ARTIFICIAL INTELLIGENCE CONCEPTS

# Ontology code

python

from owlready2 import \*

*# Create a new ontology*

onto = get\_ontology("http://example.org/chemical\_ontology.owl")

*# Define classes*

with onto:

    class Chemical(Thing):

        pass

    class ChemicalName(Chemical):

        pass

    class ChemicalSymbol(Chemical):

        pass

    class ChemicalFormula(Chemical):

        pass

*# Define properties*

with onto:

    class has\_name(Chemical >> ChemicalName, FunctionalProperty):

        pass

    class has\_symbol(Chemical >> ChemicalSymbol, FunctionalProperty):

        pass

    class has\_formula(Chemical >> ChemicalFormula, FunctionalProperty):

        pass

*# Add individuals (chemical entities)*

water = Chemical("Water")

water.has\_name = [ChemicalName("Water")]

water.has\_symbol = [ChemicalSymbol("H2O")]

water.has\_formula = [ChemicalFormula("H2O")]

carbon\_dioxide = Chemical("CarbonDioxide")

carbon\_dioxide.has\_name = [ChemicalName("Carbon Dioxide")]

carbon\_dioxide.has\_symbol = [ChemicalSymbol("CO2")]

carbon\_dioxide.has\_formula = [ChemicalFormula("CO2")]

*# Save the ontology*

onto.save(file="chemical\_ontology.owl")

```

This code creates an ontology with the following structure:

- \*\*Classes\*\*: `Chemical`, `ChemicalName`, `ChemicalSymbol`, `ChemicalFormula`.

- \*\*Properties\*\*: `has\_name`, `has\_symbol`, `has\_formula`.

- \*\*Individuals\*\*: Examples like `Water` and `CarbonDioxide`.

---

### Step 3: Load and Query the Ontology

You can load the ontology and query it to retrieve chemical information.

#### Ontology Query Code (`ontology\_query.py`)

```python

from owlready2 import \*

*# Load the ontology*

onto = get\_ontology("chemical\_ontology.owl").load()

*# Query the ontology*

def get\_chemical\_info(query):

    """

    Retrieve chemical information based on a query (name, symbol, or formula).

    """

    query = query.strip().lower()

*# Search for the chemical by name, symbol, or formula*

    for chemical in onto.Chemical.instances():

        name = chemical.has\_name[0].name.lower() if chemical.has\_name else ""

        symbol = chemical.has\_symbol[0].name.lower() if chemical.has\_symbol else ""

        formula = chemical.has\_formula[0].name.lower() if chemical.has\_formula else ""

        if query in [name, symbol, formula]:

            return {

                "name": name,

                "symbol": symbol,

                "formula": formula

            }

    return None

*# Example usage*

if \_\_name\_\_ == "\_\_main\_\_":

    query = input("Enter a chemical name, symbol, or formula: ")

    result = get\_chemical\_info(query)

    if result:

        print(f"Name: {result['name']}")

        print(f"Symbol: {result['symbol']}")

        print(f"Formula: {result['formula']}")

    else:

        print("Chemical not found.")

```

---

### Step 4: Integrate Ontology with Tkinter Chatbot

Now, integrate the ontology query functionality into your Tkinter chatbot.

#### Updated `chatbot.py`

```python

import tkinter as tk

from owlready2 import \*

*# Load the ontology*

onto = get\_ontology("chemical\_ontology.owl").load()

class ChemicalChatbot:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("Chemical Recognition Chatbot")

        self.chat\_history = tk.Text(root, state='disabled')

        self.chat\_history.pack(fill='both', expand=True)

        self.input\_frame = tk.Frame(root)

        self.input\_frame.pack(fill='x')

        self.input\_entry = tk.Entry(self.input\_frame)

        self.input\_entry.pack(side='left', fill='x', expand=True)

        self.send\_button = tk.Button(self.input\_frame, text="Send", command=self.send\_message)

        self.send\_button.pack(side='right')

    def send\_message(self):

        user\_input = self.input\_entry.get()

        self.input\_entry.delete(0, tk.END)

        if user\_input.lower() in ["exit", "quit"]:

            self.root.quit()

            return

        self.update\_chat\_history("You: " + user\_input)

        response = self.get\_response(user\_input)

        self.update\_chat\_history("Bot: " + response)

    def get\_response(self, user\_input):

        result = self.query\_ontology(user\_input)

        if result:

            return f"Name: {result['name']}\nSymbol: {result['symbol']}\nFormula: {result['formula']}"

        else:

            return "Sorry, I couldn't recognize that chemical."

    def query\_ontology(self, query):

        """

        Query the ontology for chemical information.

        """

        query = query.strip().lower()

        for chemical in onto.Chemical.instances():

            name = chemical.has\_name[0].name.lower() if chemical.has\_name else ""

            symbol = chemical.has\_symbol[0].name.lower() if chemical.has\_symbol else ""

            formula = chemical.has\_formula[0].name.lower() if chemical.has\_formula else ""

            if query in [name, symbol, formula]:

                return {

                    "name": name,

                    "symbol": symbol,

                    "formula": formula

                }

        return None

    def update\_chat\_history(self, message):

        self.chat\_history.configure(state='normal')

        self.chat\_history.insert(tk.END, message + "\n")

        self.chat\_history.configure(state='disabled')

        self.chat\_history.yview(tk.END)

if \_\_name\_\_ == "\_\_main\_\_":

    root = tk.Tk()

    app = ChemicalChatbot(root)

    root.mainloop()

```

---

### Step 5: Run the Application

1. Run the ontology creation script:

   ```bash

   python ontology.py

   ```

2. Run the chatbot application:

   ```bash

   python chatbot.py

   ```

---

### Example Interaction

- \*\*Input\*\*: `H2O`

- \*\*Output\*\*:

  ```

  Name: water

  Symbol: h2o

  Formula: h2o

  ```

- \*\*Input\*\*: `Carbon Dioxide`

- \*\*Output\*\*:

  ```

  Name: carbon dioxide

  Symbol: co2

  Formula: co2

  ```