English Dictionary Project

# Overview:

The English Dictionary project is a simple command-line application that allows users to add words, find their meanings, and update their meanings. The words and their meanings are stored in a dictionary.

# Requirements:

Python 3.6 or later

pickle module (included in Python standard library)

Project Structure:

words.txt: A file to store the words and their meanings.

english\_dictionary.py: The main Python script that contains the EnglishDictionary class and its methods.

# Methods:

## load\_words:

This method is used to load the words from the 'words.txt' file when the application starts. It checks if the 'words.txt' file exists. If it does, it opens the file in binary read mode ('rb') and loads the words using the pickle.load() function.

**def load\_words(self):**

**if os.path.exists(self.file\_name):**

**with open(self.file\_name, 'rb') as f:**

**self.words = pickle.load(f)**

**save\_words:**

This method is used to save the words to the 'words.txt' file whenever there is a change in the words. It opens the file in binary write mode ('wb') and saves the words using the pickle.dump() function.

**def save\_words(self):**

**with open(self.file\_name, 'wb') as f:**

**pickle.dump(self.words, f)**

## add\_word:

This method is used to add a new word to the dictionary. It prompts the user to enter a word and its meaning. The word and its meaning are then stored in the dictionary and saved to the 'words.txt' file.

**def add\_word(self):**

**word = input("Enter a word: ")**

**meaning = input("Enter the meaning of the word: ")**

**self.words[word] = meaning**

**self.save\_words()**

**print("Word added successfully.")**

## find\_meaning:

This method is used to find the meaning of a word. It prompts the user to enter a word. If the word is found in the dictionary, its meaning is displayed. Otherwise, an appropriate message is displayed.

**def find\_meaning(self):**

**word = input("Enter a word to find its meaning: ")**

**if word in self.words:**

**print(f"Meaning of '{word}': {self.words[word]}")**

**else:**

**print("Word not found.")**

## update\_word:

This method is used to update the meaning of a word. It prompts the user to enter a word and its updated meaning. The word and its updated meaning are then stored in the dictionary and saved to the 'words.txt' file.

**def update\_word(self):**

**word = input("Enter a word to update its meaning: ")**

**if word in self.words:**

**meaning = input("Enter the updated meaning of the word: ")**

**self.words[word] = meaning**

**self.save\_words()**

**print("Word updated successfully.")**

**else:**

**print("Word not found.")**

## run:

This method is the main method of the application. It displays the main menu and prompts the user to enter a choice. Based on the choice, the corresponding method is called. If the user selects choice 4, the application exits. Otherwise, the main menu is displayed again.

**def run(self):**

**while True:**

**print("\nMain Menu")**

**print("1. Add a new word")**

**print("2. Find the meaning")**

**print("3. Update a word")**

**print("4. Exit")**

**choice = int(input("Enter Choice: "))**

**if choice == 1:**

**self.add\_word()**

**elif choice == 2:**

**self.find\_meaning()**

**elif choice == 3:**

**self.update\_word()**

**elif choice**