University Management System

By: Abubakar Idrees (02066)

Introduction

This guide provides step-by-step instructions to set up and use the University Management System, along with a brief explanation of each function.

This is Github link for full code: https://github.com/Abubakar-idrees-01/university_management_system

1. Project Setup

Prerequisites:

- Python (>=3.8)
- SQLite3 (built into Python)

Installation Steps:

1. Clone or download the project files.
2. Ensure all necessary files are present:
    - university_management_system.py
    - student.py
    - instructor.py
    - course.py
    - person.py
3. Open a terminal and navigate to the project folder.
4. Run the command:
5. python university_management_system.py

---

2. Code Explanation

person.py (Base Class)

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

    def get_details(self):
        return f"Name: {self.name}, Age: {self.age}, City: {self.city}"
```

Explanation:

- Defines a Person class as a base class.
- Stores name, age, and city.
- get_details() returns formatted details.

---

student.py (Student Class)

```
from person import Person


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

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

    def drop_course(self, course):
        if course in self.courses:
```

```
        self.courses.remove(course)


    def get_courses(self):
        return self.courses
```

Explanation:

- Inherits from Person.
- Manages enrolled courses using enroll_course(), drop_course(), and get_courses().

---

instructor.py (Instructor Class)

```
from person import Person


class Instructor(Person):
    def __init__(self, name, age, city):
        super().__init__(name, age, city)
        self.courses_taught = []


    def assign_course(self, course):
        self.courses_taught.append(course)


    def remove_course(self, course):
        if course in self.courses_taught:
            self.courses_taught.remove(course)


    def get_courses_taught(self):
        return self.courses_taught
```

Explanation:

- Inherits from Person.
- Manages assigned courses.

---

course.py (Course Class)

```
class Course:
    def __init__(self, course_name, course_code):
        self.course_name = course_name
        self.course_code = course_code

    def get_course_info(self):
        return f"{self.course_name} ({self.course_code})"
```

Explanation:

- Defines a Course class with course_name and course_code.
- get_course_info() returns formatted course details.

---

university_management_system.py (Main System)

```
from student import Student
from instructor import Instructor
from course import Course

students = []
instructors = []
courses = []
```

Explanation:

- Imports Student, Instructor, and Course.
- Initializes empty lists to store data.

```
def add_student():
    name = input("Enter student name: ")
    age = int(input("Enter student age: "))
    city = input("Enter student city: ")
    students.append(Student(name, age, city))
```

Explanation:

- Takes user input and adds a Student object to the list.

```
def add_instructor():
    name = input("Enter instructor name: ")
    age = int(input("Enter instructor age: "))
    city = input("Enter instructor city: ")
    instructors.append(Instructor(name, age, city))
```

Explanation:

- Takes user input and adds an Instructor object to the list.

```
def add_course():
    name = input("Enter course name: ")
    code = input("Enter course code: ")
    courses.append(Course(name, code))
```

Explanation:

- Takes user input and adds a Course object to the list.

```
def display_students():
    for student in students:
        print(student.get_details())
```

Explanation:

- Loops through the list of students and prints their details.

```
def main():
  while True:
    print("\n1. Add Student\n2. Add Instructor\n3. Add Course\n4. Display Students\n5. Exit")
    choice = input("Enter choice: ")

    if choice == "1":
      add_student()
    elif choice == "2":
      add_instructor()
    elif choice == "3":
      add_course()
    elif choice == "4":
      display_students()
    elif choice == "5":
      break
```

Explanation:

- Implements a menu-based system.
- Calls functions based on user choice.

```
if __name__ == "__main__":
  main()
```

Explanation:

- Ensures the script runs only when executed directly.

---

3. Running the System

1. Start the system using python university_management_system.py.
2. Select your role (Admin, Instructor, Student).

3. Follow on-screen prompts to navigate features.

4. Screenshots & Example Workflows

(Screenshots to be added based on execution results)

Note: Ensure that SQLite database modifications persist between sessions by keeping university_database.db in the project directory.