Here are 10 essay questions based on the provided context, designed to be of medium difficulty, along with model answers:

1. **Question:** Explain the process of generating random text using bigrams, as demonstrated by the `generate_model()` function. What are the limitations of this approach, and how could the text generation be improved?

Answer: The `generate_model()` function creates random text by treating each word as a condition and creating a frequency distribution over the following words. It starts with an initial context word, prints it, and then selects the most likely following word based on the conditional frequency distribution. This new word becomes the next context. A major limitation is that this method often gets stuck in loops, repeating the same sequence of words. Improvements could involve randomly choosing the next word from available words or using higher-order n-grams (trigrams, etc.) to capture longer-range dependencies.

2. **Question:** Describe the structure and purpose of a conditional frequency distribution (CFD) in NLTK. Provide examples of how CFDs can be used in NLP tasks such as text generation or lexical analysis.

Answer: A conditional frequency distribution (CFD) is a data structure that collects frequency distributions, each one associated with a different condition. In NLTK, it is used to record the frequency of samples for each condition. For example, in text generation, a CFD can store the frequency of words that follow a given word (bigrams). In lexical analysis, it can store the frequency of part-of-speech tags for each word or the frequency of word lengths in different genres of text.

3. **Question:** Explain the significance of using a text editor for writing Python programs, as

opposed to only using the interactive interpreter. What are the advantages of saving code in a file with a `.py` extension?

Answer: Using a text editor allows for composing multi-line programs and saving them for later use, which is more efficient than retyping code in the interactive interpreter. Saving code in a `.py` file allows for easy reuse, modification, and organization of code into modules. It also enables the creation of more complex and maintainable programs.

4. **Question:** Define what a function is in Python and explain its importance in reusing code. Describe the concepts of parameters, local variables, and return values in the context of a function.

Answer: A function is a named block of code that performs a specific task. It is essential for code reuse, allowing the same code to be executed multiple times with different inputs. Parameters are special variables used to pass inputs to the function. Local variables are variables defined inside the function and are not accessible outside it. A return value is the value produced as output by the function.

5. **Question:** Explain the concept of a Python module and how it facilitates code organization. How can you access functions and variables defined in a module within another program? Provide an example.

Answer: A Python module is a file containing a collection of variable and function definitions. It allows for organizing code into logical units, promoting reusability and maintainability. To access functions and variables from a module, you can use the `import` statement (e.g., `from textproc import plural`). This makes the `plural` function available for use in your program.

6. **Question:** Define the terms "lexicon" and "lexical resource." How do lexical resources enhance NLP tasks? Give examples of different types of lexical resources and their specific uses.

Answer: A lexicon, or lexical resource, is a collection of words and/or phrases along with associated information, such as part-of-speech tags and sense definitions. Lexical resources enhance NLP tasks by providing structured information about words, aiding in tasks like spell-checking, text filtering, and language translation. Examples include wordlists (for spell-checking), stopword lists (for text filtering), and pronouncing dictionaries (for text-to-speech).

7. **Question:** Describe the purpose and usage of stop words in NLP. How does filtering out stop words contribute to text analysis and processing? Provide an example of how to calculate the content fraction of a text using stop words.

Answer: Stop words are high-frequency words (e.g., "the," "to," "and") that are often filtered out of a document before further processing because they have little lexical content and do not significantly distinguish one text from another. Filtering out stop words can improve text analysis by focusing on more meaningful words. The content fraction of a text is calculated by dividing the number of non-stop words by the total number of words in the text.

8. **Question:** Explain the concept of a lexical entry, headword (lemma), and homonyms. Provide examples to illustrate the differences between these terms.

Answer: A lexical entry consists of a headword (or lemma) along with additional information, such as the part-of-speech and sense definition. The headword (lemma) is the canonical form of a word. Homonyms are two distinct words having the same spelling but different meanings (e.g., bank as a financial institution and bank as the side of a river).

9. **Question:** Discuss the importance of choosing descriptive but short names for files. Explain the conventions for naming files in Python, including the use of lowercase letters, underscores, and the `.py` extension.

Answer: Descriptive but short file names help in quickly identifying the purpose of a file. Python file naming conventions include using all lowercase letters, separating words with underscores, and using the `.py` extension (e.g., `monty_python.py`). This ensures consistency and readability, making it easier to manage and share code.

10. **Question:** Explain what is meant by testing code interactively. How does inspecting data and invoking functions in the interactive interpreter aid in the development of larger programs?

Answer: Testing code interactively involves using the interpreter to test ideas, revise lines of code, and inspect data. This approach allows developers to quickly check the behavior of individual components and functions, making it easier to identify and fix errors before integrating the code into a larger program. It is particularly useful for understanding how functions work and for experimenting with different approaches. ?