import random

from abc import ABC, abstractmethod

# Abstract Base Class for Person

class Person(ABC):

@abstractmethod

def get\_details(self):

pass

# Student class inheriting from Person

class Student(Person):

def \_\_init\_\_(self, student\_id, name, age):

self.student\_id = student\_id

self.name = name

self.age = age

self.courses = []

def get\_details(self):

return f"Student ID: {self.student\_id}, Name: {self.name}, Age: {self.age}, Courses: {', '.join(self.courses)}"

# Lecturer class inheriting from Person

class Lecturer(Person):

def \_\_init\_\_(self, lecturer\_id, name):

self.lecturer\_id = lecturer\_id

self.name = name

self.courses = []

def get\_details(self):

return f"Lecturer ID: {self.lecturer\_id}, Name: {self.name}, Courses: {', '.join(self.courses)}"

# Course class to hold course information

class Course:

def \_\_init\_\_(self, course\_id, course\_name):

self.course\_id = course\_id

self.course\_name = course\_name

# In-memory storage

students = []

lecturers = []

courses = []

# Function to add a student

def add\_student(name, age):

student\_id = random.randint(1000, 9999) # Random Student ID

student = Student(student\_id, name, age)

students.append(student)

print(f"Student {name} added successfully with ID: {student\_id}!")

# Function to add a lecturer

def add\_lecturer(name):

lecturer\_id = random.randint(1000, 9999) # Random Lecturer ID

lecturer = Lecturer(lecturer\_id, name)

lecturers.append(lecturer)

print(f"Lecturer {name} added successfully with ID: {lecturer\_id}!")

# Function to add a course

def add\_course(course\_name):

course\_id = random.randint(1000, 9999) # Random Course ID

course = Course(course\_id, course\_name)

courses.append(course)

print(f"Course {course\_name} added successfully with ID: {course\_id}!")

# Function to assign course to a student

def assign\_course\_to\_student(student\_id, course\_id):

student = next((s for s in students if s.student\_id == student\_id), None)

course = next((c for c in courses if c.course\_id == course\_id), None)

if student and course:

student.courses.append(course.course\_name)

print(f"Course {course.course\_name} assigned to student {student.name}.")

else:

print("Student or Course not found.")

# Function to assign course to a lecturer

def assign\_course\_to\_lecturer(lecturer\_id, course\_id):

lecturer = next((l for l in lecturers if l.lecturer\_id == lecturer\_id), None)

course = next((c for c in courses if c.course\_id == course\_id), None)

if lecturer and course:

lecturer.courses.append(course.course\_name)

print(f"Course {course.course\_name} assigned to lecturer {lecturer.name}.")

else:

print("Lecturer or Course not found.")

# Function to edit student name

def edit\_student\_name(student\_id, new\_name):

student = next((s for s in students if s.student\_id == student\_id), None)

if student:

student.name = new\_name

print(f"Student ID: {student\_id} name changed to {new\_name}.")

else:

print("Student not found.")

# Function to edit course name

def edit\_course\_name(course\_id, new\_name):

course = next((c for c in courses if c.course\_id == course\_id), None)

if course:

course.course\_name = new\_name

print(f"Course ID: {course\_id} name changed to {new\_name}.")

else:

print("Course not found.")

# Function to delete student

def delete\_student(student\_id):

global students

students = [s for s in students if s.student\_id != student\_id]

print(f"Student ID: {student\_id} deleted.")

# Function to delete course

def delete\_course(course\_id):

global courses

courses = [c for c in courses if c.course\_id != course\_id]

print(f"Course ID: {course\_id} deleted.")

# Function to view all students

def view\_all\_students():

if students:

print("\nList of All Students:")

for student in students:

print(student.get\_details())

else:

print("No students available.")

# Function to view all courses

def view\_all\_courses():

if courses:

print("\nList of All Courses:")

for course in courses:

print(f"ID: {course.course\_id}, Name: {course.course\_name}")

else:

print("No courses available.")

# Main menu loop

def main():

while True:

print("\nStudent Academic Management System")

print("1. Add Student")

print("2. Add Lecturer")

print("3. Add Course")

print("4. Assign Course to Student")

print("5. Assign Course to Lecturer")

print("6. Edit Student Name")

print("7. Edit Course Name")

print("8. Delete Student")

print("9. Delete Course")

print("10. View All Students")

print("11. View All Courses")

print("12. Exit")

choice = input("Enter your choice: ")

if choice == '1':

name = input("Enter Student Name: ")

age = int(input("Enter Student Age: "))

add\_student(name, age)

elif choice == '2':

name = input("Enter Lecturer Name: ")

add\_lecturer(name)

elif choice == '3':

course\_name = input("Enter Course Name: ")

add\_course(course\_name)

elif choice == '4':

student\_id = int(input("Enter Student ID: "))

course\_id = int(input("Enter Course ID: "))

assign\_course\_to\_student(student\_id, course\_id)

elif choice == '5':

lecturer\_id = int(input("Enter Lecturer ID: "))

course\_id = int(input("Enter Course ID: "))

assign\_course\_to\_lecturer(lecturer\_id, course\_id)

elif choice == '6':

student\_id = int(input("Enter Student ID: "))

new\_name = input("Enter New Student Name: ")

edit\_student\_name(student\_id, new\_name)

elif choice == '7':

course\_id = int(input("Enter Course ID: "))

new\_name = input("Enter New Course Name: ")

edit\_course\_name(course\_id, new\_name)

elif choice == '8':

student\_id = int(input("Enter Student ID: "))

delete\_student(student\_id)

elif choice == '9':

course\_id = int(input("Enter Course ID: "))

delete\_course(course\_id)

elif choice == '10':

view\_all\_students()

elif choice == '11':

view\_all\_courses()

elif choice == '12':

print("Exiting the system.")

break

else:

print("Invalid choice. Please select a valid option.")

if \_\_name\_\_ == "\_\_main\_\_":

main()