Fehlerbehandlung:

Zwei arten von Strategien:

- LBYL (LookBeforeYouLeap)
    if else
- EAFP (Easier to ask Forgivnes than Permission)
    try catch

Zwei Arten von Fehlern:
    - neue Errors
        - Fehler gefunden, und neuen Fehler generieren
    - hoch"geblubberte" Errors
        - Fehler von einer aufgerufenen Funktion bekommen

Zwei Arten von Fehlerbehebarkeit:
    - behebar
    - nicht behebar

Vier Möglichkeite von Fehlern:

a) neuer Fehler und behebar
b) hochblubber-Fehler und behebar
c) neuer Fehler NICHT behebar
d) hochblubber-Fehler und NICHT behebar

zu a)

```
def add_song_to_database(song):
    # ...
    if song.year is None:
        song.year = 'Unknown'
    # ...
```

zu b)

```
def add_song_to_database(song):
    # ...
    try:
        artist = get_artist_from_database(song.artist)
    except NotFound:
        artist = add_artist_to_database(song.artist)
    # ...
```

zu c)

```
def add_song_to_database(song):
    # ...
    if song.name is None:
        raise ValueError('The song must have a name')
    # ...
```

zu d)

```
def new_song():
    song = get_song_from_user()
    add_song_to_database(song)
```

KEINE Fehlerbehandlung hier!! Wenn man einen Fehler nicht beheben kann, dann fägt man diesen auch nicht. prints sind nicht immer hilfreich z.B. Web-Apps. Man kann nur hoffen, dass beim Hochblubbern eine geeignete Stelle existiert, welcher den Fehler behandeln kann. Sonst:

```
import sys

def my_cli()
    # ...

if __name__ == '__main__':
    try:
        my_cli()
    except Exception as error:
        print(f"Unexpected error: {error}")
        sys.exit(1)
```

so weiss die startende Stelle z.B. Bash-Script, dass Applikation mit Fehler terminiert hat, aber nicht gecrasht.