EE 381 Project 4

William Thinh Luu
California State University of Long Beach
EE 381 Probability and Statistics, with Application on Computing
Behruz Revani, Ph.D.
11/17/2024

Part 1

Introduction

Given a collection of books in which each book has thickness W, where W is a RV, uniformly distributed between minimum a and maximum b. Calculate the mean and standard deviation of the thickness of n books, where n = 1, 5, 10, 15. Create 4 histograms that run N = 10000 experiments simulating RV S = W, and plot the normal distribution probability function to compare the histogram and plot. Have a histogram for n = 1, 5, 10, 15.

Methodology

Initialize the global variables of "a", "b", mean, and std via the example provided in the instructions. Create a function that prints the number of books, mean of the random variable, and standard deviation of the random variable, with parameters a, b, and n. The experiment function gets the RV S using numpy's library function *np.random.uniform(a, b, (N, n))* and plots the graphs using matplotlib. The normal distribution plot is also calculated using the given function of f(x).

Results

Mean thickness of a single book (cm)	Standard deviation of thickness (cm)
$\mu_w = 2.0$	$\sigma_w = 0.5774$

Number of Books n	Mean thickness of a stack of n books (cm)	Standard deviation of the thickness for n books
n=1	$\mu_w = 2.0$	$\sigma_w = 0.5774$
n=5	$\mu_w = 10.0$	$\sigma_w = 1.2909$
n=10	$\mu_w = 20.0$	$\sigma_w = 1.8257$
n=15	$\mu_w = 30.0$	$\sigma_w = 2.2361$

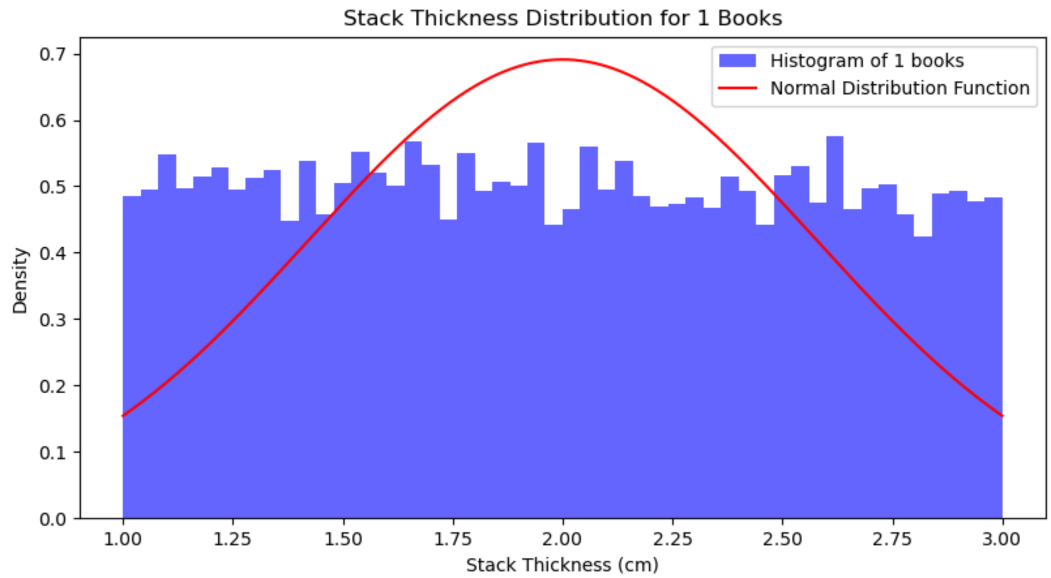

Figure 1: Simulating RV S = W, with n = 1 and N = 10,000

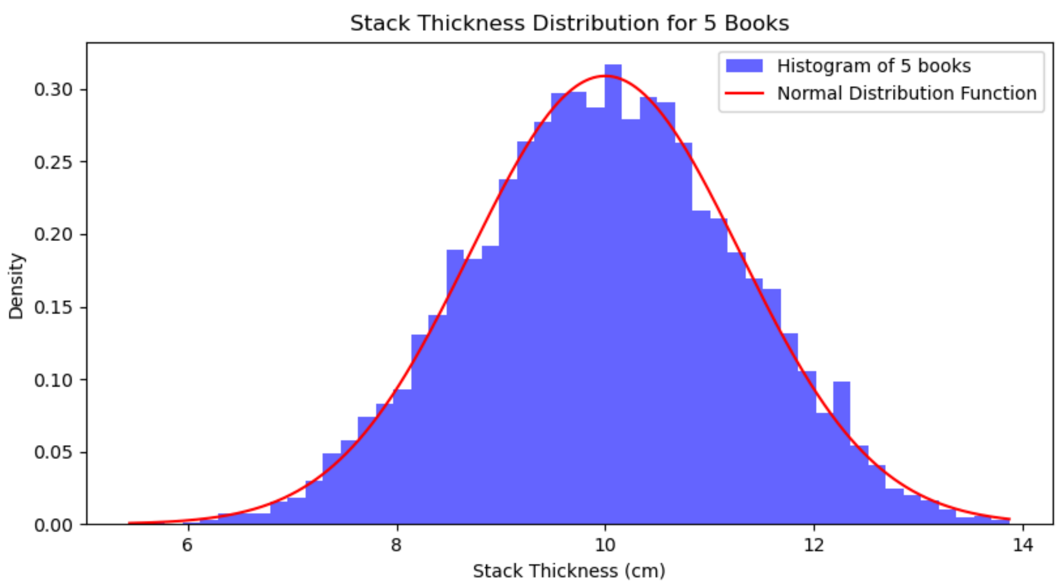

Figure 2: Simulating RV S = W, with n = 5 and N = 10,000

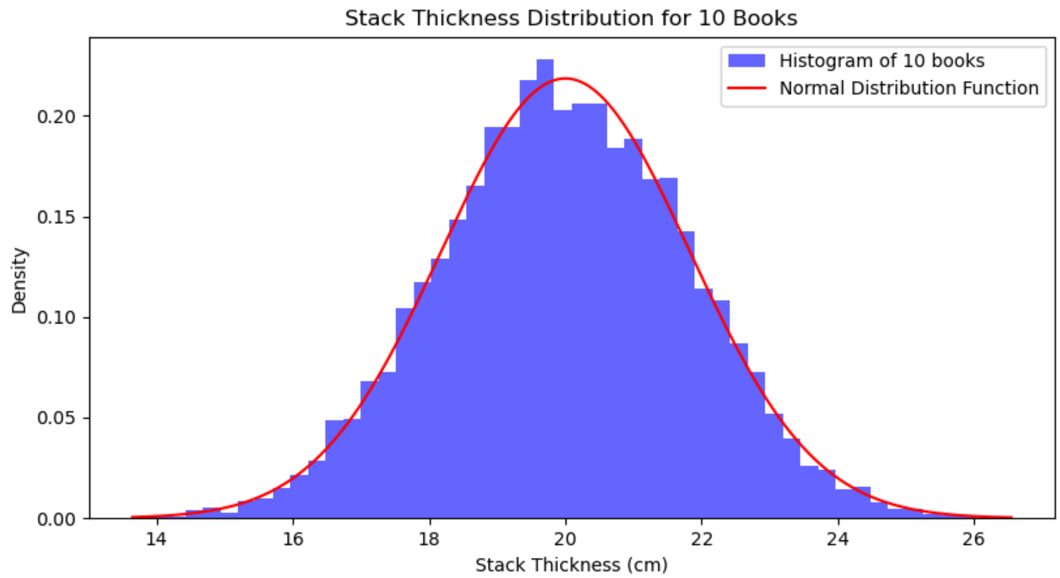

Figure 3: Simulating RV S = W, with n = 10 and N = 10,000

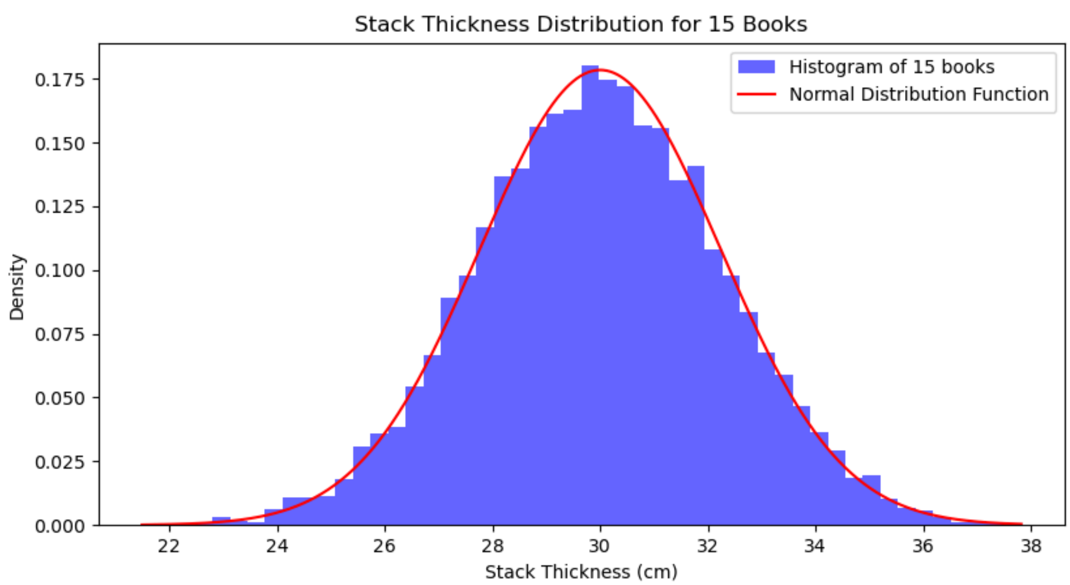

Figure 4: Simulating RV S = W, with n = 15 and N = 10,000

Conclusion

In this simulation, the thickness of a collection of books, modeled as a random variable uniformly distributed between a minimum "a" and a maximum "b", was analyzed for different values of "n". By running 10,000 experiments for each case of n = 1, 5, 10, 15, we generated histograms that depict the distribution of the sum of n books' thicknesses. As n increased, the experimental histograms began to resemble the normal distribution more closely, which is consistent with the Central Limit Theorem. The histograms for each n were compared with the theoretical normal distribution, showing that as the number of books increases, the mean and standard deviation converge toward the expected values based on the parameters "a" and "b".

Appendix 1

```python
import numpy as np
import matplotlib.pyplot as plt

# book thickness W = RV
# RV uniformly distributed between min(a) < RV < max(b)
# a, b is provided
# calculate mean and std of thickness using a and b

a = 1
b = 3
mean = (a+b)/2
std = np.sqrt((b-a)**2/12)


def uniformDist(a, b, n):
    return np.random.uniform(a, b, n)


def meanAndStd(a, b, n):
    print(f"Number of Books = {n}")
    print(f"mean = {mean * n}")
    print(f"standard deviation = {std * np.sqrt(n)}")
    print()

def experiment(a, b, N, n):
    # plotting histogram of stacks
    RV_S = np.random.uniform(a, b, (N,n))

    stacks = np.sum(RV_S, axis=1)
    plt.hist(stacks, bins=50, density=True, alpha=0.6, color='b', label=f'Histogram of {n} books')

    # plotting normal distribution probability funciton
    theoretical_mean = mean * n
    theoretical_std = std * np.sqrt(n)

    x = np.linspace(min(stacks), max(stacks), 1000)
    y = (1/(theoretical_std * np.sqrt(2 * np.pi))) * np.exp(-((x - theoretical_mean)**2) / (2 * theoretical_std**2))
    plt.plot(x, y, color="red", label = "Normal Distribution Function")

    plt.title(f"Stack Thickness Distribution for {n} Books")
    plt.xlabel("Stack Thickness (cm)")
    plt.ylabel("Density")
    plt.legend()
    plt.show()

if __name__ == '__main__':
    N = 10000
    n = [1, 5, 10, 15]

    # finding mean and standard dev for stack of n books
    for i in n:
        meanAndStd(a, b, i)
        experiment(a, b, N, i)
```

Part 2

Introduction

Simulate an exponentially distributed RV T by generating a RV U, which is uniformly distributed in [0, 1]. Using the formula: $T = (-1/\alpha)\ln(1-U)$, where $\alpha = 2$. This is considered one experiment. The PDF provided by the instructions will be used to compare the experimentally generated histogram. The experimentally generated histogram of RV T is created by performing the experiment N = 10000 times.

Methodology

First initialize a function called experiment that takes parameters N, which calculates RV U using numpy's uniform function and using RV U, calculates RV T using the formula provided in the instructions. A function called histogram takes 1 parameter of the RV T, which plots the RV T using matplotlib's histogram function. Plot the theoretical PDF which is the function f(t) provided in the instructions using matplotlib. In main, repeat the experiment N = 10000 times to create the histogram of RV T and the theoretical PDF.

Results

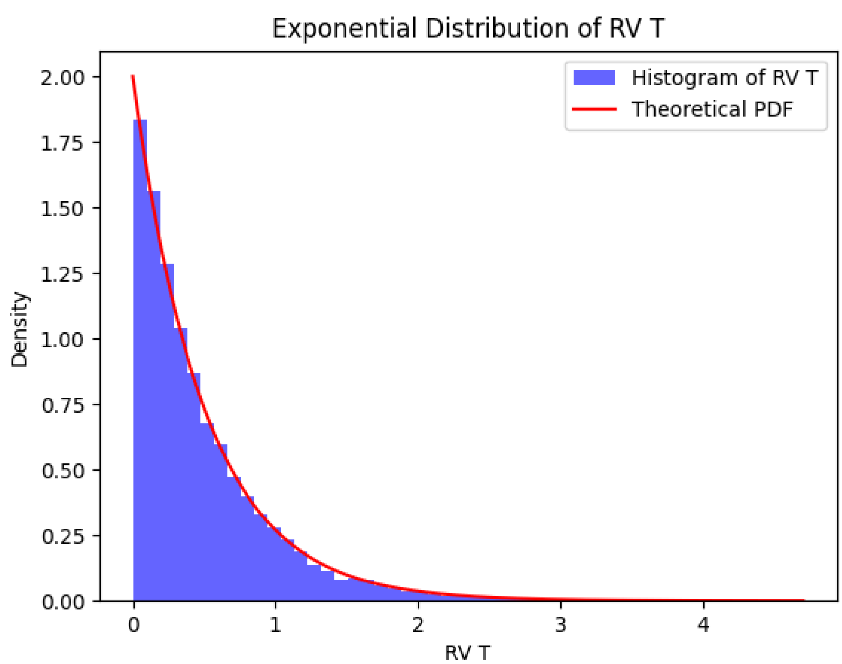

Figure 5: Exponential Distribution of RV T

Conclusion

In conclusion, the simulation of the exponentially distributed random variable was successfully carried out by generating a uniformly distributed random variable and applying the given transformation formula. By performing 10,000 experiments, we obtained an experimental histogram of the random variable, which closely matched the theoretical probability density function, which is the normal distribution function. This comparison demonstrated the accuracy of the simulation method, validating the approach of using uniform random variables to simulate an exponential distribution

Appendix 2

```
import numpy as np
import matplotlib.pyplot as plt

def experiment(N):
    RV_U =  np.random.uniform(0, 1, N)
    RV_T = (-1/2) * np.log(1-RV_U)
    return RV_T

def histogram(RV_T, bins=50):
    plt.hist(RV_T, bins=bins, density=True, alpha=0.6, color='b', label='Histogram of RV T')

    # theoretical PDF of the exponential distribution
    lambda_ = 2
    x = np.linspace(0, max(RV_T), 1000)
    pdf = lambda_ * np.exp(-lambda_ * x)

    # plot the theoretical PDF
    plt.plot(x, pdf, color='red', label='Theoretical PDF')

    # add labels and legend
    plt.title("Exponential Distribution of RV T")
    plt.xlabel("RV T")
    plt.ylabel("Density")
    plt.legend()
    plt.show()

if __name__ == '__main__':
    N = 10000
    T = experiment(N)
    histogram(T)
```

Part 3

Introduction

Given a random variable T that is the exponentially distributed lifetime of a battery, with a battery lasting an average of beta = 40 days, simulate the RV representing the lifetime of a carton of 24 batteries. Create a histogram of the RV, and plot the normal distribution over the histogram to compare results. Then create and plot the CDF of the lifetime of a carton, and find the probability that a carton will last longer than 3 years, and last between 2.0 and 2.5 years.

Methodology

To create the vector of 24 elements that represents a carton, initialize a vector using sum(np.random.exponential(beta, n)) that generates a vector in which beta = 40 and n = 24. Using plt.hist() from matplotlib, I graphed the histogram based on 50 bins relative to the vector C which holds 10,000 experiments. The CLT approximation was created by calculating the mean and standard deviation using CLT. The normal distribution is found using the mean and standard deviation formulas provided in the instructions. To create the CDF of the lifetime of a carton, F(C), use numpy's function *cdf = np.cumsum(hist * np.diff(bin_edges))* which returns an array where each element is the sum of all previous elements up to that point. The questions 1 and 2 are found by converting year into days, and using *np.interp(years, bin_centers, cdf)* to determine the probability of how long a carton can last.

Results

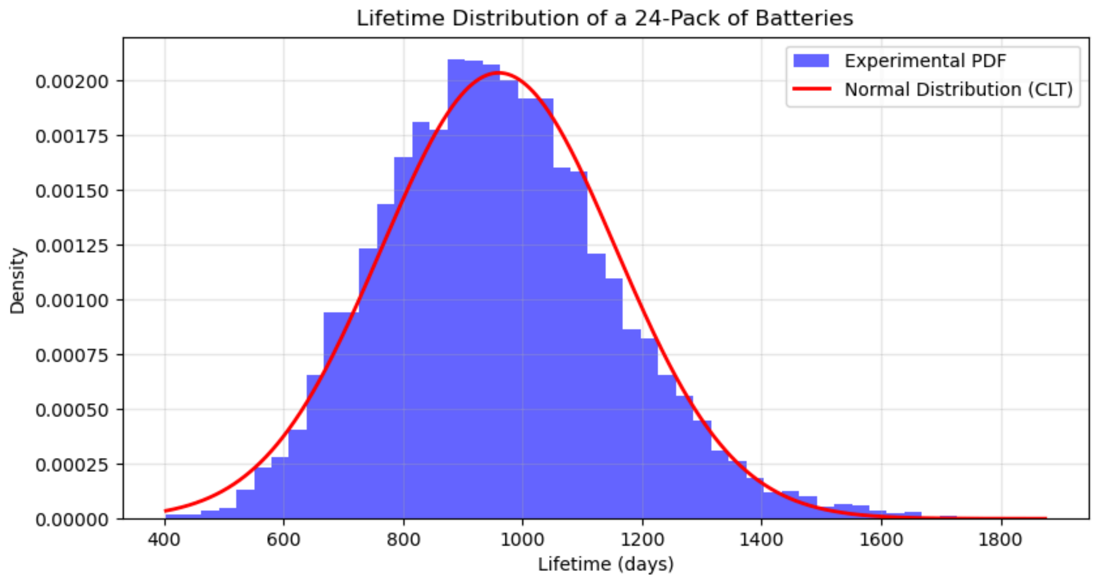

Figure 6: PDF of Lifetime of a 24-Pack of Batteries

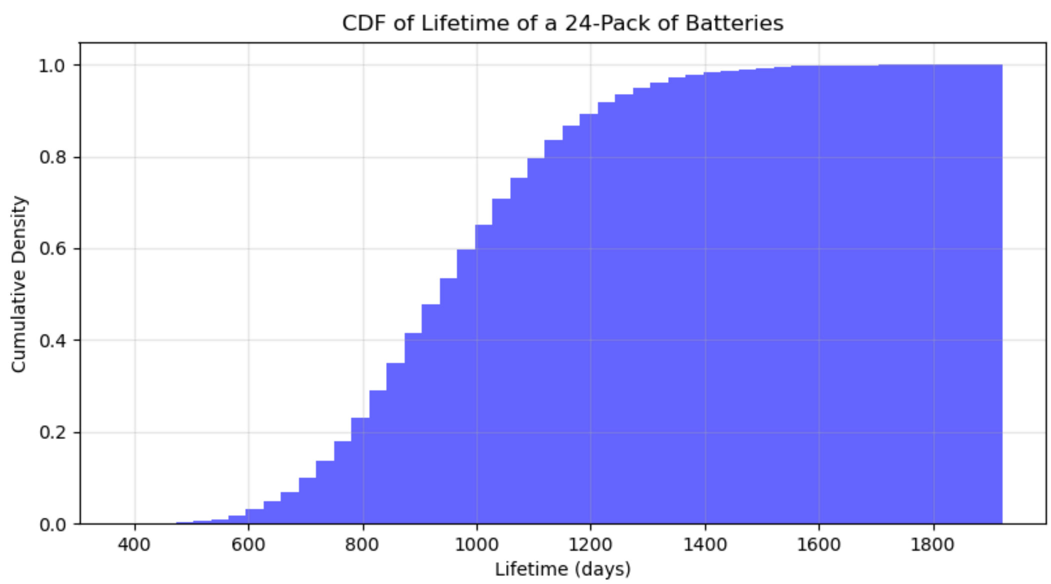

Figure 7: CDF of Lifetime of a 24-Pack of Batteries

Question	**ANS.**

1. Probability that the carton will last longer than 2.5 years	0.2861
2. Probability that the carton will last between 1.50 and 2.0 years	0.02538

Conclusion

In this analysis, we simulated the lifetime of a carton containing 24 batteries, each with an exponentially distributed lifetime of 40 days, across 10,000 experiments. The experimental probability density function (PDF) was created using histograms and compared with a normal distribution derived from the Central Limit Theorem (CLT). We also computed the cumulative distribution function (CDF) from the experimental data and used interpolation to estimate the probability that a carton would last longer than 3 years or between 2.0 and 2.5 years. The results showed that the lifetime distribution of the carton aligned well with the CLT approximation.

Appendix 3

```python
import numpy as np
import matplotlib.pyplot as plt


def normalDist(mu, sig, z):
    f = np.exp(-(z-mu) ** 2 / (2 * sig ** 2)) / (sig * np.sqrt(2* np.pi))
    return f

if __name__ == "__main__":
    beta = 40  # Mean lifetime of an individual battery
    N = 10_000  # Number of cartons to simulate
    n = 24  # Number of batteries in one carton

    # Create an array to store the lifetime of each carton
    C = np.zeros((N,))

    # Generate lifetimes for each carton (N experiments)
    for T in range(N):
        C[T] = sum(np.random.exponential(beta, n))

    # Plot the histogram of the experimental PDF
    plt.hist(C, bins=50, density=True, alpha=0.6, color='blue', label="Experimental PDF")

    # Calculate mean and standard deviation using CLT
    mu_C = n * beta  # Mean lifetime of a carton
    sigma_C = np.sqrt(n) * beta  # Standard deviation of a carton's lifetime

    # Generate points for the theoretical normal distribution (CLT approximation)
    z = np.linspace(min(C), max(C), 1000)
    normal_pdf = normalDist(mu_C, sigma_C, z)

    # Plot the theoretical normal distribution
    plt.plot(z, normal_pdf, color='red',  linewidth=2, label="Normal Distribution (CLT)")

    plt.title("Lifetime Distribution of a 24-Pack of Batteries")
    plt.xlabel("Lifetime (days)")
    plt.ylabel("Density")
    plt.legend()
    plt.grid(alpha=0.3)
    plt.show()

    # Calculate the histogram for experimental PDF again (for CDF calculation)
    hist, bin_edges = np.histogram(C, bins=50, density=True)
    bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2  # Compute bin centers

    # Calculate the CDF using the cumulative sum of the PDF
    cdf = np.cumsum(hist * np.diff(bin_edges))  # CDF as cumulative sum of PDF values

    # Plot the CDF
    plt.hist(C, bins=50, density=True, alpha=0.6, cumulative=True, color='b', label="Experimental CDF")
    plt.title("CDF of Lifetime of a 24-Pack of Batteries")
    plt.xlabel("Lifetime (days)")
    plt.ylabel("Cumulative Density")
    plt.grid(alpha=0.3)
    plt.show()

    # Calculate the probability that the carton lasts longer than 3 years
    time_3_years = 1095
    F_1095 = np.interp(time_3_years, bin_centers, cdf)
    prob_3_years = 1 - F_1095  # P(S > 1095)
    print(f"Probability that the carton lasts longer than 3 years: {prob_3_years:.4f}")

    # calculate probability that carton last between 2.0 and 2.5 years
    time_2_years = 730
    time_2_5_years = 912
    F_730 = np.interp(time_2_years, bin_centers, cdf)
    F_912 = np.interp(time_2_5_years, bin_centers, cdf)
    prob_between_2_and_2_5_years = F_912 - F_730  # P(730 <= S <= 912)
    print(f"Probability that the carton lasts between 2.0 and 2.5 years: {prob_between_2_and_2_5_years:.4f}")
```