Random Numbers for Business Simulation

As CEO, you need to simulate scenarios for strategic planning. NumPy's random functions help model uncertainty.

```python
import numpy as np

# Set seed for reproducibility
np.random.seed(42)

# Simulate dice rolls (discrete uniform)
dice_rolls = np.random.randint(1, 7, size=100)  # 100 rolls of a 6-sided
die
print(f"100 dice rolls average: {dice_rolls.mean():.2f} (expected: 3.5)")

# Simulate daily sales (continuous uniform)
daily_sales = np.random.uniform(80000, 120000, size=7)  # Week of sales
print(f"\nWeek's sales: ${daily_sales.sum():,.0f}")

# Simulate customer wait times (normal distribution)
wait_times = np.random.normal(90, 15, size=1000)  # Mean 90 sec, std 15 sec
print(f"\nWait time analysis:")
print(f"Average: {wait_times.mean():.1f} seconds")
print(f"% under 2 minutes: {(wait_times < 120).sum() / 10:.1f}%")
```

```
100 dice rolls average: 3.69 (expected: 3.5)

Week's sales: $706,223

Wait time analysis:
Average: 90.7 seconds
% under 2 minutes: 97.4%
```

Random Number Distributions

- `randint(low, high)` - Integers between low and high-1
- `uniform(low, high)` - Decimal numbers evenly distributed
- `normal(mean, std)` - Bell curve distribution (most common in nature/business)

Exercise 7.1 - Simulate Future Scenarios

Use random numbers to simulate business scenarios for strategic planning.

```python
import numpy as np
np.random.seed(200)  # For consistent results
```

```python
# YOUR CODE BELOW
# 1. Simulate 1000 dice rolls and calculate the average
dice_outcomes =
average_roll =

# 2. Simulate next quarter sales (90 days) between $100k and $150k per day
quarter_sales =
total_quarter =

# 3. Simulate customer ratings (normal distribution: mean=4.3, std=0.4)
# Generate 500 ratings, then clip to keep between 1 and 5
ratings =
ratings = np.clip(ratings, 1, 5)  # Keep in valid range
percent_satisfied =

print(f"Dice simulation (1000 rolls): {average_roll:.2f}")
print(f"Projected quarterly revenue: ${total_quarter:,.0f}")
print(f"Customer satisfaction (4+ stars): {percent_satisfied:.1f}%")
```

```python
# Test your simulations
assert 3.3 < average_roll < 3.7, f"Dice average should be near 3.5, got
{average_roll}"
assert 11000000 < total_quarter < 12000000, f"Quarter sales should be
~11.5M"
assert 70 < percent_satisfied < 80, f"Satisfaction should be ~75%"
print("Perfect! You can now simulate future scenarios for strategic
planning!")
```

Bibliography