DS2030 Data Structures and Algorithms for Data Science
Practice Set - 3

August 20th, 2025

Lab Instructions

- Create a folder named **"DS2030_<RollNo.>"** (all letters in capital) in **"home"** directory. Eg- **DS2030_142402022**
- Name the script files in the given format **"<your_roll_no>_TextEditor_Lab2.py"**
- Make sure the **folder, files, classes, functions and attributes** are named as instructed in the lab sheet.
- We will not be able to evaluate your work if the folder is not properly named or is not located in the home directory.
- Save your progress before leaving the lab.
- You are not allowed to share code with classmates or use external code. Copying will result in a fail grade.

1 Task 1: Define the Notepad Class (3 Points)

You are required to design a simple text editor (`Notepad`) using stacks to support **Undo** and **Redo**.
Attributes:

- `text` : Current text stored in the notepad.
- `undo_stack` : Stack storing previous states of the text.
- `redo_stack` : Stack storing undone states for redo.

Methods:

- `__init__()` : Initialize attributes.
- `write(new_text)` : Append new text and push old state to undo stack.
- `undo()` : Revert to the last state.
- `redo()` : Restore undone text.
- `show()` : Display the current text.

```
class Notepad:
    def __init__(self):
        """
        Initialize with empty text, undo stack, and redo stack.
        """
        # TO DO

    def write(self, new_text):
        """
        Append new_text to the current text.
        Save previous state in undo_stack and clear redo_stack.
        """
        # TO DO

    def undo(self):
        """
        Undo the last action.
        Move current text to redo_stack and restore from undo_stack.
        """
        # TO DO

    def redo(self):
        """
        Redo the last undone action.
        Move current text to undo_stack and restore from redo_stack.
        """
        # TO DO

    def show(self):
        """
        Print the current text.
        """
        # TO DO
```

2 Task 2: Extend to a Menu-Driven Program (3 Points)

Write a menu-driven program to interact with the notepad:

- `1. Write Text`
- `2. Undo`
- `3. Redo`
- `4. Show Text`
- `5. Exit`

```
def menu():
    pad = Notepad()
    while True:
        print("\n1. Write Text")
        print("2. Undo")
        print("3. Redo")
        print("4. Show Text")
        print("5. Exit")

        choice = input("Enter choice: ")

        if choice == "1":
            text = input("Enter text to write: ")
            pad.write(text)
        elif choice == "2":
            pad.undo()
        elif choice == "3":
            pad.redo()
        elif choice == "4":
            pad.show()
        elif choice == "5":
            break
        else:
            print("Invalid choice. Try again.")
```

3 Task 3: Testing (2 Points)

Write test cases to validate functionality:

- Writing text updates the notepad correctly.
- Undo reverts to the previous state.
- Redo restores the undone state.
- Edge cases: Undo with no history, Redo with no history.

```python
def test():
    pad = Notepad()
    pad.write("Hello")
    pad.write(" World")
    pad.undo()
    pad.redo()
    pad.show()

# Run test
test()
```

Sample Output

```
Current Text: Hello World
Undo performed.
Current Text: Hello
Redo performed.
Current Text: Hello World
```