**CODE:\**

import matplotlib.pyplot as plt

import numpy as np

import random

class VirtualMemory:

    def \_\_init\_\_(self, num\_pages, num\_frames):

        self.num\_pages = num\_pages

        self.num\_frames = num\_frames

        self.page\_table = {}  # Simulated page table, initially empty

        self.frames = [None] \* num\_frames  # Simulated physical memory

    def load\_page(self, page\_number):

        if page\_number in self.page\_table:

            frame\_number = self.page\_table[page\_number]

            print(f"Page {page\_number} already in Frame {frame\_number}")

        else:

            free\_frame = self.get\_free\_frame()

            if free\_frame is not None:

                self.frames[free\_frame] = page\_number

                self.page\_table[page\_number] = free\_frame

                print(f"Page {page\_number} loaded into Frame {free\_frame}")

            else:

                self.handle\_page\_fault(page\_number)

    def get\_free\_frame(self):

        for i, frame in enumerate(self.frames):

            if frame is None:

                return i

        return None

    def handle\_page\_fault(self, page\_number):

        # Replace a page using some algorithm (e.g., FIFO, LRU)

        # For simplicity, let's just pick a random frame to replace

        frame\_to\_replace = random.randint(0, self.num\_frames - 1)

        old\_page = self.frames[frame\_to\_replace]

        del self.page\_table[old\_page]

        self.page\_table[page\_number] = frame\_to\_replace

        self.frames[frame\_to\_replace] = page\_number

        print(f"Page {old\_page} evicted. Page {page\_number} loaded into Frame {frame\_to\_replace}")

    def display\_memory(self):

        fig, ax = plt.subplots()

        page\_numbers = list(self.page\_table.keys())

        frame\_numbers = list(self.page\_table.values())

        for i, frame\_number in enumerate(self.frames):

            if frame\_number is not None:

                ax.barh(i, 1, color='blue', label=f'Frame {i}: Page {frame\_number}')

            else:

                ax.barh(i, 1, color='white', label=f'Frame {i}: Empty')

        ax.set\_yticks(np.arange(self.num\_frames))

        ax.set\_yticklabels([f'Frame {i}' for i in range(self.num\_frames)])

        ax.set\_xlabel('Pages')

        ax.set\_title('Virtual Memory')

        ax.legend()

        plt.show()

# Example usage

virtual\_memory = VirtualMemory(num\_pages=10, num\_frames=4)

virtual\_memory.load\_page(0)

virtual\_memory.load\_page(1)

virtual\_memory.load\_page(2)

virtual\_memory.load\_page(3)

virtual\_memory.load\_page(4)

virtual\_memory.display\_memory()

**OUTPUT:**

