# Estrutura de dados
## Listas encadeadas

Profª. Drª.  Alana Morais

# Recapitulando - Estrutura de dados

**Dados simples:**

- padrão:
  - inteiro (int);
  - real (float);
  - caracter (str);
  - lógico (boolean).

**Dados estruturados:**

- Estáticos:
  - arrays;
  - registros;
  - arquivos;
  - conjuntos;
  - cadeias.
- Dinâmicos:
  - filas;
  - pilhas;
  - listas encadeadas;
  - árvores;
  - grafos.

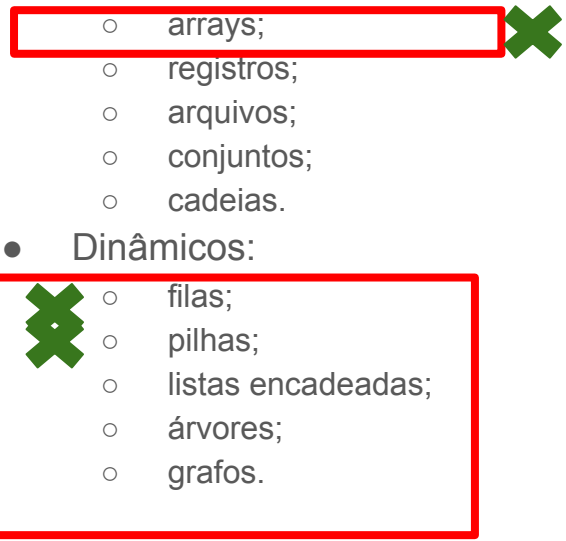
# Recapitulando - Estrutura de dados

**Dados simples:**

- padrão:
  - inteiro (int);
  - real (float);
  - caracter (str);
  - lógico (boolean).

**Dados estruturados:**

- Estáticos:
  - arrays;
  - registros;
  - arquivos;
  - conjuntos;
  - cadeias.
- Dinâmicos:
  - filas;
  - pilhas;
  - listas encadeadas;
  - árvores;
  - grafos.



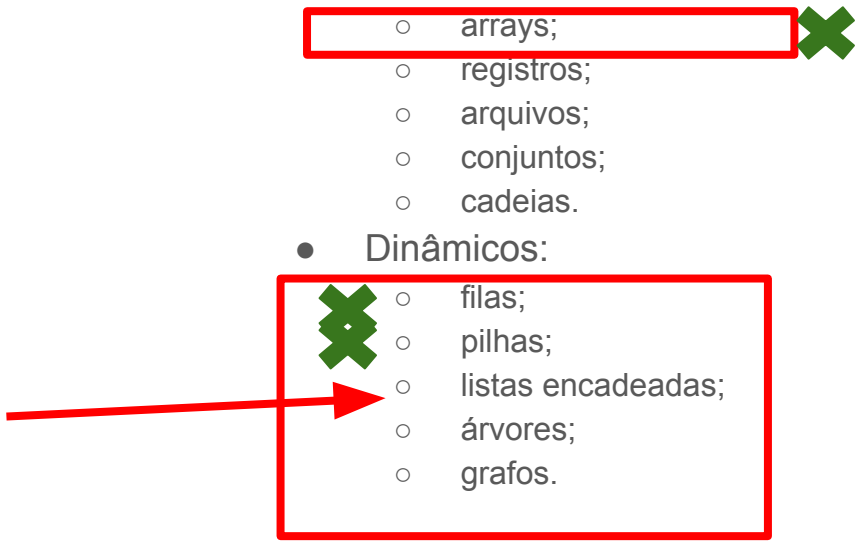
# Recapitulando - Estrutura de dados

**Dados simples:**

- padrão:
  - inteiro (int);
  - real (float);
  - caracter (str);
  - lógico (boolean).

**Dados estruturados:**

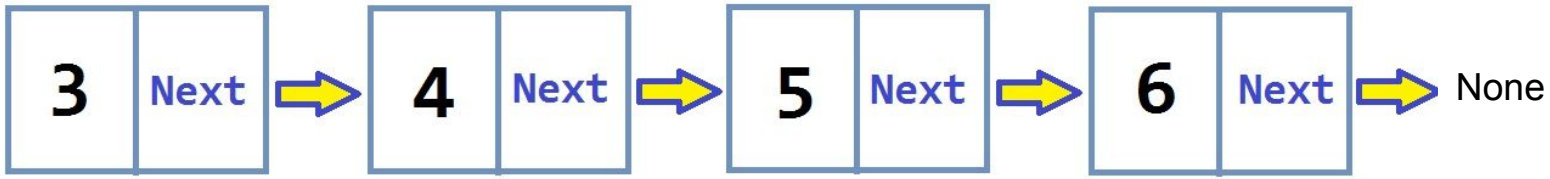
- Estáticos:
  - arrays;
  - registros;
  - arquivos;
  - conjuntos;
  - cadeias.
- Dinâmicos:
  - filas;
  - pilhas;
  - listas encadeadas;
  - árvores;
  - grafos.



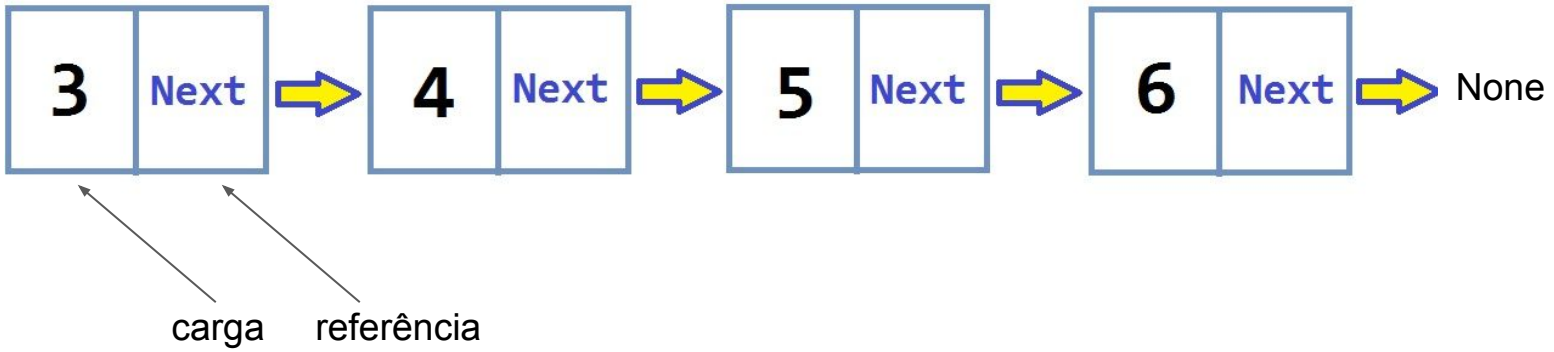
# Lista Encadeada

- Também conhecida como **lista ligada**;
- Uma lista vazia, representada por *None*;
- Um nó que contém um objeto carga e uma referência para uma lista ligada.

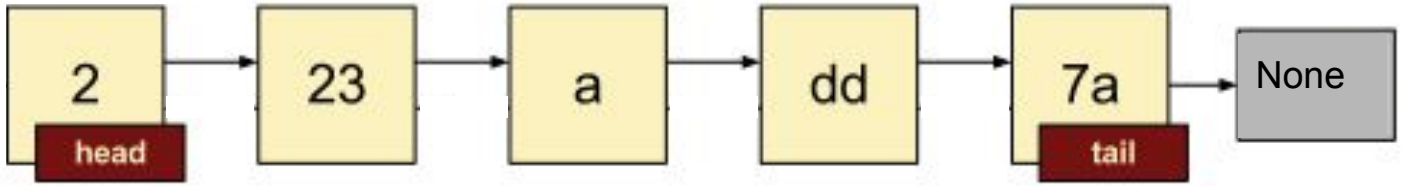
# Lista Encadeada



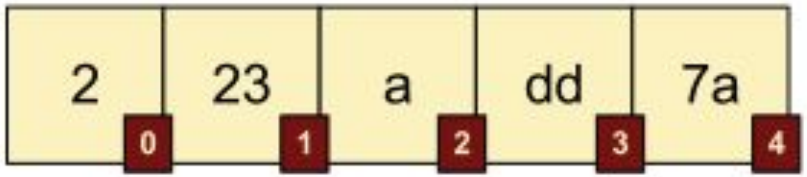
# Lista Encadeada



## Lista encadeada



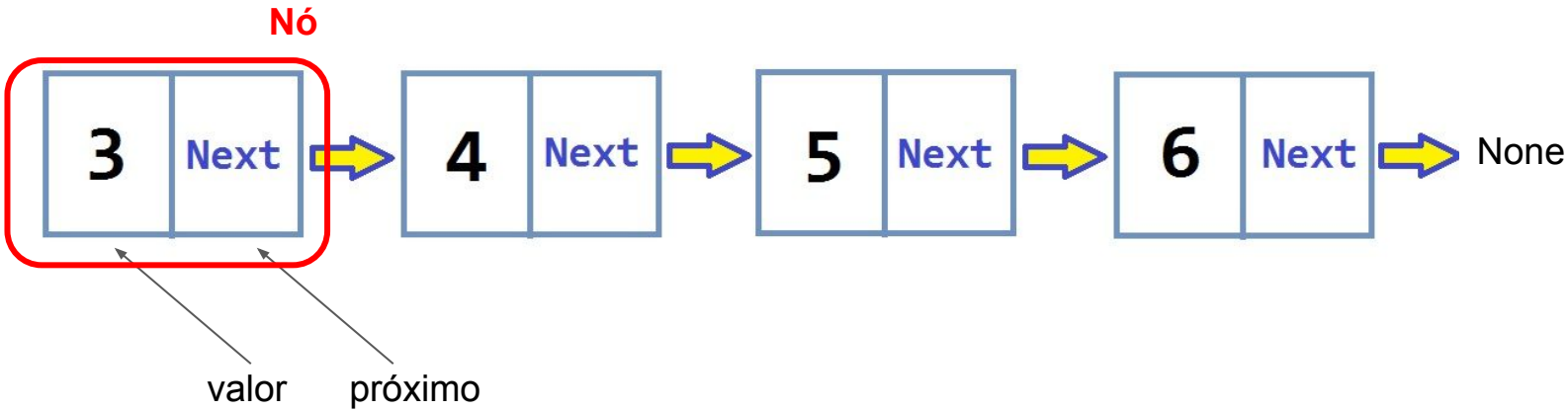

## Fila



# Operações com lista encadeada

- **Criação**;
- Inserção de elementos na lista encadeada;
- Mostrar elementos da lista encadeada;
- Remoção de elementos na lista encadeada;
- Busca na lista encadeada;

# Criação de uma lista encadeada - forma 1- Nó



# Criação de uma lista encadeada- forma 1- Nó

node.py

```
class No(object):
    def __init__(self):
        self.valor = None
        self.proximo = None
```

main.py

```
import node

def main():
    no1 = node.No()
    no2 = node.No()
    no3 = node.No()
```

# Criação de uma lista encadeada- forma 1- Nó

node.py

```
class No(object):
    def __init__(self):
        self.valor = None
        self.proximo = None

    def getValor(self):
        return self.valor

    def getProximo(self):
        return self.proximo
```

main.py

```
import node

def main():
    no1 = node.No()
    no2 = node.No()
    no3 = node.No()
```

# Criação de uma lista encadeada- forma 1- Nó

node.py

```
class No(object):
    ...
    def setValor(self, valor):
        self.valor   = valor

    def setProximo(self, prox):
         self.proximo = prox
```

main.py

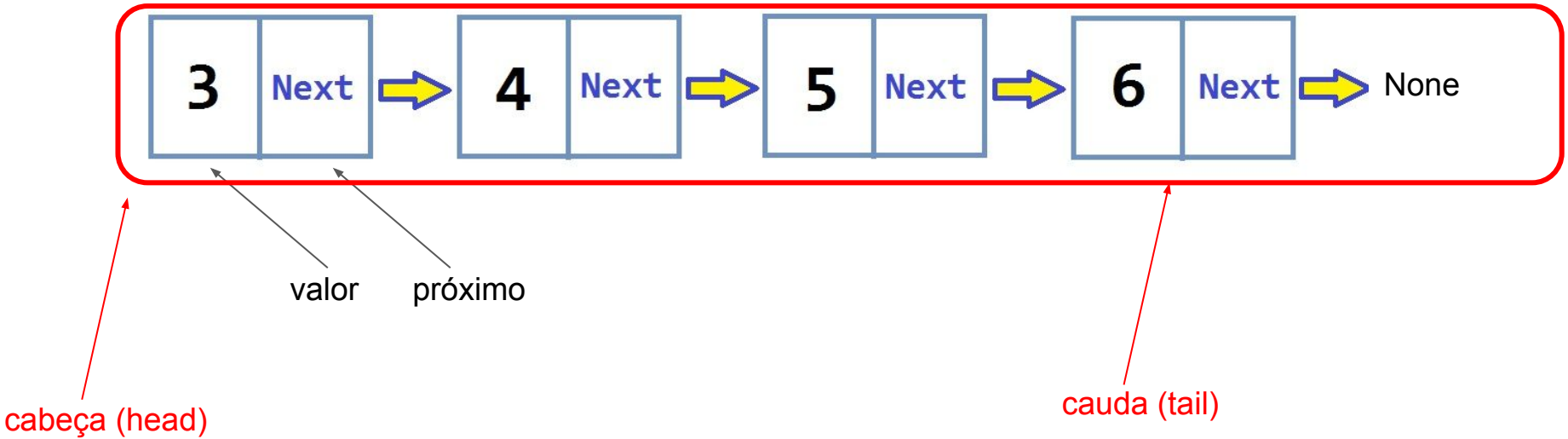
```
import node

def main():
     no1 = node.No()
     no2 = node.No()
     no3 = node.No()
```

# Exercício A

Implemente o método de criação de dois nós, utilizando uma classe chamada node.

# Criação de uma lista encadeada- forma 2- Lista



# Criação de uma lista encadeada- forma 2- Lista

node.py

```
class No:
   def __init__(self):
       self.valor = None
       self.proximo = None
```

# Criação de uma lista encadeada- forma 2- Lista

listaencadeada.py

```
import node

class ListaEncadeada(object):
    def __init__(self):
        self.cabeça = None
        self.cauda = None
```

main.py

```
import listaencadeada

def main():
    lista = listaencadeada.ListaEncadeada ()

main()
```

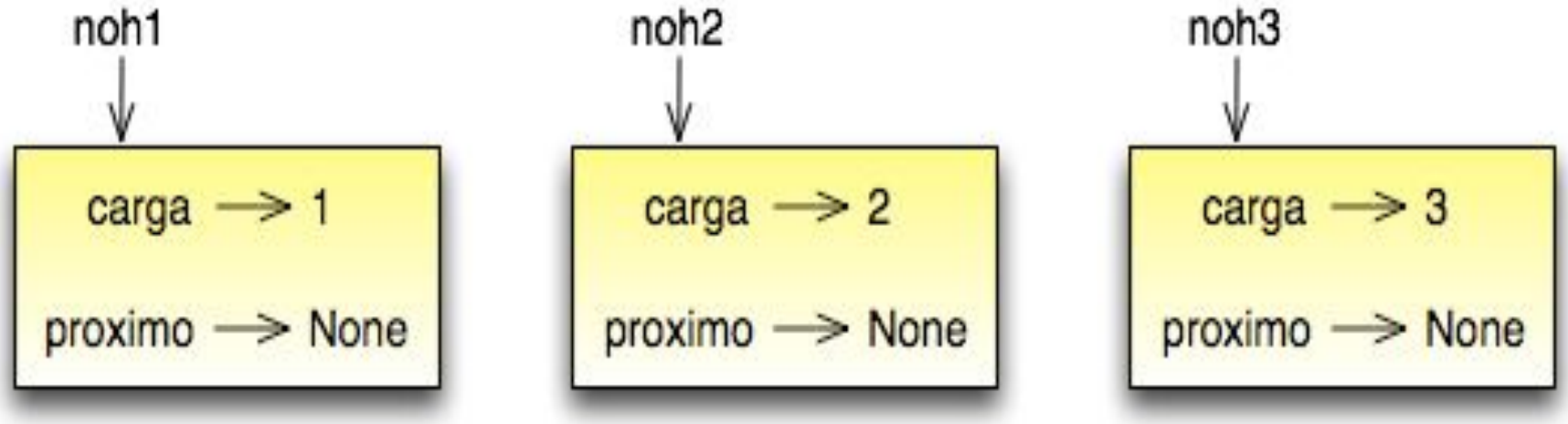
# Exercício B

Implemente o método de criação de duas listas encadeadas, utilizando uma classe chamada listaEncadeada.

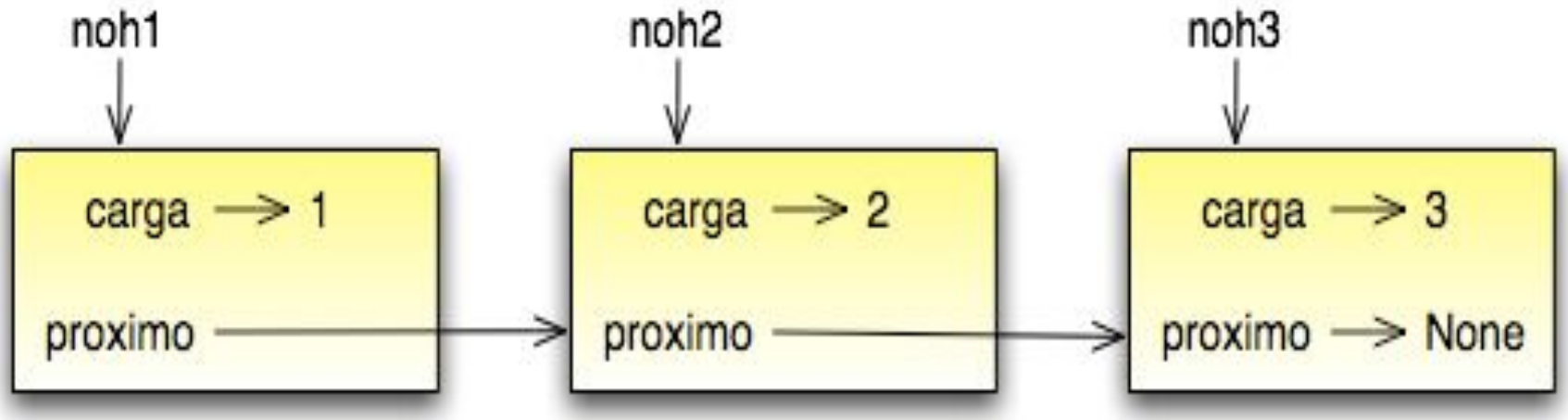
# Operações com lista encadeada

- Criação;
- **Inserção de elementos na lista encadeada;**
- Mostrar elementos da lista encadeada;
- Remoção de elementos na lista encadeada;
- Busca na lista encadeada;

# Nós


noh1
carga → 1
proximo → None
noh2
carga → 2
proximo → None
noh3
carga → 3
proximo → None

# Como fazer para ligar os nós?



# Inserção de valores na lista encadeada - Nó

node.py

```
class No (object):
    def __init__(self):
        self.valor = None
        self.proximo = None

    def inserirValor(self, valorNovo):
        self.valor = valorNovo

    def inserirProximo(self, proximoNovo):
        self.proximo = proximoNovo
```

# Inserção de valores na lista encadeada - Nó

node.py

```
class No (object):
    def __init__(self):
        self.valor = None
        self.proximo = None

    def inserirValor(self, valorNovo):
        self.valor = valorNovo

    def inserirProximo(self, proximoNovo):
        self.proximo = proximoNovo
```

main.py

```
import node

def main():
    no1 = node.No()
    no2 = node.No()

    no1.inserirValor("Aline")
    no2.inserirValor("Marques")

main()
```

# Inserção de valores na lista encadeada - Nó

node.py

```
class No (object):
    def __init__(self):
        self.valor = None
        self.proximo = None

    def inserirValor(self, valorNovo):
        self.valor = valorNovo

    def inserirProximo(self,
    proximoNovo):
         self.proximo = proximoNovo
```

main.py

```
import node

def main():
     no1 = node.No()
     no2 = node.No()

     no1.inserirValor("Aline")
     no2.inserirValor("Marques")
     no1.inserirProximo(no2)

main()
```

# Exercício C

Crie uma  lista encadeada, no qual cada nó possui nomes de cores como valor.

# Inserção de valores na lista encadeada - Lista

listaencadeada.py

```
import node

class listaEncadeada(object):
    ...
    def inserirNo(self, val):
        item = node.No()
        item.setValor(val)
        if (self.cabeça == None
                and self.cauda == None):
            self.cabeça = item
            self.cauda = item
```

```
else:
    self.cauda.setProximo(item)
item.setProximo(None)
self.cauda = item
```

# Inserção de valores na lista encadeada - Lista

main.py

```
import listaencadeada

def main():
    lista = listaencadeada.ListaEncadeada()
    lista.inserirNo("valor 1")


main()
```

# Exercício D

Crie duas  listas encadeadas: a primeira composta por números pares e a segunda por números ímpares.

# Operações com lista encadeada

- Criação;
- Inserção de elementos na lista encadeada;
- **Mostrar elementos da lista encadeada;**
- Remoção de elementos na lista encadeada;
- Busca na lista encadeada;

# Mostrar elementos da lista encadeada - Nó

node.py

```
class No (object):
    def __init__(self):
        self.valor = None
        self.proximo = None

    def inserirValor(self, valorNovo):
        self.valor = valorNovo

    def inserirProximo(self, proximoNovo):
        self.proximo = proximoNovo
```

main.py

```
import node

def mostrarLista(nos):
def main():
    no1 = node.No()
    no2 = node.No()

    no1.inserirValor("Aline")
    no2.inserirValor("Marques")

main()
```

# Mostrar elementos da lista encadeada - Nó

main.py

```
import node

def mostrarLista(no):
    while (no != None):
        print(no.getValor())
        no = no.proximo
```

```
def main():
    no1 = node.No()
    no2 = node.No()

    no1.inserirValor("Aline")
    no2.inserirValor("Marques")

    mostrarLista(no1)

main()
```

# Exercício E

Mostre os elementos da lista encadeada que você criou com a classe Nó, cujo valores são cores .

# Mostrar elementos da lista encadeada - Lista

listaencadeada.py

```
import node

class listaEncadeada(object):
    ...
    def PrintLista(self):
        no = self.cabeça
        while (no != None):
            print(no.getValor())
            no = no.proximo
```

main.py

```
import listaencadeada

def main():
    lista = listaEncadeada.listaEncadeada()
    lista.inserirNo("valor 1")
    lista.inserirNo("valor 2")
    lista.PrintLista()


main()
```

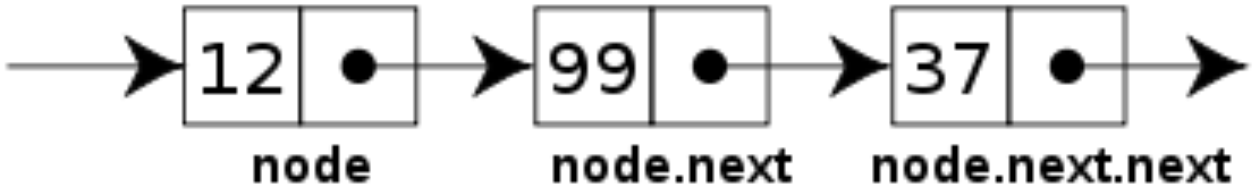
# Exercício F

Mostre os elementos da lista encadeada que você criou com a classe ListaEncadeada, sobre os valores pares e ímpares.

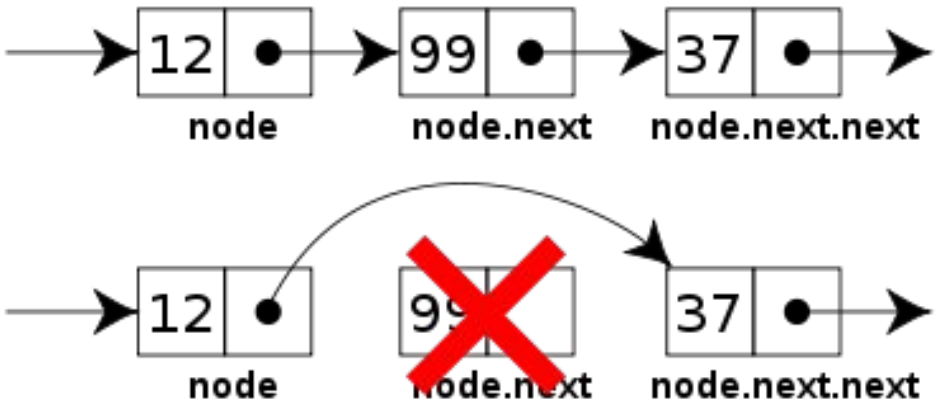
# Operações com lista encadeada

- Criação;
- Inserção de elementos na lista encadeada;
- Mostrar elementos da lista encadeada;
- **Remoção de elementos na lista encadeada;**
- Busca na lista encadeada;

# Remoção de valores na lista encadeada



# Remoção de valores na lista encadeada



# Remoção de valores na lista encadeada (informando o valor) - Nó

main.py

```
import node

def removerNo(no, item):
    while no:
        if (no.getValor() == item):
            no.setValor(no.proximo.getValor()
            no.setProximo(no.proximo.getProximo())
        no = no.proximo
...
```

```
def main():
    no1 = node.No()
    no2 = node.No()

    no1.inserirValor("Aline")
    no2.inserirValor("Marques")
    removerNo(no1,"Aline")
    mostrarLista(no1)

main()
```

# Remoção de valores na lista encadeada (informando o valor) - Lista

listaencadeada.py

```
import node

class listaEncadeada(object):
    def removerNo(self, item):
      no = self.cabeça
      while (no != None):
          if (no.getValor() == item):
              no.setValor(no.proximo.getValor())
              no.setProximo(no.proximo.getProximo())
          no = no.proximo
```

# Remoção de valores na lista encadeada (informando o valor) - Lista

main.py

```
import listaencadeada

def main():
    lista = listaEncadeada.listaEncadeada()
    lista.inserirNo("valor 1")
    lista.inserirNo("valor 2")
    lista.PrintLista()
    lista.removerNo("valor 1")
    lista.PrintLista()
main()
```

# Exercício F

Crie uma lista encadeada com nomes de pessoas. Em seguida, remova um dos nós armazenados.

# Exercício G

Crie uma lista encadeada chamada de cidades. Armazene 4 nomes de cidades diferentes. Em seguida, insira na segunda posição dessa lista encadeada o nome de uma nova cidade.

Dica: você precisará criar uma nova função dentro da classe lista encadeada que defina a posição onde a operação push vai acontecer.

# Operações com lista encadeada

- Criação;
- Inserção de elementos na lista encadeada;
- Mostrar elementos da lista encadeada;
- Remoção de elementos na lista encadeada;
- **Busca na lista encadeada;**

# Busca na lista encadeada

- Basta varrer toda a lista em busca de um determinado valor (já foi feito na solução dos exemplos anteriores)

# Dúvidas?

alanamm.prof@gmail.com