Viva

1. What is Prolog, and how is it different from other programming languages like Python or Java?

Answer:
Prolog is a declarative programming language primarily used for logic programming. Unlike imperative languages like Python or Java, where the programmer specifies *how* to solve a problem, Prolog allows the programmer to specify *what* the problem is in terms of facts, rules, and queries. The Prolog engine uses logical inference to find solutions, utilizing backtracking to explore possible answers.

---

2. Explain the concept of facts, rules, and queries in Prolog.

Answer:

- **Facts:** These are basic assertions about the world, written as `fact_name(argument).`. For example, `parent(john, mary).` asserts that John is a parent of Mary.
- **Rules:** These define relationships between facts and can be used to infer new facts. A rule is written like `head :- body.` For example, `grandparent(X, Y) :- parent(X, Z), parent(Z, Y).` states that X is a grandparent of Y if X is a parent of Z, and Z is a parent of Y.
- **Queries:** These are questions asked to the Prolog system to check if something is true, written as `?- query.` For example, `?- parent(john, mary).` checks if John is a parent of Mary.

---

3. What is backtracking in Prolog, and how does it work?

Answer:
Backtracking is the process by which Prolog searches for solutions to a query by trying different possibilities when a choice fails. When Prolog encounters a failure (i.e., a rule or fact doesn't match), it backtracks to the last successful choice point and tries a different option. This process continues until a solution is found or all possibilities are exhausted.

---

4. How does Prolog utilize a knowledge base, and what is its primary advantage in logic programming?

Answer:
In Prolog, the knowledge base consists of facts and rules that represent information about the world. The Prolog engine uses this knowledge to infer new facts and answer queries. The main advantage of Prolog is its ability to handle complex logical relationships and search spaces efficiently through its built-in backtracking mechanism. This makes Prolog suitable for problems like artificial intelligence, theorem proving, and expert systems.

Viva

Simple Facts in Prolog

1. What are facts in Prolog, and how are they represented?

Answer:
Facts are basic assertions about the world that Prolog treats as true. They are written as `predicate(argument).` For example, `likes(john, pizza).` is a fact that states John likes pizza. Facts are used to represent known information or data.

---

2. How do you assert and retract facts in Prolog?

Answer:

- **assert/1**: Adds a new fact to the knowledge base. For example, `assert(likes(john, pizza)).` adds the fact that John likes pizza.
- **retract/1**: Removes a fact from the knowledge base. For example, `retract(likes(john, pizza)).` removes the fact that John likes pizza.

---

3. Can you provide an example of a simple fact in Prolog and explain its usage?

Answer:
Example: `mother(mary, john).`
This fact asserts that Mary is the mother of John. You can use this fact in queries, like `?- mother(mary, john).`, to check if Mary is indeed John’s mother. Prolog will answer "true" if the fact exists in the knowledge base.

---

4. How does Prolog use facts to derive new information or answer queries?

Answer:
Prolog uses facts in conjunction with rules to derive new information. When a query is made, Prolog tries to match the query with facts in the knowledge base. If a matching fact exists, it returns "true." If no direct match is found, Prolog attempts to use rules to infer new facts and check if they match the query.

Viva

Writer Predicates and Create Functions

1. What is a predicate in Prolog, and how does it differ from a function in other programming languages?

Answer:
A predicate in Prolog is a relation or logical condition that can be true or false. It is used to express facts and rules. Unlike functions in imperative languages, predicates do not return values but instead express relationships or conditions that can be queried. For example, `likes(john, pizza)` is a predicate that checks if John likes pizza.

---

2. How would you write a predicate in Prolog to check if a number is even?

Answer:
A simple predicate to check if a number is even can be written as follows:

3. Can you explain the difference between a fact and a predicate in Prolog?

Answer:
A fact in Prolog is a statement that is unconditionally true, such as `likes(john, pizza).` A predicate, on the other hand, is a rule or condition that defines a relationship or logic to infer new facts, such as `father(X, Y) :- parent(X, Y), male(X).` where `father(X, Y)` can be derived using the condition provided by the rule.

4. How do you write a function to calculate the factorial of a number in Prolog?

Answer:
You can write a recursive predicate to calculate the factorial of a number in Prolog as follows:

This predicate states that the factorial of 0 is 1, and for any number greater than 0, it recursively calculates the factorial by multiplying the number by the factorial of the number minus 1.

Viva

1. What is BFS (Breadth-First Search), and how can it be implemented in Prolog?

Answer:
BFS is an algorithm used to traverse or search through a graph or tree. It explores all the neighbors of a node before moving on to the next level of nodes. In Prolog, BFS can be implemented using a queue-based approach where nodes are visited level by level. The basic idea is to visit all the nodes in the current level, then move on to the next level.

2. What are the major challenges in implementing BFS in Prolog?

Answer:
One of the major challenges is handling loops and ensuring that nodes are not revisited. Prolog doesn't have an inherent queue or built-in support for BFS, so the programmer needs to manage the queue explicitly and keep track of visited nodes to avoid infinite loops.

3. Can you explain how the `findall/3` predicate is used in the BFS implementation?

Answer:
The `findall/3` predicate in Prolog is used to collect all solutions to a goal into a list. In the BFS example, `findall(Neighbor, edge(Current, Neighbor), Neighbors)` finds all the neighbors of the current node (represented by `Current`). It gathers them into a list called `Neighbors`, which is then appended to the queue (rest of the nodes to be processed).

Viva Questions

Ques - 1
What is the Water Jug Problem?
Ans - 1
The Water Jug Problem is a classic problem in artificial intelligence where two jugs of different capacities are used to measure a required amount of water, following a set of rules.

Ques - 2
How do you approach the Water Jug Problem?
Ans - 2
The Water Jug Problem is approached using state-space search where each state represents the amount of water in both jugs. A solution is found by exploring the transitions between states using valid operations like filling, emptying, or transferring water.

Ques - 3
What are the basic operations involved in the Water Jug Problem?
Ans - 3
The basic operations include filling a jug, emptying a jug, or pouring water from one jug to another, which may either empty the first jug or fill the second one.

Ques - 4
What is the time complexity of the Water Jug Problem?
Ans - 4
The time complexity is typically O(n), where n is the number of states in the state-space. Each operation moves from one state to another, and in the worst case, the number of states is proportional to the jug capacities.

Viva Questions

Ques - 1
What data structure is used to represent the Tic-Tac-Toe board?
Ans - 1
A 2D list (or matrix) is commonly used to represent the Tic-Tac-Toe board, where each cell holds a player's mark (either 'X', 'O', or empty).

Ques - 2
How do you check if there is a winner in Tic-Tac-Toe?
Ans - 2
A winner is determined by checking all rows, columns, and diagonals for three consecutive marks of the same player ('X' or 'O').

Ques - 3
How would you handle invalid moves in the Tic-Tac-Toe game?
Ans - 3
Invalid moves can be handled by checking if the selected cell is already occupied or if the move is outside the bounds of the grid.

Ques - 4
How would you implement a computer player in Tic-Tac-Toe?
Ans - 4
A computer player can be implemented using a simple algorithm like random moves or an optimal strategy like the Minimax algorithm to select the best move.

Viva Questions

Ques - 1
What is the use of regular expressions in removing punctuation from a string?
Ans - 1
Regular expressions are used to match punctuation characters and replace them with an empty string, effectively removing them from the text.

Ques - 2
What libraries can be used to remove punctuation from a string in Python?
Ans - 2
You can use the `string` library to get a list of punctuation marks and the `re` library to use regular expressions for removing them.

Ques - 3
How do you handle whitespace characters when removing punctuation?
Ans - 3
Whitespace characters are retained while removing punctuation because they are not considered part of the defined punctuation characters.

Ques - 4
Why is string translation more efficient for removing punctuation in Python?
Ans - 4
String translation, using the `str.translate()` method, is more efficient because it directly maps each punctuation character to be removed, which avoids the overhead of regular expressions.

Viva Questions

Ques - 1
How do you split a sentence into words in Python?
Ans - 1
You can use the `split()` method to break a sentence into a list of words, which can then be sorted alphabetically.

Ques - 2
How do you sort a list of words alphabetically in Python?
Ans - 2
You can use the `sorted()` function to sort the list of words in alphabetical order.

Ques - 3
What happens if the sentence contains punctuation when sorting alphabetically?
Ans - 3
Punctuation can be removed before sorting to avoid interference with the sorting process, as it may affect the lexicographical order.

Ques - 4
Can you sort words in reverse alphabetical order?
Ans - 4
Yes, by passing the argument `reverse=True` to the `sorted()` function, you can sort the words in reverse alphabetical order.

Viva Questions

Ques - 1
What data structures are useful for implementing the Hangman game?
Ans - 1
A list can be used to track the guessed letters, and a string can represent the word to guess. A set can help manage the set of correct and incorrect guesses.

Ques - 2
How do you ensure that the user cannot guess the same letter twice in Hangman?
Ans - 2
By keeping track of previously guessed letters in a set, you can check if the letter has been guessed before and prevent repeated guesses.

Ques - 3
How do you handle incorrect guesses in Hangman?
Ans - 3
For each incorrect guess, a part of the hangman figure is drawn, and the user loses attempts. If the number of attempts reaches zero, the game ends.

Ques - 4
How can you improve the Hangman game with more features?
Ans - 4
Features like hints, a timer, or a scoring system can make the game more interesting. You can also add multiple difficulty levels by increasing the word length.

Viva Questions

Ques - 1
How does PROLOG handle state transitions in the Hangman game?
Ans - 1
PROLOG uses facts and rules to represent the game's state and logic. The game state transitions are handled by updating the facts about guesses and remaining attempts.

Ques - 2
Can you define the rules of the Hangman game in PROLOG?
Ans - 2
Yes, in PROLOG, you can define rules for checking guesses, updating the game state, and determining whether the player wins or loses.

Ques - 3
How do you check for winning conditions in Hangman using PROLOG?
Ans - 3
You check if all letters in the word are correctly guessed, ensuring that no remaining blanks are present in the word.

Ques - 4
What are the advantages of using PROLOG for implementing Hangman?
Ans - 4
PROLOG's declarative nature makes it easy to define and reason about game rules. It excels in logic-based problems like Hangman, where state transitions are simple to express.

Viva Questions

Ques - 1
What are stop words in text processing?
Ans - 1
Stop words are common words like "the," "and," and "in," that are usually removed from text because they don't add significant meaning in natural language processing.

Ques - 2
How does NLTK help in removing stop words?
Ans - 2
NLTK provides a built-in list of stop words in multiple languages that can be used to filter out stop words from a text passage.

Ques - 3
How do you read and process a text file in Python?
Ans - 3
You can use Python's `open()` function to read the file, and then use the `read()` or `readlines()` method to process the content.

Ques - 4
What is the impact of removing stop words on text analysis?
Ans - 4
Removing stop words helps in reducing the noise in text analysis, leading to more accurate results in tasks like text classification or sentiment analysis.

Viva Questions

Ques - 1
What is stemming in natural language processing?
Ans - 1
Stemming is the process of reducing words to their base or root form, such as changing "running" to "run" or "better" to "good."

Ques - 2
Which algorithm is used in NLTK for stemming?
Ans - 2
NLTK uses algorithms like PorterStemmer and LancasterStemmer for performing stemming on words.

Ques - 3
Why is stemming important in text analysis?
Ans - 3
Stemming reduces words to their root form, helping in better matching and comparison of terms in tasks like information retrieval or sentiment analysis.

Ques - 4
What is the difference between stemming and lemmatization?
Ans - 4
Stemming removes word affixes to return a root word, while lemmatization converts a word to its lemma using a dictionary, considering the word’s meaning.

Viva Questions

Ques - 1
What is POS tagging in natural language processing?
Ans - 1
POS tagging involves labeling each word in a sentence with its corresponding part of speech, such as noun, verb, adjective, etc.

Ques - 2
How does NLTK perform POS tagging?
Ans - 2
NLTK uses predefined models like the Penn Treebank tagset, along with the `pos_tag()` function, to tag words in a sentence.

Ques - 3
Why is POS tagging useful in text processing?
Ans - 3
POS tagging helps in understanding the grammatical structure of a sentence, which is important for tasks like parsing, machine translation, and named entity recognition.

Ques - 4
Can POS tagging handle ambiguous words?
Ans - 4
Yes, POS tagging uses context to resolve ambiguities by determining the most likely part of speech based on surrounding words.

Viva Questions

Ques - 1
What is lemmatization in natural language processing?
Ans - 1
Lemmatization is the process of converting a word to its base or dictionary form, such as changing "better" to "good" and "running" to "run."

Ques - 2
How does NLTK perform lemmatization?
Ans - 2
NLTK uses the WordNetLemmatizer to perform lemmatization, which requires a word’s part of speech to accurately return its lemma.

Ques - 3
What is the difference between stemming and lemmatization?
Ans - 3
Stemming cuts off affixes to return a base form, while lemmatization uses a dictionary to return the correct lemma based on context.

Ques - 4
Why is lemmatization more accurate than stemming?
Ans - 4
Lemmatization uses a vocabulary and considers the context, providing more meaningful and grammatically correct words compared to stemming, which may produce non-words.

Viva Questions

Ques - 1
What is text classification in natural language processing?
Ans - 1
Text classification is the process of assigning predefined categories or labels to a text based on its content, such as classifying emails as spam or not spam.

Ques - 2
How can NLTK help with text classification?
Ans - 2
NLTK provides tools to preprocess text, extract features, and train machine learning models like Naive Bayes or SVM for text classification.

Ques - 3
What features are commonly used in text classification?
Ans - 3
Common features include word frequency, n-grams, and part-of-speech tags, which help the model distinguish between different text categories.

Ques - 4
Why is data preprocessing important in text classification?
Ans - 4
Preprocessing steps like tokenization, stop-word removal, and stemming or lemmatization are important to clean the data and ensure better model accuracy.