# Knowledge Graph Question Answering System

## Overview

This project implements a smart question-answering system that understands natural language questions about facts like the age of a person or the population of a city and provides precise answers by querying a knowledge graph.

For example, you can ask:

* "How old is Tom Cruise?"
* "What is the population of London?"

The system interprets your question, identifies the relevant entity and the type of information requested, then dynamically fetches the answer from Wikidata, a large open knowledge graph.

## Why Knowledge Graphs?

Knowledge graphs like Wikidata and Google Knowledge Graph organize facts as interconnected entities and relationships. This allows the system to:

* Access rich, up-to-date data through flexible queries
* Discover new entities without needing hardcoded information
* Reason over data, such as calculating age from birth and death dates
* Scale to answer a wide variety of questions

## Solution Approach

Our system combines several intelligent techniques for a unique and robust experience:

* **Dynamic Entity Resolution:** Instead of hardcoding entity IDs, the system uses the Wikidata Search API to find entities based on user questions, handling unexpected names gracefully.
* **Intent Detection:** The system parses your question to understand what information you’re requesting whether it’s age, population, or other facts.
* **Semantic Reasoning:** For example, if a person is deceased, the system calculates their age at death; otherwise, it calculates their current age using birth dates.
* **Dynamic SPARQL Query Generation:** Queries are built on the fly using resolved entity IDs and detected intents to fetch precise answers.
* **Error Handling:** The system gracefully manages ambiguous or missing data and can provide brief explanations if needed.

## Project Structure

* ask\_function.py — Contains the main ask() function that answers your questions.
* utils.py — Helper functions for entity search, question parsing, and query building.
* wikidata\_api.py — Interfaces with Wikidata APIs for searching entities and running SPARQL queries.
* reasoning.py — Contains reasoning logic, such as calculating age from birth and death dates.
* main.py — Entry point to run test cases and demonstrate the system.
* app.py (optional) — A simple web interface to ask questions in your browser.

## How It Works

1. The system uses Wikidata’s Search API to convert your question’s natural language entity name (e.g., “Taylor Swift”) into a unique entity ID.
2. It constructs and runs a SPARQL query against Wikidata’s knowledge graph to retrieve relevant properties like birthdate or population.
3. It applies reasoning, such as calculating current age from birth and death dates, enhancing basic data lookup.
4. Since entities are interconnected in the graph, the system can easily be extended to answer more complex questions by exploring related data.

## Getting Started

1. Run the tests and see examples:

Bash CopyEdit - python main.py

1. (Optional) Run the web interface:

Bash CopyEdit - python app.py

Then open your browser and visit <http://127.0.0.1:5000>, where you can ask questions like:

* “How old is Barack Obama?”
* “What is the population of Mumbai?”

## Future Improvements

* Support more question types (e.g., place of birth, spouse, capital cities)
* Improve natural language understanding with advanced NLP models

## Acknowledgements

This project uses [Wikidata](https://www.wikidata.org), an open knowledge graph maintained by the Wikimedia Foundation, as the data source.

=^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^

=^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^.^= =^