PROGRAM 1

PROBABILITY

```
total_outcomes = 6
favorable_outcomes = 1
probability_4 = favorable_outcomes / total_outcomes
print(f"Probability of rolling a 4: {probability_4}")
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm, poisson, binom, expon
mean = 50
std_dev = 10
samples = np.random.normal(mean, std_dev, 1000)
plt.figure(figsize=(8, 6))
plt.hist(samples, bins=30, density=True, alpha=0.6, color='blue')
x = np.linspace(mean - 4*std_dev, mean + 4*std_dev, 100)
plt.plot(x, norm.pdf(x, mean, std_dev), 'r-', lw=2, label='Normal Distribution')
plt.title('Normal Distribution Example (Quality Control)')
plt.xlabel('Values')
plt.ylabel('Probability Density')
plt.legend()
plt.grid(True)
plt.show()
lambda_param = 5
k = 3
prob_3_events = poisson.pmf(k, lambda_param)
print(f"Probability of 3 events occurring in an hour: {prob_3_events}")
```

```
n = 10
p = 0.6
k_success = 7
prob_7_success = binom.pmf(k_success, n, p)
print(f"Probability of 7 successes out of 10 trials: {prob_7_success}")
exp_samples = np.random.exponential(scale=2, size=1000)
plt.figure(figsize=(8, 6))
plt.hist(exp_samples, bins=30, density=True, alpha=0.6, color='green')
x_exp = np.linspace(0, 10, 100)
plt.plot(x_exp, expon.pdf(x_exp, scale=2), 'r-', lw=2, label='Exponential Distribution')
plt.title('Exponential Distribution Example (Reliability Analysis)')
plt.xlabel('Values')
plt.ylabel('Probability Density')
plt.legend()
plt.grid(True)
plt.show()
```

PROGRAM 2

REGRESSION

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
np.random.seed(42)
X = np.random.rand(100, 1) * 10
y = 2 * X.squeeze() + np.random.randn(100) * 2
plt.figure(figsize=(8, 4))
plt.scatter(X, y)
plt.title('Scatter Plot')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True)
correlation_coefficient = np.corrcoef(X.squeeze(), y)[0, 1]
print(f"Correlation Coefficient: {correlation_coefficient}")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred = lin_reg.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared Score: {r2}")
plt.figure(figsize=(8, 4))
plt.scatter(X_test, y_test, color='black')
plt.plot(X_test, y_pred, color='blue', linewidth=3)
plt.title('Linear Regression Prediction')

plt.xlabel('X')

plt.ylabel('Y')

plt.grid(True)

iris = load_iris()

X_iris = iris.data[:, :2]

y_iris = iris.target

log_reg = LogisticRegression()

log_reg.fit(X_iris, y_iris)

x_min, x_max = X_iris[:, 0].min() - 1, X_iris[:, 0].max() + 1

y_min, y_max = X_iris[:, 1].min() - 1, X_iris[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1), np.arange(y_min, y_max,
0.1))

Z = log_reg.predict(np.c_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.figure(figsize=(8, 6))

plt.contourf(xx, yy, Z, alpha=0.4)

plt.scatter(X_iris[:, 0], X_iris[:, 1], c=y_iris, s=20, edgecolor='k')

plt.title('Logistic Regression (Iris dataset)')

plt.xlabel('Sepal Length')

plt.ylabel('Sepal Width')

plt.grid(True)

plt.show()
```