📚 Module 11: Percentages - Complete Notes

🎯 What You'll Learn

In this module, you'll master **percentages** — "per hundred" calculations essential for pass rates, coverage metrics, and performance comparisons.

📖 Concept Explained (Like a YouTube Video)

The Basics

Percent means "per 100." It normalizes ratios to a base of 100 for easy comparison.

```
75% means 75 out of 100, or 75/100 = 0.75

Conversions:
Fraction → %:  3/4 = 0.75 = 75%
Decimal → %:   0.85 × 100 = 85%
% → Decimal:   65% ÷ 100 = 0.65
```

Key Formulas

```
Percentage of a number:
    Result = Value × (Percent / 100)
    20% of 150 = 150 × 0.20 = 30

Percent change:
    Change % = ((New - Old) / Old) × 100
    100 → 120: ((120-100)/100) × 100 = 20% increase

Find original from percentage:
    Original = Result / (1 + Percent/100)
    Final $120 after 20% increase: 120 / 1.20 = $100 original
```

💻 Programming Connection

Code Examples

```
# Example 1: Basic Percentage Calculations

def percentage_of(percent, value):
    """Calculate percent of a value"""
    return value * (percent / 100)

print(percentage_of(20, 150))   # 30.0 (20% of 150)
print(percentage_of(8.5, 200))  # 17.0 (8.5% tax on 200)
print(percentage_of(15, 80))    # 12.0 (15% tip on 80)
```

```python
# Example 2: Percentage Change

def percentage_change(old, new):
    """Calculate percentage change from old to new"""
    if old == 0:
        return float('inf') if new > 0 else 0
    return ((new - old) / old) * 100

print(percentage_change(100, 120))  # 20.0% increase
print(percentage_change(100, 80))   # -20.0% decrease
print(percentage_change(50, 75))    # 50.0% increase
```

```python
# Example 3: Apply Percentage Change

def apply_change(value, percent):
    """Apply percentage increase (positive) or decrease (negative)"""
    return value * (1 + percent / 100)

print(apply_change(100, 20))    # 120.0 (add 20%)
print(apply_change(100, -20))   # 80.0 (subtract 20%)
print(apply_change(80, 25))     # 100.0 (add 25%)
```

```python
# Example 4: Calculate Discount

def calculate_discount(original_price, discount_percent):
    """Calculate price after discount"""
    discount = original_price * (discount_percent / 100)
    final_price = original_price - discount

    return {
        "original": original_price,
        "discount_percent": discount_percent,
        "discount_amount": discount,
        "final_price": final_price,
        "savings": f"You save ${discount:.2f}"
    }

result = calculate_discount(80, 25)
print(result)
# {'original': 80, 'discount_percent': 25, 'discount_amount': 20.0,
#  'final_price': 60.0, 'savings': 'You save $20.00'}
```

```python
# Example 5: The Percentage Trap!

# IMPORTANT: +50% then -50% does NOT equal original!
value = 100
value = apply_change(value, 50)   # 150 (up 50%)
value = apply_change(value, -50)  # 75 (down 50%)
```

```
print(value)  # 75, NOT 100!

# Why? 50% of 150 is 75, not 50!
```

🧪 SDET/Testing Application

```
# SDET Scenario: Calculate Test Pass Rate

def calculate_pass_rate(passed, failed, skipped=0):
    """Calculate test pass rate"""
    total = passed + failed + skipped
    if total == 0:
        return {"error": "No tests run"}

    pass_rate = (passed / total) * 100
    fail_rate = (failed / total) * 100
    skip_rate = (skipped / total) * 100

    return {
        "total": total,
        "passed": passed,
        "failed": failed,
        "skipped": skipped,
        "pass_rate": f"{pass_rate:.2f}%",
        "fail_rate": f"{fail_rate:.2f}%",
        "skip_rate": f"{skip_rate:.2f}%",
        "status": "✅ PASS" if pass_rate >= 95 else "❌ FAIL"
    }

result = calculate_pass_rate(95, 3, 2)
print(f"Pass rate: {result['pass_rate']}")
print(f"Status: {result['status']}")
```

```
# SDET Scenario: Performance Regression Check

def check_regression(baseline_ms, current_ms, threshold_percent=10):
    """Check if performance regressed beyond threshold"""
    change = percentage_change(baseline_ms, current_ms)

    return {
        "baseline": f"{baseline_ms}ms",
        "current": f"{current_ms}ms",
        "change": f"{change:+.2f}%",
        "threshold": f"{threshold_percent}%",
        "regressed": change > threshold_percent,
        "status": "🔴 REGRESSION" if change > threshold_percent else "🟢 OK"
    }

result = check_regression(100, 115)
```

```
print(f"Change: {result['change']}")
print(f"Status: {result['status']}")
# Change: +15.00%, Status: 🔴 REGRESSION
```

🎓 Practice Problems

Problem 1: Easy 🟢

Challenge: 85 out of 100 tests passed. What's the pass rate?

Problem 2: Medium 🟡

Scenario: Price was $80, now $100.
Challenge: What's the percentage increase?

Problem 3: Application 🔴

Scenario: Your API response time baseline was 200ms. Current is 230ms. Threshold is 10%.
Challenge: Is this a regression?

⚠️ Common Mistakes

❌ **Mistake 1**: +50% then -50% equals original
✅ **Fix**: NO! 100 → 150 → 75

❌ **Mistake 2**: Confusing percentage OF vs percentage CHANGE
✅ **Fix**:

- OF: 20% of 100 = 20
- CHANGE: 100→120 = 20% increase

🔑 Key Takeaways

✅ **Percent = Per 100** — Always compare to 100
✅ **To decimal**: Divide by 100
✅ **Change formula**: `((new - old) / old) × 100`
✅ **+X% then -X% ≠ original** — Be careful!

💾 *Save as: `Module_11_Percentages.md`*