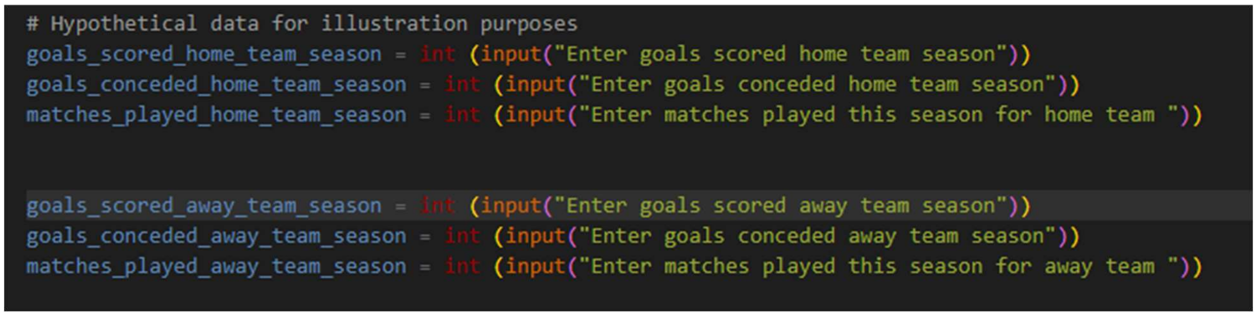
Predict Football Match Outcomes

1- Data Input

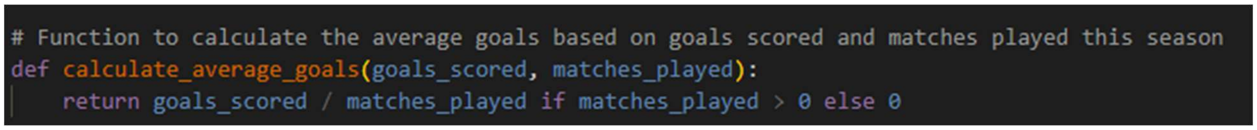
The user is prompted to input data for both home and away teams, including the goals scored, matches played, goals conceded, etc.



```python
# Hypothetical data for illustration purposes
goals_scored_home_team_season = int (input("Enter goals scored home team season"))
goals_conceded_home_team_season = int (input("Enter goals conceded home team season"))
matches_played_home_team_season = int (input("Enter matches played this season for home team "))


goals_scored_away_team_season = int (input("Enter goals scored away team season"))
goals_conceded_away_team_season = int (input("Enter goals conceded away team season"))
matches_played_away_team_season = int (input("Enter matches played this season for away team "))
```

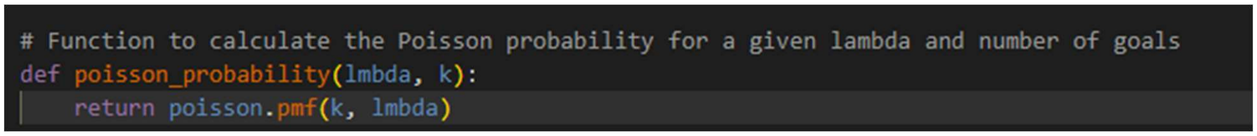
2- **Functions**

- **calculate_average_goals**: Calculates the average goals per match based on the total goals scored and matches played.



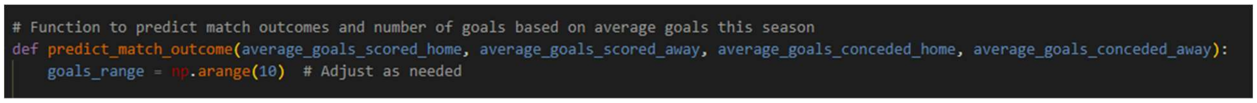
```python
# Function to calculate the average goals based on goals scored and matches played this season
def calculate_average_goals(goals_scored, matches_played):
    return goals_scored / matches_played if matches_played > 0 else 0
```

- **poisson_probability**: Calculates the Poisson probability for a given lambda and number of goals using the **pmf** (probability mass function) from the **scipy.stats.poisson** module.



```python
# Function to calculate the Poisson probability for a given lambda and number of goals
def poisson_probability(lmbda, k):
    return poisson.pmf(k, lmbda)
```

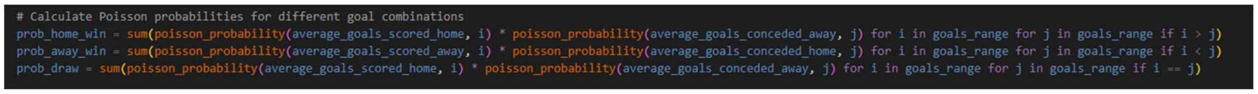
- **predict_match_outcome**: Predicts match outcomes and the number of goals based on the average goals scored and conceded by both home and away teams.



```python
# Function to predict match outcomes and number of goals based on average goals this season
def predict_match_outcome(average_goals_scored_home, average_goals_scored_away, average_goals_conceded_home, average_goals_conceded_away):
    goals_range = np.arange(10)  # Adjust as needed
```

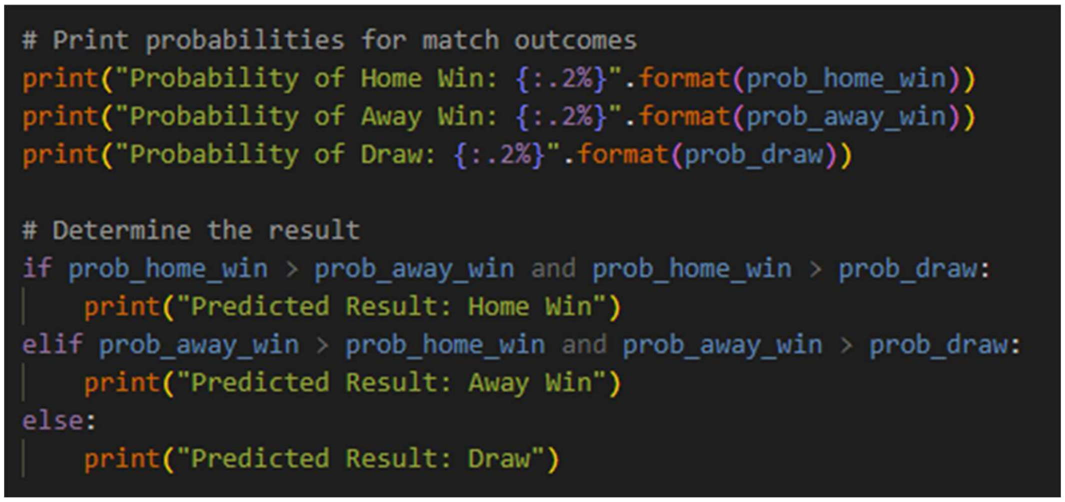
3-Prediction:

- The script calculates the average goals scored and conceded for both the home and away teams using the provided data.
- It then uses the Poisson distribution to calculate the probabilities of different goal combinations for home and away teams.



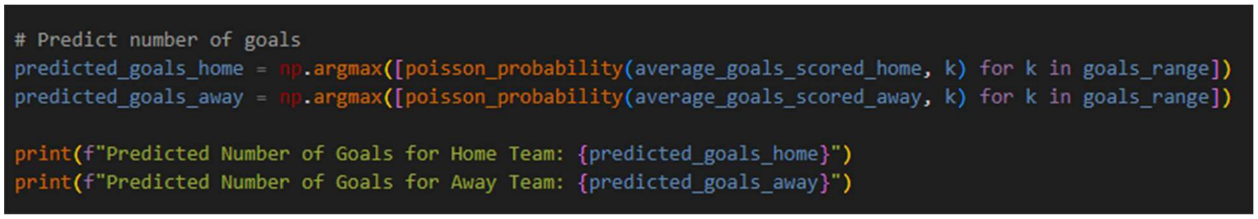
```python
# Calculate Poisson probabilities for different goal combinations
prob_home_win = sum(poisson_probability(average_goals_scored_home, i) * poisson_probability(average_goals_conceded_away, j) for i in goals_range for j in goals_range if i > j)
prob_away_win = sum(poisson_probability(average_goals_scored_away, i) * poisson_probability(average_goals_conceded_home, j) for i in goals_range for j in goals_range if i < j)
prob_draw = sum(poisson_probability(average_goals_scored_home, i) * poisson_probability(average_goals_conceded_away, j) for i in goals_range for j in goals_range if i == j)
```

- The script prints the probabilities of a home win, away win, and draw, and predicts the match result based on the highest probability.



```python
# Print probabilities for match outcomes
print("Probability of Home Win: {:.2%}".format(prob_home_win))
print("Probability of Away Win: {:.2%}".format(prob_away_win))
print("Probability of Draw: {:.2%}".format(prob_draw))

# Determine the result
if prob_home_win > prob_away_win and prob_home_win > prob_draw:
    print("Predicted Result: Home Win")
elif prob_away_win > prob_home_win and prob_away_win > prob_draw:
    print("Predicted Result: Away Win")
else:
    print("Predicted Result: Draw")
```

- Additionally, it predicts the number of goals for both the home and away teams.



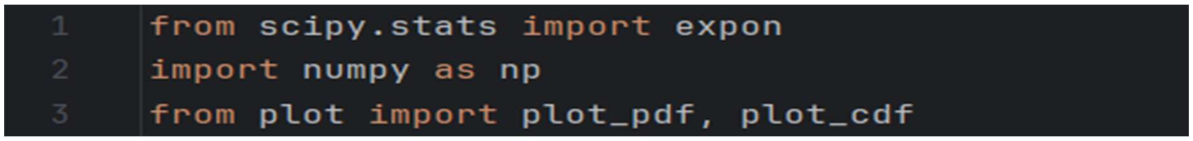
```python
# Predict number of goals
predicted_goals_home = np.argmax([poisson_probability(average_goals_scored_home, k) for k in goals_range])
predicted_goals_away = np.argmax([poisson_probability(average_goals_scored_away, k) for k in goals_range])

print(f"Predicted Number of Goals for Home Team: {predicted_goals_home}")
print(f"Predicted Number of Goals for Away Team: {predicted_goals_away}")
```

Predict the time when an Earthquake might occur.

The exponential distribution is often concerned with the amount of time until some specific event occurs. For example, the amount of time until an earthquake occurs has an exponential distribution.

Explaining Code in details:

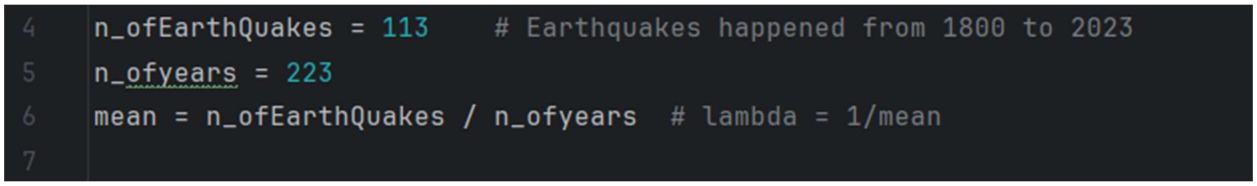
1. Importing needed libraries for exponential distribution



```
from scipy.stats import expon
import numpy as np
from plot import plot_pdf, plot_cdf
```

From "scipy.stats" we imported 'expon' for exponential distribution, "numpy" as 'np' for numerical operations and from "plot" we used 'plot_pdf', 'plot_cdf' for graphing functions.

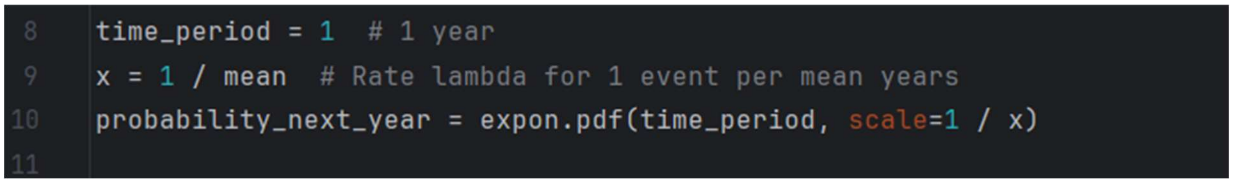
2. Setting parameter for the Exponential distribution



```
n_ofEarthQuakes = 113     # Earthquakes happened from 1800 to 2023
n_ofyears = 223
mean = n_ofEarthQuakes / n_ofyears  # lambda = 1/mean

```

Mean is the rate of earthquakes per year, which is used as the rate parameter lambda for exponential distribution.

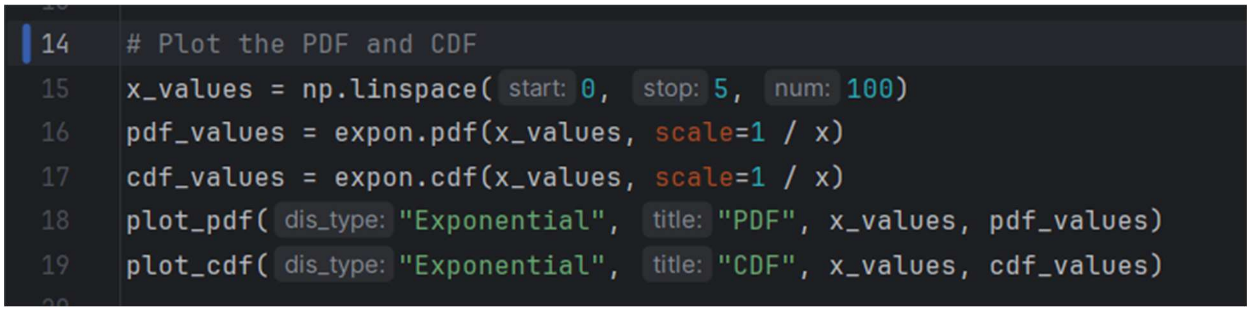
3. Setting variables, calculating PDF and Printing Probability



```
time_period = 1  # 1 year
x = 1 / mean  # Rate lambda for 1 event per mean years
probability_next_year = expon.pdf(time_period, scale=1 / x)

```

The variable "time_period" is set to next year and then calculating x parameter which is lambda and use "expon.pdf" for the Probabilty Density Function for next year.

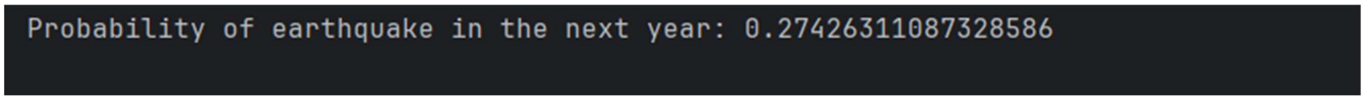
4. Graphing PDF and CDF



```
# Plot the PDF and CDF
x_values = np.linspace( start: 0,  stop: 5,  num: 100)
pdf_values = expon.pdf(x_values, scale=1 / x)
cdf_values = expon.cdf(x_values, scale=1 / x)
plot_pdf( dis_type: "Exponential",  title: "PDF", x_values, pdf_values)
plot_cdf( dis_type: "Exponential",  title: "CDF", x_values, cdf_values)
```

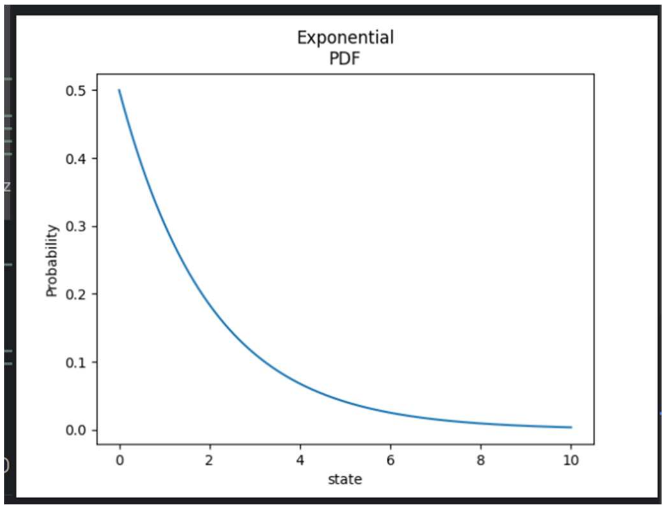
Output

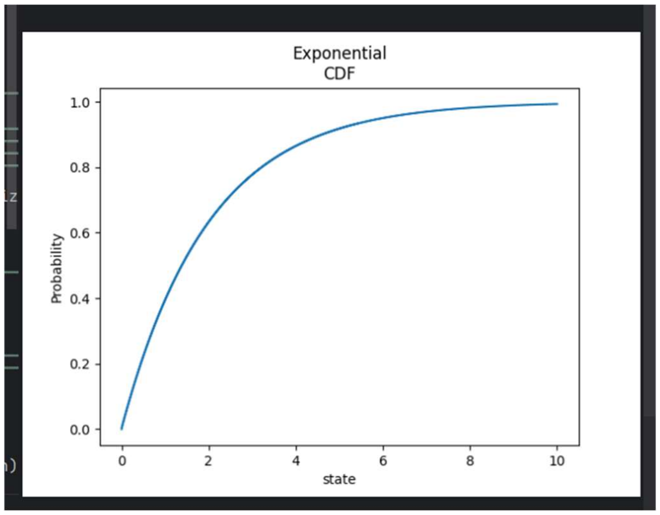
1. Probability of occurrence Earthquake in the next year.



```
Probability of earthquake in the next year: 0.27426311087328586
```

2. PDF and CDF Graphs





Traffic density of street

If the average number of cars that cross a particular street in a day is 25, then you can find the probability of 28 cars passing the street using the poisson formula.

- X is the random variable representing the number of events,
- k is the specific number of events we're interested in (in this case, 28 cars passing the street),
- λ is the average rate of events per interval.

In your example:

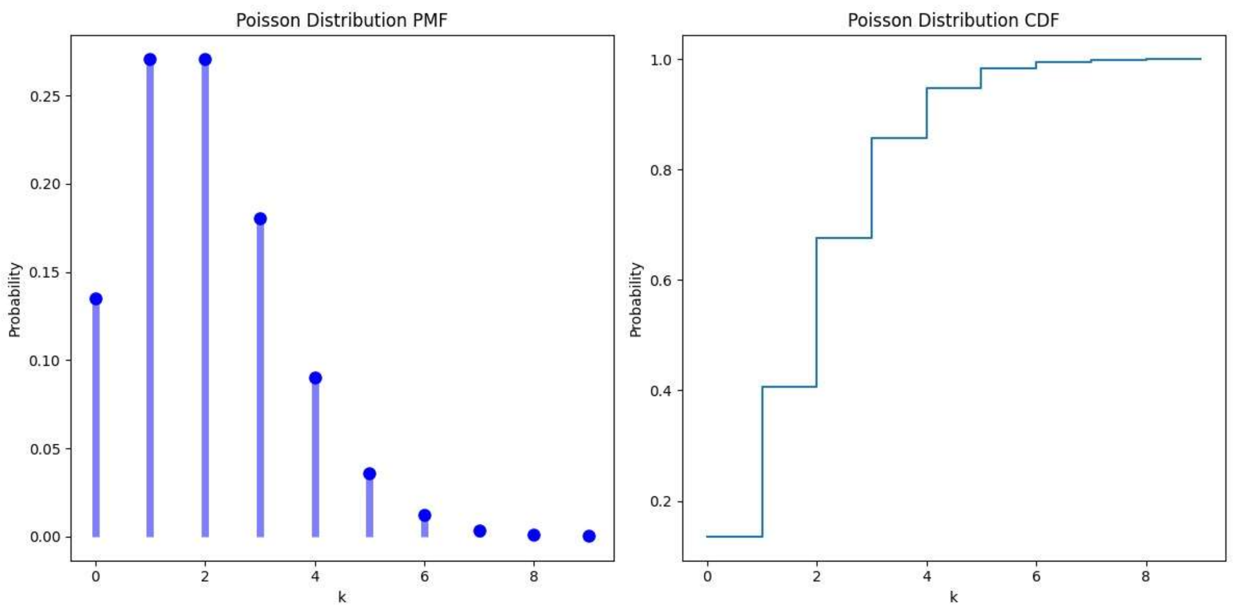
- $\lambda=25$ (average number of cars passing the street in a day),
- $k=28$ (the number of cars you want to find the probability for).

Plugging in these values, the probability of observing exactly 28 cars passing the street in a day is:

$P(X=28)= e^{-25}\cdot 25^{28}/28!$

```python
import numpy as np

import matplotlib.pyplot as plt

import time

from scipy.stats import poisson



def print_pause(message):

    print(message)

    time.sleep(2)



def poisson_drv():

    # Set the parameter

    rate_poisson = 2  # Average rate (lambda) for the Poisson distribution

    print("A service center receives an average of 2 customers per hour.")

    print_pause("What is the probability of receiving at most 3 customers in the
next hour?")

    print_pause("We can use the Poisson random variable for this scenario.")

    print_pause(f"The average rate (lambda) is {rate_poisson} customers per hour.")


    # Generate Poisson random variable

    poisson_rvs = poisson.rvs(mu=rate_poisson, size=1000)


    # Calculate PMF and CDF

    k_values = np.arange(0, 10)  # Adjust the range based on your scenario
    pmf_values = poisson.pmf(k_values, mu=rate_poisson)
    cdf_values = poisson.cdf(k_values, mu=rate_poisson)

    # Calculate mean and variance
    mean_poisson = poisson.mean(mu=rate_poisson)
    variance_poisson = poisson.var(mu=rate_poisson)

    # Display mean and variance
    print_pause(f"The MEAN value = {mean_poisson}")
    print_pause(f"The VARIANCE value = {variance_poisson}")

    cdf = poisson.cdf(3, mu=rate_poisson)
    print_pause("To calculate the probability of receiving at most 3 customers in the
next hour,")
    print_pause(f"We must calculate the CDF of 3, which is approximately {cdf:.4f}")
    time.sleep(4)

    print_pause("Now let's visualize the PMF and CDF:")
    time.sleep(3)
    show_x = int(input("Show? (1 for Yes, 0 for No): "))
    if show_x == 1:
        plt.figure(figsize=(12, 6))

        plt.subplot(121)
        plt.plot(k_values, pmf_values, "bo", ms=8, label="Poisson PMF")
        plt.vlines(k_values, 0, pmf_values, colors="b", lw=5, alpha=0.5)
        plt.title('Poisson Distribution PMF')
        plt.xlabel('k')
        plt.ylabel('Probability')

        plt.subplot(122)
        plt.step(k_values, cdf_values, where='post')
        plt.title('Poisson Distribution CDF')
        plt.xlabel('k')
        plt.ylabel('Probability')
        plt.tight_layout()
        plt.show()



print("Choose the number of Example that you want : ")
print("1- Binomial random variable ")
print("2- Poisson random variable ")
choose = int(input("Choose: "))
if choose == 1:
    binomial_drv()
elif choose == 2:
    poisson_drv()
else:
```

```
print("Invalid choice. Please choose 1 or 2.")
```


Poisson Distribution PMF
Probability
k
Poisson Distribution CDF
Probability
k