Live de Python #28

Collections #1 - Deque e Namedtuple

Nome:
Eduardo Mendes

Email:
mendesxeduardo@gmail.com

GIT:
github.com/z4r4tu5tr4

Deque

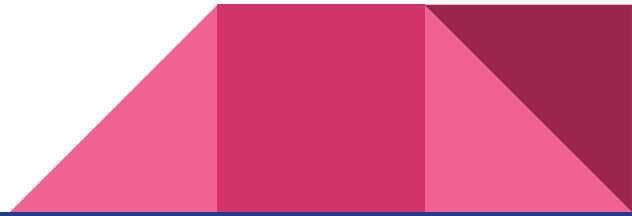
Afinal, o que é uma deque?

- Sequência mutável
- Uma generalização entre uma lista/pilha/fila
- MAAAAAAAS, é uma lista
  - Como assim?

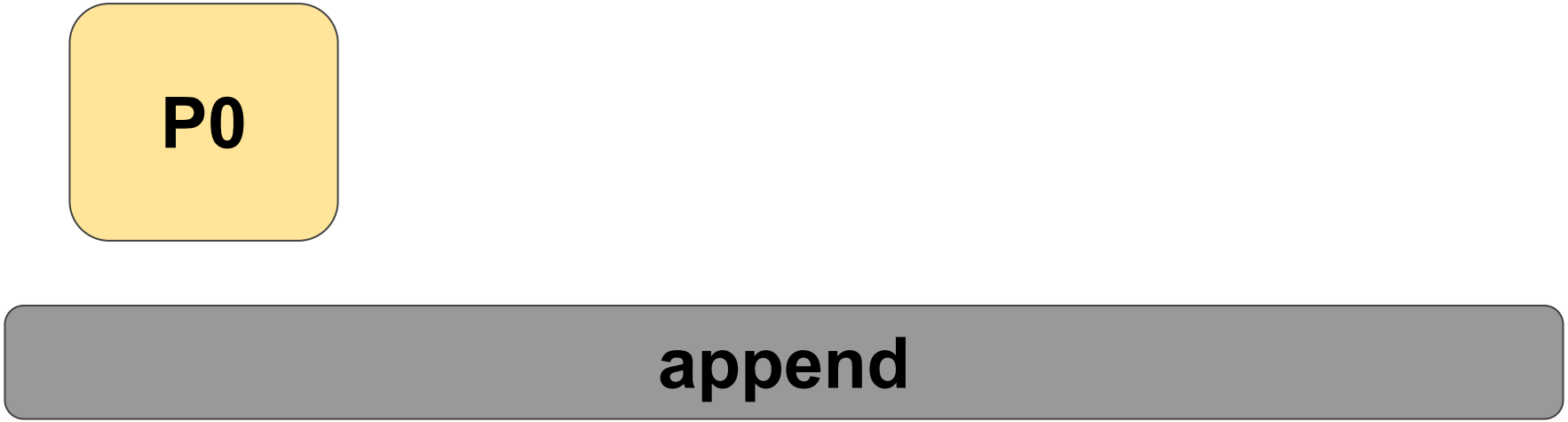
- Ela é uma lista mais legal que uma lista, saca?

Então, no princípio, antes de tudo, fez-se a lista

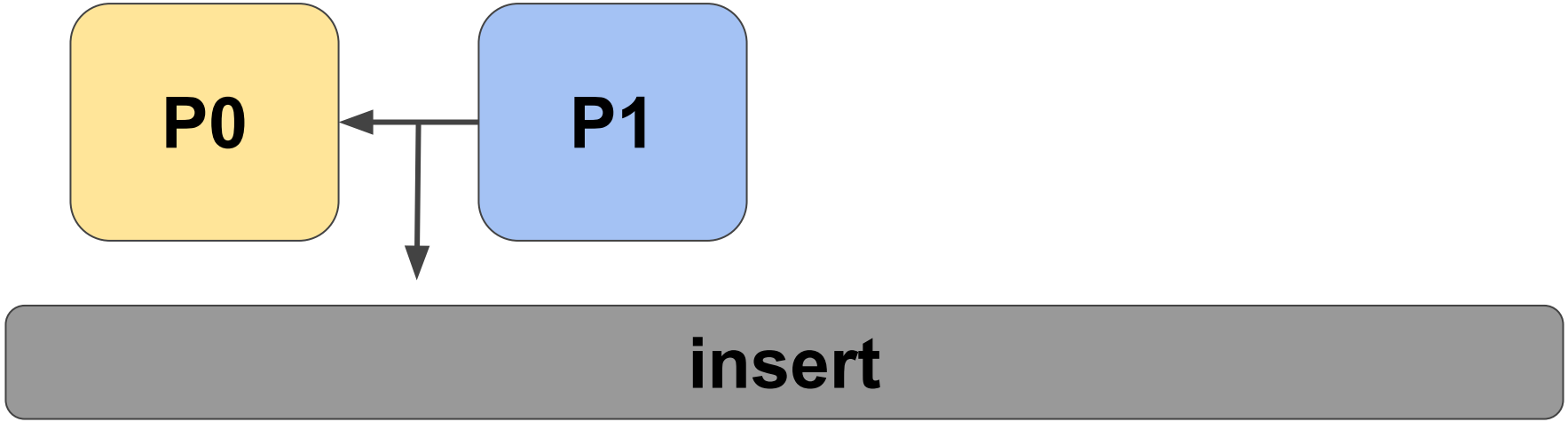
- Sequência mutável
- Acessível por index/slice
- Aceita resignação em qualquer posição
- Têm MUITOS métodos
- Mas só vamos ver os necessários para entender a deque



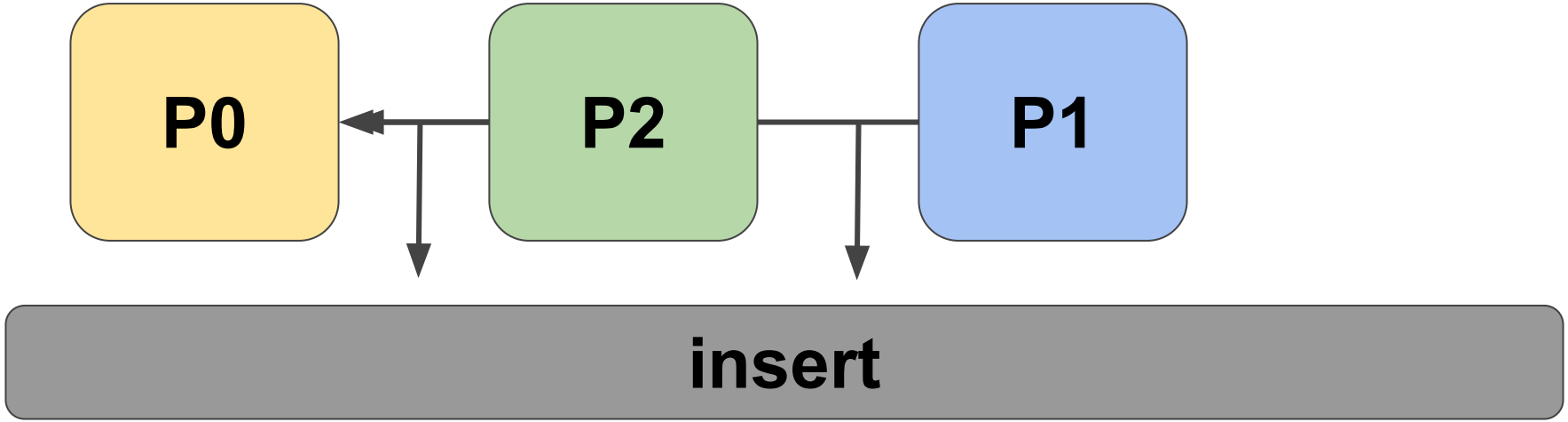
Lista ilustrada [0]



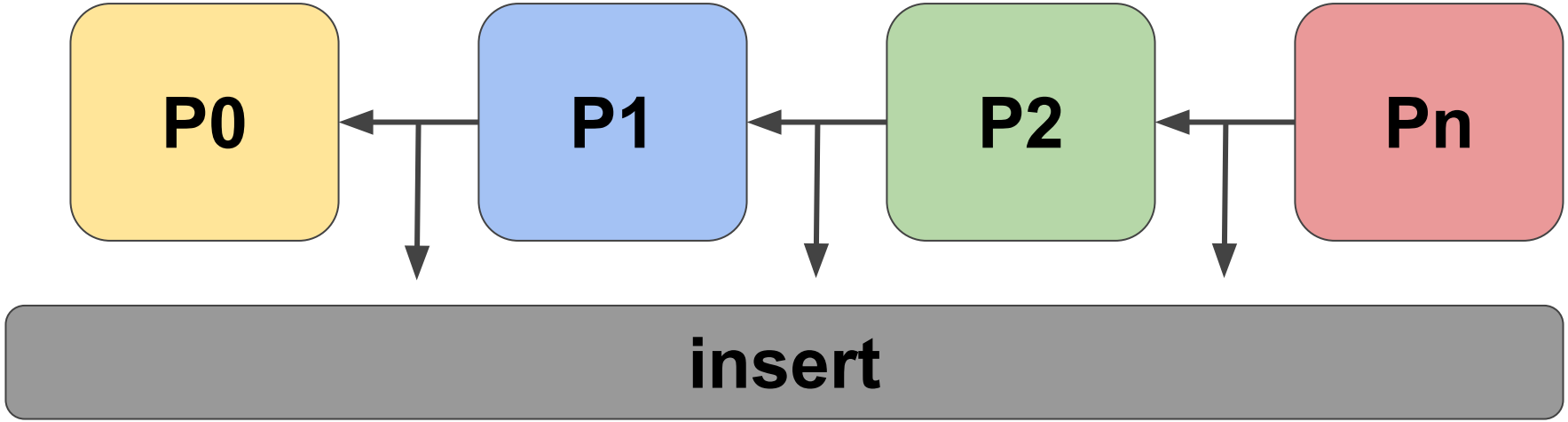
Lista ilustrada [1]



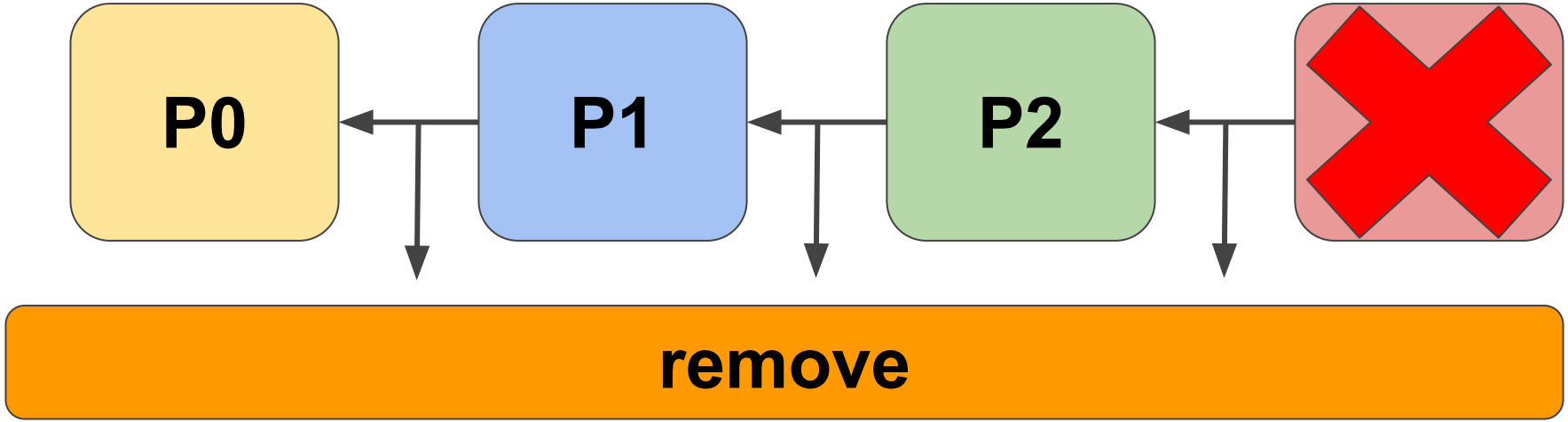
Lista ilustrada [2]



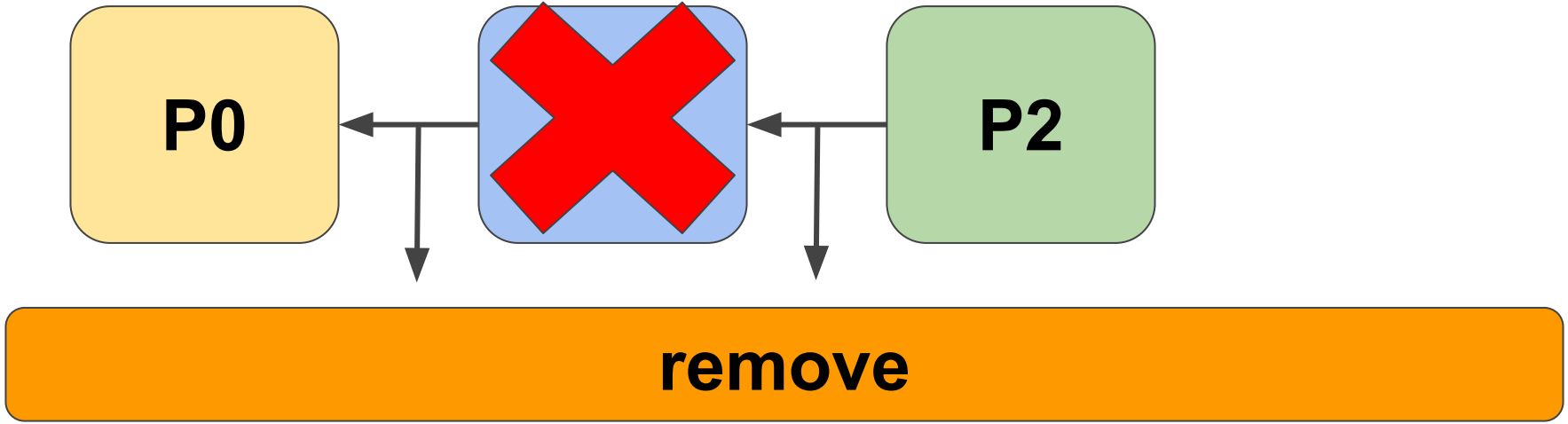
Lista ilustrada [3]



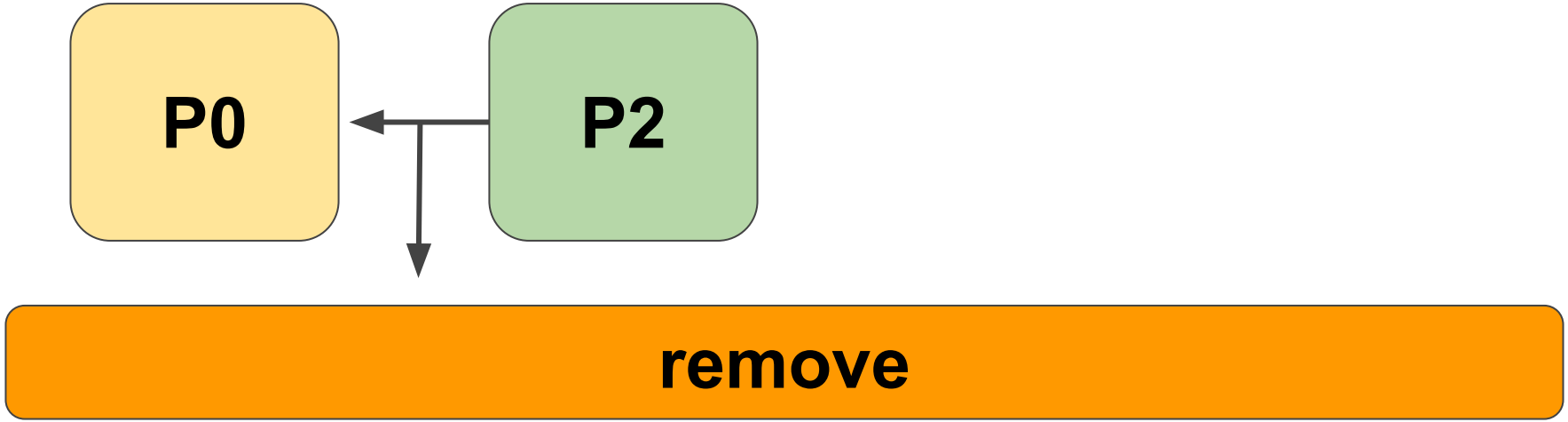
Lista ilustrada [4]



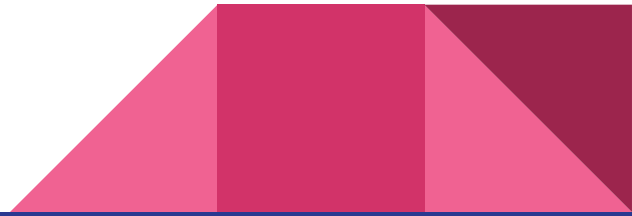
Lista ilustrada [5]



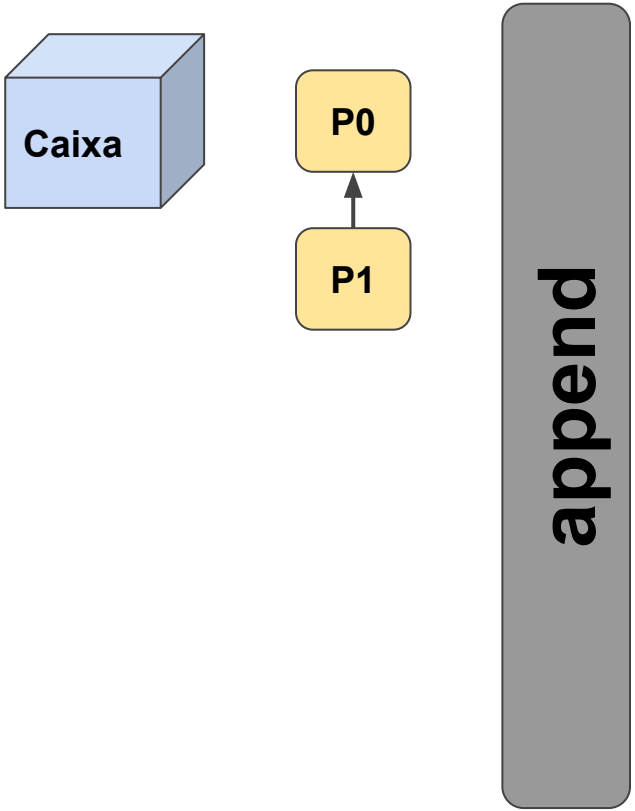
Lista ilustrada [6]



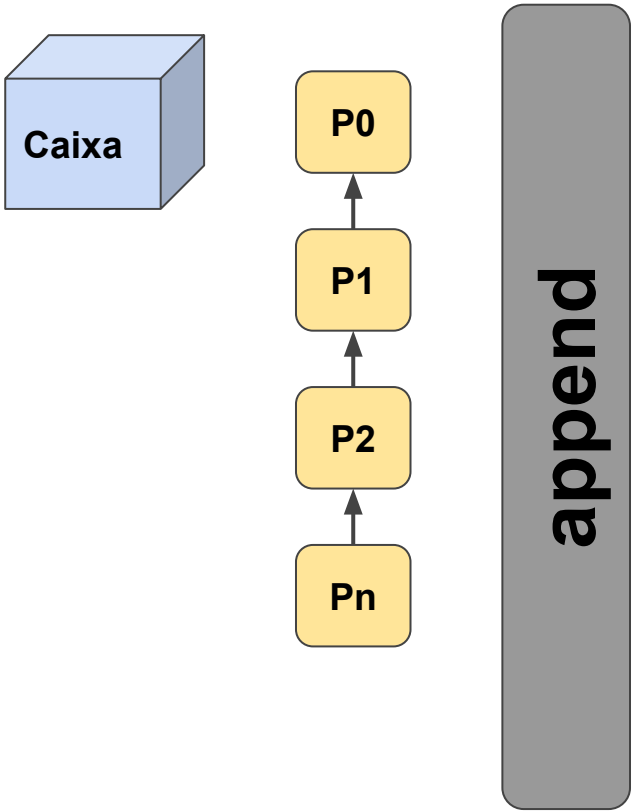
Código



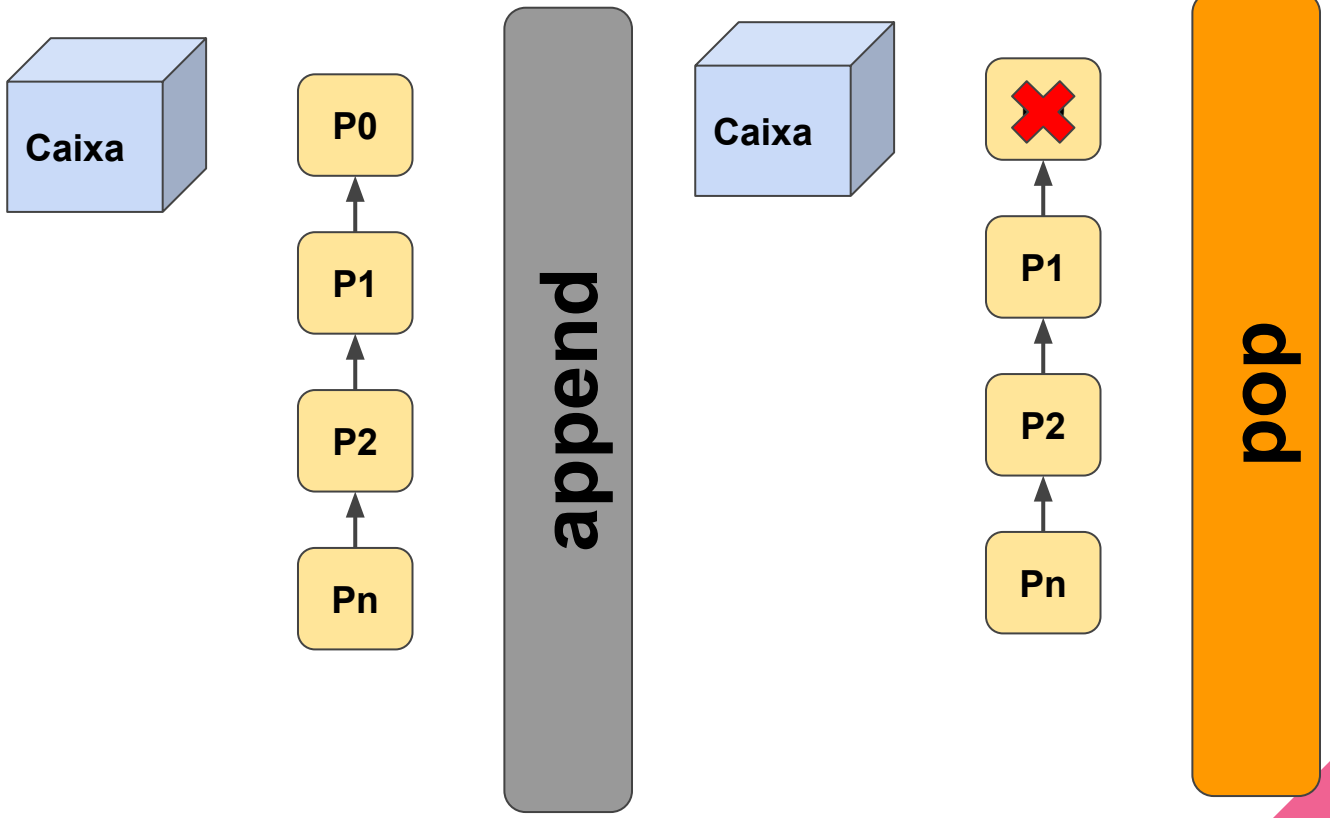
Fila ilustrada [0]



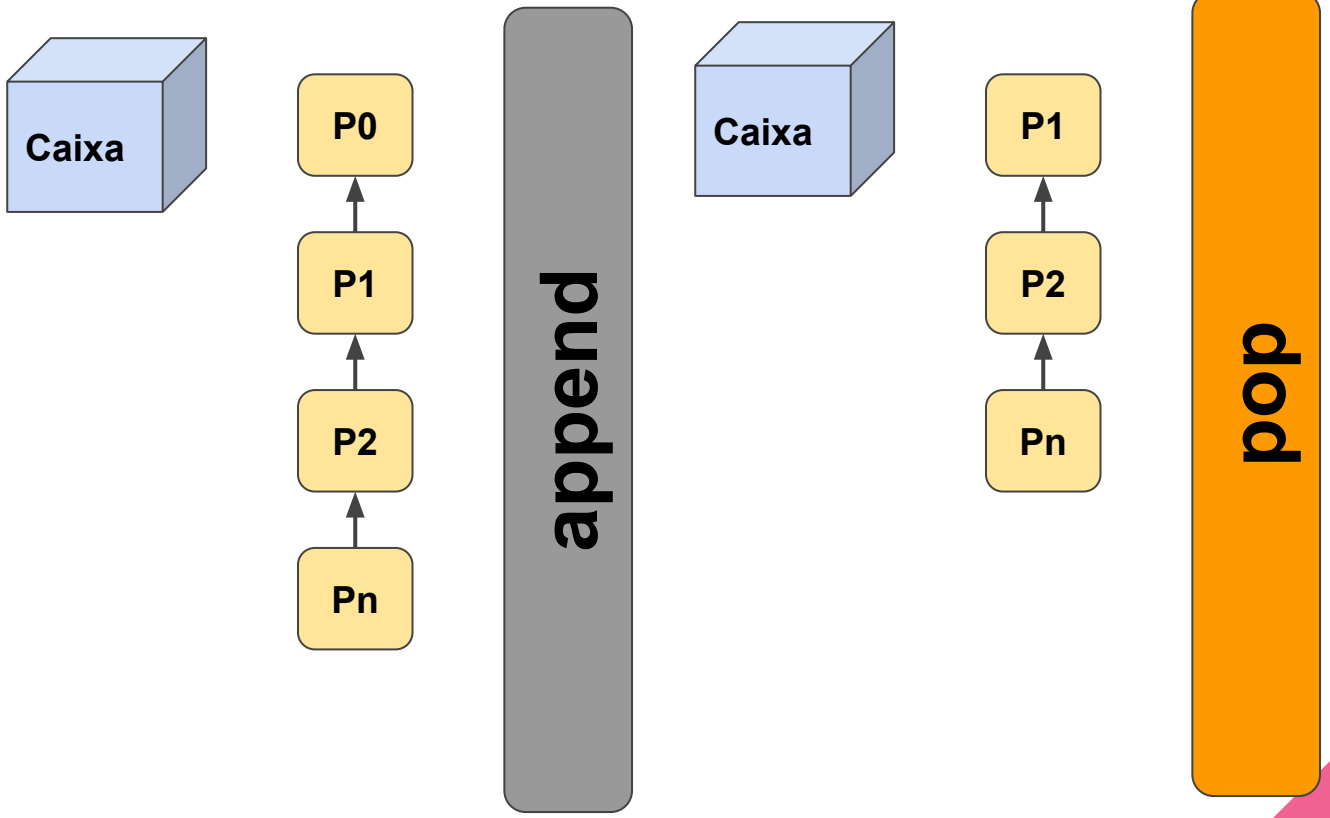
Fila ilustrada [1]



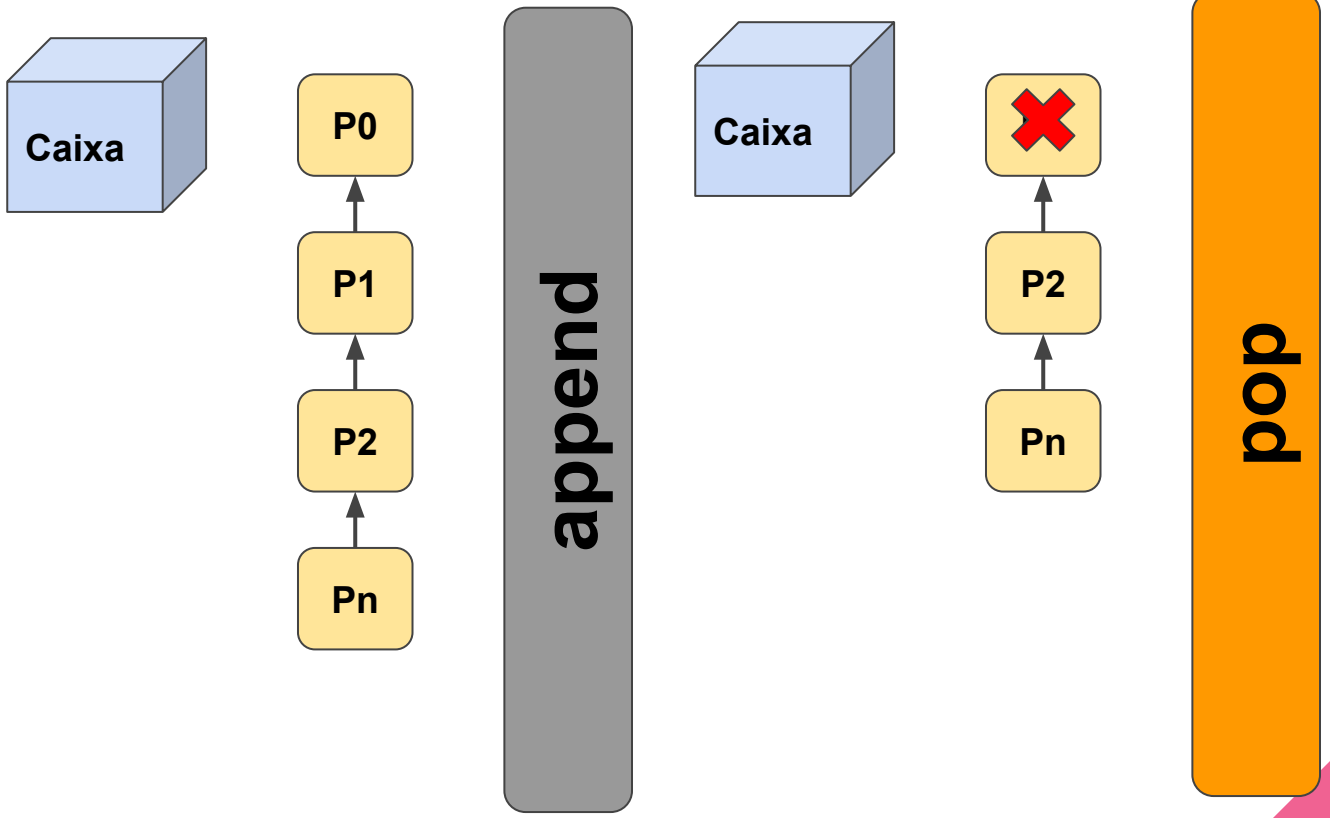
Fila ilustrada [2]



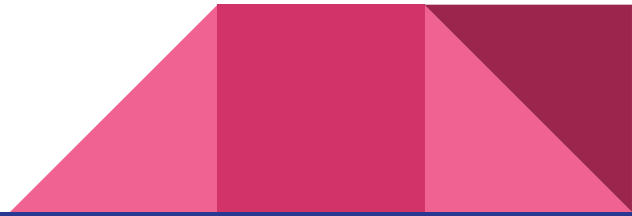
Fila ilustrada [3]



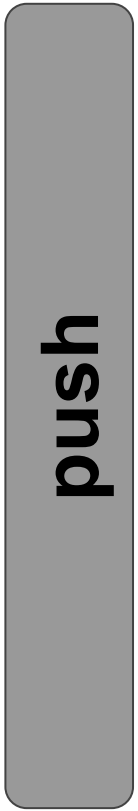
Fila ilustrada [4]

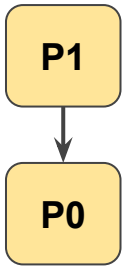


Código

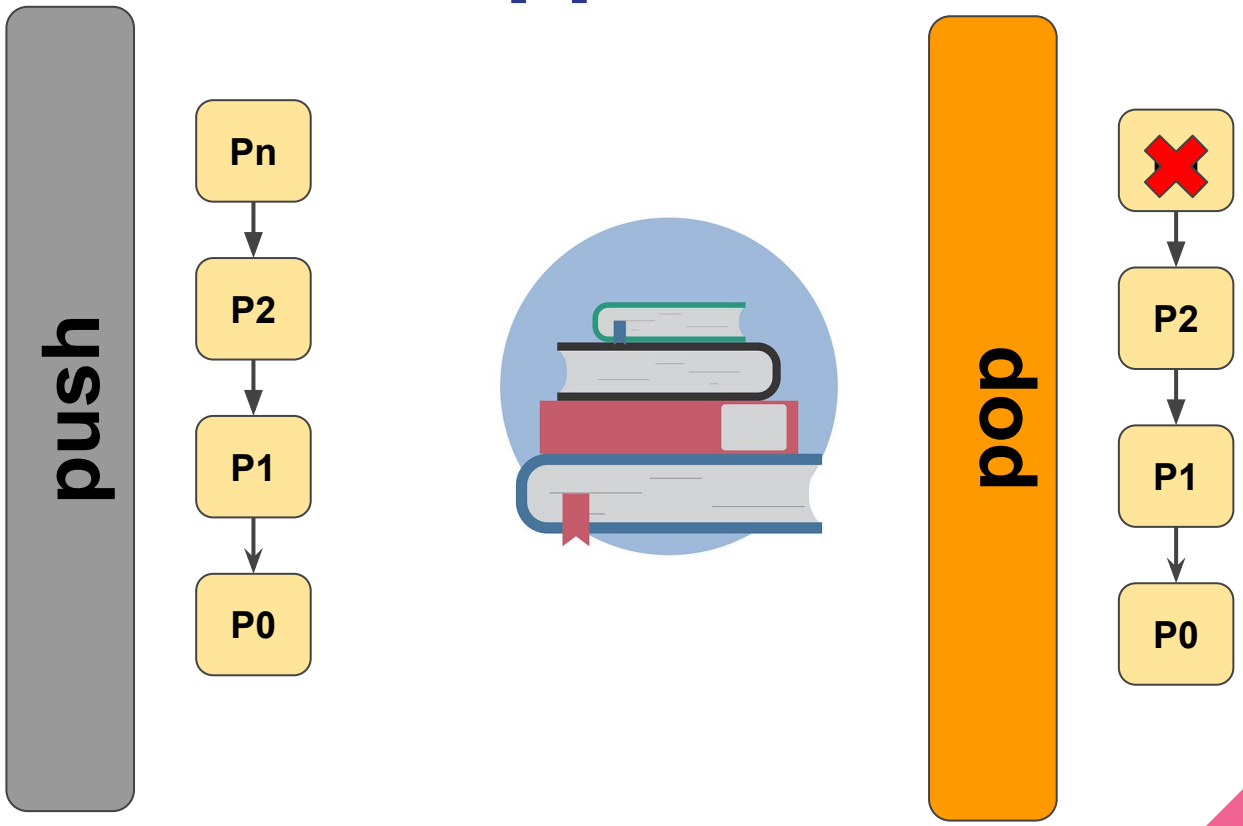


Pilha ilustrada [0]

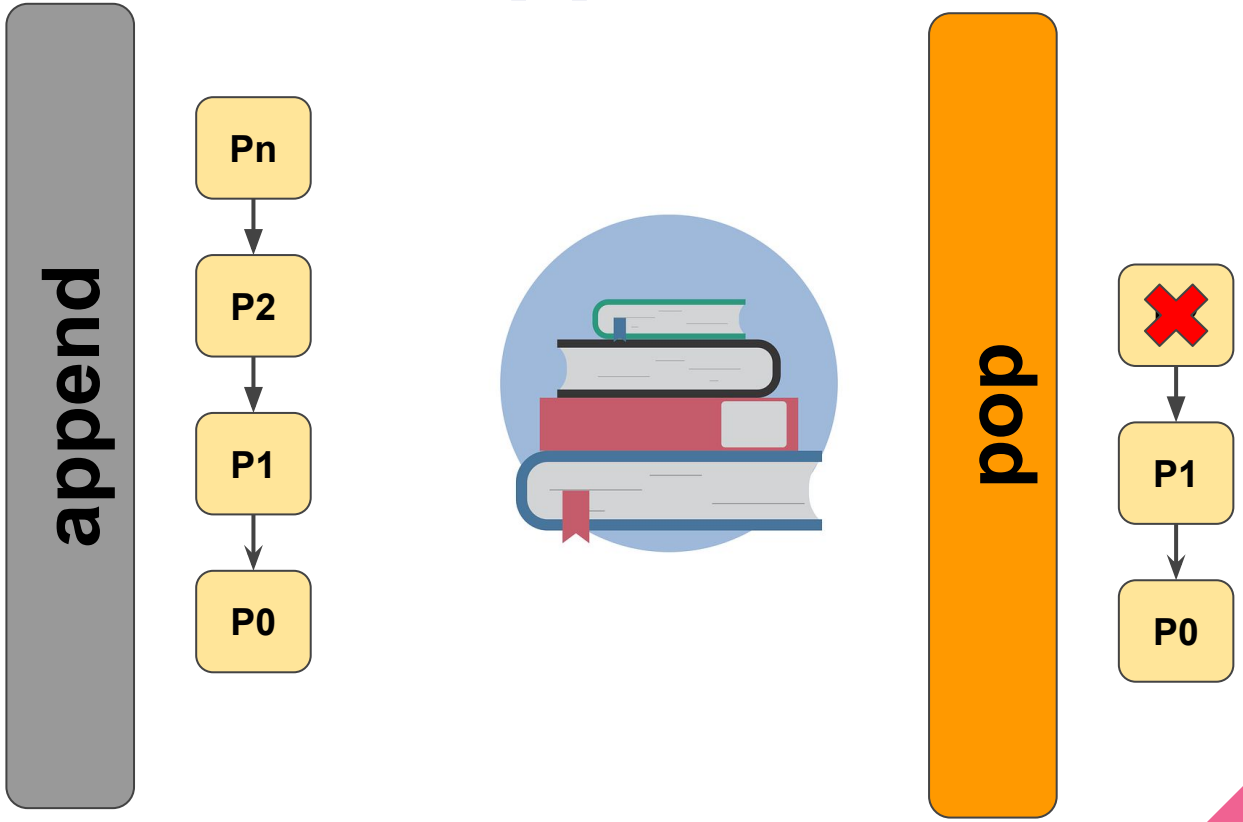




Pilha ilustrada [1]

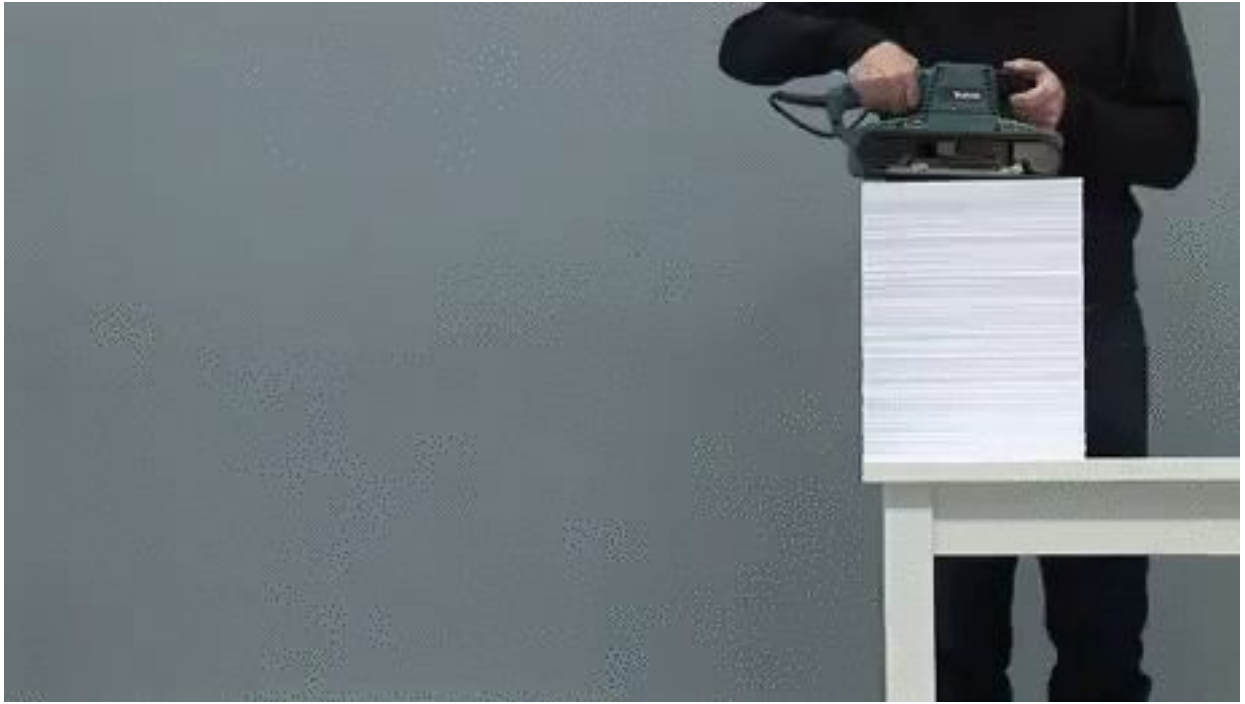


Pilha ilustrada [2]

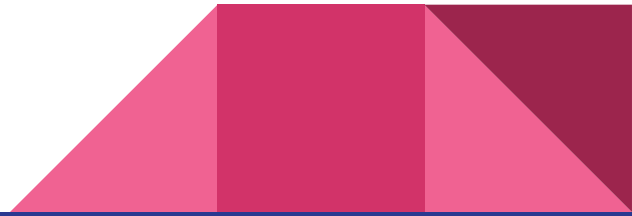


Tipo isso

Ou isso



Código



Afinal, o que é uma deque?

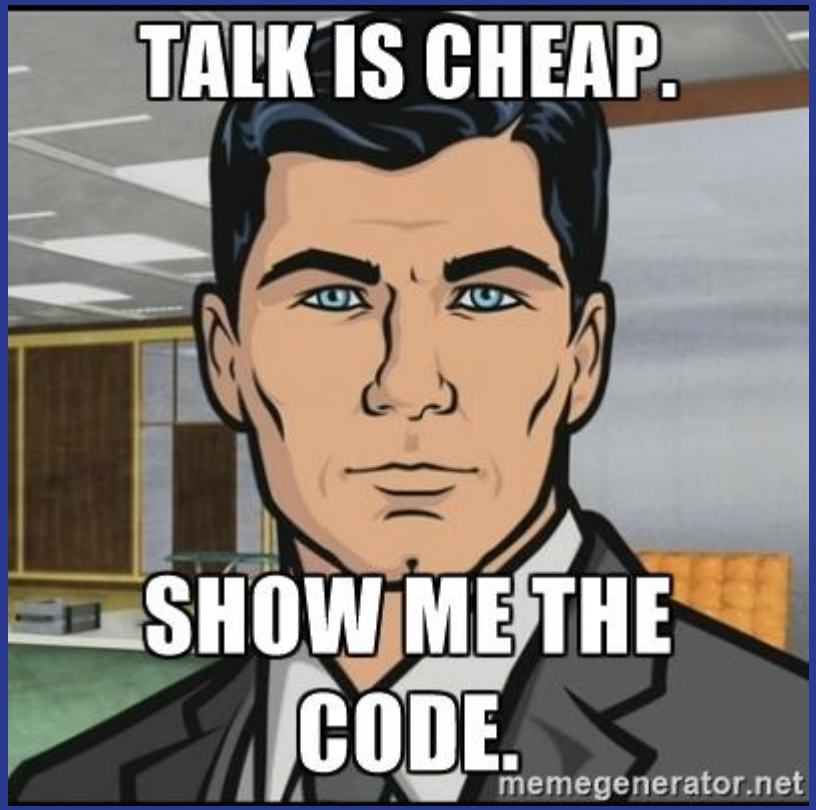
- Uma:
  - lista
  - pilha
  - fila
- Isso mesmo, tudo junto

Hã????

- Você pode:
  - Adicionar do lado que quiser
  - Remover do lado que quiser
  - Definir um tamanho máximo
  - Rotacionar a lista
    - Não é uma implementação cíclica

É tipo isso

MAAAAAAAAA
AAAAAAAAAAA
SSSSSSSS



Namedtuple

O que é uma tupla em python?

- Sequência imutável
  - Não aceita resignação de novos itens após sua definição

```
In: t = (1,2,3,4)

In: t[0] = 5

TypeError: 'tuple' object does not support item assignment
```

Tuplas são acessíveis via index/slice

```
In: t = (1, 2, 3, 4)

In: t[0]

Out: 1
```

```
In: t = (1, 2, 3, 4)

In: t[0:3]

Out: (1, 2, 3)
```

```
In: t = (1, 2, 3, 4)

In: t[::2]

Out: (1, 3)
```

Tuplas executam dois métodos

```
In: t = (1, 2, 3, 4)

In: t.count(1)

Out: 1
```

```
In: t = (1, 2, 3, 4)

In: t.index(3)

Out: 2
```

Então, o que é uma namedtuple?

- Sequência imutável
  - Não aceita resignação de novos itens após sua definição
  - Os elementos estão acessíveis como atributos

```
In: n_t = namedtuple('jogador', ['nome','time','camisa'])

In: n_t('Ronaldo', 'Brasil', 9)

Out: jogador(nome='Ronaldo', time='Brasil', camisa=9)
```

São acessíveis via index/slice/atributo

```
In: j = n_t('Ronaldo', 'Brasil', 9)
```

```
In: j[0]

Out: ‘Ronaldo’
```

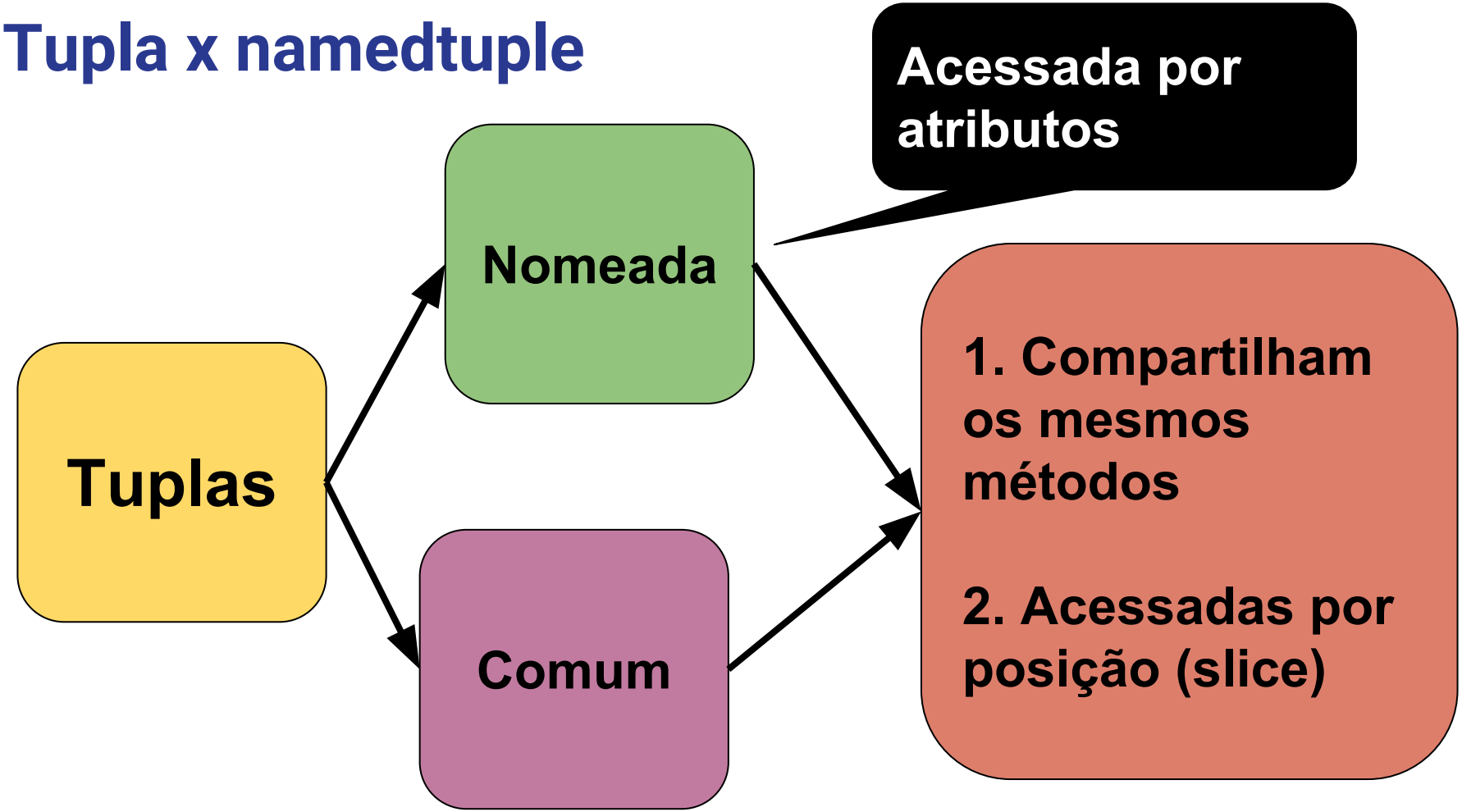
```
In: j[::2]

Out: (‘Ronaldo’, 9)
```

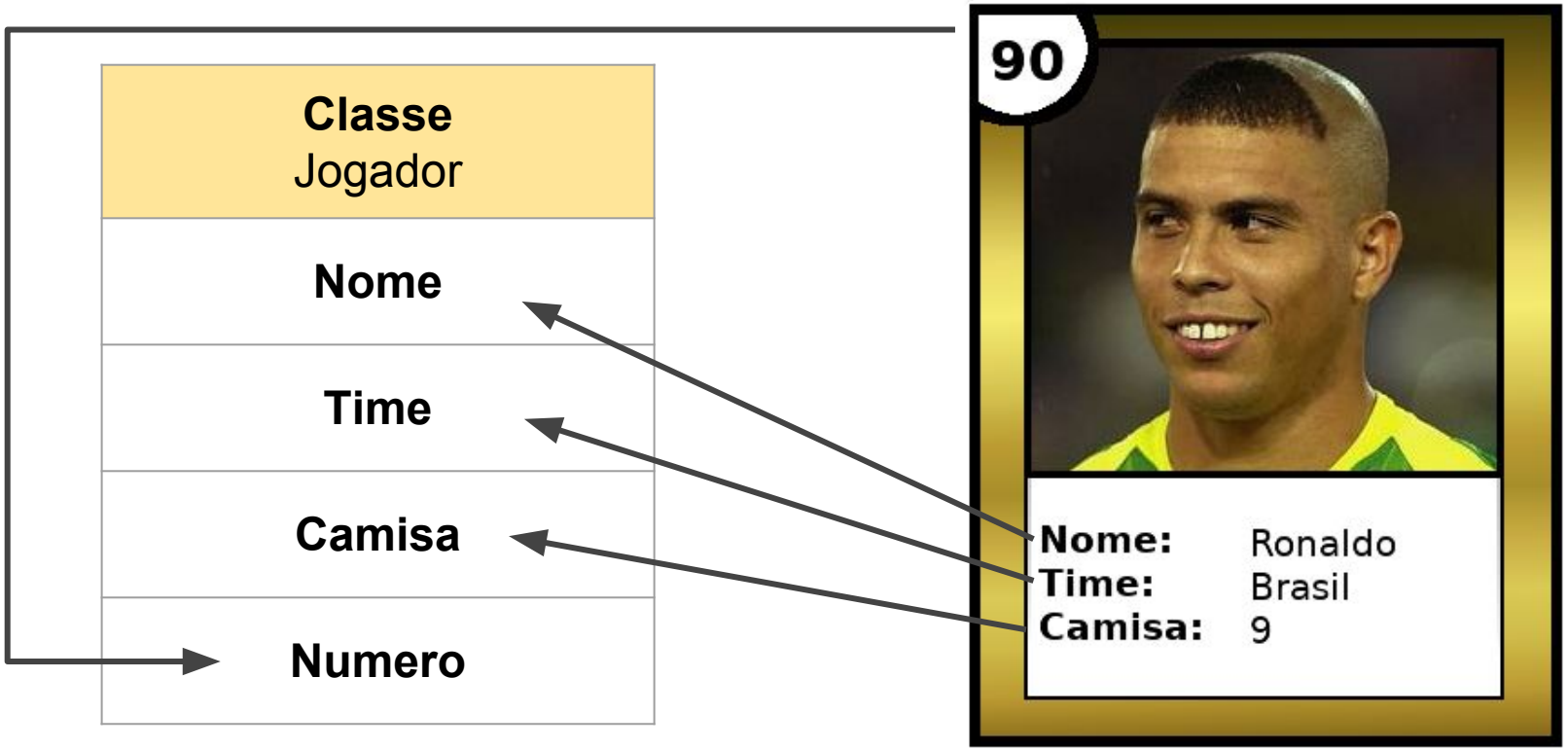
```
In: j.nome

Out: ‘Ronaldo’
```

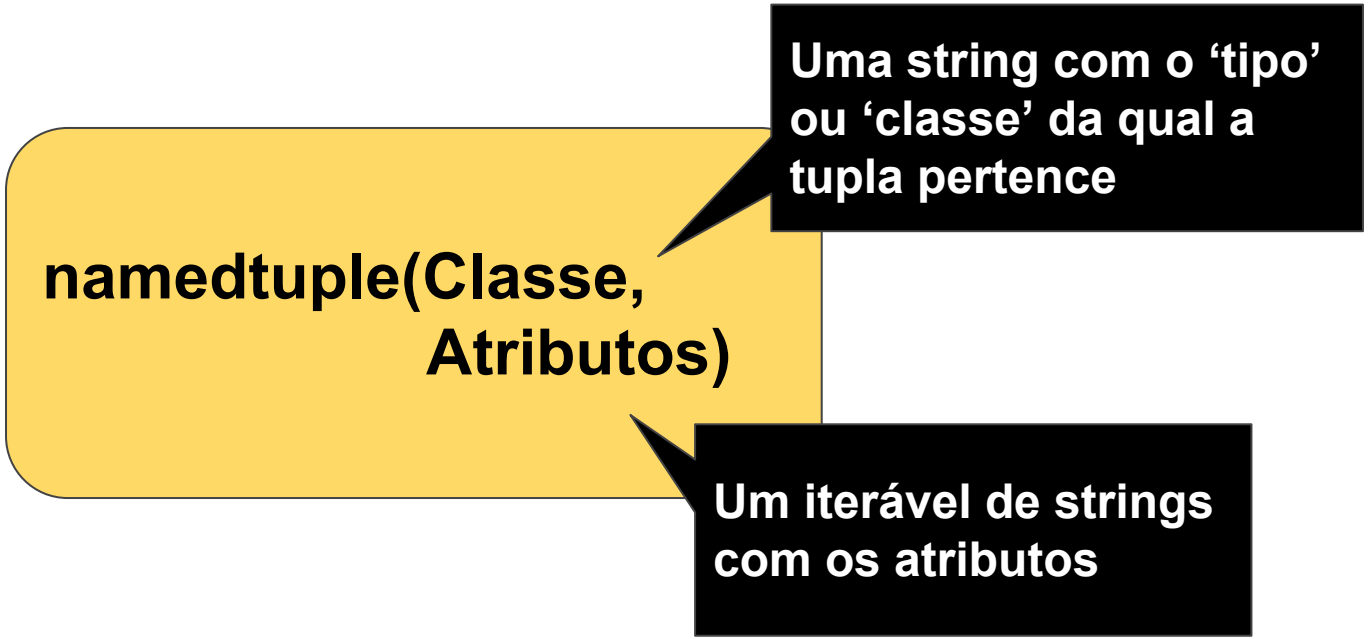
Tupla x namedtuple



Exemplo: Figurinhas da copa [0]

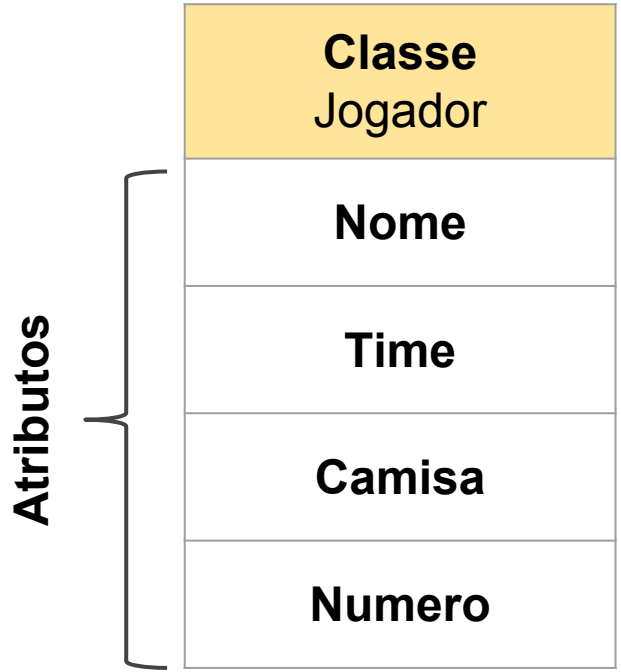


Inicializando uma namedtuple



Exemplo: Figurinhas da copa [1]

```
namedtuple('jogador', ['nome',
                       'time'
                       'camisa',
                       'Numero’])
```



Exemplo: Figurinhas da copa [2]

Classe Jogador	
Nome	Ronaldo
Time	Brasil
Camisa	9
Numero	90

Exemplo: Figurinhas da copa [3]

```
jogador = t('Ronaldo,
            'Brasil',
            9,
            90])
```

Classe Jogador	
Nome	Ronaldo
Time	Brasil
Camisa	9
Numero	90

MAAAAAAAAAAAAASSSSS, espera aí...

Posso fazer tudo isso com classes, não?

SIM
SIM
SIM
SIM
SIM
SIM

```python
class jogador:
    def __init__(self, nome, time, n):
        self.nome = nome
        self.time = time
        self.n = n
        self._t = (self.nome, self.time, self.n)

    def __repr__(self):
        return 'Jogador(nome={}, time={}, n={})'.format(self.nome,
                                                        self.time,
                                                        self.n)


    def __getitem__(self, n):
        return self._t[n]

    def __len__(self):
        return len(self._j)

    def count(self, att):
        return len([x for x in self._t if x == att])

    def index(self, att):
        for x, y in enumerate(self._t):
            if y == att:
                return x

```

SIM
SIM
SIM
SIM
SIM
SIM

MAAAAAAAAAAAAASSSSS,

eu prefiro assim...

```
1    from collections import namedtuple
2
3    jogador = namedtuple('jogador', 'Nome Time Camisa Numero'. split())
4
```