Student Class & Simple OOP Demo

Tuần 28 - Thứ 5 chiều (14:00-17:00)

Mục tiêu buổi học:

- Hiểu cách tạo và sử dụng class Student
- Thực hành tạo objects từ class
- Làm quen với methods trong class
- Xây dựng demo đơn giản với OOP

1. Ôn tập: Class và Object cơ bản

Từ buổi sáng, chúng ta đã học:

- Class là một khuôn mẫu (template)
- Object là một thực thể được tạo từ class
- Methods là các hàm bên trong class

In [ ]:
```python
# Ví dụ class đơn giản từ buổi sáng
class Person:
    def __init__(self, name):
        self.name = name

    def say_hello(self):
        print(f"Xin chào, tôi là {self.name}")

# Tạo object
person1 = Person("An")
person1.say_hello()
```

2. Xây dựng Class Student

Bây giờ chúng ta sẽ tạo một class Student phức tạp hơn để quản lý thông tin học sinh.

In [ ]:
```python
class Student:
    def __init__(self, name, age, student_id):
        self.name = name
        self.age = age
        self.student_id = student_id
        self.grades = []  # Danh sách điểm số

    def introduce(self):
        print(f"Xin chào, tôi là {self.name}, {self.age} tuổi")
        print(f"Mã sinh viên: {self.student_id}")
```

```
    def add_grade(self, grade):
        if 0 <= grade <= 10:
            self.grades.append(grade)
            print(f"Đã thêm điểm {grade} cho {self.name}")
        else:
            print("Điểm phải từ 0 đến 10")

    def show_grades(self):
        if self.grades:
            print(f"Điểm của {self.name}: {self.grades}")
        else:
            print(f"{self.name} chưa có điểm nào")

# Demo tạo student
student1 = Student("Nguyễn Văn A", 20, "SV001")
student1.introduce()
```

📝 TODO 1: Tạo Student Object

Hãy tạo một student mới với thông tin của bạn và gọi method introduce()

In [ ]:
```
# TODO 1: Tạo student với thông tin của bạn
# my_student = Student("Tên của bạn", tuổi, "Mã SV")
# my_student.introduce()

# Viết code ở đây:
```

3. Thêm Methods cho Student Class

Chúng ta sẽ thêm các methods để quản lý điểm số:

In [ ]:
```
# Demo thêm điểm
student1.add_grade(8.5)
student1.add_grade(9.0)
student1.add_grade(7.5)
student1.show_grades()
```

📝 TODO 2: Thêm điểm cho Student

Thêm một vài điểm cho student bạn vừa tạo và hiển thị danh sách điểm

In [ ]:
```
# TODO 2: Thêm điểm cho student của bạn
# my_student.add_grade(điểm1)
# my_student.add_grade(điểm2)
# my_student.show_grades()

# Viết code ở đây:
```

4. Mở rộng Student Class - Tính điểm trung bình

Thêm method tính điểm trung bình:

In [ ]:
```python
class StudentAdvanced:
    def __init__(self, name, age, student_id):
        self.name = name
        self.age = age
        self.student_id = student_id
        self.grades = []

    def introduce(self):
        print(f"Xin chào, tôi là {self.name}, {self.age} tuổi")
        print(f"Mã sinh viên: {self.student_id}")

    def add_grade(self, grade):
        if 0 <= grade <= 10:
            self.grades.append(grade)
            print(f"Đã thêm điểm {grade} cho {self.name}")
        else:
            print("Điểm phải từ 0 đến 10")

    def show_grades(self):
        if self.grades:
            print(f"Điểm của {self.name}: {self.grades}")
        else:
            print(f"{self.name} chưa có điểm nào")

    def calculate_average(self):
        if self.grades:
            average = sum(self.grades) / len(self.grades)
            return round(average, 2)
        else:
            return 0

    def get_grade_info(self):
        if self.grades:
            avg = self.calculate_average()
            print(f"Thông tin điểm của {self.name}:")
            print(f"Các điểm: {self.grades}")
            print(f"Điểm trung bình: {avg}")

            # Xếp loại dựa trên điểm trung bình
            if avg >= 8.5:
                grade_level = "Xuất sắc"
            elif avg >= 7.0:
                grade_level = "Khá"
            elif avg >= 5.5:
                grade_level = "Trung bình"
            else:
                grade_level = "Yếu"

            print(f"Xếp loại: {grade_level}")
        else:
            print(f"{self.name} chưa có điểm nào")
```

```
# Demo
student2 = StudentAdvanced("Trần Thị B", 19, "SV002")
student2.introduce()
student2.add_grade(8.0)
student2.add_grade(9.5)
student2.add_grade(7.5)
student2.get_grade_info()
```

📝 TODO 3: Tạo Student Advanced

Tạo một StudentAdvanced mới và thêm ít nhất 4 điểm, sau đó hiển thị thông tin chi tiết

In [ ]:
```
# TODO 3: Tạo StudentAdvanced và thêm điểm
# advanced_student = StudentAdvanced("Tên", tuổi, "Mã")
# Thêm 4 điểm
# Hiển thị thông tin

# Viết code ở đây:
```

5. Tạo nhiều Students - Class Demo

Tạo một danh sách nhiều students và quản lý chúng:

In [ ]:
```
# Tạo danh sách students
students = []

# Thêm students
student_a = StudentAdvanced("Nguyễn Văn An", 20, "SV001")
student_b = StudentAdvanced("Trần Thị Bình", 19, "SV002")
student_c = StudentAdvanced("Lê Văn Cường", 21, "SV003")

students.append(student_a)
students.append(student_b)
students.append(student_c)

# Thêm điểm cho từng học sinh
student_a.add_grade(8.5)
student_a.add_grade(9.0)
student_a.add_grade(7.5)

student_b.add_grade(9.5)
student_b.add_grade(8.0)
student_b.add_grade(9.0)

student_c.add_grade(6.5)
student_c.add_grade(7.0)
student_c.add_grade(8.5)

# Hiển thị thông tin tất cả students
print("=== THÔNG TIN TẤT CẢ HỌC SINH ===")
for i, student in enumerate(students, 1):
```

```
    print(f"\n--- Học sinh {i} ---")
    student.get_grade_info()
```

📝 TODO 4: Tạo lớp học của riêng bạn

Tạo một danh sách 3 students với thông tin và điểm số khác nhau, sau đó hiển thị thông tin tất cả

In [ ]:
```
# TODO 4: Tạo lớp học của riêng bạn
# my_class = []
# Tạo 3 students
# Thêm điểm cho mỗi student
# Hiển thị thông tin tất cả

# Viết code ở đây:
```

6. Tìm kiếm Students

Tạo functions để tìm kiếm students theo các tiêu chí khác nhau:

In [ ]:
```
def find_student_by_id(students, student_id):
    """Tìm học sinh theo mã sinh viên"""
    for student in students:
        if student.student_id == student_id:
            return student
    return None

def find_students_by_grade_range(students, min_grade, max_grade):
    """Tìm học sinh có điểm trung bình trong khoảng"""
    result = []
    for student in students:
        avg = student.calculate_average()
        if min_grade <= avg <= max_grade:
            result.append(student)
    return result

def get_top_students(students, top_n=3):
    """Lấy top N học sinh có điểm cao nhất"""
    # Tạo danh sách (student, average) và sắp xếp
    student_averages = []
    for student in students:
        avg = student.calculate_average()
        student_averages.append((student, avg))

    # Sắp xếp theo điểm trung bình giảm dần
    student_averages.sort(key=lambda x: x[1], reverse=True)

    # Lấy top N
    return [student for student, avg in student_averages[:top_n]]

# Demo tìm kiếm
print("=== TÌM KIẾM DEMO ===")
```

```
# Tìm theo ID
found_student = find_student_by_id(students, "SV002")
if found_student:
    print(f"Tìm thấy: {found_student.name}")
    found_student.get_grade_info()

# Tìm theo khoảng điểm
good_students = find_students_by_grade_range(students, 8.0, 10.0)
print(f"\nHọc sinh có điểm từ 8.0-10.0: {len(good_students)} người")
for student in good_students:
    print(f"- {student.name}: {student.calculate_average()}")

# Top students
top_students = get_top_students(students, 2)
print(f"\nTop 2 học sinh:")
for i, student in enumerate(top_students, 1):
    print(f"{i}. {student.name}: {student.calculate_average()}")
```

📝 TODO 5: Tìm kiếm trong lớp học của bạn

Sử dụng các functions tìm kiếm với lớp học bạn tạo ở TODO 4

In [ ]:
```
# TODO 5: Tìm kiếm trong lớp học của bạn
# Tìm 1 học sinh theo ID
# Tìm học sinh có điểm từ 7.0-9.0
# Tìm top 2 học sinh có điểm cao nhất

# Viết code ở đây:
```

7. Mini Project: Student Management System

Tạo một hệ thống quản lý học sinh đơn giản với menu:

In [ ]:
```
class StudentManager:
    def __init__(self):
        self.students = []

    def add_student(self, name, age, student_id):
        """Thêm học sinh mới"""
        # Kiểm tra ID đã tồn tại chưa
        for student in self.students:
            if student.student_id == student_id:
                print(f"Mã sinh viên {student_id} đã tồn tại!")
                return False

        new_student = StudentAdvanced(name, age, student_id)
        self.students.append(new_student)
        print(f"Đã thêm học sinh {name} (ID: {student_id})")
        return True

    def show_all_students(self):
```

```
        """Hiển thị tất cả học sinh"""
        if not self.students:
            print("Chưa có học sinh nào!")
            return

        print("=== DANH SÁCH HỌC SINH ===")
        for i, student in enumerate(self.students, 1):
            print(f"{i}. {student.name} (ID: {student.student_id}) - Tuổi: {student
            print(f"   Điểm TB: {student.calculate_average()}")

    def find_student(self, student_id):
        """Tìm học sinh theo ID"""
        for student in self.students:
            if student.student_id == student_id:
                return student
        return None

    def add_grade_to_student(self, student_id, grade):
        """Thêm điểm cho học sinh"""
        student = self.find_student(student_id)
        if student:
            student.add_grade(grade)
            return True
        else:
            print(f"Không tìm thấy học sinh có ID: {student_id}")
            return False

# Demo Student Manager
manager = StudentManager()

# Thêm học sinh
manager.add_student("Nguyễn Văn A", 20, "SV001")
manager.add_student("Trần Thị B", 19, "SV002")
manager.add_student("Lê Văn C", 21, "SV003")

# Thêm điểm
manager.add_grade_to_student("SV001", 8.5)
manager.add_grade_to_student("SV001", 9.0)
manager.add_grade_to_student("SV002", 9.5)
manager.add_grade_to_student("SV002", 8.0)

# Hiển thị tất cả
manager.show_all_students()
```

📝 TODO 6: Tạo Student Manager của riêng bạn

Tạo một StudentManager mới và thực hiện các thao tác sau:

1. Thêm 3 học sinh
2. Thêm điểm cho mỗi học sinh
3. Hiển thị danh sách tất cả học sinh
4. Tìm một học sinh cụ thể và hiển thị thông tin chi tiết

In [ ]:
```python
# TODO 6: Tạo Student Manager của riêng bạn
# my_manager = StudentManager()
# Thêm 3 học sinh
# Thêm điểm cho mỗi học sinh
# Hiển thị danh sách
# Tìm và hiển thị thông tin chi tiết 1 học sinh

# Viết code ở đây:
```

8. Tương tác với Menu (Bonus)

Tạo menu đơn giản để tương tác với Student Manager:

In [ ]:
```python
def show_menu():
    print("\n=== HỆ THỐNG QUẢN LÝ HỌC SINH ===")
    print("1. Thêm học sinh mới")
    print("2. Hiển thị tất cả học sinh")
    print("3. Thêm điểm cho học sinh")
    print("4. Tìm học sinh theo ID")
    print("5. Thoát")
    print("=" * 35)

def run_student_management():
    manager = StudentManager()

    # Thêm một số dữ liệu mẫu
    manager.add_student("Nguyễn Văn A", 20, "SV001")
    manager.add_student("Trần Thị B", 19, "SV002")
    manager.add_grade_to_student("SV001", 8.5)
    manager.add_grade_to_student("SV002", 9.0)

    print("Chào mừng đến với hệ thống quản lý học sinh!")
    print("(Đã có sẵn 2 học sinh mẫu)")

    while True:
        show_menu()
        choice = input("Nhập lựa chọn của bạn (1-5): ")

        if choice == "1":
            name = input("Nhập tên học sinh: ")
            age = int(input("Nhập tuổi: "))
            student_id = input("Nhập mã sinh viên: ")
            manager.add_student(name, age, student_id)

        elif choice == "2":
            manager.show_all_students()

        elif choice == "3":
            student_id = input("Nhập mã sinh viên: ")
            grade = float(input("Nhập điểm (0-10): "))
            manager.add_grade_to_student(student_id, grade)

        elif choice == "4":
```

```
            student_id = input("Nhập mã sinh viên cần tìm: ")
            student = manager.find_student(student_id)
            if student:
                print(f"\nTìm thấy học sinh:")
                student.get_grade_info()
            else:
                print(f"Không tìm thấy học sinh có ID: {student_id}")

        elif choice == "5":
            print("Cảm ơn bạn đã sử dụng hệ thống!")
            break

        else:
            print("Lựa chọn không hợp lệ, vui lòng thử lại!")

        input("\nNhấn Enter để tiếp tục...")

# Chạy demo (uncomment dòng dưới để chạy)
# run_student_management()
```

9. Tóm tắt bài học

Trong buổi học này, chúng ta đã:

✅ Kiến thức đã học:

- Tạo class Student với các attributes cơ bản (name, age, student_id, grades)
- Sử dụng method `__init__()` để khởi tạo object
- Tạo các methods để quản lý thông tin học sinh
- Tính toán điểm trung bình và xếp loại
- Quản lý nhiều objects trong danh sách
- Tìm kiếm và lọc dữ liệu
- Xây dựng StudentManager class để quản lý tập trung

💡 Khái niệm quan trọng:

- **Class**: Khuôn mẫu để tạo objects
- **Object**: Thực thể được tạo từ class
- **Attributes**: Thuộc tính của object (name, age, grades...)
- **Methods**: Hành động mà object có thể thực hiện
- **self**: Tham chiếu đến object hiện tại

🎯 Lợi ích của OOP:

- Code dễ đọc và dễ hiểu
- Dữ liệu được tổ chức rõ ràng
- Dễ bảo trì và mở rộng
- Tái sử dụng code hiệu quả

📝 TODO 7: Bài tập cuối buổi

Tạo một class Teacher tương tự Student với các thông tin:

- name, age, teacher_id
- subjects (danh sách môn dạy)
- Các methods: introduce(), add_subject(), show_subjects()

Sau đó tạo 2 teachers và thêm môn học cho họ.

In [ ]:
```python
# TODO 7: Tạo class Teacher
class Teacher:
    def __init__(self, name, age, teacher_id):
        # Viết code ở đây
        pass

    def introduce(self):
        # Viết code ở đây
        pass

    def add_subject(self, subject):
        # Viết code ở đây
        pass

    def show_subjects(self):
        # Viết code ở đây
        pass

# Tạo 2 teachers và test
# teacher1 = Teacher("Nguyễn Văn X", 35, "GV001")
# teacher2 = Teacher("Trần Thị Y", 30, "GV002")
# Thêm môn học và test các methods
```

10. Bài tập về nhà

Bài 1: Mở rộng Student class

Thêm các methods sau vào Student class:

- `remove_grade(index)` : Xóa điểm tại vị trí index
- `update_grade(index, new_grade)` : Cập nhật điểm tại vị trí index
- `get_highest_grade()` : Trả về điểm cao nhất
- `get_lowest_grade()` : Trả về điểm thấp nhất

Bài 2: Tạo School class

Tạo class School để quản lý cả Students và Teachers:

- Attributes: school_name, students, teachers
- Methods: add_student(), add_teacher(), show_school_info()

Bài 3: Thống kê

Viết functions để:

- Tìm học sinh có điểm cao nhất trong trường
- Tính điểm trung bình của cả lớp
- Đếm số học sinh theo từng xếp loại

💡 Gợi ý cho buổi sau:

Buổi học tiếp theo chúng ta sẽ học về:

- Methods và **init** nâng cao
- Bank Account System project
- Cách tổ chức code OOP tốt hơn

Kết thúc bài học W28 - T5 chiều

Chúc các bạn học tốt! 🎓