Here are some ideas I have for "mini projects." These should hopefully guide the learning process to some tangible results that will be applicable (either directly or indirectly through the knowledge gained) to the final project.

- **Python Documentation Searcher**

Task	Analyze Python Documentation
Purpose	Demonstrate understanding of basic indexing algorithms and proecedures
Specification	Index the python documentation (a cluster of html files) and enable a keyword search
Comments	Develop a simple data structure that will allow for reverse-indexing and quickly compute tf-idf of keywords. The html files may be preprocessed using pythons standard `http` library. The inteface could allow you to index some new files, whose results are stored (added to) a json. A query would take some keywords and compute tf-idf from this data structure.

- **Python Documentation Bot**

Task	Create a bot for Interactive IR (IIR)
Purpose	Demonstrate basic use of the MS Bot Framework
Specification	Create a bot which will enter a dialogue with the user and help them satisfy their information need.
Comments	It seems obvious to use the previous project or an existing search capability to iteratively respond to user queries and refinements, but it may be better to abstract this interaction (using dependency injection, for example), and focus on the interaction mechanisms. This would include maintaining conversation state (not bot state - use the bot as a reducer and keep it clean), and reaction from user input. The MS "Cards" infrastructure would likely be useful to help guide the user (if so desired). Otherwise, NLP will be required to interact with the user.

- **Web-Connected Documentation Bot**

Task	Enhance an IIR bot with network capabilities
Purpose	Demonstrate an ability to implement network enhanced behavior of an interactive bot and take advantage of *context*
Specification	Take an IIR bot and utilize results or tools obtained from the internet
Comments	The bot will never do what google does better than google. Don't try to be google! Use google (for example) to enhance the services being provided. Here is also a good opportunity to utilize *context,* i.e. make use of assumptions about the user that a general purpose search engine could not. An example would be to prioritize results from stack-overflow or add keywords to searches to google to help obtain better results.

- **NLP Bot**

Task	Implement NLP into a bot for some purpose
Purpose	Gain familiarity and utility with NLP for use with bots
Specification	Take a bot of choice and enhance with NLP capabilities
Comments	This could be a good opportunity to make the previous bots actually useful, in particular through keyword-expansion or query-categorization. An example of keyword-expansion would be adding or replacing search keywords with synonyms. This can be made powerful by making assumptions about the context in which it is being used (for a programmer "class" almost certainly doesn't refer to social-status or lectures somewhere). Query-categorization could be, for example, determining if the user would benefit most from API documentation, an answer on stack-overflow, a tutorial from some website, an nice example on github, et cetera.

- **Smart Bot**

Task	Incorporate Machine-Learning into some BOT
Purpose	Be a badass
Specification	Take a bot of choice and enhance with ML capabilities
Comments	The basic idea is to use techniques from ML to enhance the bot. This may be out of scope, but one idea would be to have a neural network for each user kept in state and have it learn what is most beneficial for the user in question. This may require feedback from the user, or some smart metrics. It could also be used to learn which bot-state-transition lead to the quickest end of conversation (assuming that's a good thing), for example. **I'm not 100% sure about this, requires more thought**