Mini-project 2

General guide

Starting database in background

```
mongod --port 1111 --dbpath /Users/<USER>/Desktop/db_temp
```

Initializing database (phase 1)

```
# From root of project directory
cd Source
python load_json.py dblp-ref-10.json 1111
```

Running the program

```
# From root of project directory
cd Source
python main.py 1111
```

Software design

This program was designed to be easily adapted to different interfaces by abstracting all the application logic into it's own layer. Our interface makes calls to the application layer to get/set data, leaving the responsibility of displaying information and handling input to the interface layer.
This, as well, makes it easier to create unit tests since the interface can be ignored; tests are run on the data returned by the application layer.

Testing strategy

Ideally, given more time, we would have written unit tests for our entire application layer (data access layer); though for the most part we ended up having to manually run and verify our queries and their integrity.
For "search for articles" we tested whether the 'and' semantics were met by including more keywords and seeing if the total results stayed the same or went down. We made sure stopwords weren't included by searching with terms like "the".
For "search for authors" we tested that each following result was correctly sorted by checking through our interface.

Source code quality

Our source code contains docstrings on files and methods as an aide. We attempted to keep nesting to a minimum to avoid complexity. Only one point of return is used in most functions so as to not accidentally bypass code that might be executed at the end of a function (as well as avoiding complexity). Input/output operations are kept in batches for runtime performance (a benefit of abstracting all database operations to a separate layer).

Project breakdown

- Phase 1
  - group effort via pair programming
- Phase 2
  - Search for articles: Connor
  - Search for authors: Leon, Connor
  - List the venues: Brandon, Leon
  - Add an article: Brandon

For phase 2, the people responsible for a particular task were responsible for implementing both the interface and application layers.

Aggregate hours spent

- Connor: 15 hours
  - 10 on search for articles
  - 2 on refactoring
  - 1 on frame/general structure of application
  - 2 on various
- Leon: 12 hours
- Brandon: 13 hours
  - 3 on adding article
  - 10 on list venues

Coordination

We kept our code synchronized through GitHub. Significant problems that couldn't be immediately solved were documented in GitHub issues. Communication was facilitated through text and discord.