# Управление университетом

**Описание задачи**:
Создайте программу для управления университетом. В университете есть студенты, преподаватели и курсы.

**Требования**:

1. Создайте классы для студентов, преподавателей и курсов.
2. Реализуйте возможность регистрации студентов и преподавателей.
3. Реализуйте систему назначения преподавателей на курсы и записи студентов на курсы.
4. Храните информацию о студентах, преподавателях и курсах.

**Решение**:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

class Student(Person):
    def __init__(self, name, age, student_id):
        super().__init__(name, age)
        self.student_id = student_id
        self.courses = []

    def enroll(self, course):
        self.courses.append(course)
        print(f"{self.name} enrolled in {course.name}.")

class Teacher(Person):
    def __init__(self, name, age, specialty):
        super().__init__(name, age)
        self.specialty = specialty
        self.courses = []

    def assign_course(self, course):
        self.courses.append(course)
        print(f"{self.name} assigned to {course.name}.")

class Course:
    def __init__(self, name, code):
        self.name = name
        self.code = code
        self.students = []
        self.teacher = None

    def add_student(self, student):
        self.students.append(student)
        student.enroll(self)

    def set_teacher(self, teacher):
        self.teacher = teacher
        teacher.assign_course(self)

class University:
    def __init__(self):
        self.students = []
        self.teachers = []
        self.courses = []

    def register_student(self, student):
        self.students.append(student)
```

```
    def register_teacher(self, teacher):
        self.teachers.append(teacher)

    def add_course(self, course):
        self.courses.append(course)

    def assign_teacher_to_course(self, teacher_name, course_code):
        teacher = next((t for t in self.teachers if t.name == teacher_name), None)
        course = next((c for c in self.courses if c.code == course_code), None)

        if teacher and course:
            course.set_teacher(teacher)
        else:
            print("Teacher or Course not found.")

    def enroll_student_in_course(self, student_id, course_code):
        student = next((s for s in self.students if s.student_id == student_id), None)
        course = next((c for c in self.courses if c.code == course_code), None)

        if student and course:
            course.add_student(student)
        else:
            print("Student or Course not found.")

# Пример использования
university = University()

student1 = Student("Alice", 20, 1)
student2 = Student("Bob", 22, 2)
teacher1 = Teacher("Dr. Smith", 50, "Mathematics")
teacher2 = Teacher("Dr. Jones", 45, "Physics")

course1 = Course("Calculus", "MATH101")
course2 = Course("Physics", "PHYS101")

university.register_student(student1)
university.register_student(student2)
university.register_teacher(teacher1)
university.register_teacher(teacher2)
university.add_course(course1)
university.add_course(course2)

university.assign_teacher_to_course("Dr. Smith", "MATH101")
university.enroll_student_in_course(1, "MATH101")
university.enroll_student_in_course(2, "PHYS101")
```

**Описание решения**:

1. **Класс** `Person`: Базовый класс для студентов и преподавателей, содержит общие свойства.
2. **Класс** `Student`: Наследует от `Person`, добавляет идентификатор студента и список курсов.
3. **Класс** `Teacher`: Наследует от `Person`, добавляет специальность и список курсов.
4. **Класс** `Course`: Представляет курс, содержит название, код, список студентов и ссылку на преподавателя.
5. **Класс** `University`: Управляет списками студентов, преподавателей и курсов, реализует методы для регистрации, назначения и записи.