📚 Module 15: Simple Equations - Complete Notes

🎯 What You'll Learn

In this module, you'll master **solving equations** — finding unknown values by working backward from known results.

📖 Concept Explained (Like a YouTube Video)

The Basics

An **equation** has an equals sign — both sides must balance. Solving means finding the unknown value.

```
3x + 5 = 20

Step 1: Subtract 5 from both sides
        3x = 15

Step 2: Divide both sides by 3
        x = 5

Verify: 3(5) + 5 = 15 + 5 = 20 ✓
```

The Golden Rule

Whatever you do to one side, do to the other.

💻 Programming Connection

Code Examples

```python
# Example 1: Solve Linear Equation

def solve_linear(a, b, c):
    """
    Solve: ax + b = c
    Solution: x = (c - b) / a
    """
    if a == 0:
        return None if b != c else "infinite"
    return (c - b) / a

# 3x + 5 = 20
print(solve_linear(3, 5, 20))  # 5.0

# 2x - 10 = 0
print(solve_linear(2, -10, 0))  # 5.0
```

```python
# Example 2: Reverse Calculation (Find What You Need)

def find_required_score(current_avg, num_tests, target_avg):
    """
    What score needed on next test to reach target average?

    Equation: (current_total + x) / (num_tests + 1) = target_avg
    Solution: x = target_avg × (num_tests + 1) - current_total
    """
    current_total = current_avg * num_tests
    required = target_avg * (num_tests + 1) - current_total
    return required

# Current avg 80 on 4 tests, want 85 avg
result = find_required_score(80, 4, 85)
print(f"Need to score: {result}")  # 105
```

```python
# Example 3: Validate a Solution

def validate_equation(a, b, c, x):
    """Check if x is correct solution for ax + b = c"""
    left_side = a * x + b
    right_side = c
    return abs(left_side - right_side) < 0.0001

# Verify: 3x + 5 = 20, x = 5
print(validate_equation(3, 5, 20, 5))   # True
print(validate_equation(3, 5, 20, 4))   # False
```

```python
# Example 4: Find Original Price from Discounted

def find_original_price(final_price, discount_percent):
    """
    final = original × (1 - discount/100)
    original = final / (1 - discount/100)
    """
    return final_price / (1 - discount_percent / 100)

# Sale price is $80 after 20% discount
original = find_original_price(80, 20)
print(f"Original price: ${original}")  # $100.0
```

🧪 SDET/Testing Application

```python
# SDET Scenario: Calculate Required Pass Count

def required_passes_for_rate(total_tests, target_rate):
```

```
    """How many tests must pass to achieve target rate?"""
    # passed / total = target_rate
    # passed = total × target_rate
    return int(total_tests * target_rate)

# Need 80% pass rate on 50 tests
result = required_passes_for_rate(50, 0.80)
print(f"Need {result} passes out of 50")  # 40
```

```
# SDET Scenario: Find Max Allowed Failures

def max_failures_for_rate(total_tests, min_pass_rate):
    """Max failures while maintaining minimum pass rate"""
    min_passes = int(total_tests * min_pass_rate)
    max_failures = total_tests - min_passes
    return max_failures

# 100 tests, need 95% pass rate
result = max_failures_for_rate(100, 0.95)
print(f"Can afford {result} failures")  # 5
```

🎓 Practice Problems

Problem 1: Easy 🟢

Challenge: Solve for x: `2x + 6 = 20`

Problem 2: Medium 🟡

Scenario: Total bill is $108. Tax was 8% of original price.
Challenge: What was the original price?

Problem 3: Application 🔴

Scenario: Your pass rate is 92%. You need 95%. You have 100 tests.
Challenge: How many more tests need to pass?

🔑 Key Takeaways

✅ **Equation = Balance** — Both sides equal
✅ **Isolate the unknown** — Move everything else
✅ **Inverse operations** — Undo to solve
✅ **Verify** — Plug back in to check

💾 *Save as:* `Module_15_Simple_Equations.md`