This is a Python project that utilizes a backward chaining inference engine for First-Order Logic (FOL). It contains a fully grounded knowledge base consisting of facts and rules, and therefore all predicates must be employed with certain constants (i.e., Parent(John, Mary) rather than variables). The system is designed to determine whether a query can be logically inferred from the provided knowledge using a goal-directed reasoning approach. Backward chaining starts with a question and then goes backward, searching for rules that close the question and recursively attempting to prove each premise of the rule. If all subgoals can be proven—directly from facts or indirectly by the use of other rules—the starting goal is concluded to be true. This recursive approach underlies multi-step reasoning by chains of logical linkages. In our code, a class KnowledgeBase is created to hold facts and rules, and a function backward_chain is employed for query evaluation. For preventing infinite loops or repeated checks, the algorithm employs a visited set to mark already tried goals while using recursion. A simple example based on family relationships was used in an attempt to test the system, e.g., facts like Parent(John, Mary) and rules like Grandparent(X, Z) $\leftarrow$ Parent(X, Y) $\land$ Parent(Y, Z), though these were managed manually. These queries tested the system's ability to deduce new knowledge by logical inference. Although this engine does not support variables, pattern matching, or automatic rule instantiation, it does well to demonstrate the overall mechanics of backward chaining logic in FOL. It can be a stepping stone towards more advanced systems that include variable unification, full parser for FOL syntax, and file input/output processing. To conclude, this implementation satisfies the basic objectives of modeling reasoning through backward chaining and provides a clear and flexible foundation for future research in AI and logic programming.