# 📚 Module 14: Variables & Expressions - Complete Notes

## 🎯 What You'll Learn

In this module, you'll master **variables and expressions** — named containers and formulas that are the foundation of all programming.

---

## 📖 Concept Explained (Like a YouTube Video)

### The Basics

A **variable** is a named container that holds a value. An **expression** combines variables and operations to produce a result.

```
Variable:     x = 5         (container named x holds 5)
Expression:   3x + 2        (formula using x)
Evaluation:   3(5) + 2 = 17 (substitute and calculate)
```

### Key Terms

```
Term        Definition              Example
─────────────────────────────────────────────
Variable    Named value             x, price, count
Constant    Fixed value             π, MAX_SIZE
Coefficient Multiplier of variable  3 in 3x
Expression  Combination of terms    3x + 2y − 5
```

---

## 💻 Programming Connection

### Code Examples

```
# Example 1: Variables as Containers

# Simple assignment
price = 19.99
quantity = 3
name = "Widget"

# Variables can change
count = 0
count = count + 1  # Now 1
count += 1         # Now 2

# Constants (by convention, ALL_CAPS)
MAX_RETRIES = 3
```

```python
TIMEOUT_SECONDS = 30
TAX_RATE = 0.08
```

```python
# Example 2: Expressions (Formulas)

# Arithmetic expression
base_price = 100
tax_rate = 0.08
discount = 20

final_price = base_price * (1 + tax_rate) - discount
print(f"Final price: ${final_price}")  # $88.00

# Boolean expression
age = 25
is_adult = age >= 18
print(f"Is adult: {is_adult}")  # True
```

```python
# Example 3: Building Formulas as Functions

def calculate_bmi(weight_kg, height_m):
    """BMI = weight / height²"""
    return weight_kg / (height_m ** 2)

def compound_interest(principal, rate, years):
    """A = P × (1 + r)^t"""
    return principal * ((1 + rate) ** years)

def celsius_to_fahrenheit(c):
    """F = (9/5)C + 32"""
    return (9/5) * c + 32

# Use them
print(calculate_bmi(70, 1.75))          # 22.86
print(compound_interest(1000, 0.05, 10)) # 1628.89
print(celsius_to_fahrenheit(25))        # 77.0
```

```python
# Example 4: Template Variables (Configuration)

# Config as variables
BASE_URL = "https://api.example.com"
VERSION = "v1"
TIMEOUT = 30

def build_url(endpoint):
    """Build full URL from template"""
    return f"{BASE_URL}/{VERSION}/{endpoint}"
```

```python
print(build_url("users"))   # https://api.example.com/v1/users
print(build_url("orders"))  # https://api.example.com/v1/orders
```

---

## 🧪 SDET/Testing Application

```python
# SDET Scenario: Test Configuration

# Test config as variables
BASE_URL = "https://test.api.com"
MAX_RETRIES = 3
TIMEOUT_MS = 5000
EXPECTED_STATUS = 200

def test_config_summary():
    return {
        "base_url": BASE_URL,
        "max_retries": MAX_RETRIES,
        "timeout": f"{TIMEOUT_MS}ms",
        "expected_status": EXPECTED_STATUS
    }
```

```python
# SDET Scenario: Dynamic Test Data

def generate_test_user(user_id, domain="test.com"):
    """Generate test user with formula-based data"""
    return {
        "id": user_id,
        "email": f"user{user_id}@{domain}",
        "username": f"testuser_{user_id}",
        "age": 20 + (user_id % 50)  # Expression: varies 20-69
    }

# Generate 3 test users
for i in range(1, 4):
    user = generate_test_user(i)
    print(user)
```

---

## 🎓 Practice Problems

### Problem 1: Easy 🟢

**Challenge**: In `5x + 3y - 7`, identify: variables, coefficients, constant.

### Problem 2: Medium 🟡

**Challenge**: Write a function `calculate_total(price, quantity, tax_rate)` that returns the total after tax.

#### Problem 3: Application 🔴

**Scenario**: Test coverage formula: `coverage = covered_lines / total_lines * 100`
**Challenge**: Create a function and calculate coverage for 847 covered out of 1000 total.

---

## 🔑 Key Takeaways

✅ **Variable = Named container** — Holds a value
✅ **Expression = Formula** — Combines variables and operations
✅ **Constants = Fixed values** — Use ALL_CAPS
✅ **Functions = Reusable formulas** — Take inputs, return output

---

💾 *Save as:* `Module_14_Variables_Expressions.md`