**Explanation:**

1. **Import necessary libraries:**
   * selenium: For web browser automation.
   * webdriver.Chrome (or webdriver.Firefox, etc.): To control the Chrome browser.
   * Keys: To simulate keyboard actions (e.g., pressing Enter).
   * By: To locate elements on the web page using different criteria.
   * time: To introduce delays for page loading.
2. **Define the scrape\_google\_results function:**
   * Takes the query as input.
   * Initializes the webdriver instance.
   * Navigates to the Google search page.
   * Locates the search bar and enters the query.
   * Presses Enter to initiate the search.
   * Waits for a short time to allow the search results to load.
   * Locates the search result elements using their class name.
   * Iterates through each result:
     + Extracts the title, url, and snippet of each result using appropriate locators.
     + Stores the extracted information in a dictionary.
     + Appends the dictionary to the result\_data list.
   * Closes the browser window.
   * Returns the result\_data list.
3. **Example usage:**
   * Sets the query to "Python programming".
   * Calls the scrape\_google\_results function with the query.
   * Prints the extracted information for each search result.

**Note:**

* This is a basic example and may need adjustments depending on the specific structure of the Google search results page, which can change over time.
* Ensure you have the necessary Selenium driver (e.g., chromedriver) installed and in your system's PATH.
* This script may violate Google's Terms of Service. Use it responsibly and ethically.
* Consider using a headless browser (like webdriver.Chrome(options=chrome\_options)) to avoid opening a visible browser window.

This script provides a foundation for scraping Google search results using Selenium. You can further enhance it by:

* Handling potential exceptions (e.g., NoSuchElementException) more gracefully.
* Implementing pagination to scrape more than the first page of results.
* Adding features to filter results based on specific criteria.
* Using more robust and reliable locators (e.g., XPath) to find elements.
* Improving the code's readability and maintainability.