第 3 次隨堂-隨堂-QZ3

學號：112111207
姓名：陳品霖
作業撰寫時間：40 (mins，包含程式撰寫時間)
最後撰寫文件日期：2024/11/24

本份文件包含以下主題：(至少需下面兩項，若是有多者可以自行新增)

- 說明內容
- 個人認為完成作業須具備觀念

說明程式與內容

1. `push(item)`：新增資料到堆疊

- 創建新節點，並將其設為新的堆疊頂端。
- 可視為 `addNodeToStack` 的整合版。

```
def push(self, item: int) -> None:
        new_node = Node(item)       # 建立新節點
        new_node.link = self.top    # 將新節點的 link 指向目前的頂端
        self.top = new_node         # 更新頂端為新節點
        print(f"Pushed {item} to stack.")
```

2. `pop()`：刪除頂端節點並返回資料

- 移除頂端節點並更新頂端指標。
- 若堆疊為空，則拋出 `STACK_EMPTY` 異常。
- 可視為 `delStack` 的整合版。

```
def pop(self) -> int:
        if self.top is None:        # 檢查堆疊是否為空
            raise Exception("STACK_EMPTY")
        removed_node = self.top     # 暫存頂端節點
        self.top = self.top.link    # 更新頂端為下一個節點
        data = removed_node.data    # 取出頂端節點的資料
        del removed_node            # 釋放記憶體
        print(f"Popped {data} from stack.")
        return data
```

3.`peek()`：查看頂端資料

- 不刪除節點，只返回目前堆疊頂端的資料。

```
    def peek(self) -> int:
            if self.top is None:
                raise Exception("STACK_EMPTY")
            return self.top.data
```

4. `is_empty()`：檢查堆疊是否為空

- 返回布林值，表示堆疊是否為空。

```
    def is_empty(self) -> bool:
            return self.top is None
```

5. `display()`：列印堆疊

- 顯示堆疊中的所有資料（從頂端到底部）。

```
    def display(self) -> None:
            current = self.top
            print("Stack (top to bottom):", end=" ")
            while current:
                print(current.data, end=" -> ")
                current = current.link
            print("NULL")
```

1. 請參閱投影片 Topic5 的第 31 至 35 頁，請用物件導向方式進行新增與刪除。 (請參照題目 pdf)

```
class Node:
    def __init__(self, data=None):
        self.data = data  # 儲存節點資料
        self.link = None  # 指向下一個節點

class LinkedStack:
    def __init__(self):
        self.top = None  # 初始化堆疊頂端為空

    # 新增節點到堆疊
    def push(self, item: int) -> None:
        new_node = Node(item)       # 建立新節點
        new_node.link = self.top    # 將新節點的 link 指向目前的頂端
        self.top = new_node         # 更新頂端為新節點
        print(f"Pushed {item} to stack.")

    # 刪除堆疊頂端的節點
    def pop(self) -> int:
        if self.top is None:        # 檢查堆疊是否為空
```

```
            raise Exception("STACK_EMPTY")
        removed_node = self.top     # 暫存頂端節點
        self.top = self.top.link    # 更新頂端為下一個節點
        data = removed_node.data    # 取出頂端節點的資料
        del removed_node            # 釋放記憶體
        print(f"Popped {data} from stack.")
        return data

    # 查看堆疊頂端資料（不刪除）
    def peek(self) -> int:
        if self.top is None:
            raise Exception("STACK_EMPTY")
        return self.top.data

    # 檢查堆疊是否為空
    def is_empty(self) -> bool:
        return self.top is None

    # 列印整個堆疊
    def display(self) -> None:
        current = self.top
        print("Stack (top to bottom):", end=" ")
        while current:
            print(current.data, end=" -> ")
            current = current.link
        print("NULL")
```

個人認為完成作業須具備觀念

應熟悉資料結構中的鏈結串列與堆疊的基本特性，包括堆疊的後進先出（LIFO）原則及鏈結串列中節點的新增與刪除過程。再來，需掌握如何實現堆疊的基本操作，例如新增節點（Push）與刪除節點（Pop），並了解節點記憶體的分配與釋放方法。還須了解 Python 程式語言的類別（Class）與物件導向設計的應用。