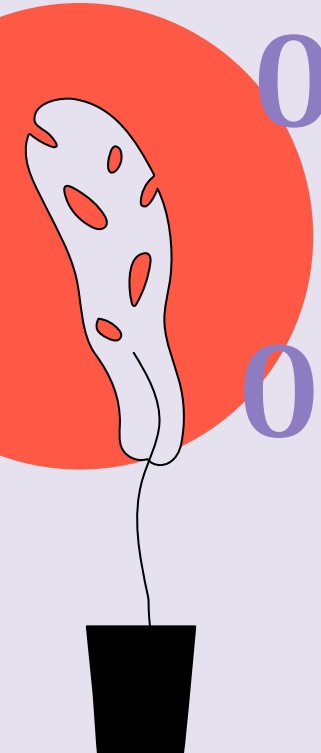
Algoritmos e Estruturas de Dados

Aula 02 - Listas

Prof. Victor Lundgren

Sumário



01

Listas Simples

TADs e listas

TADs

- Tipos Abstratos de Dados (TADs) são uma especificação:
  - Dados + Operações.
- TADs implicam em conceitos matemáticos (i.e. algebras) independentes das suas implementações;
- A implementação ocorre em torno dos tipos de dados da linguagem alvo por meio de estruturas de dados.

Exemplo: [ints], (+, -, *, /)

TADs em Python

- Utilizaremos definições de classes;
- Estas são **encapsuladas**, não sabemos como serão implementadas;
- Algumas estruturas são suportadas como *built-ins* de Python;
- **Não** utilizaremos os métodos prontos!

cont_regressiva = [5, 4, 3, 2, 1]

Listas

- Em Listas nós tem um encadeamento sequencial dos dados do conjunto;
- Podemos adicionar elementos ou removê-los;

Temos a sequência: [A0, A1, ..., A(**n**-1)], onde a sequência possui tamanho **n**. Uma lista com **n** = 0 é uma lista vazia.
O elemento Ai+1 sucede o elemento Ai, que sucede o elemento Ai-1.

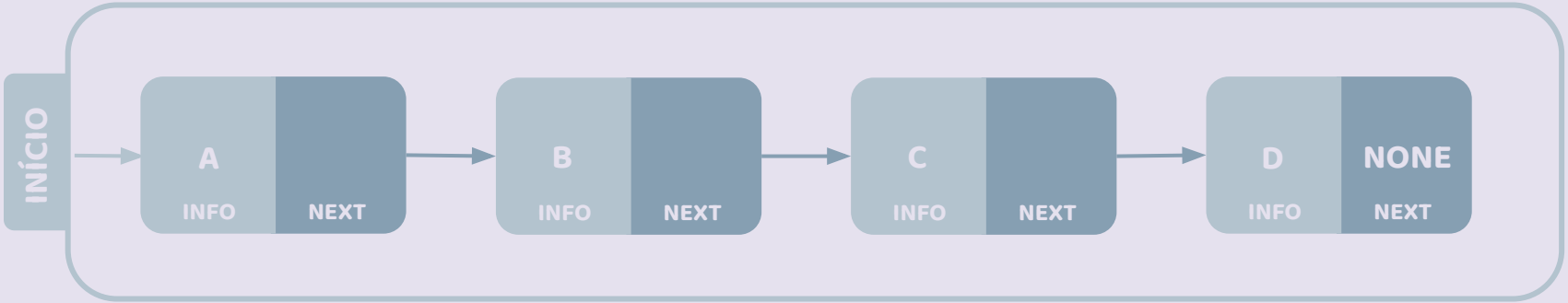
Listas

Operações

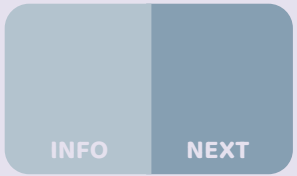
- imprimir_lista()
- esvaziar()
- inserir(valor)
- remover(valor)
- procurar_iesimo(iesimo)

Listas - Intuição

class Lista:



class No:



Listas

Nó (node) p

Um objeto do tipo Node:

- Informação: dados úteis armazedos em p.info
- Próximo: referência ao próximo nó em p.next

Fim da Lista

O fim da Lista é marcado pelo nó no qual
p.next == None
Ou seja, onde o próximo nó for nulo

Informação encadeada

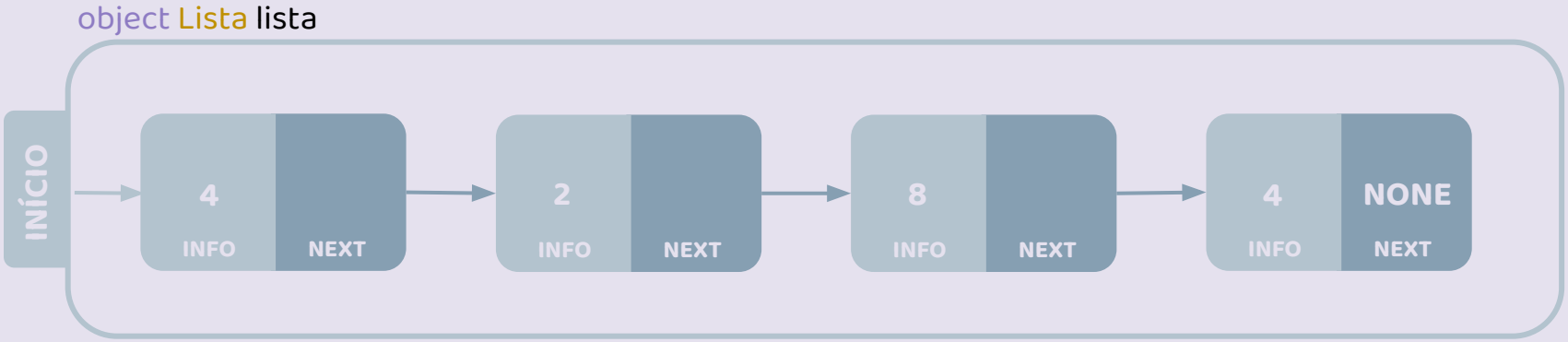
Caso p.next != None
Então p.next.info aponta para a porção de informação nó seguinte ao nó referenciado por p

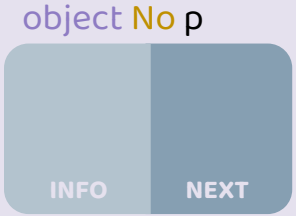
Listas com *list* do Python

Em Python temos listas como *built-ins*. Podemos apenas usar uma variável (mais eficiente), ou criar uma classe intermediária simples.

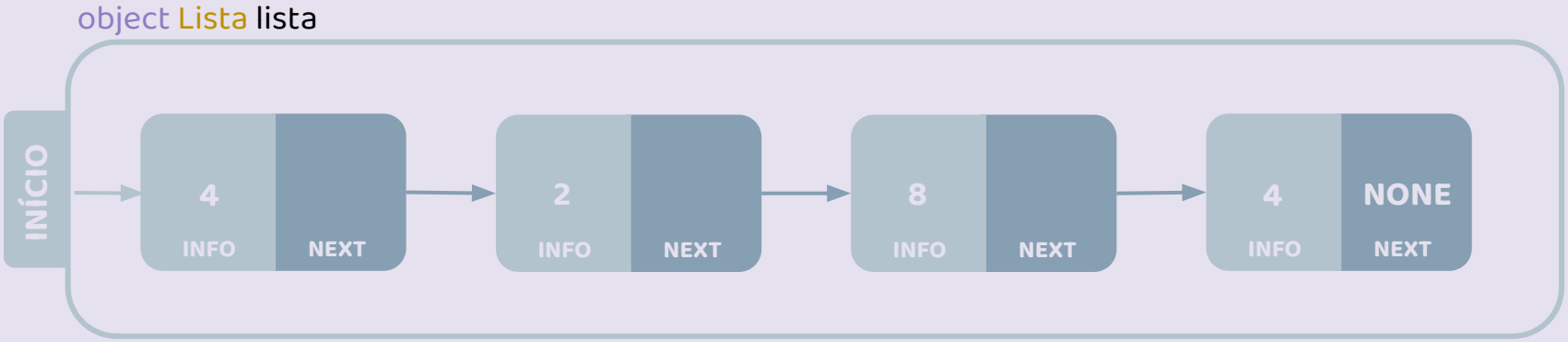
- Custo das funções.

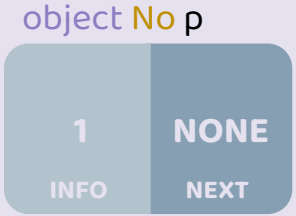
Listas - Inserção no Início



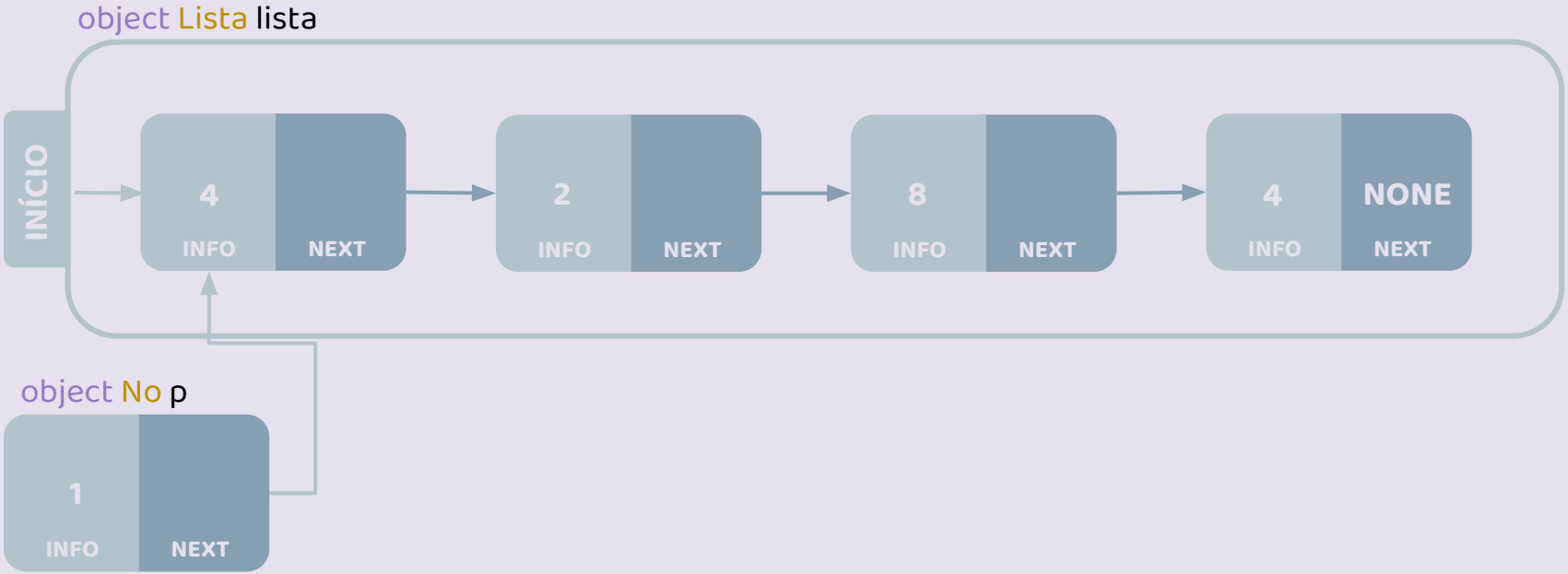


Listas - Inserção no Início

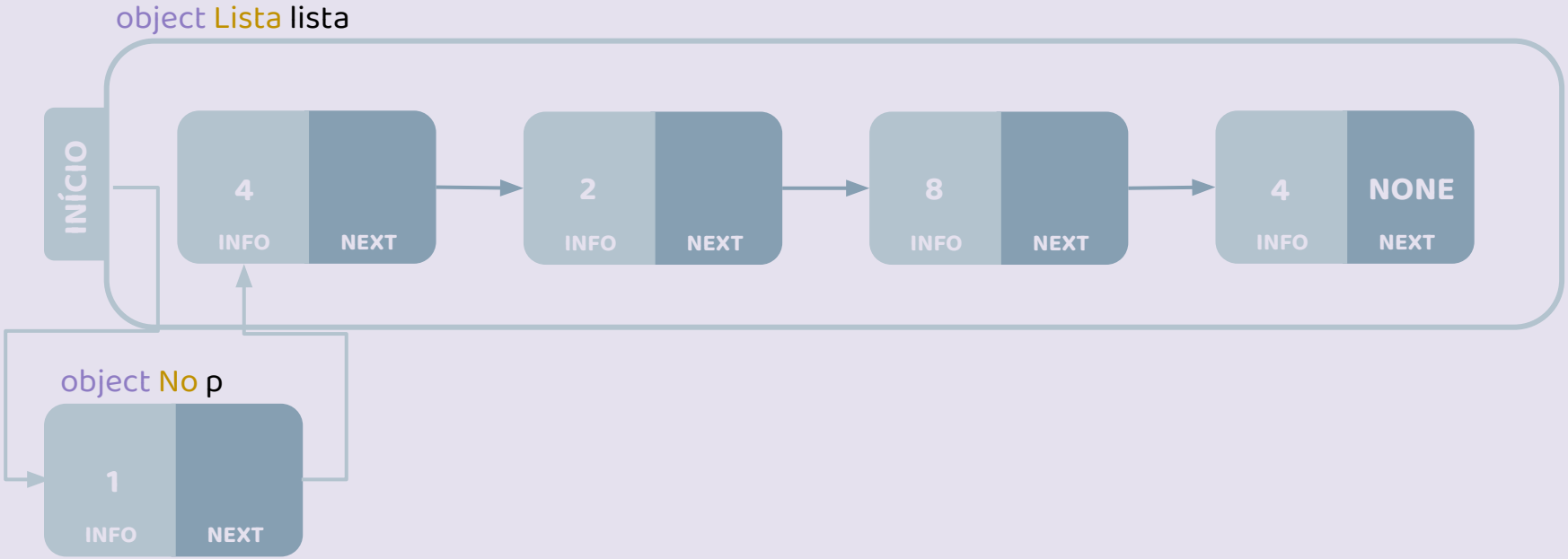




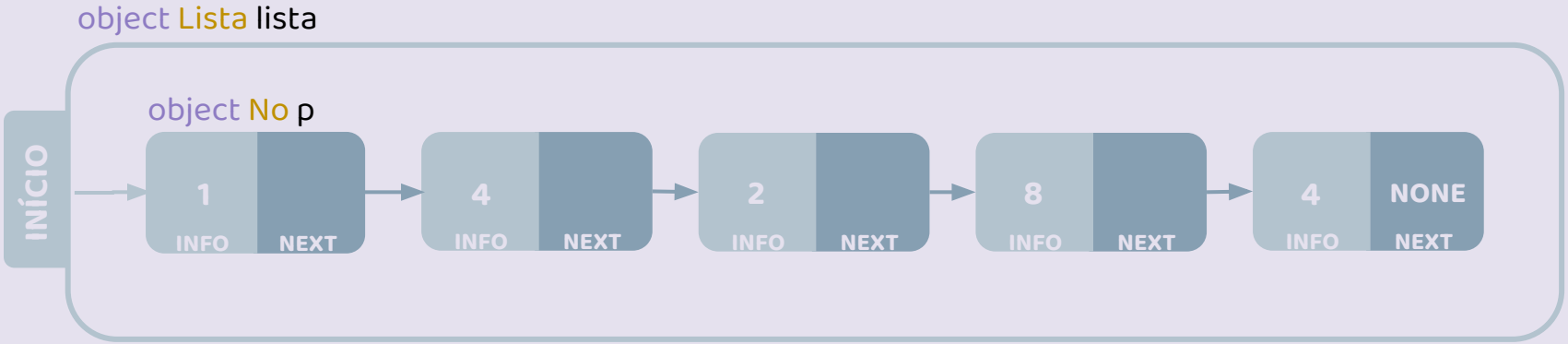
Listas - Inserção no Início



Listas - Inserção no Início

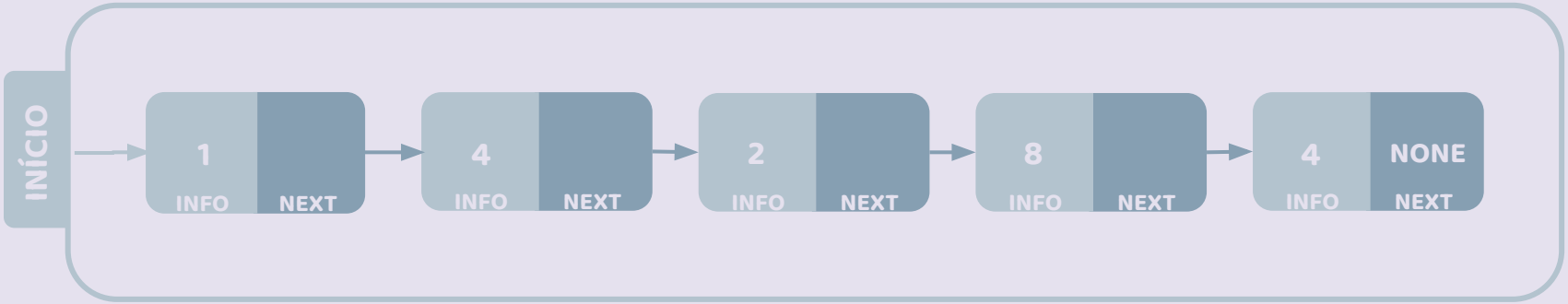


Listas - Inserção no Início

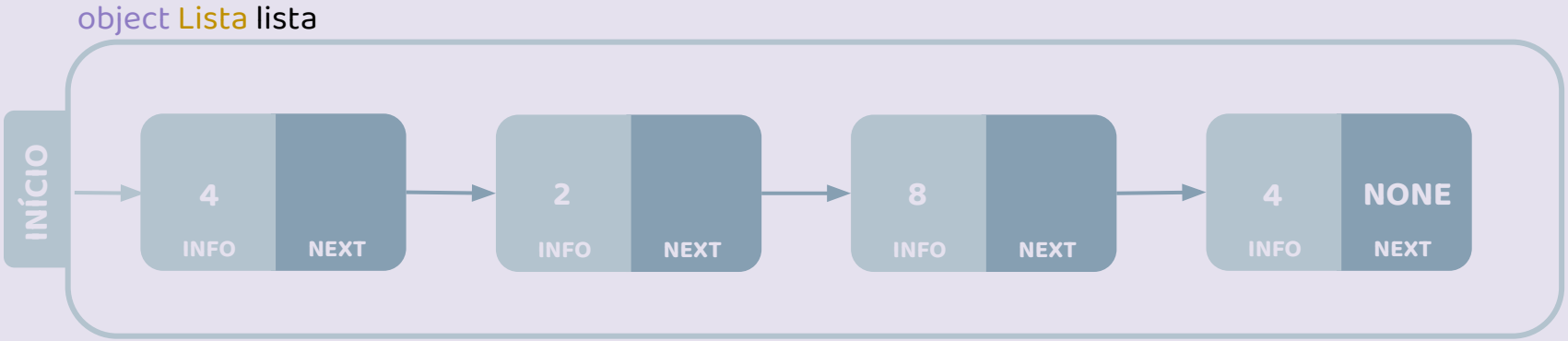


Listas - Inserção

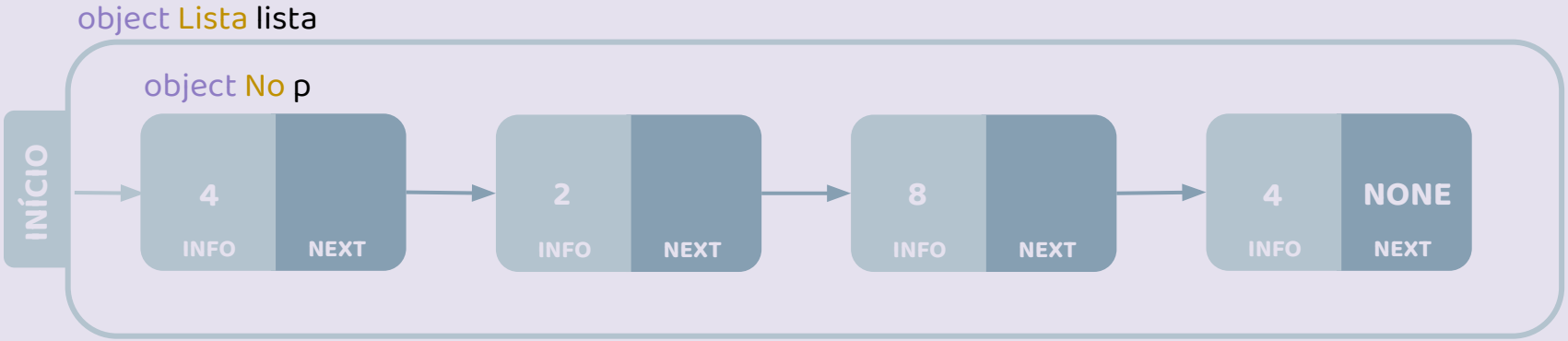
object Lista lista



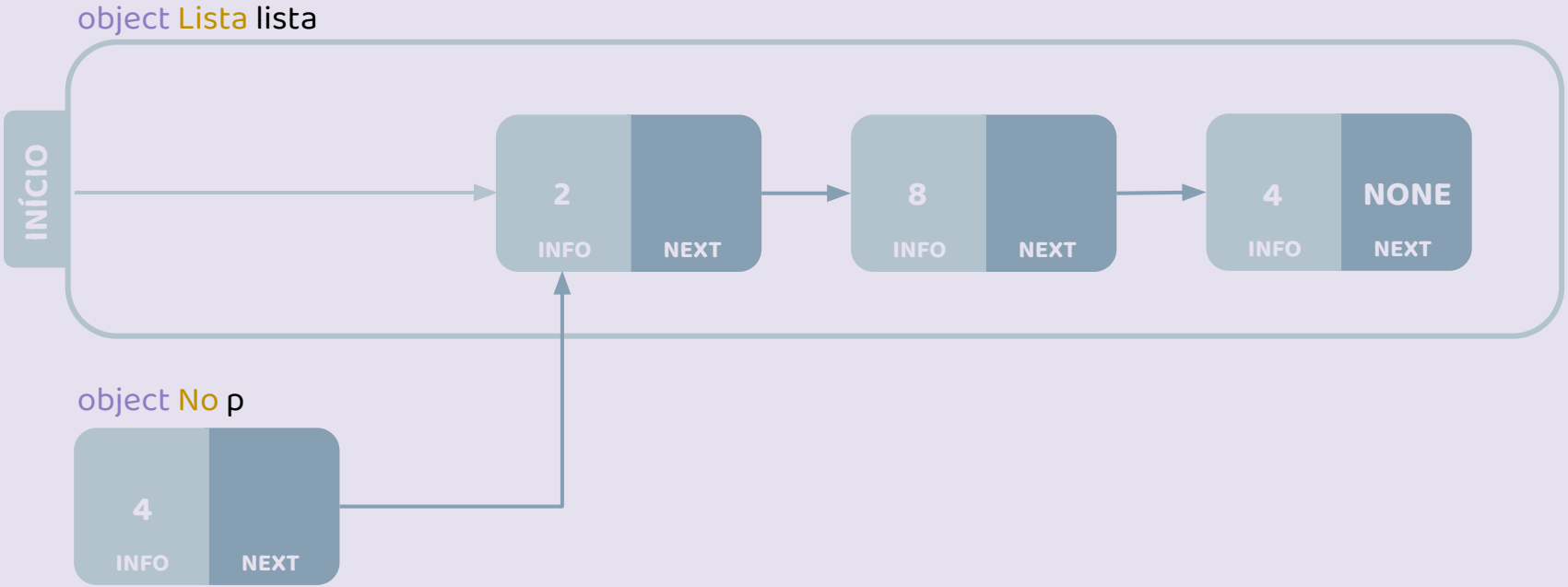
Listas - Remoção do Início



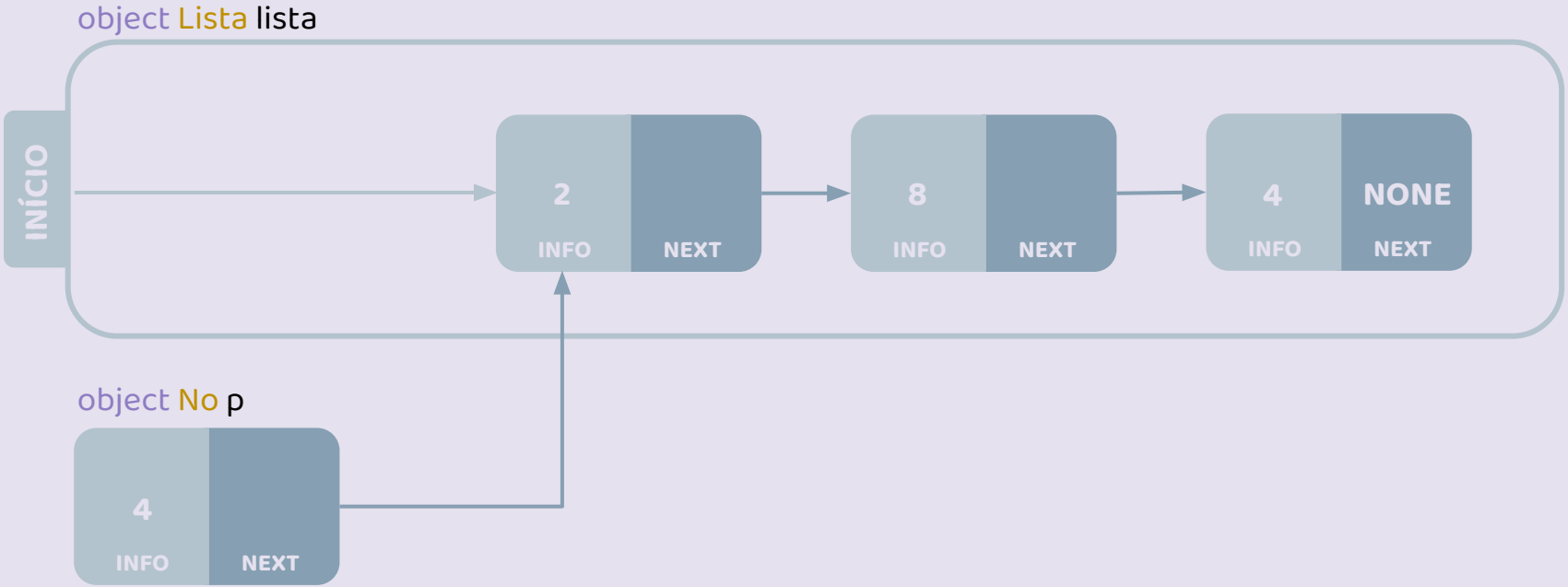
Listas - Remoção do Início



Listas - Remoção do Início

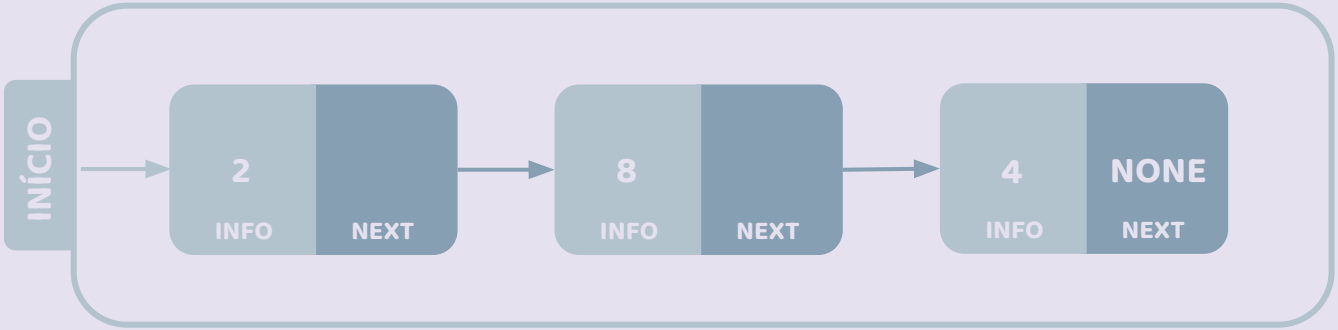


Listas - Remoção do Início



Listas - Remoção do Início

object Lista lista



info = 4

Linguagens de Programação

Ins./Rem. no Fim

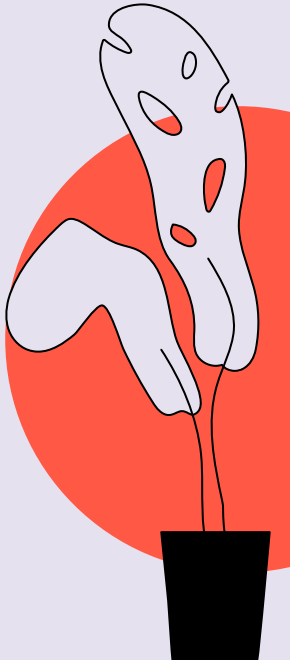
Percorrer toda a lista?
Manter o fim como var.?
Se manter como atualizar?
E se inserir na posição x?

Lista ordenada

Como adicionar?
Como remover?

Como iterar sobre a lista? (Iterator)

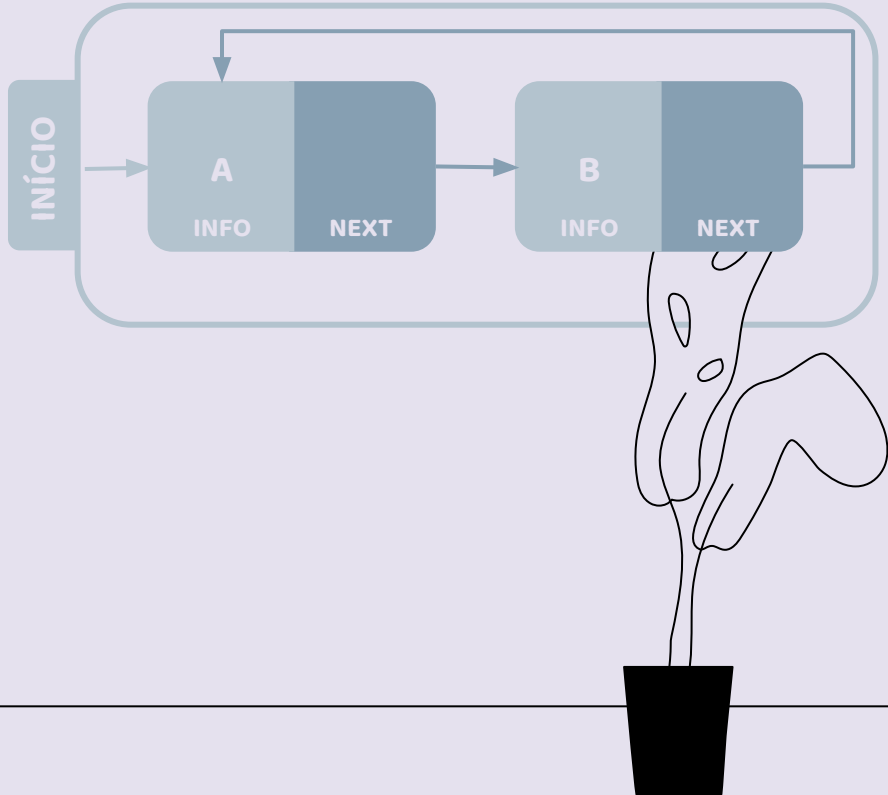
Vamos implementar!



02
Lista Circular

Listas Circular

- Listas simples pode apresentar problemas em percorrer do fim ao início;
- Para isso podemos referência o primeiro nó da Lista como **p.prox** do último nó;
- Remoção é alterada;
- Inserção se mantém;

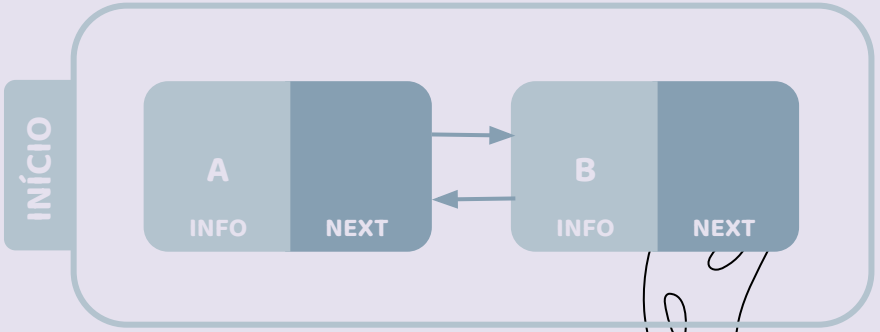


03

Lista Dupla

Listas Duplas

- Também chamadas de Listas Duplamente Encadeadas;
- Nas listas simples, temos dificuldade em encontrar o nó **anterior** ao atual;
- Também podem ser circular;
- Maior custo de memória;
- Simplifica processo de remoção;



Exercício

Crie uma lista ordenada e duplamente encadeada que possua os seguintes métodos:

- inserir;
- remover;
- imprimir_lista;
- procurar_iesimo;

Depois adicione os seguintes valores à lista: **5**,**3**,**4**,**2**.
Depois remova o **2º** valor e adicione o valor **6**;
Por fim imprima a lista.

O resultado da impressão foi **[2, 4, 5, 6]**?