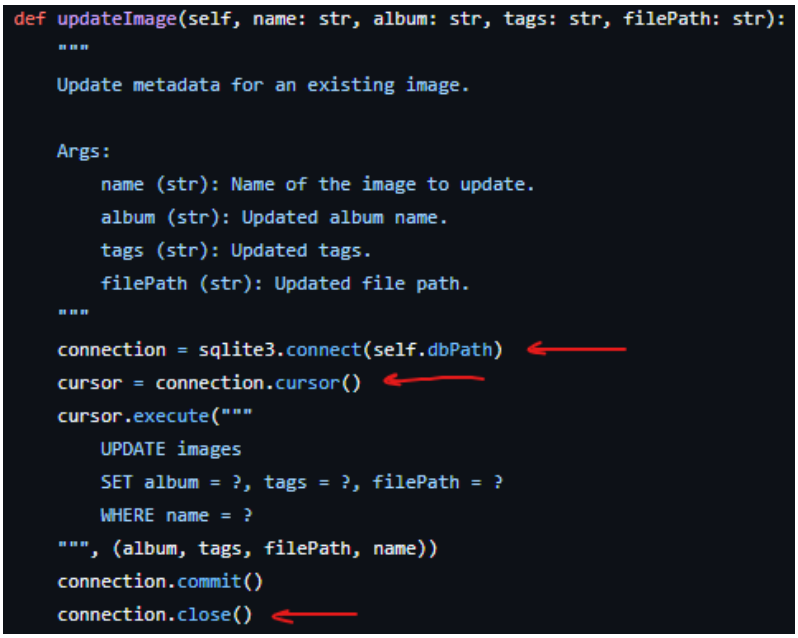
Code Review

Author - Oasis
Reviewers - Luka, Satwik
Scribe - Grace (Kanyanat)

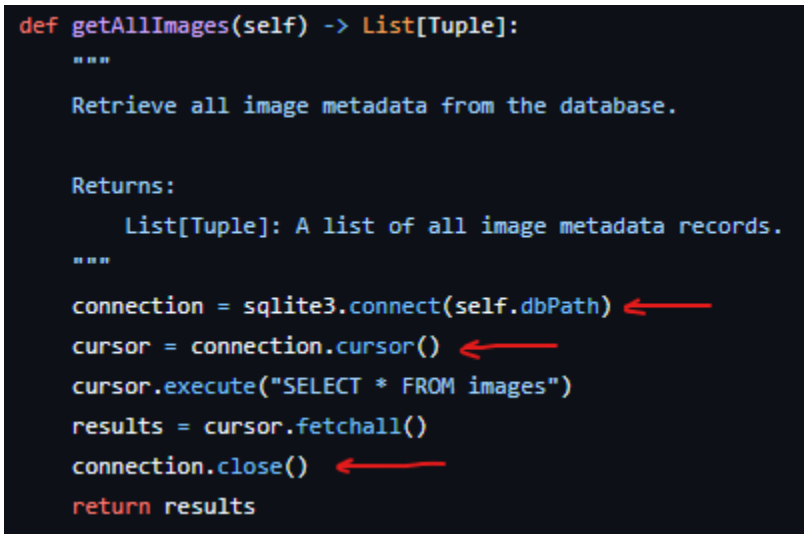
Issues:

- Make the connection central since connecting to the database every time will be time consuming. If an exception occurs between connect() and close(), the connection might remain open.



```
def updateImage(self, name: str, album: str, tags: str, filePath: str):
    """
    Update metadata for an existing image.

    Args:
        name (str): Name of the image to update.
        album (str): Updated album name.
        tags (str): Updated tags.
        filePath (str): Updated file path.
    """
    connection = sqlite3.connect(self.dbPath)
    cursor = connection.cursor()
    cursor.execute("""
        UPDATE images
        SET album = ?, tags = ?, filePath = ?
        WHERE name = ?
    """, (album, tags, filePath, name))
    connection.commit()
    connection.close()
```



```
def getAllImages(self) -> List[Tuple]:
    """
    Retrieve all image metadata from the database.

    Returns:
        List[Tuple]: A list of all image metadata records.
    """
    connection = sqlite3.connect(self.dbPath)
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM images")
    results = cursor.fetchall()
    connection.close()
    return results
```

- No error handling

```python
def getAllImages(self) -> List[Tuple]:
    """
    Retrieve all image metadata from the database.

    Returns:
        List[Tuple]: A list of all image metadata records.
    """
    connection = sqlite3.connect(self.dbPath)
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM images")
    results = cursor.fetchall()
    connection.close()
    return results
```

- No input validation

```python
def insertImage(self, name: str, album: str, tags: str, filePath: str):
    """
    Insert a new image record into the database.

    Args:
        name (str): Name of the image.
        album (str): Album the image belongs to.
        tags (str): Tags associated with the image.
        filePath (str): Path to the actual image file.
    """
    connection = sqlite3.connect(self.dbPath)
    cursor = connection.cursor()
    cursor.execute("""
        INSERT INTO images (name, album, tags, filePath)
        VALUES (?, ?, ?, ?)
    """, (name, album, tags, filePath))
    connection.commit()
    connection.close()
```

- The database could be designed better, works for small databases but doesn't really scale well. Maybe add indexing for albums?

```
def _createTable(self):
    """Create the images table if it does not already exist."""
    connection = sqlite3.connect(self.dbPath)
    cursor = connection.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS images (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT NOT NULL,
            album TEXT,
            tags TEXT,
            filePath TEXT NOT NULL
        )
    """)
    connection.commit()
    connection.close()
```

- Any schema change would require altering the _createTable method manually. Making a schema file might help.

```
def _createTable(self):
    """Create the images table if it does not already exist."""
    connection = sqlite3.connect(self.dbPath)
    cursor = connection.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS images (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT NOT NULL,
            album TEXT,
            tags TEXT,
            filePath TEXT NOT NULL
        )
    """)
    connection.commit()
    connection.close()
```

- The implementation for the SearchEngine and the Similarity Search should be wary of similar issues when dealing with Databases.

```python
def getImageByName(self, name: str) -> Optional[Tuple]:
    """
    Retrieve a single image record by name.

    Args:
        name (str): Name of the image to search for.

    Returns:
        Optional[Tuple]: The image record if found, else None.
    """
    connection = sqlite3.connect(self.dbPath)
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM images WHERE name = ?", (name,))
    result = cursor.fetchone()
    connection.close()
    return result
```

Strengths:

- Good documentation and follows the coding standards. Both module-level and class-level docstrings are descriptive and formatted properly. The function docstrings follow consistent conventions with clear explanations of parameters and return values.
- Database operations are neatly encapsulated inside the class. (Search operations will be implemented elsewhere)
- Code is very readable
- Type hint inclusion