**Detailed Explanation**

1. **Import Statements and API Key Configuration**

* **import tkinter as tk**: This line imports the tkinter module, a standard Python interface to the Tk GUI toolkit, and aliases it as **tk** for easier reference throughout the code.
* **from tkinter import filedialog, messagebox**: Specifically imports the **filedialog** and **messagebox** modules from tkinter. **filedialog** is used to open the file explorer for selecting files, while **messagebox** is used to display alerts and information to the user.
* **import openai**: Imports the OpenAI Python client library, enabling the script to interact with OpenAI's API for utilizing AI models such as GPT-3.5.
* **openai.api\_key**: The script sets the OpenAI API key directly. This key is essential for authenticating requests to OpenAI's API. The provided string is a placeholder and should be replaced with a valid API key by the user.

2. **Function Definitions**

* **chat\_gpt\_interaction(prompt)**: This function is designed to interact with the ChatGPT model from OpenAI. It constructs a message payload that includes a system message to define the assistant's role and the user's prompt. It then calls the OpenAI API to generate a response based on the prompt. The function handles exceptions by displaying an error message through a messagebox. This abstraction simplifies the interaction with OpenAI's API elsewhere in the code.
* **resume\_score(resume\_text, job\_description, mandatory\_keywords)**: This function constructs a specific prompt that integrates the resume text, job description, and mandatory keywords to evaluate the resume. It uses the **chat\_gpt\_interaction** function to send this prompt to the GPT model and interprets the response to extract a numerical score. If the response cannot be interpreted as a numerical score, it shows the response to the user for manual evaluation. This function encapsulates the logic for leveraging AI to analyze and score a resume against a job description.
* **upload\_resume() and upload\_job\_description()**: These functions provide the GUI functionality to allow users to upload their resume and job descriptions, respectively. They open a file dialog where the user can choose a file, and then they read and display the content of these files in the respective text widgets in the GUI. This design facilitates a user-friendly way to input the necessary documents for analysis.
* **calculate\_score()**: Orchestrates the process of collecting inputs (resume and job description texts, along with mandatory keywords), invoking the resume scoring function, and handling the scoring process. It's triggered by a GUI button and is a key interaction point for the user, initiating the evaluation process.

3. **GUI Setup**

* The script sets up a basic graphical user interface using tkinter. It initializes the main window (**root**), sets its title, and creates a series of widgets (labels, text boxes, and buttons) organized in a grid layout. Each widget serves a specific purpose:
  + **Labels** (**resume\_label**, **job\_description\_label**) provide textual information to guide the user.
  + **Text widgets** (**resume\_text**, **job\_description\_text**) are multi-line text input areas where the resume and job description can be displayed or directly edited.
  + **Buttons** (**upload\_resume\_button**, **upload\_job\_description\_button**, **calculate\_button**) enable the user to upload documents and initiate the resume scoring process.
* This setup creates an intuitive interface for users to interact with, enabling them to easily upload their resume and job description, and calculate the score based on the criteria defined by the mandatory keywords.

4. **Main Event Loop**

* By calling **root.mainloop()**, the script enters the Tk event loop, which waits for events from the user (such as clicks and input) and responds accordingly. This loop keeps the application running and responsive to user actions.