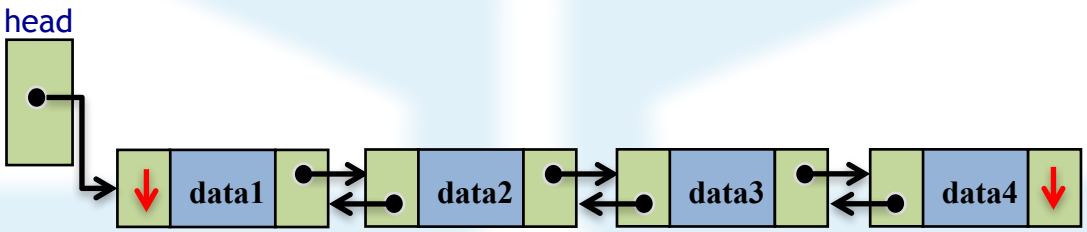
ساختمان داده‌ها و الگوریتم‌ها

لیست دو پیوندی

D o u b l y L i n k e d L i s t



Dr. *Ali Valinejad*

valinejad.ir
valinejad@umz.ac.ir

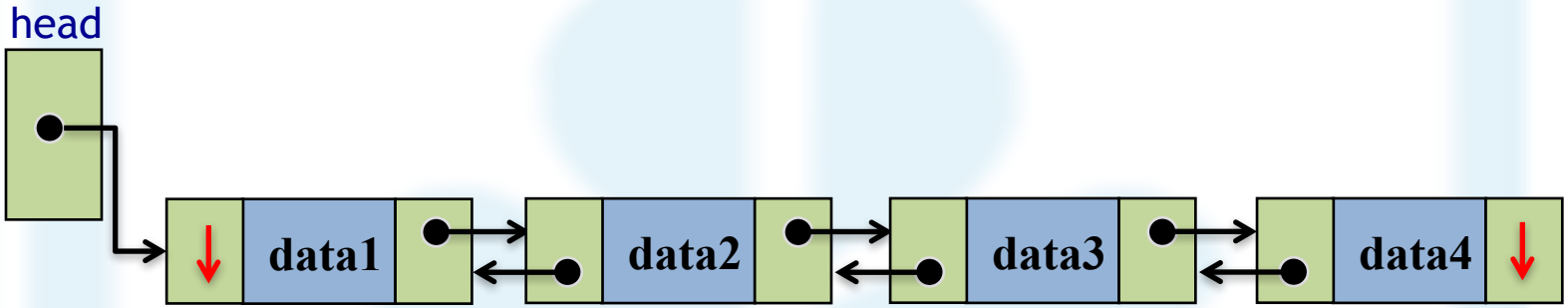
فهرست مطالب

تعریف لیست دو پیوندی(Doubly Linked List)

- **لیست دو پیوندی** (یا لیست دو طرفه) نوعی ساختار داده‌ای است که در آن یک مجموعه از عناصر به نام **گره** (Node)، طوری قرار می‌گیرند که هر گره آدرس گره‌های قبلی و بعدی را در خود ذخیره می‌کند. این گره‌ها می‌توانند در مکان‌های مختلف حافظه قرار داشته باشند.
- هر گره از لیست دو پیوندی شامل سه بخش مختلف است:
  ۱- *اشاره‌گر* به گره قبلی
  ۲- بخش *داده*
  ۳- *اشاره‌گر* به گره بعدی

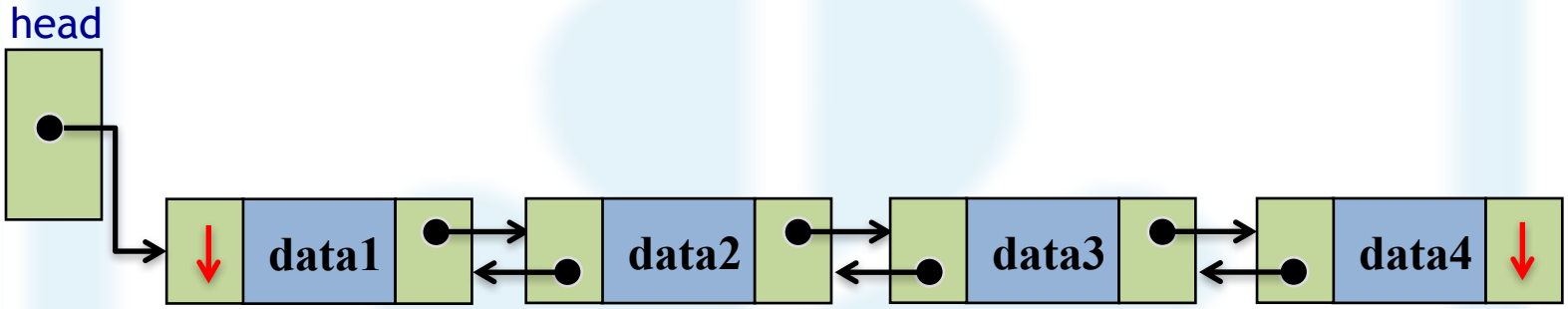
- هرگره خود یک ساختار داده‌ای است.
- ترتیب خطی گره‌های یک لیست دو پیوندی، توسط اشارگرها تعیین می‌گردد.

نمایش لیست دو پیوندی(Doubly Linked List)



- دسترسی به گره ابتدایی لیست با استفاده از یک اشاره‌گر خارجی صورت می‌پذیرد.
- بخش اشاره‌گر prev از اولین گره، هیچ آدرسی را در خود ذخیره نمی‌کند.
- بخش اشاره‌گر next از آخرین گره، هیچ آدرسی را در خود ذخیره نمی‌کند.
- پیکان رو به پایین نشانگر پیوند **NULL** یا **None** است.

نمایش لیست دو پیوندی(Doubly Linked List)



- دسترسی مستقیم به گره‌ها وجود ندارد.
- دسترسی به گره ابتدایی توسط اشاره‌گر head امکان‌پذیر است. سایر گره‌ها توسط بخش اشاره‌گر از گره قبلی قابل دسترسی هستند.
- برخلاف آرایه که یک ساختار دسترسی مستقیم است، لیست پیوندی یک ساختار داده‌ای دسترسی متوالی است.

نوع داده انتزاعی لیست دو پیوندی

نوع داده انتزاعی لیست دو پیوندی

تعریف:

- **لیست دو پیوندی** (یا لیست دو طرفه) نوعی ساختار داده‌ای است که در آن یک مجموعه از عناصر به نام **گره** (Node)، طوری قرار می‌گیرند که هر گره آدرس گره‌های قبلی و بعدی را در خود ذخیره می‌کند. این گره‌ها می‌توانند در مکان‌های مختلف حافظه قرار داشته باشند.

عملیات اصلی:

- ایجاد لیست خالی،
- امتحان خالی بودن لیست،
- درج یک گره به لیست،
- حذف یک گره از لیست،
- پیمایش لیست،
- نمایش لیست

پیاده‌سازی لیست دو پیوندی

نکته کلیدی:

- قبل از پیاده‌سازی لیست پیوندی، لازم است تا نمایش (نحوه پیاده‌سازی) هر گره لیست را مشخص کنیم.
- برای نمایش یک گره از لیست دو پیوندی می‌توان کلاسی برای آن نوشت.

پیاده‌سازی لیست دو پیوندی

نوع داده انتزاعی یک گره از لیست دو پیوندی

تعریف گره:

گره لیست، یک ساختار داده‌ای، شامل یک بخش **داده** و دو بخش **آدرس** است. بخش داده برای ذخیره‌سازی اطلاعات داده‌ای هر گره، و دو بخش آدرس برای ذخیره‌سازی آدرس گره‌های قبلی و بعدی لیست بکار می‌رود.

- **اعضای داده‌ای:**
  - تعریف قسمت داده
  - تعریف اشاره‌گرها
- **اعضای تابعی:**
  - تعریف تابع init

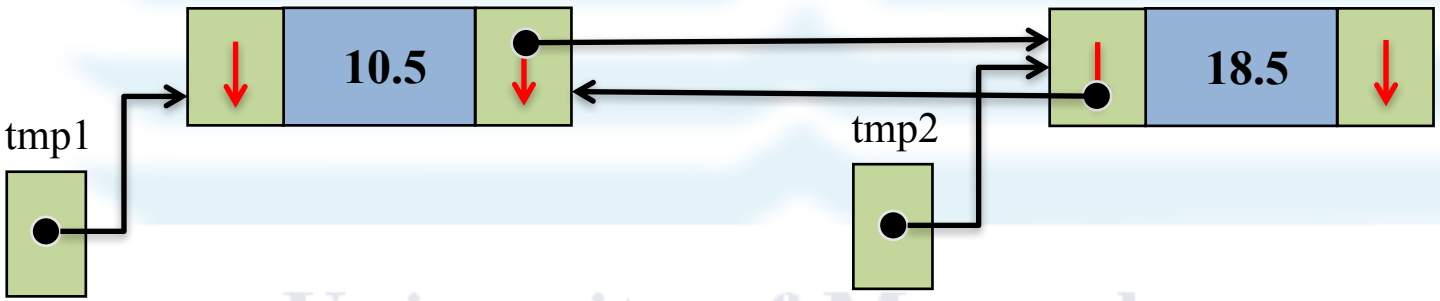
پیاده‌سازی یک گره از لیست دو پیوندی

نمایش یک گره از لیست دو پیوندی

```
class DoublyLinkedListNode:
    def __init__(self, prev=None, data=0.0, next=None):
        self._prev = prev
        self._data = data
        self._next = next
```


```
tmp1=SinglyLinkedListNode(None ,10.5, None)
tmp2=SinglyLinkedListNode(None , 18.5, None)
tmp1._next=tmp2
tmp2._prev=tmp1
```

print(tmp1._next._data)
????



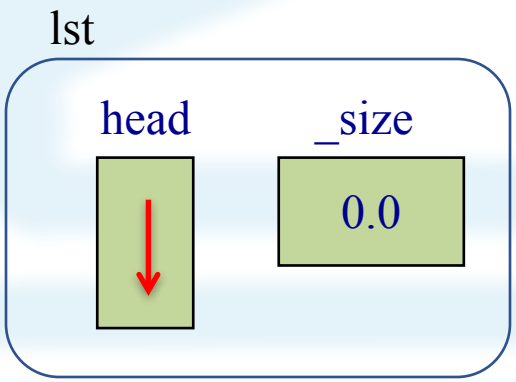
پیاده‌سازی لیست دو پیوندی

نمایش لیست دو پیوندی

```
class DoublyLinkedList:
    class DoublyLinkedListNode:
        def __init__(self, prev=None, data=0.0, next=None):
            self._prev = prev
            self._data = data
            self._next = next
    def __init__(self):
        self.head = None
        self._size = 0
```

- تعریف تابع **init**

```
lst = DoublyLinkedList()
```



پیاده‌سازی لیست دو پیوندی

نمایش لیست دو پیوندی

```
class DoublyLinkedList:
    class DoublyLinkedListNode:
        def __init__(self, prev=None, data=0.0, next=None):
            self._prev = prev
            self._data = data
            self._next = next
    def __init__(self):
        self.head = None
        self._size = 0

    def __len__(self):
        return self._size
```

- تعریف تابع init
- تعریف تابع __len__

پیاده‌سازی لیست دو پیوندی

نمایش لیست دو پیوندی

```
class DoublyLinkedList:
    class DoublyLinkedListNode:
        def __init__(self, prev=None, data=0.0, next=None):
            self._prev = prev
            self._data = data
            self._next = next
    def __init__(self):
        self.head = None
        self._size = 0

    def __len__(self):
        return self._size

    def is_empty(self):
        return self._size == 0
```

- تعریف تابع init
- تعریف تابع __len__
- تعریف تابع is_empty

پیاده‌سازی لیست دو پیوندی

- **تابع درج یک گره در ابتدای لیست (با داشتن داده مربوط به گره):**

```
def Add2FirstByValue(self, data):
    tmp = self.DoublyLinkedListNode(None, data, None)
    if self._size == 0:
        self.head = tmp
    self._size += 1
    return self.head
```

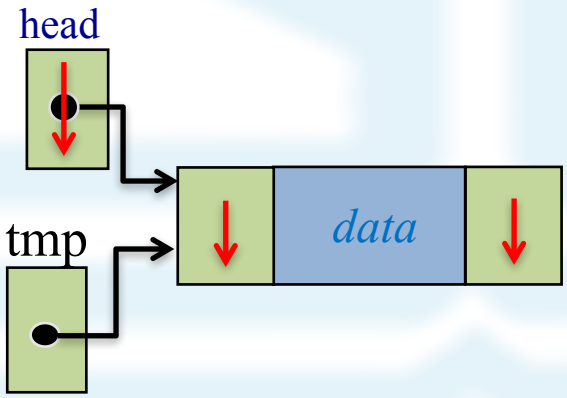
گام۱: تعریف گره

گام۲: اضافه کردن گره به لیست

- دستورات لازم در صورتی که لیست خالی باشد:

گام۳: اضافه کردن یک واحد به _size

گام۴: برگرداندن ارجاع به گره اول لیست به عنوان خروجی تابع



پیاده‌سازی لیست دو پیوندی

- **تابع درج یک گره در ابتدای لیست (با داشتن داده مربوط به گره):**

```
def Add2FirstByValue(self, data):
    tmp = self.DoublyLinkedListNode(None, data, None)
    if self._size == 0:
        self.head = tmp
    else:
        tmp._next = self.head
        self.head._prev=tmp
        self.head = tmp
    self._size += 1
    return self.head
```

گام۱: تعریف گره

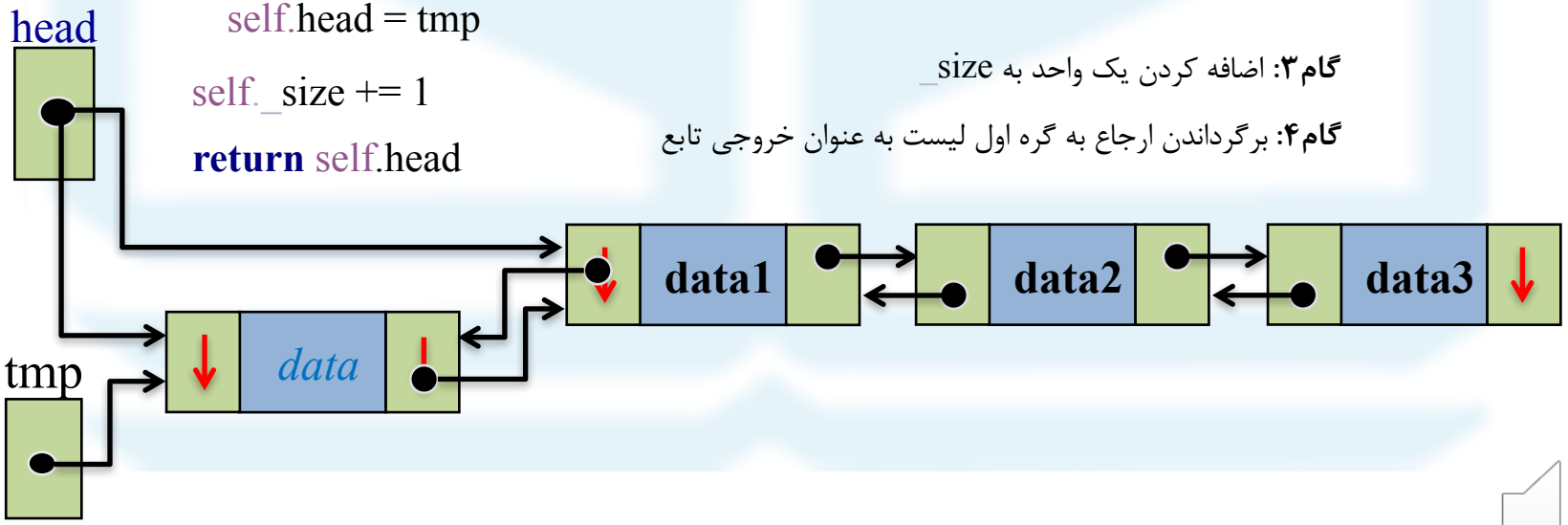
گام۲: اضافه کردن گره به لیست

- دستورات لازم در صورتی که لیست خالی باشد:

- دستورات لازم در صورتی که لیست خالی نباشد:

گام۳: اضافه کردن یک واحد به _size

گام۴: برگرداندن ارجاع به گره اول لیست به عنوان خروجی تابع

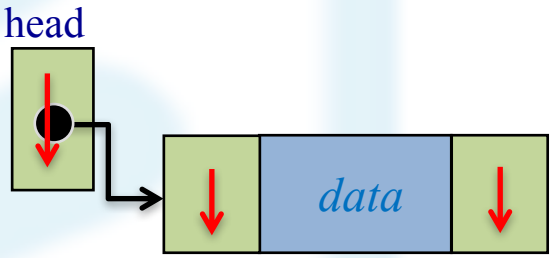


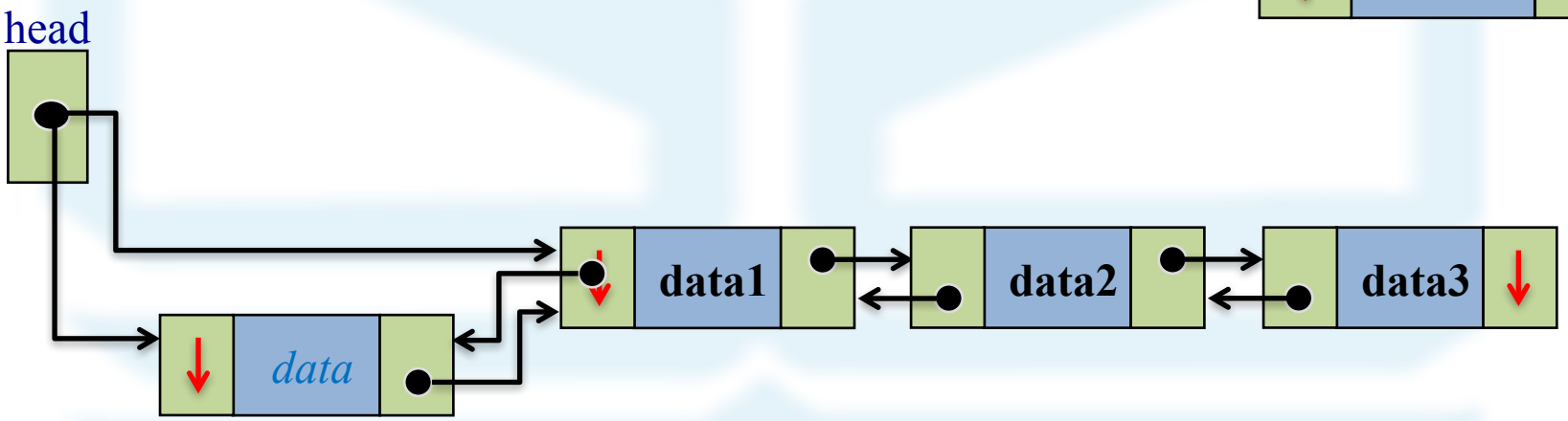
پیاده‌سازی لیست دو پیوندی

قطعه برنامه بهینه

- تابع درج یک گره در ابتدای لیست (با داشتن داده مربوط به گره):

```
def Add2FirstByValue(self, data):
    self.head = self.DoublyLinkedListNode(None, data, self.head )
    if self._size != 0:
        self.head._next._perv = self.head
    self._size += 1
    return self.head
```





پیاده‌سازی لیست دو پیوندی

- **تابع درج یک گره در ابتدای لیست (با داشتن آدرس گره):**

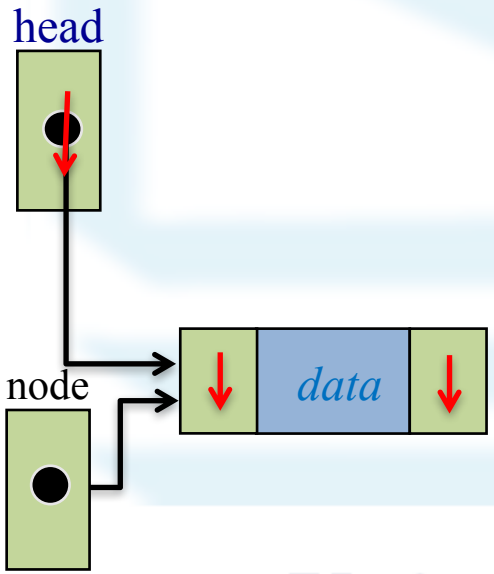
```
def Add2FirstByRefrence(self, node):
    if self._size == 0:
        node._next = None
        node._prev = None
        self.head = node
    self._size += 1
    return self.head
```

گام۱: اضافه کردن گره به لیست

- دستورات لازم در صورتی که لیست خالی باشد:

گام۲: اضافه کردن یک واحد به _size

گام۳: برگرداندن ارجاع به گره اول لیست به عنوان خروجی تابع



پیاده‌سازی لیست دو پیوندی

- **تابع درج یک گره در ابتدای لیست (با داشتن آدرس گره):**

```
def Add2FirstByRefrence(self, node):
    if self._size == 0:
        node._next = None
        node._prev = None
        self.head = node
    else:
        node._prev = None
        node._next = self.head
        self.head._perv = node
        self._head = node
    self._size += 1
    return self.head
```

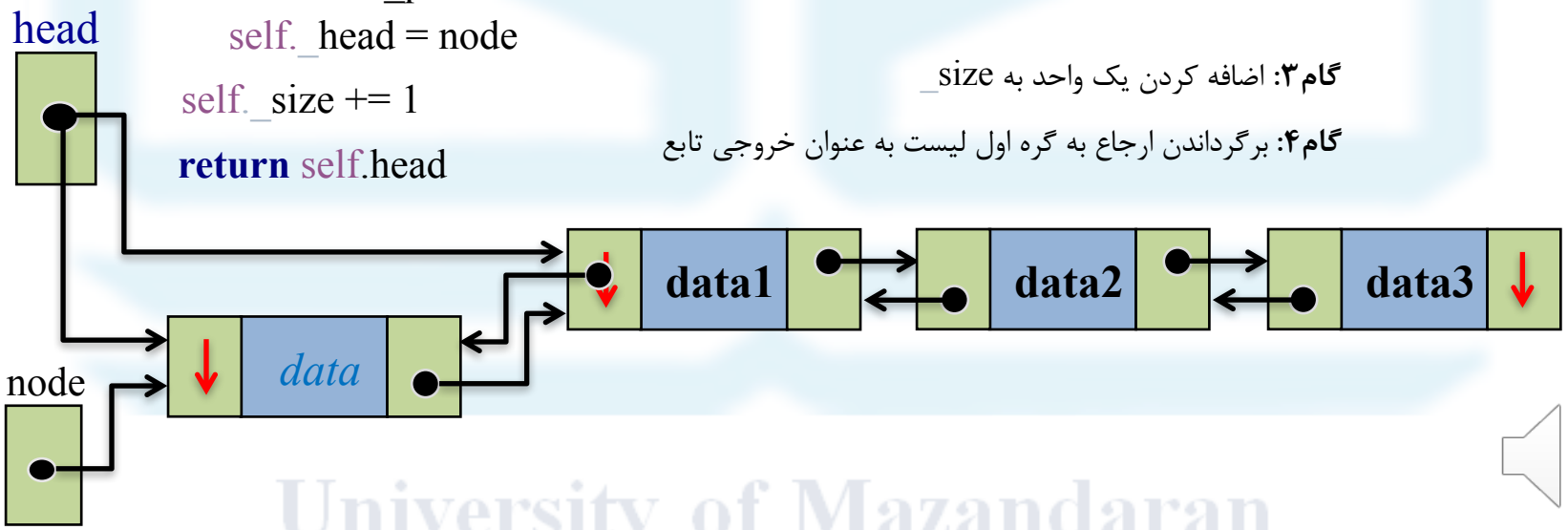
گام۱: اضافه کردن گره به لیست

- دستورات لازم در صورتی که لیست خالی باشد:

- دستورات لازم در صورتی که لیست خالی نباشد:

گام۳: اضافه کردن یک واحد به _size

گام۴: برگرداندن ارجاع به گره اول لیست به عنوان خروجی تابع



پیاده‌سازی لیست دو پیوندی

قطعه برنامه بهینه‌تر

تمرین: تابع درج یک گره در ابتدای لیست را به صورت بهینه بنویسید.

```
def Add2First(self, inp):
    if type(self.head) == type(inp):
```

?

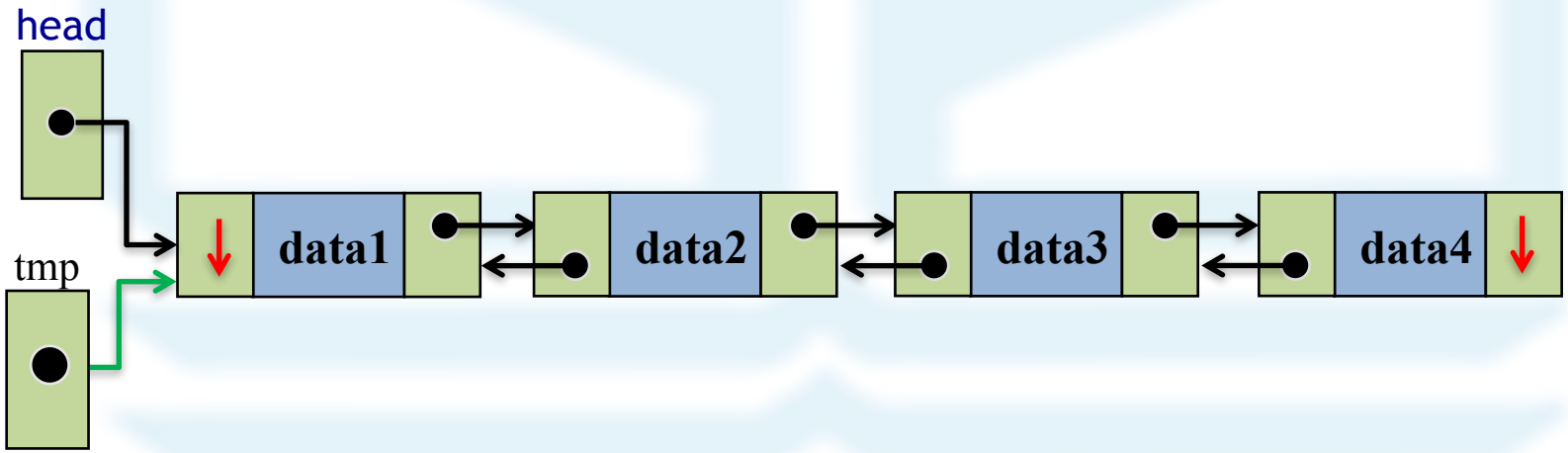
پیاده‌سازی لیست دو پیوندی

- **تابع دسترسی به اولین گره از لیست دو پیوندی غیر خالی:**

```
def ReturnFirstNode(self):
    tmp = self.head
    return tmp
```

گام۱: تعریف اشاره گری به اولین گره لیست

گام۲: برگرداندن ارجاع به گره اول لیست به عنوان خروجی تابع



پیاده‌سازی لیست دو پیوندی

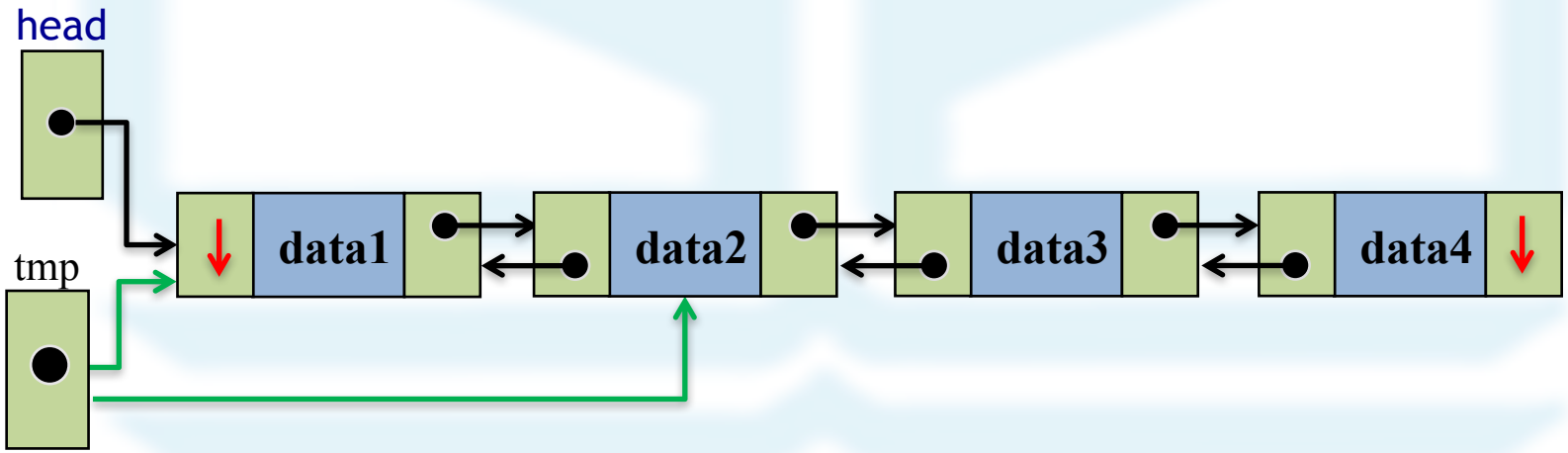
- تابع دسترسی به دومین گره از لیست دو پیوندی با بیش از یک گره:

```
def ReturnSecondNode(self):
    tmp = self.head
    tmp =  tmp._next
    return tmp
```

گام۱: تعریف اشاره گری به اولین گره لیست

گام۲: تغییر اشاره گره tmp تا زمانی که به دومین گره لیست اشاره کند

گام۳: برگرداندن ارجاع به گره دوم لیست به عنوان خروجی تابع



پیاده‌سازی لیست دو پیوندی

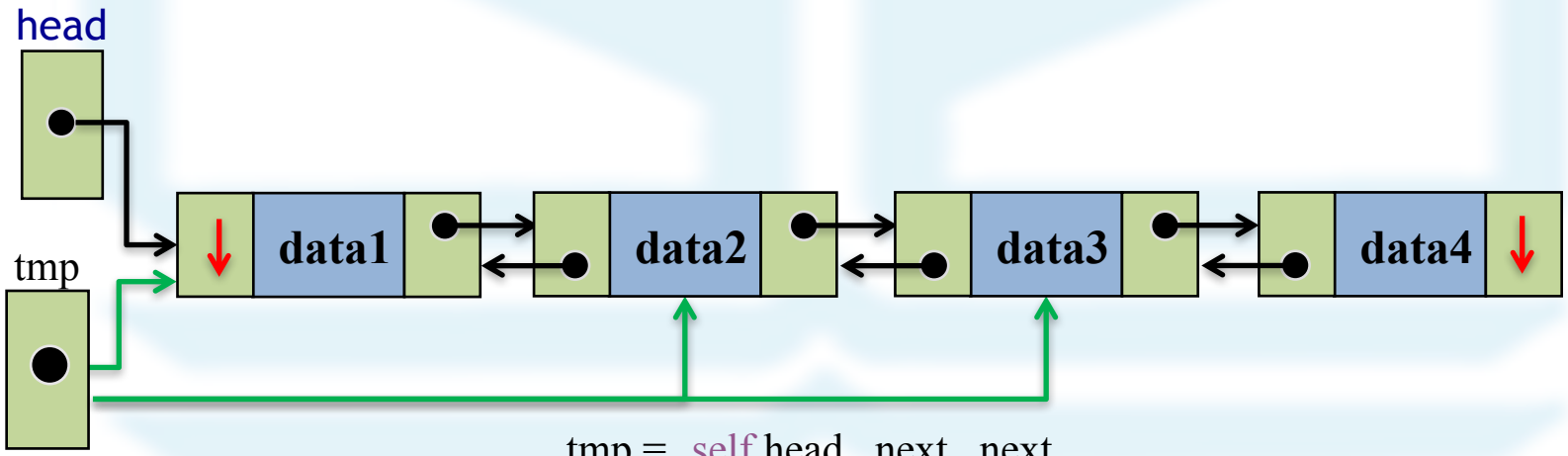
- **تابع دسترسی به سومین گره از لیست دو پیوندی با بیش از دو گره:**

```
def ReturnThirdNode(self):
    tmp = self.head
    tmp =  tmp._next
    tmp =  tmp._next
    return tmp
```

گام۱: تعریف اشاره گری به اولین گره لیست

گام۲: تغییر اشاره گره tmp تا زمانی که به سومین گره لیست اشاره کند

گام۳: برگرداندن ارجاع به گره سوم لیست به عنوان خروجی تابع:



tmp = self.head._next._next

پیاده‌سازی لیست دو پیوندی

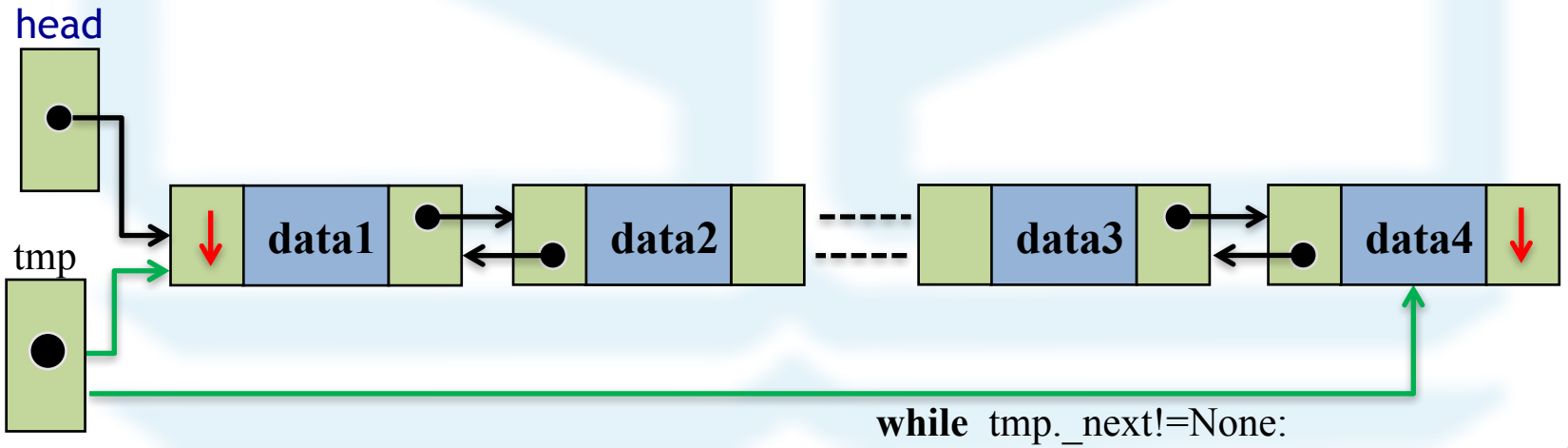
- تابع دسترسی به آخرین گره از لیست دو پیوندی غیر خالی:

```
def ReturnLastNode(self):
    tmp = self.head
    tmp =  tmp._next
          :
    tmp =  tmp._next
    return tmp
```

گام۱: تعریف اشاره گری به اولین گره لیست

گام۲: تغییر اشاره گره tmp تا زمانی که به آخرین گره لیست اشاره کند

گام۳: برگرداندن ارجاع به گره آخر لیست به عنوان خروجی تابع:



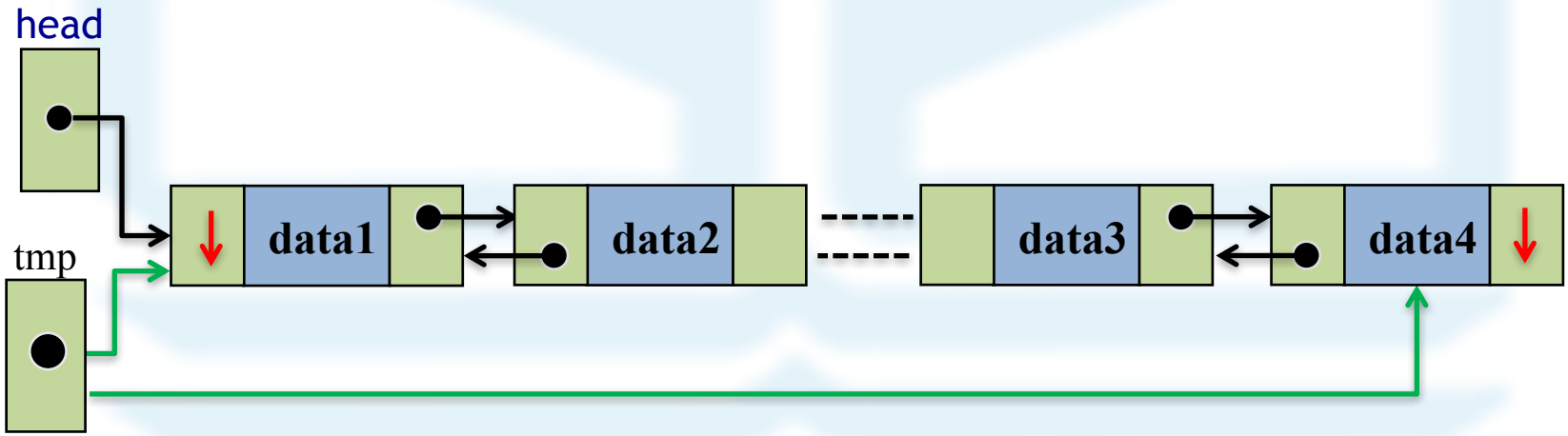

```
while  tmp._next!=None:
    tmp =  tmp._next
```

پیاده‌سازی لیست دو پیوندی

قطعه برنامه بهینه

- تابع دسترسی به آخرین گره از لیست دو پیوندی:

```
def ReturnLastNode(self):
    tmp = self.head
    if self.head:
        while  tmp._next != None:
            tmp =  tmp._next
    return tmp
```



پیاده‌سازی لیست دو پیوندی

- **تابع جستجوی یک گره با مقدار داده‌ای x:**

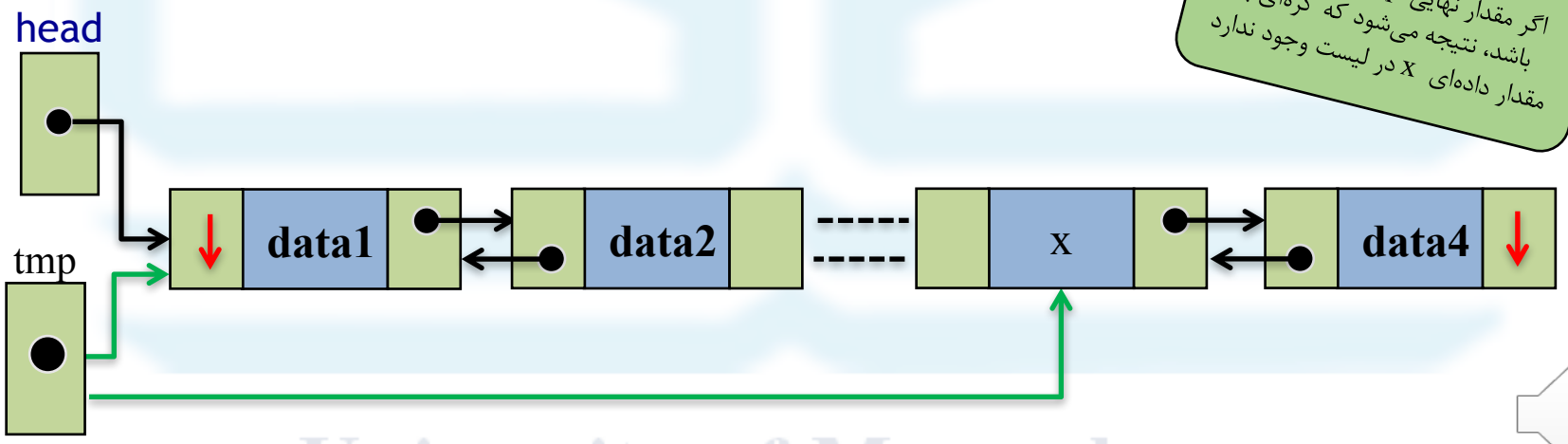
```
def SearchNodeByValue(self,x):
    tmp = self.head
    while  tmp  !=  None:
        if tmp._data  ==  x:
            break
        tmp =  tmp._next
    return tmp
```

گام۱: تعریف اشاره گری به اولین گره لیست

گام۲: تغییر اشاره گره tmp تا زمانی که به گره مورد نظر و یا None اشاره کند.

گام۳: برگرداندن ارجاع مناسب:

اگر مقدار نهایی tmp برابر None باشد، نتیجه می‌شود که گره‌ای با مقدار داده‌ای x در لیست وجود ندارد



پیاده‌سازی لیست دو پیوندی

- **تابع جستجوی یک گره با ارجاع:**

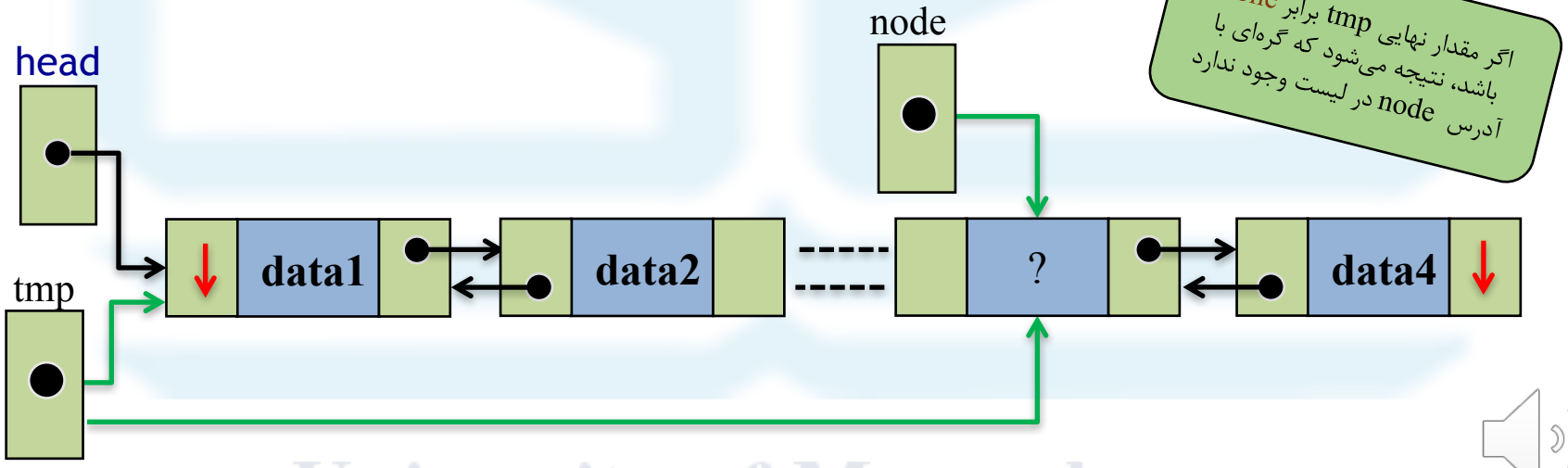
```
def SearchNodeByReference(self,node):
    tmp = self.head
    while  tmp  !=  None:
        if tmp  ==  node:
            break
        tmp =  tmp._next
    return tmp
```

گام۱: تعریف اشاره گری به اولین گره لیست

گام۲: تغییر اشاره گره tmp تا زمانی که به گره مورد نظر و یا None اشاره کند.

گام۳: برگرداندن ارجاع مناسب:

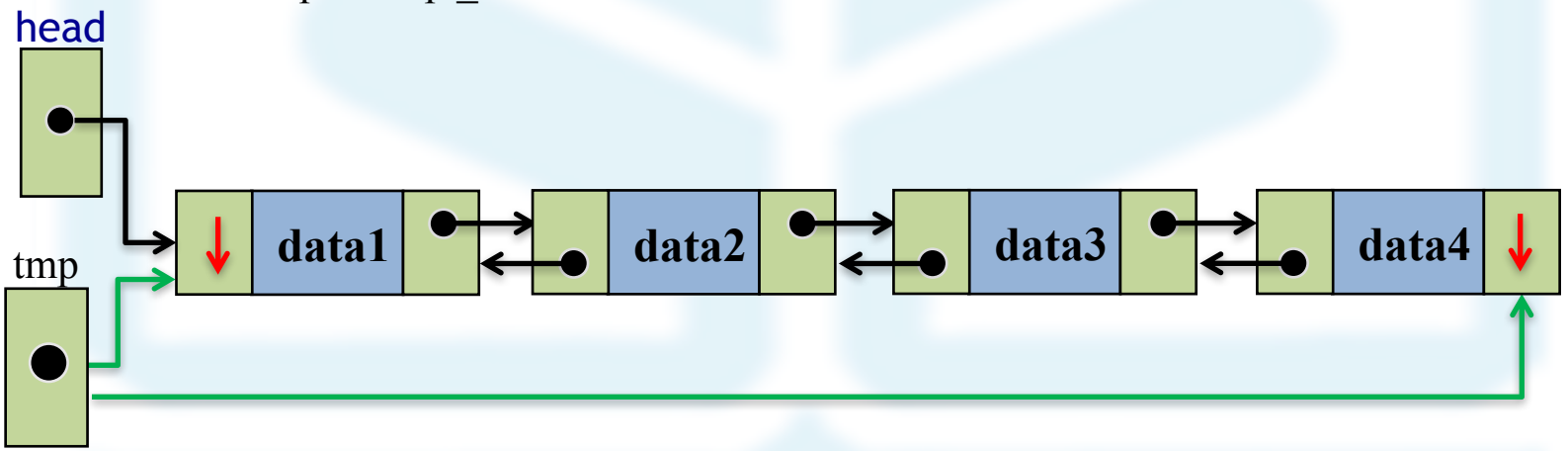
اگر مقدار نهایی tmp برابر None باشد، نتیجه می‌شود که گره‌ای با آدرس node در لیست وجود ندارد



پیاده‌سازی لیست دو پیوندی

- تابع چاپ داده گره‌های لیست دو پیوندی:

```
def Show(self):
    tmp = self.head
    while  tmp != None:
        print(tmp._data,end='→')
        tmp =  tmp._next
```



data1→data2→data3→data4→

پیاده‌سازی لیست دو پیوندی

تمرین

۱- تابعی بنویسید که گره‌ای با مقدار داده‌ای x را جستجو نماید و در صورت وجود، آدرس گره قبلی آن را برگرداند.

۲- تابعی بنویسید که گره‌ای با آدرس معلوم را جستجو نماید و در صورت وجود، آدرس گره قبلی آن را برگرداند.

۳- تابعی بنویسید که گره‌ای با مقدار داده‌ای x را جستجو نماید و در صورت وجود، آدرس گره بعدی آن را برگرداند.

۴- تابعی بنویسید که گره‌ای با آدرس معلوم را جستجو نماید و در صورت وجود، آدرس گره بعدی آن را برگرداند.