

Movimento LILO em Python

Mário Leite

Os movimentos REAIS em pilhas, são três FILO, FIFO e LIFO; mas, inventaram o tal **LILO** (Last In, Last Out - Último a Entrar Último a Sair); talvez para incluir mais um arranjo com a sigla começando com a letra "L". Mas, se considerarmos "LILO", isso não seria natural para uma pilha, pois uma pilha é, por definição, LIFO; porém, podemos simular um comportamento LILO em uma pilha com certas manipulações:

Exemplo técnico: simulação de fila usando duas pilhas.

Para obter comportamento FIFO/LILO com pilhas:

- **Pilha A** (entrada): insere elementos.
- **Pilha B** (saída): quando precisar remover, se B estiver vazia, transfere todos os elementos de A para B (invertendo a ordem).

Assim, o primeiro elemento inserido (Last In se considerarmos a inversão) será o último a sair? Na verdade, essa implementação simula uma fila (FIFO) usando duas pilhas, então o comportamento resultante é **First In, First Out**, o que é equivalente a Last In, Last Out (LILO) se olharmos pelo fim da fila. O resumo da lógica é o seguinte:

- Pilha natural: LIFO.
- Fila natural: FIFO (ou LILO, que é a mesma coisa, mas olhando pelo final da fila).
- Se quisermos "LILO" em pilha" estaremos, provavelmente, buscando uma fila implementada com pilhas, que é um exercício clássico de ciência da computação.

Na verdade, o movimento **LILO** em pilha é apenas uma ilusão conceitual. O que existe, de verdade, é mostrado na **figura 1** (tirado da Internet). A **figura 2** mostra a saída de um programa que faz a simulação de um **LILO**.

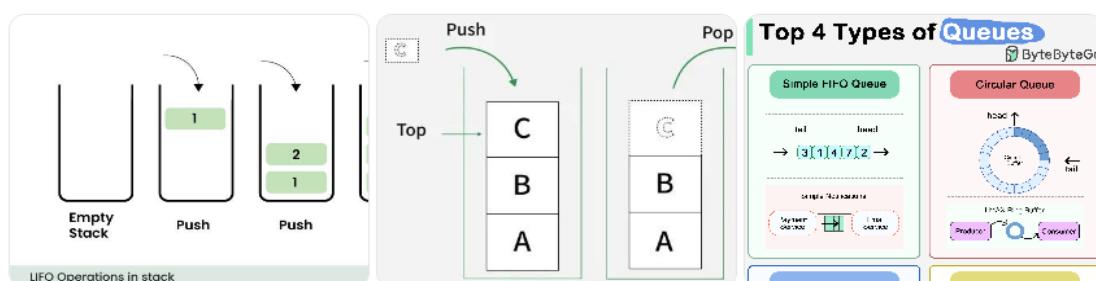


Figura 1 - O verdadeiro movimento em uma pilha

Então... de onde vem o tal "LILO"?

LILO (Last In, Last Out) às vezes aparece em:

- explicações confusas,
- interpretações visuais equivocadas,
- tentativas de descrever "ordem de inserção vs observação".

Mas isto não define nenhuma estrutura de dados funcional.

Se algo fosse realmente **LILO**, significaria:

- o último a entrar também sair por último
- isso não é pilha
- isso não é fila
- isso não é nada útil do ponto de vista algorítmico

The screenshot shows the Python IDLE Shell 3.13.0 interface. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The window displays the following text:

```
Python 3.13.0 (tags/v3.13.0:60403a5, Oct  7 2024, 09:38:07) [MSC v.1941 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> ====== RESTART: D:/Cantinho da Programação/Códigos/Python/MovimentoLILO.py ======
Inserido: 10 | Estrutura: [10]
Inserido: 20 | Estrutura: [10, 20]
Inserido: 30 | Estrutura: [10, 20, 30]

Remoções (simulação LILO):
Removido: 10 | Estrutura: [20, 30]
Removido: 20 | Estrutura: [30]
Removido: 30 | Estrutura: []
Estrutura vazia!
>>> |
```

Ln: 14 Col: 0

Figura 2 - Saída do programa “MovimentoLILO”

Nota1: Postagem baseada no livro: “*1001 Programas em Python Para Você Aprender Praticando - Volume 3: Nível Avançado*”. Publicado pelo na “Amazon” e no “Clube de Autores” <https://www.amazon.com.br/Curso-B%C3%A1sico-Programa%C3%A7%C3%A3o-Teoria-Pr%C3%A1tica/dp/8539908700>

Nota2: Acesse o *link* abaixo para ver meus mais recentes livros de Python publicado pelo “Clube de Autores”, no formato impresso, da coleção “*1001 Programas em Python Para Você Aprender Praticando*”:

Volume1: Nível Básico (500 programas)
Volume2: Nível Intermediário (300 programas)
Volume3: Nível Avançado (201 programas)
<https://clubedeautores.com.br/livros/autores/mario-leite>

Para adquirir PDF dos livros: marleite@gamil.com

```

"""
=====
MovimentoLILO.py
Simulação ARTIFICIAL de LILO (Last In, Last Out).
ATENÇÃO: isto NÃO é uma pilha real
=====

class ClsLILO:
    def __init__(self):
        self.Lista = []

    def Inserir(self, item):
        self.Lista.append(item)
        print(f"Inserido: {item} | Estrutura: {self.Lista}")

    def Remover(self):
        if self.EstaVazia():
            print("Estrutura vazia!")
            return None

        #Se houver mais de um elemento,
        #remove SEMPRE o mais antigo
        if len(self.Lista) > 1:
            item = self.Lista.pop(0)
        else:
            #Só sobra o último inserido
            item = self.Lista.pop()

        print(f"Removido: {item} | Estrutura: {self.Lista}")
        return item

    def EstaVazia(self):
        return len(self.Lista) == 0

=====
#Programa principal

Lilo = ClsLILO()

#Inserções
Lilo.Inserir(10)
Lilo.Inserir(20)
Lilo.Inserir(30)

print("\nRemoções (simulação LILO):")
Lilo.Remover()      #10
Lilo.Remover()      #20
Lilo.Remover()      #30 ← último a entrar sai por último

Lilo.Remover()      #estrutura vazia
#Fim do programa "MovimentoLILO" -----

```