Overview of the Features

1. Create a New Post
2. View All Posts
3. View a Single Post (with comments)
4. Edit a Post
5. Add a Comment to a Post
6. Search Posts (by title)
7. Delete a Post
8. Exit the Program

The **data structure** to store each post will be a **dictionary** of the form:

```
{
  post_id: {
    "title": ...,
    "content": ...,
    "comments": [ ... ]  # a list of comments (strings)
  },
  ...
}
```

We will track posts using an integer counter, so each post ID is unique.

Step-by-Step Implementation

1. Initialize Variables

Create an empty dictionary for blog posts, e.g.:

```
blog_posts = {}
```

Create a variable to store the next available post ID, for example:
python

```
next_post_id = 1
```

2. Main `while` Loop for the Menu

We will use an **infinite loop** (`while True`) to repeatedly show the menu options. The user can exit by selecting the "Exit" option.

Inside this loop:

- Print the menu.
- Take the user’s choice.
- Use `if/elif/else` statements to handle each menu option.

A possible menu layout:

```
while True:
    print("\n==== BLOG MENU ====")
    print("1. Create New Post")
    print("2. View All Posts")
    print("3. View Single Post")
    print("4. Edit a Post")
    print("5. Add a Comment")
    print("6. Search Posts")
    print("7. Delete a Post")
    print("8. Exit")

    choice = input("Choose an option (1-8): ")

    if choice == "1":
        # Create New Post
        ...
    elif choice == "2":
        # View All Posts
        ...
    elif choice == "3":
        # View Single Post
        ...
    elif choice == "4":
        # Edit a Post
        ...
    elif choice == "5":
        # Add a Comment
        ...
    elif choice == "6":
        # Search Posts
        ...
    elif choice == "7":
        # Delete a Post
        ...
    elif choice == "8":
        print("Exiting the program. Goodbye!")
        break
    else:
        print("Invalid choice. Try again.")
```

---

3. Detailed Menu Features

Below are the **implementation details** and **hints** for each option. Students should fill in the code themselves, but the guidelines below will help them figure out how to implement each step.

Option 1: Create a New Post

- Prompt the user for a **title** and **content**.

Create a dictionary for the new post. For example:

```
new_post = {
    "title": post_title,
    "content": post_content,
    "comments": []
}
```

- Add `new_post` to `blog_posts` using the key `next_post_id`.
- Increment `next_post_id` by 1 so the next post has a unique ID.
- **Hint**: Remember to use `input()` to get user input for title and content.

Example steps:

```
post_title = input("Enter post title: ")
post_content = input("Enter post content: ")

blog_posts[next_post_id] = {
    "title": post_title,
    "content": post_content,
    "comments": []
}

next_post_id += 1
print("New post created successfully!")
```

Option 2: View All Posts

- If blog_posts is empty, print a message indicating no posts exist.

Otherwise, use a **for loop** to iterate over the keys in blog_posts:

```
for pid in blog_posts:
    print(f"Post ID: {pid} | Title: {blog_posts[pid]['title']}")
```

- This will show only a quick summary. If you want to show content as well, you can print that, but it might be more readable to just show the title here, then let the user choose to view the full content (option 3).

Option 3: View a Single Post (with comments)

- Prompt the user for the **post ID**.
- Check if that post ID exists in blog_posts.
  - If it does, print the post details (title, content).
  - Then print each comment from the comments list, if any exist.
  - If no comments, print a message: "No comments yet."
- If the post does not exist, print an error message.

Hint: Don't forget to convert the user's input to int, since our blog_posts dictionary keys are integers:

```
post_id = int(input("Enter the ID of the post you want to view: "))
if post_id in blog_posts:
    print("Title:", blog_posts[post_id]["title"])
    print("Content:", blog_posts[post_id]["content"])
    comments_list = blog_posts[post_id]["comments"]
    if len(comments_list) == 0:
        print("No comments yet.")
    else:
        print("\n--- Comments ---")
        for comment in comments_list:
            print("-", comment)
else:
    print("Post not found.")
```

Option 4: Edit a Post

- Prompt the user for the post ID.
- If it exists, show the current title and content.
- Ask the user for a new title and new content.
    - If the user wants to leave the title or content unchanged, you can let them enter something like `"<leave>"` or an empty string to skip changing that field. This is optional and might add complexity.
- Update the post in `blog_posts` with the new title and/or content.
- If the post ID doesn't exist, print an error message.

Example:

```
post_id = int(input("Enter the ID of the post to edit: "))
if post_id in blog_posts:
    current_title = blog_posts[post_id]["title"]
    current_content = blog_posts[post_id]["content"]

    print("Current Title:", current_title)
    print("Current Content:", current_content)

    new_title = input("Enter new title (or press Enter to leave unchanged): ")
    new_content = input("Enter new content (or press Enter to leave unchanged): ")

    # If new title is not empty, update
    if new_title != "":
        blog_posts[post_id]["title"] = new_title

    # If new content is not empty, update
    if new_content != "":
        blog_posts[post_id]["content"] = new_content

    print("Post updated successfully!")
else:
    print("Post not found.")
```

Option 5: Add a Comment to a Post

- Ask the user for the post ID.
- Check if the post exists.
- If it does, prompt for the comment text.
- Append the comment (a simple string) to the `comments` list in that post's dictionary.
- If it doesn't exist, print an error message.

Example:

```
post_id = int(input("Enter the ID of the post you want to comment on: "))
if post_id in blog_posts:
    new_comment = input("Enter your comment: ")
    blog_posts[post_id]["comments"].append(new_comment)
    print("Comment added!")
else:
    print("Post not found.")
```

Option 6: Search Posts (by title)

- Ask the user for a **search term** (a string).
- Iterate over all posts in `blog_posts` to see if the **search term** appears in the post's **title** (you can do a simple substring check with `in`).
    - For example: `if search_term.lower() in blog_posts[pid]["title"].lower():`
- Print out any matching posts (show ID and title).
- If no matches, print a message saying "No matches found."

Example:

```
search_term = input("Enter a keyword to search for in titles: ")
found_any = False
for pid in blog_posts:
    title_lower = blog_posts[pid]["title"].lower()
    if search_term.lower() in title_lower:
        print(f"Match found -> Post ID: {pid}, Title: {blog_posts[pid]['title']}")
        found_any = True

if not found_any:
    print("No matches found.")
```

Option 7: Delete a Post

- Prompt the user for the post ID.
- Check if it exists.
  - If yes, use `del blog_posts[post_id]` to remove it.
  - Print a confirmation message.
- If it doesn't exist, print an error message.

Example:

```
post_id = int(input("Enter the ID of the post to delete: "))
if post_id in blog_posts:
    del blog_posts[post_id]
    print("Post deleted successfully.")
else:
    print("Post not found.")
```

Option 8: Exit

- Simply `break` out of the `while True:` loop.
- Optional: Print a goodbye message.