Lab 7

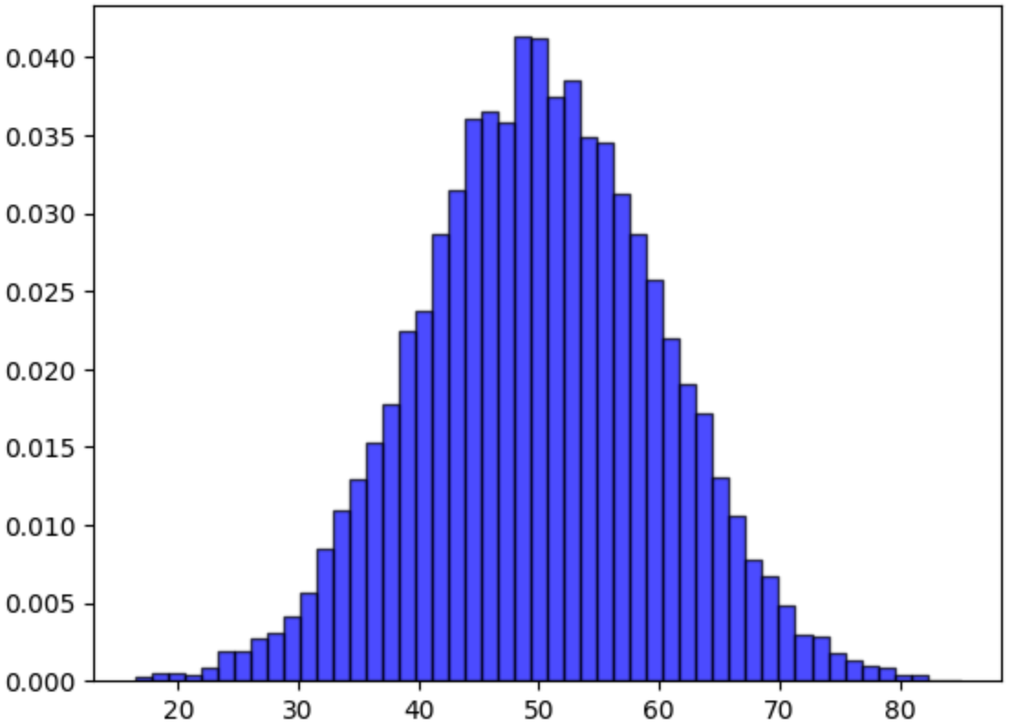
Daniel Mehta n01753264

In [16]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import scipy.special as sp
```

1. Generate and Visualize a Normal Distribution

In [7]:
```
mean = 50
std_dev = 10
num_samples = 10000

data = np.random.normal(mean, std_dev, num_samples)
plt.hist(data, bins=50, color='blue', edgecolor='black', alpha=0.7, density=True)
plt.show()
```



2. Compute Probability Density Function (PDF)

In [15]:
```
x=30
mean=30
std_dev=5

coefficient = 1/(std_dev *np.sqrt(2 * np.pi))
exponent = np.exp(-((x - mean) ** 2) / (2 * std_dev ** 2))
manual_pdf = coefficient * exponent

scipy_pdf = stats.norm.pdf(x, 30, 5)

print(f"Manual PDF at x = {x}: {manual_pdf}")
print(f"SciPy PDF at x = {x}: {scipy_pdf}")
```

```
Manual PDF at x = 30: 0.07978845608028655
SciPy PDF at x = 30: 0.07978845608028654
```

3. Compute Cumulative Distribution Function (CDF)

In [17]:
```
x=45
mean=40
std_dev=8

manual_cdf= 0.5 * (1 + sp.erf((x - mean)/(std_dev * np.sqrt(2))))
scipy_cdf = stats.norm.cdf(x, mean, std_dev)

print(f"Manual CDF at x = {x}: {manual_cdf}")
print(f"SciPy CDF at x = {x}: {scipy_cdf}")
```

```
Manual CDF at x = 45: 0.7340144709512995
SciPy CDF at x = 45: 0.7340144709512995
```

4. Generate and Compare PDF and CDF Plots

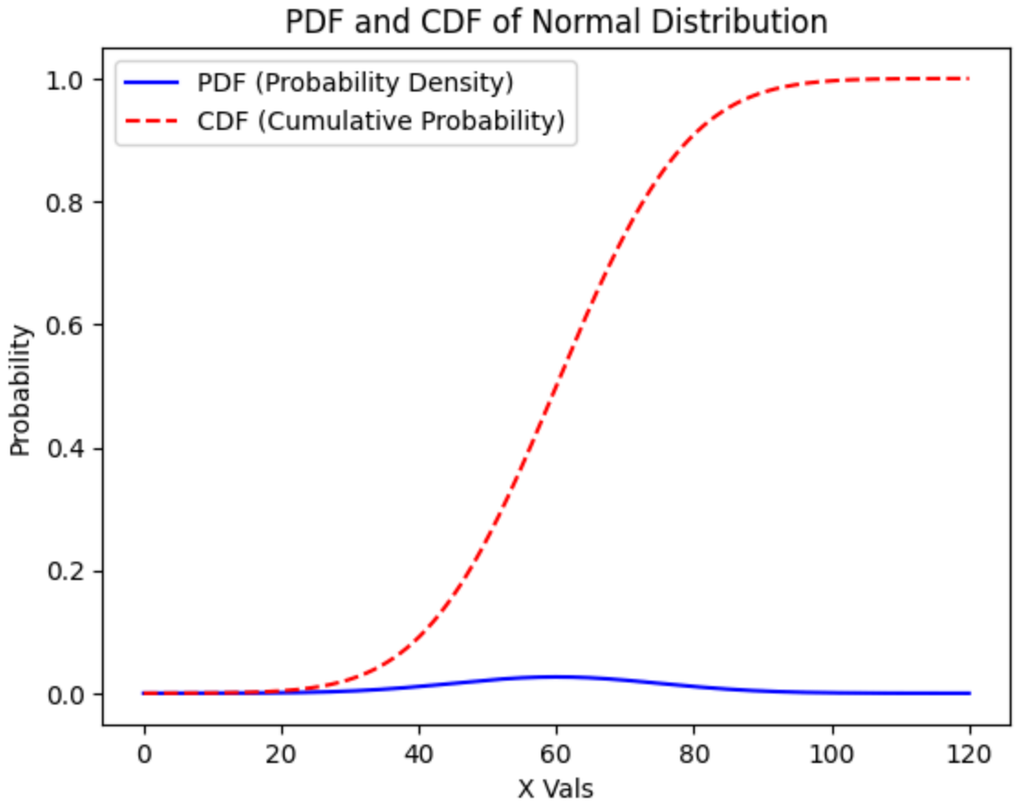
In [22]:
```
mean=60
std_dev=15
x_values = np.linspace(0, 120, 500)

pdf_values = stats.norm.pdf(x_values, mean, std_dev)
cdf_values = stats.norm.cdf(x_values, mean, std_dev)

plt.plot(x_values, pdf_values, label="PDF (Probability Density)", color='blue')
plt.plot(x_values, cdf_values, label="CDF (Cumulative Probability)", linestyle='dashed', color='red')

plt.xlabel("X Vals")
plt.ylabel("Probability")
plt.title("PDF and CDF of Normal Distribution")
plt.legend()

plt.show()
```



5. Calculate Probability of an Interval

In [25]:
```
mean = 70
std_dev = 12
x1 = 55
x2 = 85

P_X_leq_85 = stats.norm.cdf(x2, mean, std_dev)
P_X_leq_55 = stats.norm.cdf(x1, mean, std_dev)

probability = P_X_leq_85 - P_X_leq_55

print(f"P(55 <= X <= 85) = {probability}")
```

```
P(55 <= X <= 85) = 0.7887004526662893
```

6. Simulate a Standard Normal Distribution

In [30]:
```
data = np.random.normal(0, 1, 10000)

calculated_mean = np.mean(data)
calculated_std_dev = np.std(data)

print(f"Calculated Mean: {calculated_mean}")
print(f"Calculated Standard Deviation: {calculated_std_dev}")
```

```
Calculated Mean: -6.398725082514076e-05
Calculated Standard Deviation: 0.9942983245176708
```

7. Standardization (Z-Score Calculation)

In [37]:
```
data = np.array([45, 50, 55, 60, 65, 70, 75, 80, 85, 90])
mean = 65
std_dev = 12

z_scores = (data - mean) / std_dev

print("Z-Scores:\n", z_scores)
```

```
Z-Scores:
 [-1.66666667 -1.25       -0.83333333 -0.41666667  0.          0.41666667
  0.83333333  1.25        1.66666667  2.08333333]
```

In [ ]:
```

```