Detailed Explanation and Implementations for Key Topics

1. Probability & Random Variables

Probability Basics

- Probability measures the likelihood of an event occurring.
- Example: Tossing a coin twice. Sample space $S = \{\text{HH, HT, TH, TT}\}$. Probability of getting at least one head:

$$P(\text{at least one head}) = 1 - P(\text{no heads}) = 1 - P(\text{TT}) = 1 - \frac{1}{4} = \frac{3}{4}.$$

Python Implementation: Probability of Rolling an Even Number

Listing 1: Probability Basics

```python
# Probability of rolling an even number on a die
favorable_outcomes = 3  # {2, 4, 6}
total_outcomes = 6
probability = favorable_outcomes / total_outcomes
print(f"Probability of rolling an even number: {probability}")
```

Expected Value (Discrete Random Variable)

$$E[X] = \sum_{x \in S} x \cdot P(X = x)$$

Listing 2: Expected Value of a Dice Roll

```python
# Expected value of a dice roll
outcomes = [1, 2, 3, 4, 5, 6]
probabilities = [1/6] * 6  # Uniform distribution
expected_value = sum(x * p for x, p in zip(outcomes, probabilities))
print(f"Expected Value: {expected_value}")
```

Variance

$$\text{Var}(X) = E[(X - E[X])^2] = E[X^2] - (E[X])^2$$

Listing 3: Variance of a Dice Roll

```python
# Variance of a dice roll
mean = expected_value
variance = sum((x - mean)**2 * p for x, p in zip(outcomes, probabilities))
print(f"Variance: {variance}")
```

2. Finite Markov Chains

Python Implementation: Stationary Distribution

Listing 4: Markov Chain: Stationary Distribution

```python
import numpy as np

# Example: Weather Markov Chain (Sunny or Rainy)
P = np.array([
    [0.8, 0.2],  # Transition from Sunny
    [0.5, 0.5]   # Transition from Rainy
])

```

```
# Stationary distribution
eigvals, eigvecs = np.linalg.eig(P.T)
stationary = eigvecs[:, np.isclose(eigvals, 1)]
stationary = stationary / stationary.sum()  # Normalize
print(f"Stationary Distribution: {stationary.flatten()}")
```

3. Concentration of Measure

Markov's Inequality

$$P(X \geq a) \leq \frac{E[X]}{a}, \, a > 0$$

Listing 5: Markov's Inequality

```
E_X = 10  # Expected value
a = 20
P = E_X / a
print(f"Markov's Inequality: P(X >= {a}) <= {P}")
```

Chebyshev's Inequality

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

Listing 6: Chebyshev's Inequality

```
mean = 50
std_dev = 10
k = 3
P = 1 / k**2
print(f"Chebyshev's Inequality: P(|X - {mean}| >= {k * std_dev}) <= {P}")
```

Hoeffding's Inequality

$$P(|\bar{X} - \mu| \geq \epsilon) \leq 2 \exp\left(-\frac{2n\epsilon^2}{(b-a)^2}\right)$$

Listing 7: Hoeffding's Inequality

```
import math

n = 100  # Number of samples
epsilon = 0.1
a, b = 0, 1  # Bounds of the random variable
P = 2 * math.exp(-2 * n * epsilon**2 / (b - a)**2)
print(f"Hoeffding's Inequality: P(|X  -   | >= {epsilon}) <= {P}")
```

4. Random Variable Generation

Python Implementation: Inverse Transform Sampling

Listing 8: Inverse Transform Sampling for Exponential Distribution

```
import numpy as np

# Exponential distribution
lambda_param = 1.0
u = np.random.uniform(0, 1, 1000)
x = -np.log(1 - u) / lambda_param
print(f"Generated Random Variables (Exponential): {x[:5]}")
```

5. Regression

Linear Regression

$$y = \beta_0 + \beta_1 x + \epsilon$$

Listing 9: Linear Regression

```
from sklearn.linear_model import LinearRegression
import numpy as np

# Example dataset
X = np.array([[1], [2], [3], [4], [5]])  # Independent variable
y = np.array([3, 4, 2, 5, 6])            # Dependent variable

# Fit linear regression model
model = LinearRegression().fit(X, y)
print(f"Intercept: {model.intercept_}")
print(f"Slope: {model.coef_[0]}")
```

Logistic Regression

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

Listing 10: Logistic Regression

```
from sklearn.linear_model import LogisticRegression

# Example dataset
X = np.array([[1], [2], [3], [4], [5]])  # Independent variable
y = np.array([0, 0, 0, 1, 1])            # Binary target variable

# Fit logistic regression model
logistic_model = LogisticRegression().fit(X, y)
print(f"Logistic Regression Coefficients: {logistic_model.coef_}")
```