📚 Module 17: Inequalities - Complete Notes

🎯 What You'll Learn

In this module, you'll master **inequalities** — range constraints and boundary conditions essential for validation.

📖 Concept Explained (Like a YouTube Video)

The Basics

Inequalities define **ranges** instead of exact values.

```
x > 5     : Any value greater than 5 (excludes 5)
x >= 5    : Any value 5 or greater (includes 5)
x < 10    : Any value less than 10 (excludes 10)
x <= 10   : Any value 10 or less (includes 10)

5 < x < 10: Any value between 5 and 10 (excludes both)
5 <= x <= 10: Any value from 5 to 10 (includes both)
```

The Key Difference

```
>  EXCLUDES the boundary (strict)
>= INCLUDES the boundary

Number line for x > 5:
    ──────○═══════════>
          5    (open circle = excluded)

Number line for x >= 5:
    ──────●═══════════>
          5    (filled circle = included)
```

💻 Programming Connection

Code Examples

```python
# Example 1: Range Validation

def is_in_range(value, min_val, max_val, inclusive=True):
    """Check if value is within range"""
    if inclusive:
        return min_val <= value <= max_val
    else:
        return min_val < value < max_val

print(is_in_range(5, 1, 10))          # True
```

```
print(is_in_range(10, 1, 10))         # True (inclusive)
print(is_in_range(10, 1, 10, False))  # False (exclusive)
```

```
# Example 2: Boundary Testing Cases

def generate_boundary_tests(min_val, max_val):
    """Generate boundary test cases"""
    return [
        {"value": min_val - 1, "expected": "invalid", "type": "below_min"},
        {"value": min_val, "expected": "valid", "type": "at_min"},
        {"value": min_val + 1, "expected": "valid", "type": "above_min"},
        {"value": (min_val + max_val) // 2, "expected": "valid", "type": "middle"},
        {"value": max_val - 1, "expected": "valid", "type": "below_max"},
        {"value": max_val, "expected": "valid", "type": "at_max"},
        {"value": max_val + 1, "expected": "invalid", "type": "above_max"},
    ]

# Age validation (18-65)
cases = generate_boundary_tests(18, 65)
for case in cases:
    print(f"Age {case['value']}: {case['expected']} ({case['type']})")
```

```
# Example 3: Categorize by Range

def categorize_score(score):
    """Categorize score into grade bands"""
    if score >= 90:
        return "A"
    elif score >= 80:  # 80 <= score < 90
        return "B"
    elif score >= 70:  # 70 <= score < 80
        return "C"
    elif score >= 60:  # 60 <= score < 70
        return "D"
    else:              # score < 60
        return "F"

# Test boundaries
for score in [59, 60, 69, 70, 79, 80, 89, 90]:
    print(f"Score {score}: Grade {categorize_score(score)}")
```

```
# Example 4: Compound Inequalities

def check_valid_password_length(length):
    """Password must be 8-20 characters"""
    min_len = 8
    max_len = 20

    if length < min_len:
```

```
        return f"Too short (minimum {min_len})"
    elif length > max_len:
        return f"Too long (maximum {max_len})"
    else:
        return "Valid length"

print(check_valid_password_length(5))   # Too short
print(check_valid_password_length(15))  # Valid length
print(check_valid_password_length(25))  # Too long
```

🧪 SDET/Testing Application

```
# SDET Scenario: Response Time Validation

def validate_response_time(actual_ms, max_allowed_ms):
    """Validate response time against SLA"""
    return {
        "actual": actual_ms,
        "threshold": max_allowed_ms,
        "passed": actual_ms <= max_allowed_ms,  # <= because exactly max is OK
        "margin": max_allowed_ms - actual_ms
    }

result = validate_response_time(185, 200)
print(f"Passed: {result['passed']}, Margin: {result['margin']}ms")
```

```
# SDET Scenario: Age Eligibility Testing

def test_age_validation(validate_func, min_age, max_age):
    """Generate and run age validation tests"""
    test_cases = [
        (min_age - 1, False, "Below minimum"),
        (min_age, True, "At minimum"),
        (min_age + 1, True, "Just above minimum"),
        ((min_age + max_age) // 2, True, "Middle"),
        (max_age - 1, True, "Just below maximum"),
        (max_age, True, "At maximum"),
        (max_age + 1, False, "Above maximum"),
    ]

    results = []
    for age, expected, description in test_cases:
        actual = validate_func(age)
        passed = actual == expected
        results.append({
            "age": age,
            "expected": expected,
            "actual": actual,
            "passed": "✅" if passed else "❌",
```

```
            "description": description
        })

    return results

def is_valid_age(age):
    return 18 <= age <= 65

results = test_age_validation(is_valid_age, 18, 65)
for r in results:
    print(f"Age {r['age']}: {r['passed']} ({r['description']})")
```

🎓 Practice Problems

Problem 1: Easy 🟢

Challenge: A test passes if response time <= 200ms. Is 200ms a pass or fail?

Problem 2: Medium 🟡

Scenario: Valid port range is 1-65535.
Challenge: List all boundary test values you should test.

Problem 3: Application 🔴

Scenario: HTTP 2xx status codes (200-299) are success.
Challenge: Write a function that returns "success", "redirect" (3xx), "client_error" (4xx), or "server_error" (5xx).

🔑 Key Takeaways

✅ `>=` and `<=` **INCLUDE boundary**
✅ `>` and `<` **EXCLUDE boundary**
✅ **Range = Two conditions with AND**
✅ **Boundary testing** — Always test at, just below, just above

---

💾 *Save as:* `Module_17_Inequalities.md`