ADVANCED LOOP PATTERNS & MINI PROJECT 1: CALCULATOR

---

1. ADVANCED LOOP PATTERNS

1.1 Nested Loops (Vòng lặp lồng nhau)

In [ ]:
```python
# Pattern 1: Bảng cửu chương
print("=== BẢNG CỬU CHƯƠNG ===")
for i in range(2, 10):  # Từ 2 đến 9
    print(f"\nBảng cửu chương {i}:")
    for j in range(1, 11):  # Từ 1 đến 10
        result = i * j
        print(f"{i} x {j} = {result}")
    print("-" * 15)
```

In [ ]:
```python
# Pattern 2: Vẽ hình tam giác
print("=== VẼ CÁC HÌNH ===")

# Tam giác vuông
print("\n1. Tam giác vuông:")
for i in range(1, 6):
    for j in range(i):
        print("*", end="")
    print()  # Xuống dòng

# Tam giác cân
print("\n2. Tam giác cân:")
for i in range(1, 6):
    # In khoảng trắng
    for space in range(5 - i):
        print(" ", end="")
    # In dấu sao
    for star in range(2 * i - 1):
        print("*", end="")
    print()  # Xuống dòng
```

In [ ]:
```python
# Pattern 3: Ma trận số
print("=== MA TRẬN SỐ ===")

# Ma trận đơn vị
size = 5
print(f"\nMa trận đơn vị {size}x{size}:")
for i in range(size):
    for j in range(size):
        if i == j:
            print("1", end=" ")
        else:
```

```
            print("0", end=" ")
    print()

# Ma trận số tăng dần
print(f"\nMa trận số tăng dần:")
number = 1
for i in range(4):
    for j in range(4):
        print(f"{number:2d}", end=" ")
        number += 1
    print()
```

1.2 Loop với Multiple Conditions

In [ ]:
```
# Pattern 4: Tìm kiếm với nhiều điều kiện
def find_special_numbers(start, end):
    """Tìm số vừa chia hết cho 3 vừa chia hết cho 5"""
    found_numbers = []
    current = start

    while current <= end:
        # Kiểm tra nhiều điều kiện
        if current % 3 == 0 and current % 5 == 0:
            found_numbers.append(current)
            print(f"Tìm thấy: {current}")

            # Dừng khi tìm được 3 số
            if len(found_numbers) >= 3:
                print("Đã tìm đủ 3 số, dừng lại!")
                break

        current += 1

    return found_numbers

# Test hàm
result = find_special_numbers(1, 100)
print(f"Kết quả: {result}")
```

In [ ]:
```
# Pattern 5: Processing data với skip conditions
def process_student_scores(scores):
    """Xử lý điểm học sinh với các điều kiện đặc biệt"""
    total = 0
    count = 0
    excellent_count = 0

    for i, score in enumerate(scores):
        print(f"\nXử lý học sinh {i+1}: {score} điểm")

        # Bỏ qua điểm không hợp lệ
        if score < 0 or score > 10:
            print("  ⚠️ Điểm không hợp lệ, bỏ qua")
            continue
```

```
        # Bỏ qua học sinh vắng mặt (điểm = 0)
        if score == 0:
            print("  ❌ Học sinh vắng mặt, bỏ qua")
            continue

        # Xử lý điểm hợp lệ
        total += score
        count += 1

        if score >= 8.5:
            excellent_count += 1
            print(f"  🌟 Học sinh xuất sắc!")
        elif score >= 6.5:
            print(f"  ✅ Học sinh khá")
        else:
            print(f"  📈 Cần cải thiện")

    # Tính kết quả
    if count > 0:
        average = total / count
        print(f"\n=== KẾT QUẢ ===")
        print(f"Số học sinh hợp lệ: {count}")
        print(f"Điểm trung bình: {average:.2f}")
        print(f"Số học sinh xuất sắc: {excellent_count}")
        print(f"Tỷ lệ xuất sắc: {excellent_count/count*100:.1f}%")
    else:
        print("Không có điểm hợp lệ để tính")

# Test với dữ liệu mẫu
test_scores = [8.5, -1, 7.2, 0, 9.5, 15, 6.0, 8.8, 0, 7.5]
process_student_scores(test_scores)
```

2. MINI PROJECT 1: CALCULATOR NÂNG CAO

2.1 Phân tích yêu cầu

Tính năng cần có:

1. ✅ Menu chính với while loop
2. ✅ Phép tính cơ bản (+, -, *, /)
3. ✅ Phép tính nâng cao (^, √, %)
4. ✅ Lịch sử tính toán
5. ✅ Validation input
6. ✅ Xử lý lỗi (chia cho 0, sqrt số âm)
7. ✅ Interface thân thiện

2.2 Implementation Step by Step

In [ ]:

```
# Step 1: Utility Functions
import math
from datetime import datetime
```

```
def get_number(prompt):
    """Lấy số từ người dùng với validation"""
    while True:
        try:
            return float(input(prompt))
        except ValueError:
            print("❌ Vui lòng nhập một số hợp lệ!")

def get_positive_number(prompt):
    """Lấy số dương từ người dùng"""
    while True:
        num = get_number(prompt)
        if num >= 0:
            return num
        print("❌ Số phải không âm!")

def format_result(result):
    """Format kết quả đẹp"""
    if result == int(result):
        return str(int(result))
    else:
        return f"{result:.6f}".rstrip('0').rstrip('.')

def add_to_history(operation, result, history):
    """Thêm vào lịch sử"""
    timestamp = datetime.now().strftime("%H:%M:%S")
    record = f"[{timestamp}] {operation} = {format_result(result)}"
    history.append(record)
    return record

# Test utility functions
print("✅ Utility functions đã sẵn sàng!")
print(f"Test format: {format_result(3.0)} | {format_result(3.14159)}")
```

In [ ]:

```
# Step 2: Basic Operations
def basic_operations(history):
    """Xử lý các phép tính cơ bản"""
    print("\n🔢 === PHÉP TÍNH CƠ BẢN ===")
    print("1. Cộng (+)")
    print("2. Trừ (-)")
    print("3. Nhân (*)")
    print("4. Chia (/)")

    choice = input("\nChọn phép tính (1-4): ")

    if choice not in ["1", "2", "3", "4"]:
        print("❌ Lựa chọn không hợp lệ!")
        return

    # Lấy hai số
    num1 = get_number("Nhập số thứ nhất: ")
    num2 = get_number("Nhập số thứ hai: ")

    # Thực hiện phép tính
    operations = {
```

```
        "1": ("+", lambda x, y: x + y),
        "2": ("-", lambda x, y: x - y),
        "3": ("*", lambda x, y: x * y),
        "4": ("/", lambda x, y: x / y if y != 0 else None)
    }

    operator, func = operations[choice]

    # Xử lý chia cho 0
    if choice == "4" and num2 == 0:
        print("❌ Không thể chia cho 0!")
        return

    result = func(num1, num2)
    operation_str = f"{format_result(num1)} {operator} {format_result(num2)}"

    # Hiển thị kết quả
    print(f"\n✅ Kết quả: {operation_str} = {format_result(result)}")

    # Lưu lịch sử
    record = add_to_history(operation_str, result, history)
    print(f"📝 Đã lưu: {record}")

# Test basic operations
test_history = []
print("✅ Basic operations function sẵn sàng!")
```

In [ ]:

```
# Step 3: Advanced Operations
def advanced_operations(history):
    """Xử lý các phép tính nâng cao"""
    print("\n🚀 === PHÉP TÍNH NÂNG CAO ===")
    print("1. Lũy thừa (x^y)")
    print("2. Căn bậc hai (√x)")
    print("3. Phần trăm (x% của y)")
    print("4. Giai thừa (x!)")
    print("5. Logarit (log x)")

    choice = input("\nChọn phép tính (1-5): ")

    if choice == "1":  # Lũy thừa
        base = get_number("Nhập cơ số: ")
        exponent = get_number("Nhập số mũ: ")
        result = base ** exponent
        operation_str = f"{format_result(base)}^{format_result(exponent)}"

    elif choice == "2":  # Căn bậc hai
        num = get_positive_number("Nhập số (≥0): ")
        result = math.sqrt(num)
        operation_str = f"√{format_result(num)}"

    elif choice == "3":  # Phần trăm
        percent = get_number("Nhập phần trăm: ")
        total = get_number("Nhập tổng số: ")
        result = (percent / 100) * total
        operation_str = f"{format_result(percent)}% của {format_result(total)}"
```

```
    elif choice == "4":  # Giai thừa
        num = get_number("Nhập số nguyên không âm: ")
        if num < 0 or num != int(num):
            print("❌ Giai thừa chỉ tính được cho số nguyên không âm!")
            return

        num = int(num)
        if num > 20:
            print("❌ Số quá lớn! Chỉ tính giai thừa đến 20")
            return

        result = math.factorial(num)
        operation_str = f"{num}!"

    elif choice == "5":  # Logarit
        num = get_number("Nhập số (>0): ")
        if num <= 0:
            print("❌ Logarit chỉ tính được cho số dương!")
            return
        result = math.log10(num)
        operation_str = f"log({format_result(num)})"

    else:
        print("❌ Lựa chọn không hợp lệ!")
        return

    # Hiển thị kết quả
    print(f"\n✅ Kết quả: {operation_str} = {format_result(result)}")

    # Lưu lịch sử
    record = add_to_history(operation_str, result, history)
    print(f"📝 Đã lưu: {record}")

print("✅ Advanced operations function sẵn sàng!")
```

In [ ]:

```
# Step 4: History Management
def show_history(history):
    """Hiển thị lịch sử tính toán"""
    if not history:
        print("\n📜 Chưa có lịch sử tính toán nào!")
        return

    print(f"\n📜 === LỊCH SỬ TÍNH TOÁN ({len(history)} phép tính) ===")

    # Hiển thị tối đa 10 phép tính gần nhất
    recent_history = history[-10:] if len(history) > 10 else history

    for i, record in enumerate(recent_history, 1):
        print(f"{i:2d}. {record}")

    if len(history) > 10:
        print(f"\n... và {len(history) - 10} phép tính cũ hơn")

def clear_history(history):
    """Xóa lịch sử"""
    if not history:
```

```
        print("\n📜 Lịch sử đã trống!")
        return

    confirm = input(f"\n⚠️ Bạn có chắc muốn xóa {len(history)} phép tính? (y/N): ")
    if confirm.lower() in ['y', 'yes', 'có', 'c']:
        history.clear()
        print("✅ Đã xóa toàn bộ lịch sử!")
    else:
        print("❌ Hủy bỏ xóa lịch sử")

def export_history(history):
    """Xuất lịch sử ra text"""
    if not history:
        print("\n📜 Không có lịch sử để xuất!")
        return

    print("\n📄 === XUẤT LỊCH SỬ ===")
    print("Copy đoạn text sau để lưu:")
    print("-" * 50)
    print(f"LỊCH SỬ CALCULATOR - {datetime.now().strftime('%d/%m/%Y %H:%M')}")
    print("-" * 50)
    for i, record in enumerate(history, 1):
        print(f"{i}. {record}")
    print("-" * 50)
    print(f"Tổng cộng: {len(history)} phép tính")

print("✅ History management functions sẵn sàng!")
```

In [ ]:

```
# Step 5: Main Calculator Program
def calculator_main():
    """Chương trình máy tính chính"""
    print("🧮" * 20)
    print("🧮     CALCULATOR NÂNG CAO     🧮")
    print("🧮      Tuần 26 - Python       🧮")
    print("🧮" * 20)

    history = []  # Lưu lịch sử tính toán

    while True:
        print("\n" + "="*50)
        print("📋 MENU CHÍNH")
        print("="*50)
        print("1. 🔢 Phép tính cơ bản (+, -, *, /)")
        print("2. 🚀 Phép tính nâng cao (^, √, %, !, log)")
        print("3. 📜 Xem lịch sử tính toán")
        print("4. 🗑️  Xóa lịch sử")
        print("5. 📄 Xuất lịch sử")
        print("6. ℹ️  Thông tin chương trình")
        print("7. 👋 Thoát")
        print("="*50)

        if history:
            print(f"📊 Đã thực hiện: {len(history)} phép tính")

        choice = input("\n🎯 Chọn chức năng (1-7): ").strip()
```

```
        if choice == "1":
            basic_operations(history)

        elif choice == "2":
            advanced_operations(history)

        elif choice == "3":
            show_history(history)

        elif choice == "4":
            clear_history(history)

        elif choice == "5":
            export_history(history)

        elif choice == "6":
            show_info()

        elif choice == "7":
            print("\n" + "🎉" * 30)
            print("🎉   CẢM ƠN BẠN ĐÃ SỬ DỤNG!    🎉")
            if history:
                print(f"🎉  Bạn đã thực hiện {len(history)} phép tính  🎉")
            print("🎉         HẸN GẶP LẠI!         🎉")
            print("🎉" * 30)
            break

        else:
            print("❌ Lựa chọn không hợp lệ! Vui lòng chọn từ 1-7")
            continue

        # Hỏi có muốn tiếp tục không
        if choice in ["1", "2"]:
            input("\n⏎ Nhấn Enter để tiếp tục...")

def show_info():
    """Hiển thị thông tin chương trình"""
    print("\n" + "ℹ️" * 30)
    print("ℹ️        THÔNG TIN CHƯƠNG TRÌNH        ℹ️")
    print("ℹ️" * 30)
    print("📝 Tên: Calculator Nâng Cao")
    print("🗓️ Phiên bản: 1.0 (Tuần 26)")
    print("👨‍🏫 Giảng viên: Ngọ Công Bình, Nguyễn Văn Anh")
    print("🎯 Mục đích: Học Python cơ bản")
    print("\n🔧 Tính năng:")
    print("   ✅ Phép tính cơ bản (+, -, *, /)")
    print("   ✅ Phép tính nâng cao (^, √, %, !, log)")
    print("   ✅ Lịch sử tính toán với timestamp")
    print("   ✅ Validation input và xử lý lỗi")
    print("   ✅ Interface thân thiện")
    print("\n💡 Kiến thức sử dụng:")
    print("   📚 Variables, Data types, Input/Output")
    print("   📚 Operators, Conditions (if/else)")
    print("   📚 Loops (for, while) và Nested Loops")
    print("   📚 Functions, Exception Handling")
    print("   📚 Modules (math, datetime)")
```

```
    print("" * 30)

# Chạy chương trình chính
if __name__ == "__main__":
    calculator_main()
```

3. BÀI TẬP THỰC HÀNH

3.1 Bài tập về Nested Loops

In [ ]:
```
# Bài tập 1: Vẽ kim tự tháp số
def draw_number_pyramid(height):
    """Vẽ kim tự tháp số"""
    print(f"\n=== KIM TỰ THÁP SỐ (cao {height}) ===")

    for i in range(1, height + 1):
        # In khoảng trắng
        spaces = " " * (height - i)

        # In số tăng dần
        numbers_up = ""
        for j in range(1, i + 1):
            numbers_up += str(j)

        # In số giảm dần
        numbers_down = ""
        for j in range(i - 1, 0, -1):
            numbers_down += str(j)

        # Ghép và in
        full_line = spaces + numbers_up + numbers_down
        print(full_line)

# Test
draw_number_pyramid(5)
```

In [ ]:
```
# Bài tập 2: Ma trận xoắn ốc
def create_spiral_matrix(size):
    """Tạo ma trận xoắn ốc"""
    matrix = [[0] * size for _ in range(size)]

    # Định hướng: phải, xuống, trái, lên
    directions = [(0, 1), (1, 0), (0, -1), (-1, 0)]
    current_dir = 0

    row, col = 0, 0

    for num in range(1, size * size + 1):
        matrix[row][col] = num

        # Tính vị trí tiếp theo
        next_row = row + directions[current_dir][0]
        next_col = col + directions[current_dir][1]
```

```
        # Kiểm tra có cần đổi hướng không
        if (next_row < 0 or next_row >= size or
            next_col < 0 or next_col >= size or
            matrix[next_row][next_col] != 0):
            current_dir = (current_dir + 1) % 4
            next_row = row + directions[current_dir][0]
            next_col = col + directions[current_dir][1]

        row, col = next_row, next_col

    return matrix

def print_matrix(matrix):
    """In ma trận đẹp"""
    size = len(matrix)
    max_num = size * size
    width = len(str(max_num)) + 1

    for row in matrix:
        for num in row:
            print(f"{num:>{width}}", end="")
        print()

# Test
print("\n=== MA TRẬN XOẮN ỐC ===")
spiral = create_spiral_matrix(6)
print_matrix(spiral)
```

In [ ]:

```
# Bài tập 3: Tìm số nguyên tố bằng Sieve of Eratosthenes
def sieve_of_eratosthenes(limit):
    """Tìm tất cả số nguyên tố <= limit bằng thuật toán Sieve"""
    if limit < 2:
        return []

    # Khởi tạo mảng boolean
    is_prime = [True] * (limit + 1)
    is_prime[0] = is_prime[1] = False

    # Thuật toán Sieve
    for i in range(2, int(limit**0.5) + 1):
        if is_prime[i]:
            # Đánh dấu tất cả bội số của i
            for j in range(i * i, limit + 1, i):
                is_prime[j] = False

    # Thu thập số nguyên tố
    primes = []
    for i in range(2, limit + 1):
        if is_prime[i]:
            primes.append(i)

    return primes

def display_primes(primes, per_line=10):
    """Hiển thị số nguyên tố thành dạng bảng"""
```

```
    print(f"\nTìm được {len(primes)} số nguyên tố:")
    print("-" * 50)

    for i, prime in enumerate(primes):
        print(f"{prime:4d}", end=" ")
        if (i + 1) % per_line == 0:
            print()  # Xuống dòng

    if len(primes) % per_line != 0:
        print()  # Xuống dòng cuối

# Test
print("=== TÌM SỐ NGUYÊN TỐ ===")
limit = 100
primes = sieve_of_eratosthenes(limit)
display_primes(primes)
print(f"\nSố nguyên tố lớn nhất <= {limit}: {max(primes)}")
```

3.2 Bài tập nâng cao cho Calculator

In [ ]:

```
# Bài tập 4: Thêm tính năng Memory cho Calculator
class CalculatorMemory:
    """Quản lý bộ nhớ cho calculator"""

    def __init__(self):
        self.memory = 0
        self.memory_history = []

    def memory_store(self, value):
        """Lưu giá trị vào memory"""
        self.memory = value
        timestamp = datetime.now().strftime("%H:%M:%S")
        self.memory_history.append(f"[{timestamp}] MS: {format_result(value)}")
        return f"💾 Đã lưu {format_result(value)} vào memory"

    def memory_recall(self):
        """Lấy giá trị từ memory"""
        timestamp = datetime.now().strftime("%H:%M:%S")
        self.memory_history.append(f"[{timestamp}] MR: {format_result(self.memory)}
        return self.memory

    def memory_add(self, value):
        """Cộng giá trị vào memory"""
        self.memory += value
        timestamp = datetime.now().strftime("%H:%M:%S")
        self.memory_history.append(f"[{timestamp}] M+: {format_result(value)} → {fo
        return f"💾 Memory: {format_result(self.memory)}"

    def memory_subtract(self, value):
        """Trừ giá trị từ memory"""
        self.memory -= value
        timestamp = datetime.now().strftime("%H:%M:%S")
        self.memory_history.append(f"[{timestamp}] M-: {format_result(value)} → {fo
        return f"💾 Memory: {format_result(self.memory)}"
```

```
    def memory_clear(self):
        """Xóa memory"""
        self.memory = 0
        timestamp = datetime.now().strftime("%H:%M:%S")
        self.memory_history.append(f"[{timestamp}] MC: Memory cleared")
        return "🗑️ Đã xóa memory"

    def show_memory_status(self):
        """Hiển thị trạng thái memory"""
        return f"💾 Memory hiện tại: {format_result(self.memory)}"

    def show_memory_history(self):
        """Hiển thị lịch sử memory"""
        if not self.memory_history:
            return "📜 Chưa có thao tác memory nào!"

        result = "\n💾 === LỊCH SỬ MEMORY ===\n"
        for record in self.memory_history[-10:]:  # 10 thao tác gần nhất
            result += f"{record}\n"
        return result

# Test memory functions
memory = CalculatorMemory()
print(memory.memory_store(42))
print(memory.memory_add(8))
print(memory.show_memory_status())
print(memory.show_memory_history())
```

In [ ]:

```
# Bài tập 5: Calculator với tính năng Unit Conversion
def unit_converter():
    """Chuyển đổi đơn vị"""
    conversions = {
        "1": {
            "name": "Độ dài",
            "units": {
                "mm": 0.001, "cm": 0.01, "m": 1, "km": 1000,
                "inch": 0.0254, "ft": 0.3048, "yard": 0.9144, "mile": 1609.34
            }
        },
        "2": {
            "name": "Khối lượng",
            "units": {
                "mg": 0.001, "g": 1, "kg": 1000, "ton": 1000000,
                "oz": 28.3495, "lb": 453.592
            }
        },
        "3": {
            "name": "Nhiệt độ",
            "special": True  # Xử lý đặc biệt
        }
    }

    print("\n🔄 === CHUYỂN ĐỔI ĐƠN VỊ ===")
    print("1. Độ dài (mm, cm, m, km, inch, ft, yard, mile)")
    print("2. Khối lượng (mg, g, kg, ton, oz, lb)")
    print("3. Nhiệt độ (°C, °F, K)")
```

```
    choice = input("\nChọn loại chuyển đổi (1-3): ")

    if choice in ["1", "2"]:
        conversion = conversions[choice]
        print(f"\n{conversion['name']} - Đơn vị có sẵn:")
        for unit in conversion['units'].keys():
            print(f"  • {unit}")

        value = get_number("\nNhập giá trị: ")
        from_unit = input("Từ đơn vị: ").lower().strip()
        to_unit = input("Đến đơn vị: ").lower().strip()

        if from_unit not in conversion['units'] or to_unit not in conversion['units
            print("❌ Đơn vị không hợp lệ!")
            return

        # Chuyển về đơn vị cơ sở rồi chuyển sang đơn vị đích
        base_value = value * conversion['units'][from_unit]
        result = base_value / conversion['units'][to_unit]

        print(f"\n✅ Kết quả: {format_result(value)} {from_unit} = {format_result(r

    elif choice == "3":
        print("\nNhiệt độ - Đơn vị: C (Celsius), F (Fahrenheit), K (Kelvin)")

        temp = get_number("Nhập nhiệt độ: ")
        from_unit = input("Từ đơn vị (C/F/K): ").upper().strip()
        to_unit = input("Đến đơn vị (C/F/K): ").upper().strip()

        if from_unit not in ['C', 'F', 'K'] or to_unit not in ['C', 'F', 'K']:
            print("❌ Đơn vị không hợp lệ!")
            return

        # Chuyển tất cả về Celsius trước
        if from_unit == 'F':
            celsius = (temp - 32) * 5/9
        elif from_unit == 'K':
            celsius = temp - 273.15
        else:
            celsius = temp

        # Chuyển từ Celsius sang đơn vị đích
        if to_unit == 'F':
            result = celsius * 9/5 + 32
        elif to_unit == 'K':
            result = celsius + 273.15
        else:
            result = celsius

        print(f"\n✅ Kết quả: {format_result(temp)}°{from_unit} = {format_result(re

    else:
        print("❌ Lựa chọn không hợp lệ!")
```

```
# Test unit converter
print("✅ Unit converter function sẵn sàng!")
```

4. TỔNG KẾT VÀ BÀI TẬP VỀ NHÀ

4.1 Kiến thức đã học

✅ Advanced Loop Patterns

- Nested loops (vòng lặp lồng nhau)
- Loop với multiple conditions
- Pattern recognition trong loops
- Break và continue trong nested loops

✅ Mini Project: Calculator

- Thiết kế menu với while loop
- Function organization
- Error handling và validation
- Data persistence (history)
- User experience design

4.2 Bài tập về nhà

In [ ]:
```
# HOMEWORK 1: Password Generator
# Tạo chương trình sinh password ngẫu nhiên với các tùy chọn:
# - Độ dài password (8-50 ký tự)
# - Bao gồm chữ hoa, chữ thường, số, ký tự đặc biệt
# - Đảm bảo password mạnh (ít nhất 1 loại ký tự mỗi loại)
# - Lưu lịch sử password đã tạo
# - Kiểm tra độ mạnh password

def password_generator_homework():
    """
    TODO: Implement password generator
    Gợi ý:
    1. import random, string
    2. Tạo character sets cho từng loại
    3. Dùng loop để tạo password
    4. Validate password requirements
    5. Rate password strength (Weak/Medium/Strong)
    """
    pass

print("📝 HOMEWORK 1: Password Generator")
print("Yêu cầu: Tạo chương trình sinh password với đầy đủ tính năng")
```

In [ ]:
```
# HOMEWORK 2: ASCII Art Generator
# Tạo chương trình vẽ ASCII art với các pattern:
# - Tạo text banner từ string input
```

```
# - Vẽ các hình 2D (heart, diamond, star)
# - Animation đơn giản (rotating patterns)
# - Save ASCII art to file

def ascii_art_generator_homework():
    """
    TODO: Implement ASCII art generator
    Gợi ý:
    1. Tạo dictionary cho ASCII patterns của từng chữ cái
    2. Dùng nested loops để render text
    3. Implement geometric shapes với math
    4. Use time.sleep() cho animation
    5. File I/O để save art
    """
    pass

print("📝 HOMEWORK 2: ASCII Art Generator")
print("Yêu cầu: Tạo chương trình vẽ ASCII art và animation")
```

In [ ]:

```
# HOMEWORK 3: Mini Game - Number Guessing với AI
# Nâng cấp game đoán số với AI player:
# - Player vs Computer: Computer đoán số của player
# - Computer vs Player: Player đoán số của computer
# - Computer vs Computer: Xem 2 AI chơi với nhau
# - Thống kê performance (số lần đoán, thời gian)
# - Difficulty levels (Easy/Medium/Hard)

def number_guessing_ai_homework():
    """
    TODO: Implement AI number guessing game
    Gợi ý:
    1. Binary search algorithm cho AI
    2. Random strategy vs Optimal strategy
    3. Game statistics tracking
    4. Multiple game modes
    5. Interactive gameplay
    """
    pass

print("📝 HOMEWORK 3: AI Number Guessing Game")
print("Yêu cầu: Game đoán số với AI player và multiple modes")
```