Doctor / Patients Scheduler Project

William Chang
Ventakatanagasatyakiran Pasupuleti
Philip Raymond

Project Details

Doctor / Patients Scheduler:

- Python program built around a doctor / patients scheduling program. We will expect around 16 patients during a usual 8 hour work day.
- There will be multiple classes from patient, doctor, appointment, and login classes to properly structure our variables.

Strategy and Approach

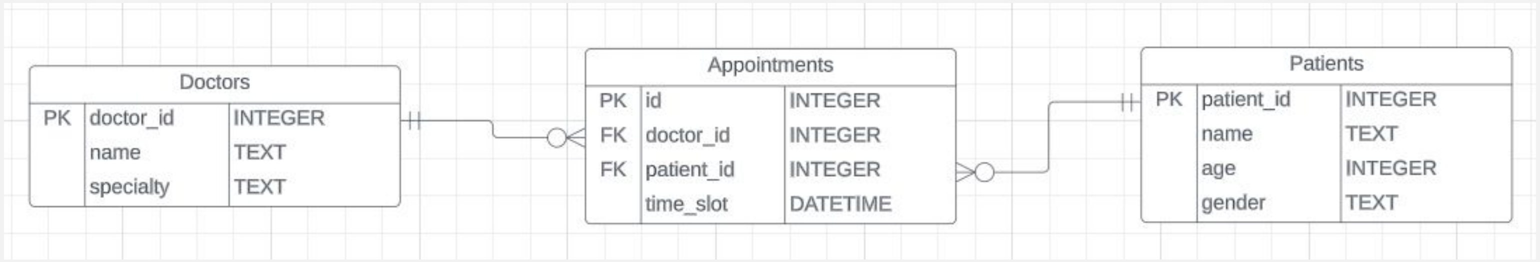
4-step strategy and approach

- Building databases
- Utilize CRUD method in code
- Visualize UI in PyCharm
- Error Handling and User Exceptions

Ethical and Critical Thinking

- Proper CRUD utilization
- Readability and Accessibility
- Error Handling

Entity Relationship Diagram



Our Code

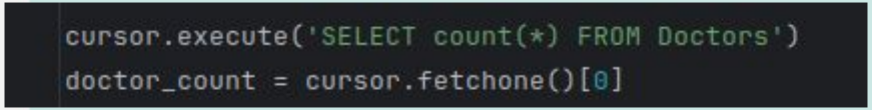
Switching screens! Please hold :)

SQL & Database - CRUD

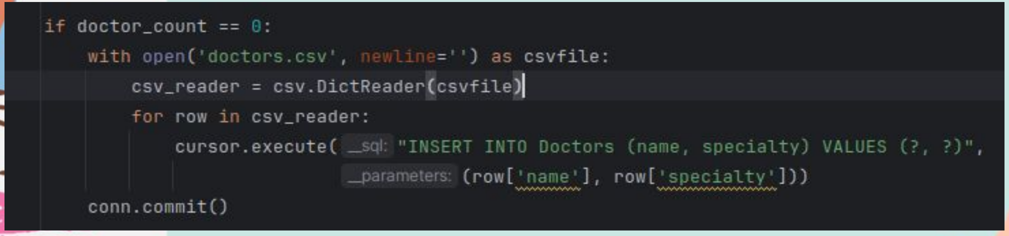
- Create:
  - Add patient
  - Add appointment
- Read
  - View patients with password
  - View appointments
- Update
  - Update appointment
- Delete
  - Delete appointment

File IO

- Create tables (if they don't exist)
- Get total record count
- If count = 0 then populate table



```
cursor.execute('SELECT count(*) FROM Doctors')
doctor_count = cursor.fetchone()[0]
```

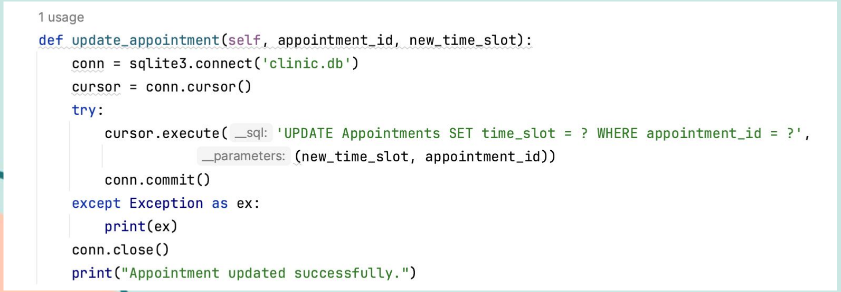


```
if doctor_count == 0:
    with open('doctors.csv', newline='') as csvfile:
        csv_reader = csv.DictReader(csvfile)
        for row in csv_reader:
            cursor.execute( __sql: "INSERT INTO Doctors (name, specialty) VALUES (?, ?)",
                            __parameters: (row['name'], row['specialty']))
    conn.commit()
```

Error Handling

Try & Except

- In our functions throughout our code we have try and exception handling to test code and handle user errors

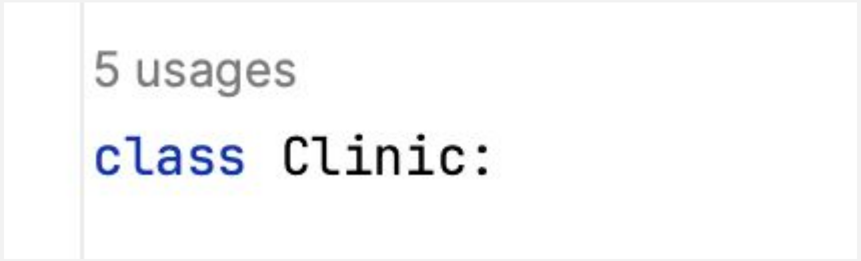


```
1 usage
def update_appointment(self, appointment_id, new_time_slot):
    conn = sqlite3.connect('clinic.db')
    cursor = conn.cursor()
    try:
        cursor.execute( __sql: 'UPDATE Appointments SET time_slot = ? WHERE appointment_id = ?',
                        __parameters: (new_time_slot, appointment_id))
        conn.commit()
    except Exception as ex:
        print(ex)
    conn.close()
    print("Appointment updated successfully.")
```

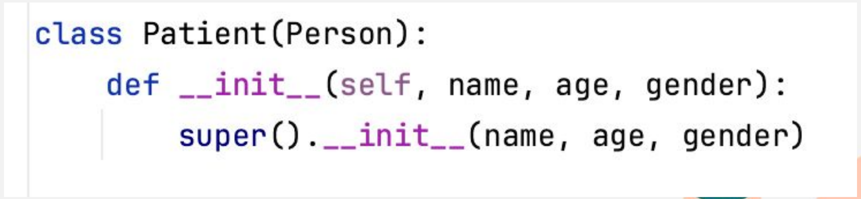
Object Oriented Programming

2 Different classes:

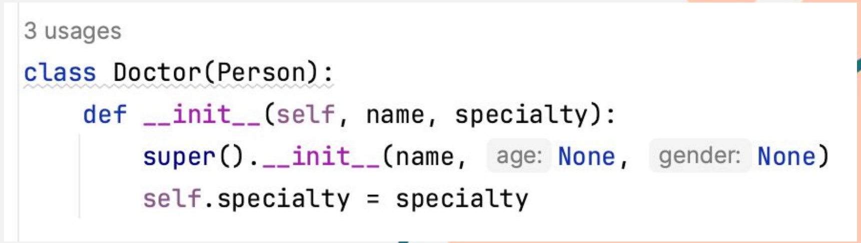
- Person class
  - Inheritance to Patient and Doctor
- Clinic class
  - Appointment functions



```
5 usages
class Clinic:
```

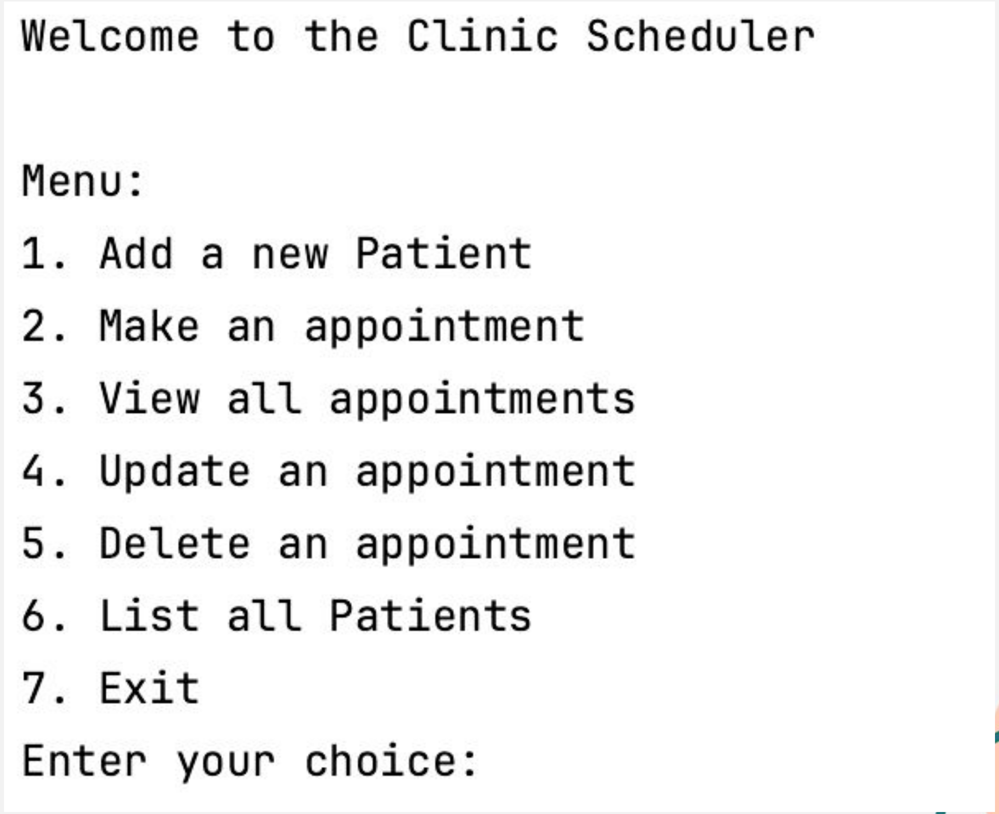


```
class Patient(Person):
    def __init__(self, name, age, gender):
        super().__init__(name, age, gender)
```



```
3 usages
class Doctor(Person):
    def __init__(self, name, specialty):
        super().__init__(name, age: None, gender: None)
        self.specialty = specialty
```

UI



```
Welcome to the Clinic Scheduler


Menu:
1. Add a new Patient
2. Make an appointment
3. View all appointments
4. Update an appointment
5. Delete an appointment
6. List all Patients
7. Exit
Enter your choice:
```

Conclusion

- Simple doctor / patients appointment scheduler built within PyCharm and SQLite
    - Class Structure:
        - Person - Patient & Doctor, Clinic - Appointments
    - CRUD method
        - Ensured 30 minute intervals for appointments
    - Organized our UI for PyCharm
    - Established error handling and user exceptions through our functions