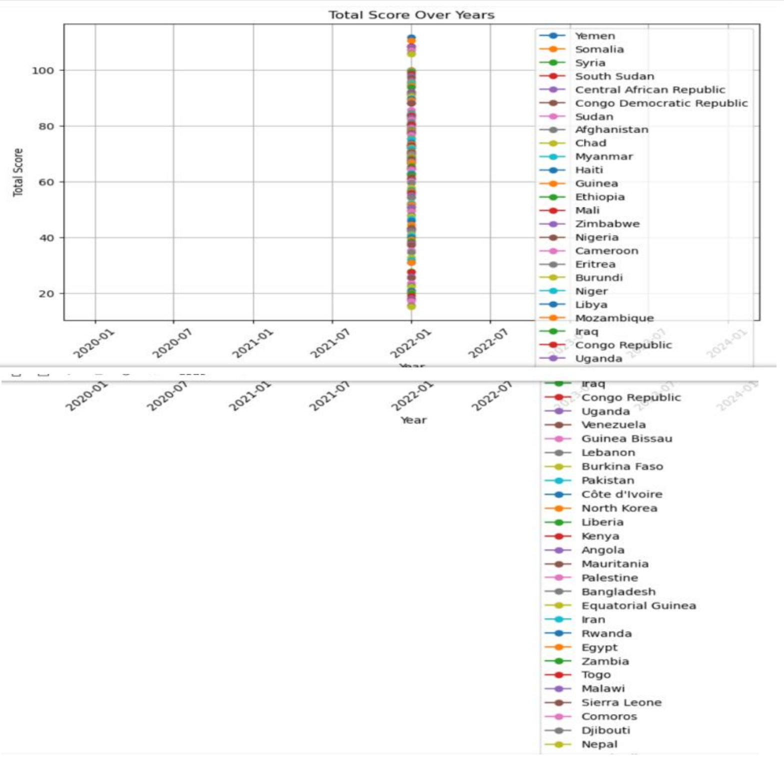


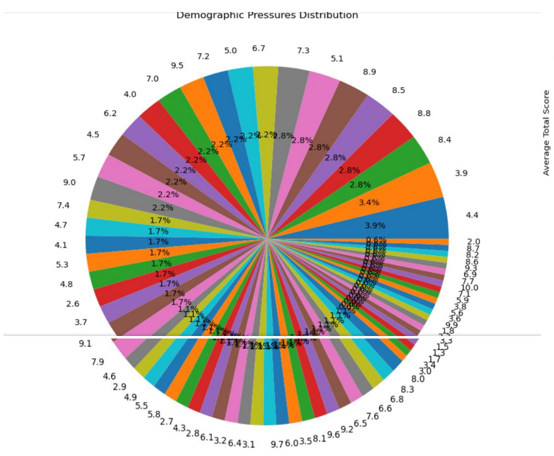
jupyter task3 Last Checkpoint: yesterday
File Edit View Run Kernel Settings Help
Code

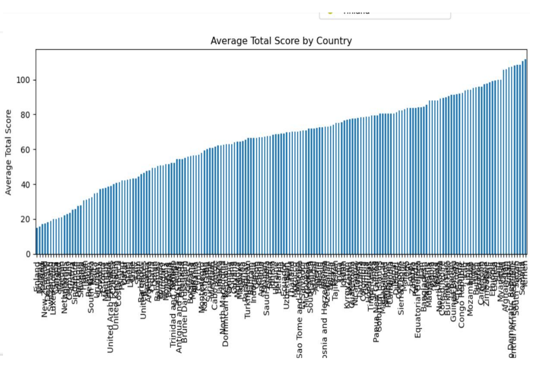
```python
[1]: import pandas as pd
     import matplotlib.pyplot as plt

     # Load the dataset
     file_path = r"C:\Users\umeh0\OneDrive\Desktop\fsi-2022-download.xlsx"
     df = pd.read_excel(file_path)

     # Display the first few rows of the dataframe and column names
     print(df.head())
     print(df.columns)

     # Convert 'Year' column to datetime
     df['Year'] = pd.to_datetime(df['Year'])

     # 1. Line plot for Total score over Year
     plt.figure(figsize=(10, 6))
     for country in df['Country'].unique():
         country_data = df[df['Country'] == country]
         plt.plot(country_data['Year'], country_data['Total'], marker='o', label=country)
     plt.title('Total Score Over Years')
     plt.xlabel('Year')
     plt.ylabel('Total Score')
     plt.grid(True)
     plt.legend()
     plt.xticks(rotation=45)
     plt.tight_layout()
     plt.show()

     # 2. Bar plot for Total score by Country
     plt.figure(figsize=(10, 6))
     df.groupby('Country')['Total'].mean().sort_values().plot(kind='bar')
     plt.title('Average Total Score by Country')
     plt.xlabel('Country')
plt.figure(figsize=(10, 6))
df.groupby('Country')['Total'].mean().sort_values().plot(kind='bar')
plt.title('Average Total Score by Country')
plt.xlabel('Country')
plt.ylabel('Average Total Score')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()

# 3. Pie chart for Demographic Pressures distribution
plt.figure(figsize=(8, 8))
df['S1: Demographic Pressures'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Demographic Pressures Distribution')
plt.ylabel('')
plt.tight_layout()
plt.show()

# 4. Scatter plot for Total vs. Economic Inequality
plt.figure(figsize=(10, 6))
plt.scatter(df['Total'], df['E2: Economic Inequality'])
plt.title('Total Score vs. Economic Inequality')
plt.xlabel('Total Score')
plt.ylabel('Economic Inequality')
plt.grid(True)
plt.tight_layout()
plt.show()

# 5. Histogram for Total Score Distribution
plt.figure(figsize=(10, 6))
df['Total'].plot(kind='hist', bins=20, edgecolor='black')
plt.title('Total Score Distribution')
plt.xlabel('Total Score')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```

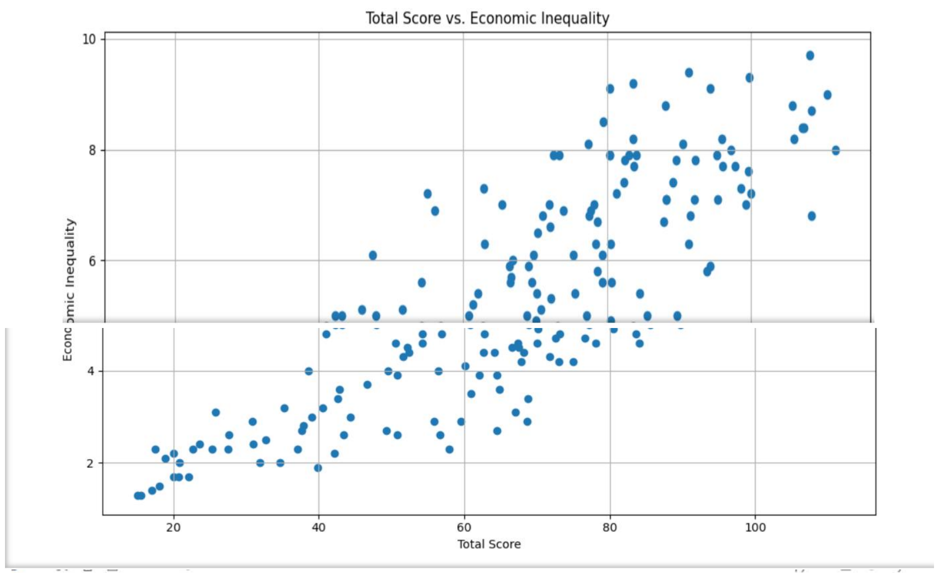
➢ Output :

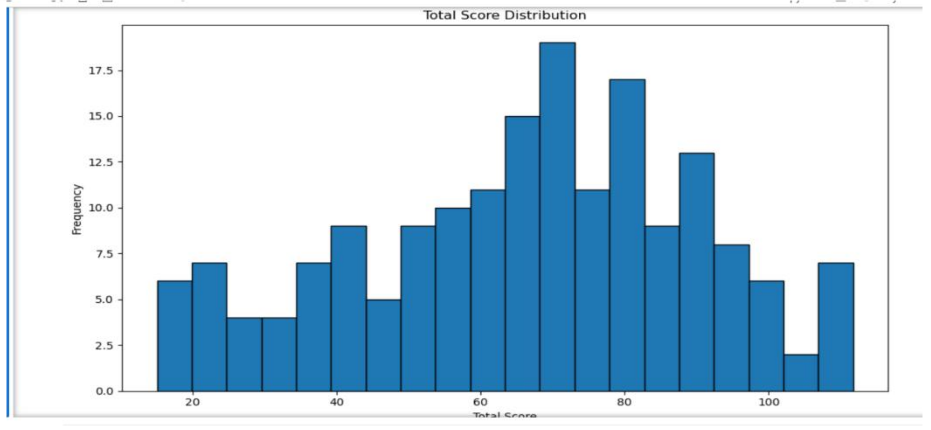
```
                    Country        Year Rank  Total  C1: Security Apparatus  \
0                     Yemen  2022-01-01  1st  111.7                     9.1
1                   Somalia  2022-01-01  2nd  110.5                     9.4
2                     Syria  2022-01-01  3rd  108.4                     9.5
3               South Sudan  2022-01-01  3rd  108.4                     9.8
4  Central African Republic  2022-01-01  5th  108.1                     8.3

   C2: Factionalized Elites  C3: Group Grievance  E1: Economy  \
0                      10.0                  9.1          9.9
1                      10.0                  8.4          9.1
2                       9.9                  9.4          9.3
3                       9.2                  8.5          8.9
4                       9.7                  8.4          8.2

   E2: Economic Inequality  E3: Human Flight and Brain Drain  \
0                      8.0                               6.7
1                      9.0                               8.7
2                      6.8                               8.1
3                      8.7                               6.6
4                      9.7                               6.5

   P1: State Legitimacy  P2: Public Services  P3: Human Rights  \
0                   9.9                  9.9               9.9
1                   9.5                  9.9               8.8
2                  10.0                  9.3               9.4
3                   9.6                  9.8               8.6
4                   9.2                 10.0               9.4

   S1: Demographic Pressures  S2: Refugees and IDPs  X1: External Intervention
0                        9.9                    9.9                        9.4
1                       10.0                    8.7                        9.0
2                        7.3                    9.4                       10.0
3                        9.6                   10.0                        9.1
```







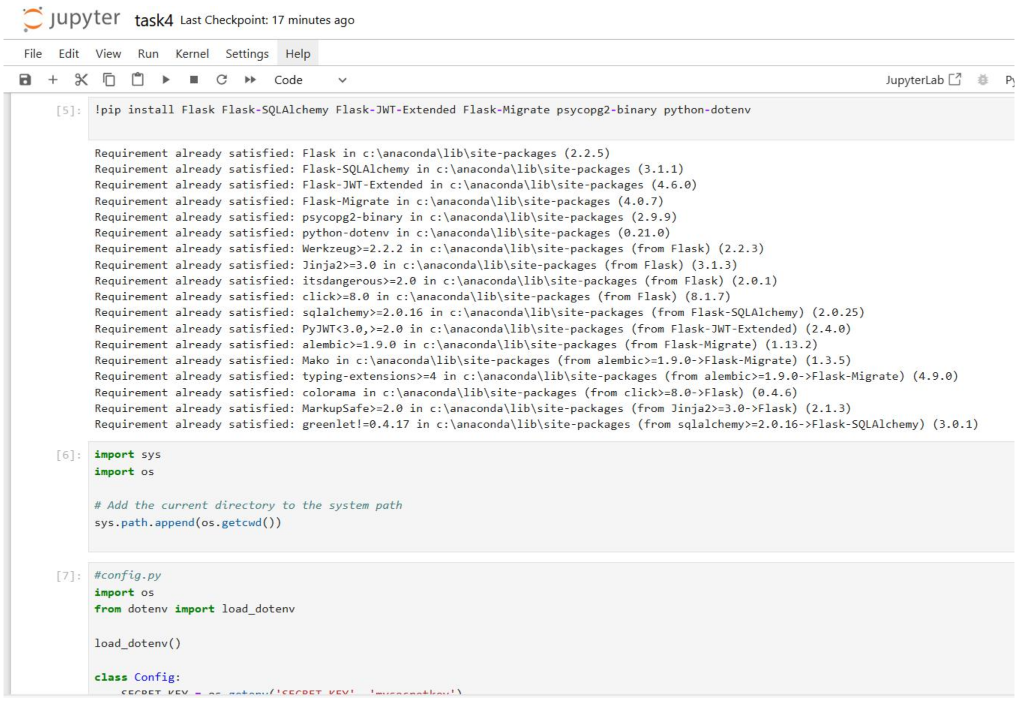




➢ Task 4:

Develop a RESTful API using Flask or Django to perform CRUD operations on a database and authenticate users.

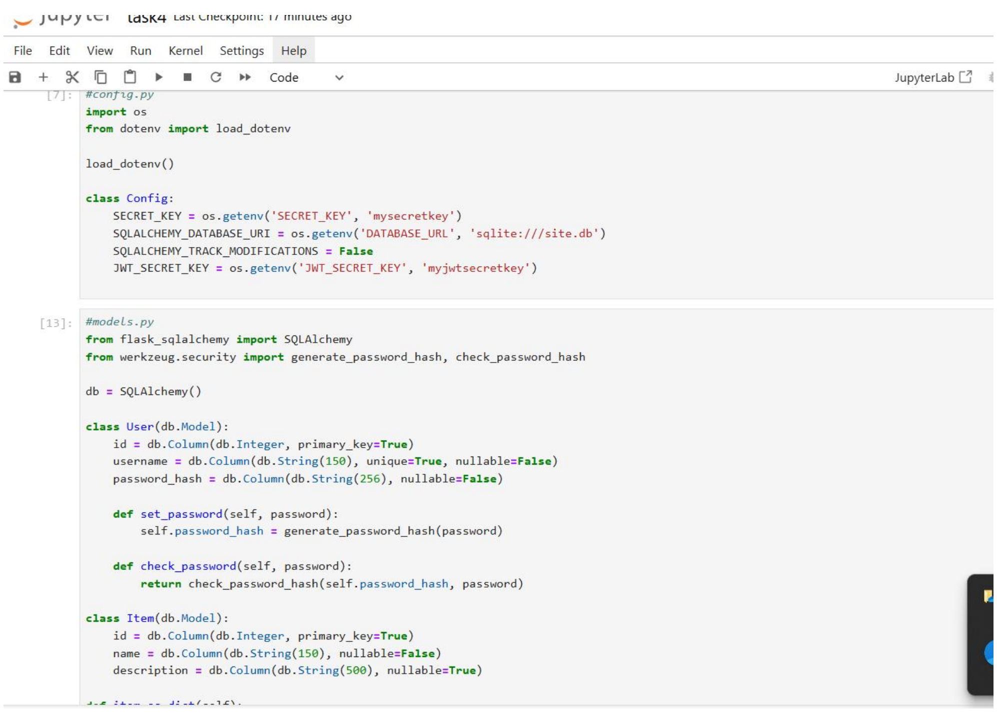
➢ Code :


jupyter task4 Last Checkpoint: 17 minutes ago
File Edit View Run Kernel Settings Help
Code
JupyterLab

```
[5]: !pip install Flask Flask-SQLAlchemy Flask-JWT-Extended Flask-Migrate psycopg2-binary python-dotenv

Requirement already satisfied: Flask in c:\anaconda\lib\site-packages (2.2.5)
Requirement already satisfied: Flask-SQLAlchemy in c:\anaconda\lib\site-packages (3.1.1)
Requirement already satisfied: Flask-JWT-Extended in c:\anaconda\lib\site-packages (4.6.0)
Requirement already satisfied: Flask-Migrate in c:\anaconda\lib\site-packages (4.0.7)
Requirement already satisfied: psycopg2-binary in c:\anaconda\lib\site-packages (2.9.9)
Requirement already satisfied: python-dotenv in c:\anaconda\lib\site-packages (0.21.0)
Requirement already satisfied: Werkzeug>=2.2.2 in c:\anaconda\lib\site-packages (from Flask) (2.2.3)
Requirement already satisfied: Jinja2>=3.0 in c:\anaconda\lib\site-packages (from Flask) (3.1.3)
Requirement already satisfied: itsdangerous>=2.0 in c:\anaconda\lib\site-packages (from Flask) (2.0.1)
Requirement already satisfied: click>=8.0 in c:\anaconda\lib\site-packages (from Flask) (8.1.7)
Requirement already satisfied: sqlalchemy>=2.0.16 in c:\anaconda\lib\site-packages (from Flask-SQLAlchemy) (2.0.25)
Requirement already satisfied: PyJWT<3.0,>=2.0 in c:\anaconda\lib\site-packages (from Flask-JWT-Extended) (2.4.0)
Requirement already satisfied: alembic>=1.9.0 in c:\anaconda\lib\site-packages (from Flask-Migrate) (1.13.2)
Requirement already satisfied: Mako in c:\anaconda\lib\site-packages (from alembic>=1.9.0->Flask-Migrate) (1.3.5)
Requirement already satisfied: typing-extensions>=4 in c:\anaconda\lib\site-packages (from alembic>=1.9.0->Flask-Migrate) (4.9.0)
Requirement already satisfied: colorama in c:\anaconda\lib\site-packages (from click>=8.0->Flask) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\anaconda\lib\site-packages (from Jinja2>=3.0->Flask) (2.1.3)
Requirement already satisfied: greenlet!=0.4.17 in c:\anaconda\lib\site-packages (from sqlalchemy>=2.0.16->Flask-SQLAlchemy) (3.0.1)
```

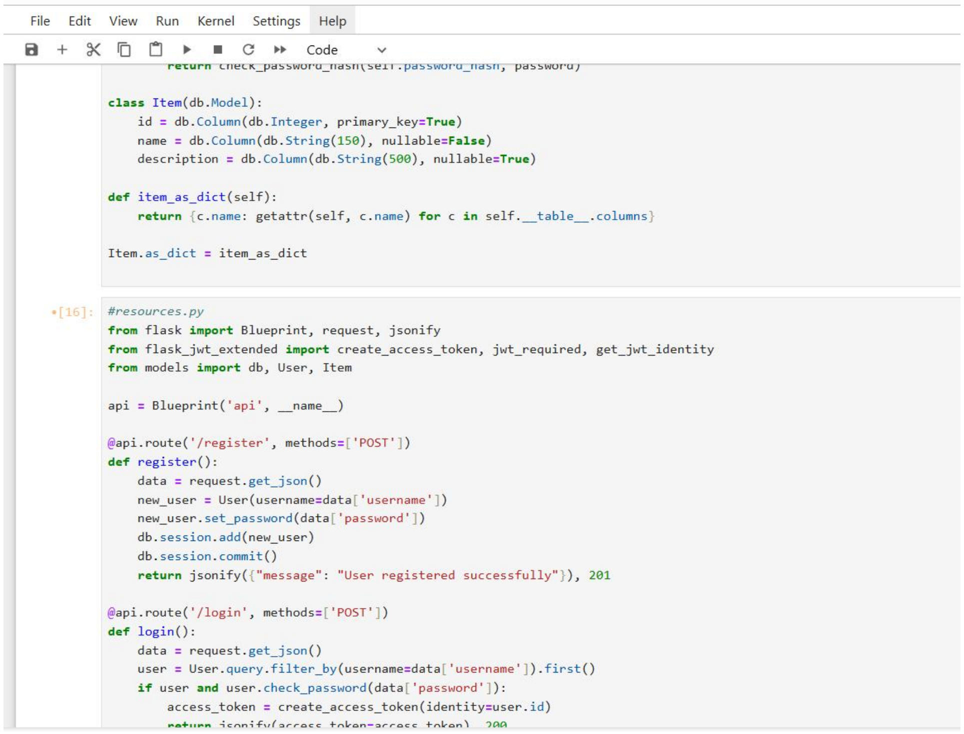
```
[6]: import sys
     import os

     # Add the current directory to the system path
     sys.path.append(os.getcwd())
```

```
[7]: #config.py
     import os
     from dotenv import load_dotenv

     load_dotenv()

     class Config:
         SECRET_KEY = os.getenv('SECRET_KEY', 'mysecretkey')
```


jupyter task4 Last Checkpoint: 17 minutes ago
File Edit View Run Kernel Settings Help
Code
JupyterLab

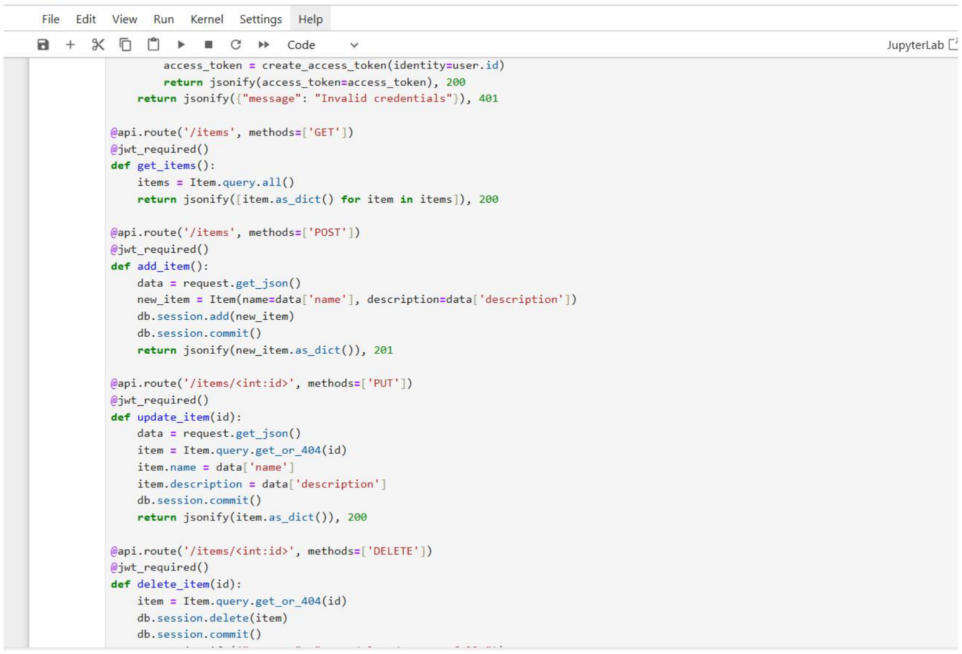
```
[7]: #config.py
     import os
     from dotenv import load_dotenv

     load_dotenv()

     class Config:
         SECRET_KEY = os.getenv('SECRET_KEY', 'mysecretkey')
         SQLALCHEMY_DATABASE_URI = os.getenv('DATABASE_URL', 'sqlite:///site.db')
         SQLALCHEMY_TRACK_MODIFICATIONS = False
         JWT_SECRET_KEY = os.getenv('JWT_SECRET_KEY', 'myjwtsecretkey')
```

```
[13]: #models.py
      from flask_sqlalchemy import SQLAlchemy
      from werkzeug.security import generate_password_hash, check_password_hash

      db = SQLAlchemy()

      class User(db.Model):
          id = db.Column(db.Integer, primary_key=True)
          username = db.Column(db.String(150), unique=True, nullable=False)
          password_hash = db.Column(db.String(256), nullable=False)

          def set_password(self, password):
              self.password_hash = generate_password_hash(password)

          def check_password(self, password):
              return check_password_hash(self.password_hash, password)

      class Item(db.Model):
          id = db.Column(db.Integer, primary_key=True)
          name = db.Column(db.String(150), nullable=False)
          description = db.Column(db.String(500), nullable=True)

          def item_to_dict(self):
```

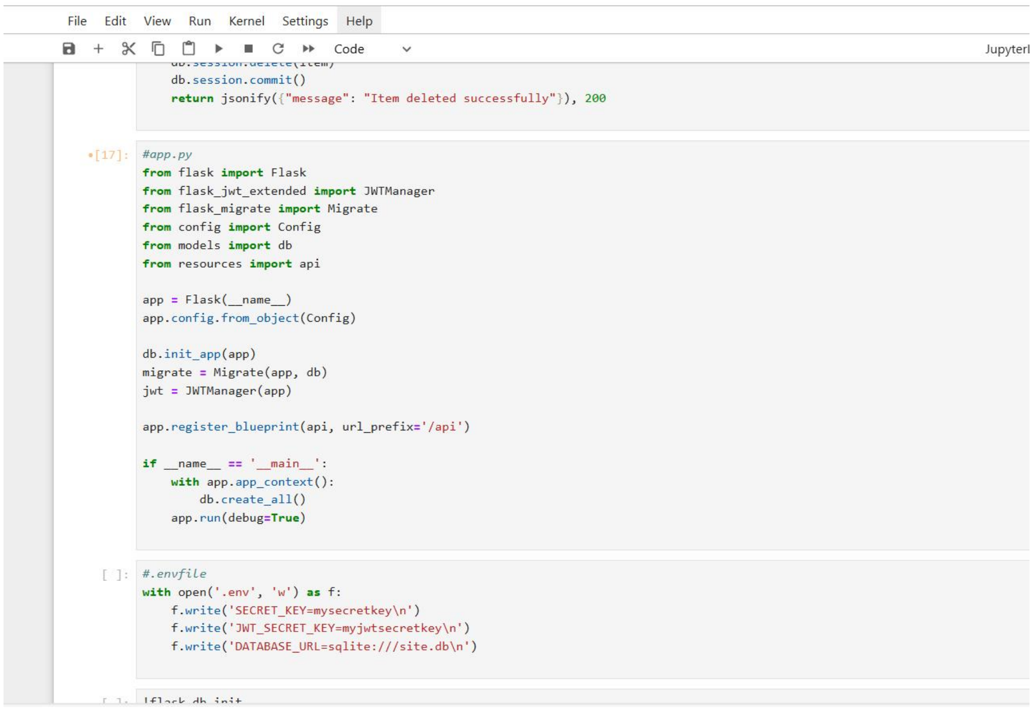


```python
        return check_password_hash(self.password_hash, password)

class Item(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(150), nullable=False)
    description = db.Column(db.String(500), nullable=True)

def item_as_dict(self):
    return {c.name: getattr(self, c.name) for c in self.__table__.columns}

Item.as_dict = item_as_dict
```

```python
*[16]: #resources.py
from flask import Blueprint, request, jsonify
from flask_jwt_extended import create_access_token, jwt_required, get_jwt_identity
from models import db, User, Item

api = Blueprint('api', __name__)

@api.route('/register', methods=['POST'])
def register():
    data = request.get_json()
    new_user = User(username=data['username'])
    new_user.set_password(data['password'])
    db.session.add(new_user)
    db.session.commit()
    return jsonify({"message": "User registered successfully"}), 201

@api.route('/login', methods=['POST'])
def login():
    data = request.get_json()
    user = User.query.filter_by(username=data['username']).first()
    if user and user.check_password(data['password']):
        access_token = create_access_token(identity=user.id)
        return jsonify(access_token=access_token), 200
```



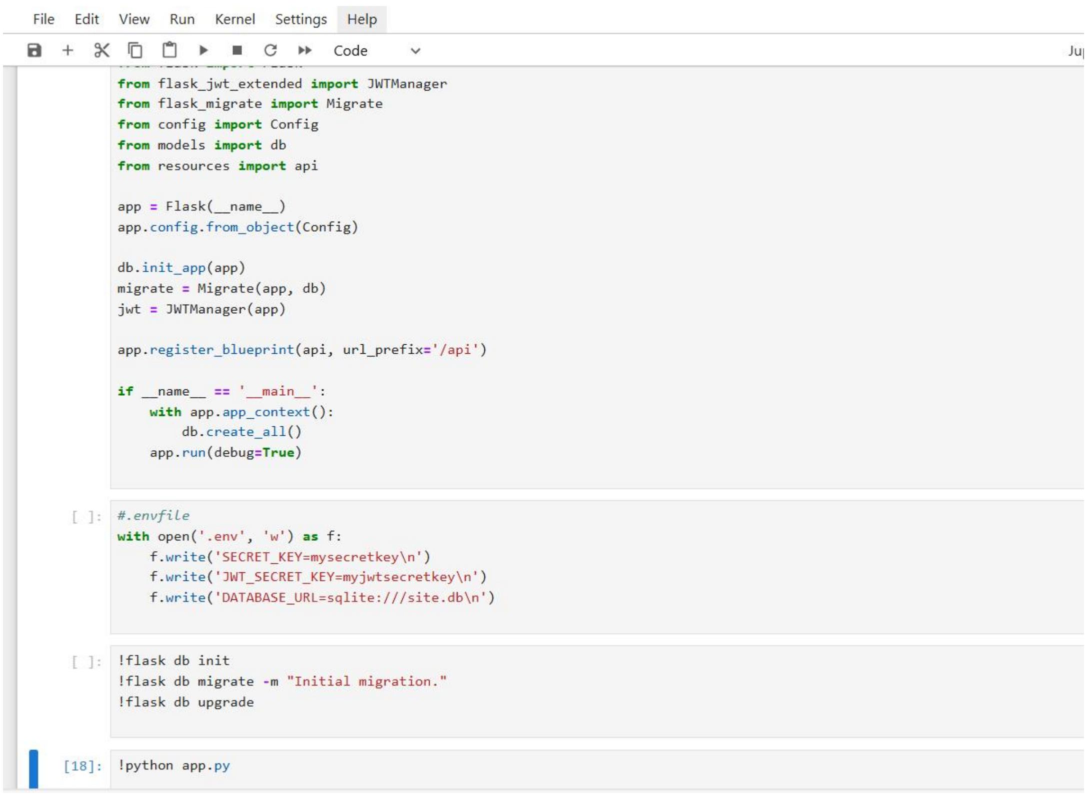
```python
        access_token = create_access_token(identity=user.id)
        return jsonify(access_token=access_token), 200
    return jsonify({"message": "Invalid credentials"}), 401

@api.route('/items', methods=['GET'])
@jwt_required()
def get_items():
    items = Item.query.all()
    return jsonify([item.as_dict() for item in items]), 200

@api.route('/items', methods=['POST'])
@jwt_required()
def add_item():
    data = request.get_json()
    new_item = Item(name=data['name'], description=data['description'])
    db.session.add(new_item)
    db.session.commit()
    return jsonify(new_item.as_dict()), 201

@api.route('/items/<int:id>', methods=['PUT'])
@jwt_required()
def update_item(id):
    data = request.get_json()
    item = Item.query.get_or_404(id)
    item.name = data['name']
    item.description = data['description']
    db.session.commit()
    return jsonify(item.as_dict()), 200

@api.route('/items/<int:id>', methods=['DELETE'])
@jwt_required()
def delete_item(id):
    item = Item.query.get_or_404(id)
    db.session.delete(item)
    db.session.commit()
```



```
    db.session.commit()
    return jsonify({"message": "Item deleted successfully"}), 200
```

```
•[17]: #app.py
from flask import Flask
from flask_jwt_extended import JWTManager
from flask_migrate import Migrate
from config import Config
from models import db
from resources import api

app = Flask(__name__)
app.config.from_object(Config)

db.init_app(app)
migrate = Migrate(app, db)
jwt = JWTManager(app)

app.register_blueprint(api, url_prefix='/api')

if __name__ == '__main__':
    with app.app_context():
        db.create_all()
    app.run(debug=True)
```

```
[ ]: #.envfile
with open('.env', 'w') as f:
    f.write('SECRET_KEY=mysecretkey\n')
    f.write('JWT_SECRET_KEY=myjwtsecretkey\n')
    f.write('DATABASE_URL=sqlite:///site.db\n')
```



```
from flask_jwt_extended import JWTManager
from flask_migrate import Migrate
from config import Config
from models import db
from resources import api

app = Flask(__name__)
app.config.from_object(Config)

db.init_app(app)
migrate = Migrate(app, db)
jwt = JWTManager(app)

app.register_blueprint(api, url_prefix='/api')

if __name__ == '__main__':
    with app.app_context():
        db.create_all()
    app.run(debug=True)
```

```
[ ]: #.envfile
with open('.env', 'w') as f:
    f.write('SECRET_KEY=mysecretkey\n')
    f.write('JWT_SECRET_KEY=myjwtsecretkey\n')
    f.write('DATABASE_URL=sqlite:///site.db\n')
```

```
[ ]: !flask db init
!flask db migrate -m "Initial migration."
!flask db upgrade
```

```
[18]: !python app.py
```