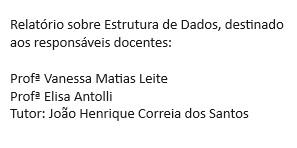
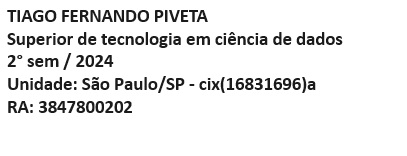
# Relatório Acadêmico: Estrutura de Dados



## 1. Introdução

Este relatório acadêmico visa explicar em detalhes o código Python fornecido, focando em cada uma das classes e funções, suas relações e o funcionamento geral da estrutura de dados implementada. O código em questão trata de uma implementação simples de uma lista   
encadeada (Linked List) em Python, abordando conceitos fundamentais de estrutura de dados e suas operações básicas.

## 2. Explicação Detalhada do Código

A implementação apresentada no código é composta por duas classes principais: a classe 'Node' e a classe 'LinkedList'.   
Cada uma dessas classes tem responsabilidades específicas na construção e manipulação da lista encadeada.

Classe Node:

A classe 'Node' representa um único elemento (nó) na lista encadeada. Ela contém dois atributos principais:  
- data: Armazena o dado que o nó contém.  
- next: Uma referência para o próximo nó na lista. Inicialmente, é definido como 'None'.

Classe LinkedList:

A classe 'LinkedList' é responsável por gerenciar a lista como um todo. Ela contém métodos para adicionar nós, imprimir a lista, e obter o   
tamanho da lista. Os principais métodos são:  
- \_\_init\_\_: Inicializa uma nova lista encadeada, com 'head' apontando para 'None' e tamanho (size) igual a zero.  
- append: Adiciona um novo nó ao final da lista.  
- print\_list: Percorre e imprime todos os nós da lista.  
- getTamanhoLista: Retorna o tamanho atual da lista encadeada.

## 3. Relações Entre as Classes e Métodos

A relação entre as classes 'Node' e 'LinkedList' é fundamental para o funcionamento da lista encadeada. Cada instância da   
classe 'LinkedList' pode conter múltiplos objetos 'Node', conectados entre si através do atributo 'next'. O método 'append' é responsável por   
inserir novos nós ao final da lista, mantendo a estrutura encadeada, enquanto 'print\_list' percorre a lista a partir do 'head' e exibe os   
dados armazenados.

O método 'getTamanhoLista' oferece uma forma eficiente de verificar quantos nós estão presentes na lista, retornando o valor do atributo 'size'.

## 4. Ilustrações do Código

Abaixo estão imagens que ilustram partes importantes do código, com destaque para as classes e métodos discutidos   
anteriormente. Essas imagens ajudam a visualizar melhor como o código está estruturado e a compreender suas funcionalidades.

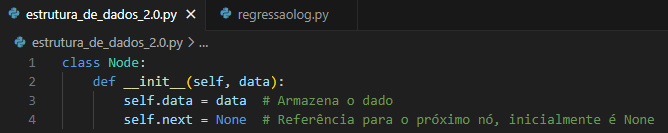
Imagem do código da classe Node.

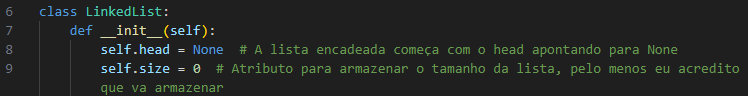
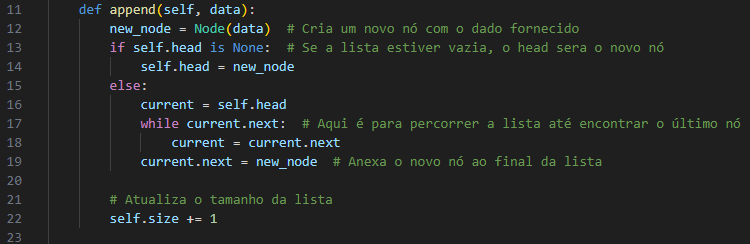
Imagem do código da classe LinkedList.  


Imagem do código do método append.  


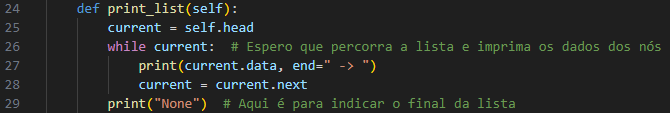
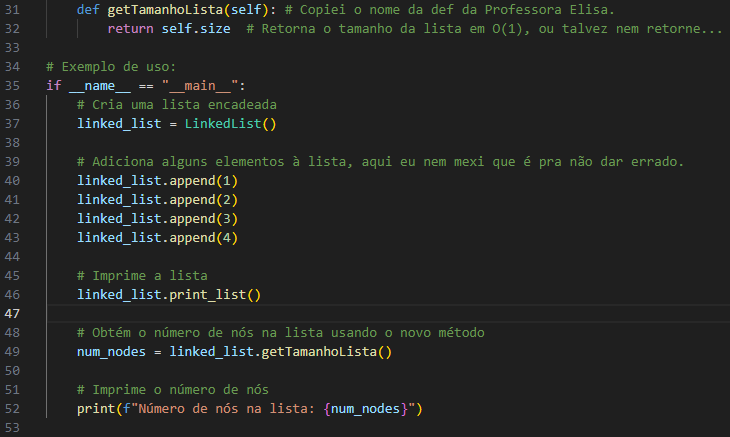
Imagem do código do método print\_list.

Imagem do código do método getTamanhoLista.  


## 5. Conclusão

Neste relatório, foi realizada uma análise detalhada do código Python que implementa uma lista encadeada.   
Através do estudo das classes 'Node' e 'LinkedList', pudemos observar como elementos individuais de dados são organizados e interligados em uma estrutura   
sequencial. A classe 'Node', simples em sua concepção, é a base para construir estruturas mais complexas, enquanto a classe 'LinkedList' serve como   
um container que gerencia e manipula esses nós de forma eficiente.  
  
As operações discutidas, como inserção de elementos (através do método 'append') e a impressão dos elementos da lista (com 'print\_list'), demonstram   
os aspectos fundamentais de listas encadeadas: o gerenciamento dinâmico de memória e a flexibilidade em operações de inserção e remoção de elementos.  
  
As estruturas de dados, como as listas encadeadas, são fundamentais para a ciência da computação e programação, fornecendo a base para muitos   
algoritmos e aplicações. Elas permitem o armazenamento e manipulação eficientes de dados, adaptando-se a diferentes cenários e necessidades   
de otimização.  
  
Refletindo sobre a importância das estruturas de dados, é evidente que a compreensão e a prática dessas implementações são essenciais para o desenvolvimento   
de habilidades em programação e para a solução de problemas complexos. A lista encadeada, embora simples, representa a essência do pensamento algorítmico   
e da organização de dados em sistemas computacionais. Com o aprofundamento no estudo de estruturas mais avançadas, como árvores e grafos, percebe-se que   
o conhecimento adquirido com listas encadeadas serve como um alicerce indispensável para a evolução em áreas mais complexas da computação.  
  
Em suma, o exercício de analisar e implementar listas encadeadas contribui significativamente para a formação de um programador ou cientista da   
computação, proporcionando não apenas uma compreensão técnica, mas também uma apreciação das elegantes soluções que estruturas de dados podem oferecer para   
os desafios de manipulação e organização de informações.