**Name : dua fatima**

**Roll No : 007**

**Submitted To : Sir Rasikh Ali**

**Documentary of the Code**

**🔹 Code Overview**

This program analyzes a list of movies with their budgets:

* It calculates the **average budget** of all movies.
* It compares each movie’s budget with the average.
* It shows which movies are **above average** and which are **below average**.

**🔹 Step-by-Step Explanation**

1. **Class Definition**
2. class MovieAnalysis:
   * A class is created to organize the code.
   * It will handle average calculation and report generation.
3. **Initialization**
4. def \_\_init\_\_(self, movies):
5. self.movies = movies
6. self.average = self.\_calculate\_average()
   * The constructor (\_\_init\_\_) stores the list of movies.
   * It immediately calls \_calculate\_average() to compute the average budget.
7. **Private Method: Calculate Average**
8. def \_calculate\_average(self):
9. total = 0
10. for \_, budget in self.movies:
11. total += budget
12. return total / len(self.movies)
    * Goes through each movie and adds up the budgets.
    * Divides total budget by the number of movies to find the **average**.
    * The underscore \_ means the title is not used in this loop, only budget is needed.
13. **Generate Report**
14. def show\_report(self):
15. print("Movie Budget Analysis")
16. print(f"Average Budget = ${self.average:,.0f}\n")
    * Prints the **title** of the report.
    * Shows the calculated **average budget** with commas (e.g., $379,000,000).
17. **Classify Movies**
18. above, below = [], []
19. for title, budget in self.movies:
20. diff = budget - self.average
21. if diff > 0:
22. above.append((title, diff))
23. else:
24. below.append((title, abs(diff)))
    * Creates two lists: above and below.
    * If the movie’s budget is **greater than average → goes to above**.
    * If it’s **less than average → goes to below**.
    * Stores the difference from average for display.
25. **Print Results**
26. print("Movies ABOVE average:")
27. for title, diff in above:
28. print(f" - {title} (+${diff:,.0f})")
29. print("Movies BELOW average:")
30. for title, diff in below:
31. print(f" - {title} (-${diff:,.0f})")
    * Prints the movies in each category with their difference in budget.
    * + sign for above, - sign for below.
32. **Movie Data**
33. movies = [
34. ("Eternal Sunshine of the Spotless Mind", 20000000),
35. ("Memento", 9000000),
36. ("Requiem for a Dream", 4500000),
37. ("Pirates of the Caribbean: On Stranger Tides", 379000000),
38. ("Avengers: Age of Ultron", 365000000),
39. ("Avengers: Endgame", 356000000),
40. ("Incredibles 2", 200000000)
41. ]
    * A list of **movies with their budgets** (title, budget).
42. **Run the Report**
43. report = MovieAnalysis(movies)
44. report.show\_report()
    * Creates an object of MovieAnalysis.
    * Calls show\_report() to display the analysis.

**✅ Full Code with Comments**

# MOVIE ANALYSIS PROGRAM

class MovieAnalysis:

def \_\_init\_\_(self, movies):

# Store movie data (title, budget)

self.movies = movies

# Calculate average budget when object is created

self.average = self.\_calculate\_average()

def \_calculate\_average(self):

# Private method: calculate average budget

total = 0

for \_, budget in self.movies: # Only use budget, ignore title

total += budget

return total / len(self.movies)

def show\_report(self):

# Print report header

print("Movie Budget Analysis")

print(f"Average Budget = ${self.average:,.0f}\n")

# Separate movies above and below average

above, below = [], []

for title, budget in self.movies:

diff = budget - self.average

if diff > 0:

above.append((title, diff))

else:

below.append((title, abs(diff)))

# Show movies with above-average budgets

print("Movies ABOVE average:")

for title, diff in above:

print(f" - {title} (+${diff:,.0f})")

# Show movies with below-average budgets

print("Movies BELOW average:")

for title, diff in below:

print(f" - {title} (-${diff:,.0f})")

# Sample dataset: movies with their budgets

movies = [

("Eternal Sunshine of the Spotless Mind", 20000000),

("Memento", 9000000),

("Requiem for a Dream", 4500000),

("Pirates of the Caribbean: On Stranger Tides", 379000000),

("Avengers: Age of Ultron", 365000000),

("Avengers: Endgame", 356000000),

("Incredibles 2", 200000000)

]

# Create report object and show results

report = MovieAnalysis(movies)

report.show\_report()