Python Code of Theatre Event Calendar

```
from datetime import datetime


class Event:
    """Represents a theater event with ID, name, start time, end time, and (optional) description."""

    def __init__(self, event_id, name, start_time, end_time, description=""):
        self.event_id = event_id
        self.name = name
        self.start_time = datetime.strptime(start_time, '%Y-%m-%d %H:%M')
        self.end_time = datetime.strptime(end_time, '%Y-%m-%d %H:%M')
        self.description = description

    def __repr__(self):
        return f"Event(ID: {self.event_id}, Name: '{self.name}', Start: {self.start_time}, End: {self.end_time}, Description: '{self.description}')"


class EventCalendar:
    """Manages a collection of theater events with methods for adding, getting, updating, deleting, and listing events."""

    def __init__(self, calendar_name):
        self.calendar_name = calendar_name
        self.events = {}

    def add_event(self, event):
        """Adds an event to the calendar, checking for duplicate IDs."""
        if event.event_id in self.events:
            raise ValueError("Event ID already exists")
        self.events[event.event_id] = event
```

```
    def get_event(self, event_id):
        """Returns an event by its ID, or None if not found."""
        return self.events.get(event_id)

    def update_event(self, event_id, **kwargs):
        """Updates an event's details based on provided arguments (name, start_time, end_time, description)."""
        if event_id not in self.events:
            raise ValueError("Event not found")

        event = self.events[event_id]
        if 'name' in kwargs:
            event.name = kwargs['name']
        if 'start_time' in kwargs:
            event.start_time = datetime.strptime(kwargs['start_time'], '%Y-%m-%d %H:%M')
        if 'end_time' in kwargs:
            event.end_time = datetime.strptime(kwargs['end_time'], '%Y-%m-%d %H:%M')
        if 'description' in kwargs:
            event.description = kwargs['description']

    def delete_event(self, event_id):
        """Deletes an event from the calendar, raising an error if not found."""
        if event_id in self.events:
            del self.events[event_id]
        else:
            raise ValueError("Event not found")

    def list_events(self):
        """Returns a list of all events in the calendar."""
        return list(self.events.values())
```

```
    def search_events(self, criteria):
        """Searches events based on criteria (name substring or time range)."""
        matching_events = []
        for event in self.events.values():
            if criteria.lower() in event.name.lower():  # Search by name substring (case-insensitive)
                matching_events.append(event)
            elif (datetime.now() >= event.start_time) and (datetime.now() <= event.end_time):
                # Search for events happening right now
                matching_events.append(event)
        return matching_events

    def __repr__(self):
        return f"EventCalendar(Name: '{self.calendar_name}', Events: {len(self.events)})"

def main():
    """Example usage of the EventCalendar class."""
    theater_calendar = EventCalendar("My Theater Calendar")

    # Add some sample theater events
    event1 = Event(1, "Hamlet", "2024-10-12 19:00", "2024-10-12 21:00", "A classic Shakespearean tragedy.")
    event2 = Event(2, "The Lion King", "2024-10-20 14:00", "2024-10-20 16:00", "A beloved Disney musical.")
    event3 = Event()
```