**Title: Report for Task 1.**

# Objective

To Build a python program which returns a .txt file of Wikipedia explanation after taking a input of any word.

This program:

* Accepts a search query from the user.
* Displays top Wikipedia search suggestions.
* Lets the user pick a topic.
* Fetches the Wikipedia summary of that topic.
* cleans the title and saves the content as a .txt file in the current working directory (CWD).

# Purpose

Takes user input and returns the top 3 Wikipedia search results.

Details:

* Prompts the user for a keyword.
* Uses wikipedia.search() to get matching topics.
* Returns:
  + list of suggestions (if found),
  + -1 if no results or error occurs (graceful fallback).
* Catches **any exception** and avoids crashing.
* Returns a **status code** -1 for the main function to act on.

# Functions

* + 1. def get\_search\_word(options: list):

Displays the search suggestions and allows the user to select one.

**Features:**

* Displays 3 results (1, 2, 3).
* Option 0: Exit program.
* Option 4: Re-do search (returns -2).
* Validates input and handles incorrect values with clear messages.

**Error Handling:**

* Uses try-except to catch ValueError (non-integer input).
* Avoids crashing by validating input bounds.
  + 1. def get\_summary(search\_word):

Fetches the **Wikipedia summary** for the selected topic.

**Features:**

* Uses wikipedia.page(search\_word, auto\_suggest=False) for exact page.
* Handles DisambiguationError:
  + Prompts the user with **alternative disambiguation options**.
  + Returns chosen summary.
  + Again supports 0 for exit and validates choices.
    1. def write\_summary\_to\_txt(summary: str, filename: str):

Writes the summary string to a .txt file named after the topic.

**Features:**

* Opens file in **write mode** ("w").
* Ensures **UTF-8 encoding** for Unicode content.
* Appends **extra newline** at the end for readability.

**Error Handling:**

* Wrapped in a try-except block.
* Catches and prints any error during file creation (e.g., permission denied, invalid paths).
  + 1. def clean\_filename(name: str) -> str:

Sanitizes filenames by replacing invalid characters with underscores \_.

**Details:**

* Uses regex: r'[\\/\*?:"<>|]' to match restricted characters.
* Replaces them with \_.

Windows/macOS/Linux have strict rules on allowed characters in filenames. This function guarantees **cross-platform compatibility**

* + 1. def main():

Acts as the **entry point** and orchestrates all logic.

**Loop:**

* Keeps prompting the user until a valid result is selected.
* Exits when:
  + No search result.
  + User selects exit option.
  + Summary is successfully saved.

**Logic Flow:**

1. Get user input → search results
2. Let user pick one
3. Fetch summary
4. Sanitize title
5. Save file
6. Exit

**Robust Error Handling:**

* Every step is covered with try-except blocks.
* Uses **return codes** (-1, -2) to control flow.
* Graceful exit on exceptions.

# Key feature

1. **Independency**
   1. All logic is **modularized** into self-contained functions.
   2. Each function:
   3. Does one task.
   4. Has **clear input/output**.
   5. Can be **reused or tested independently**.
2. **Robust Error Handling**
   1. **No raw crashes** — everything is caught and reported gracefully.
   2. Uses **user-friendly messages**.
   3. Fallback codes like -1 or -2 ensure the flow doesn’t break.
3. User Interaction
   1. Clear prompts.
   2. Input validation.
   3. Allows retries or exits.
   4. Feedback messages (e.g., getting explanation for..., file saved to...).
4. Sample

