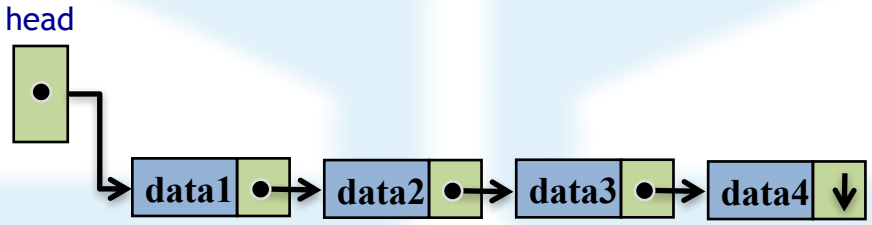
ساختمان داده‌ها و الگوریتم‌ها

لیست تک پیوندی

S i n g l y L i n k e d L i s t



Dr. *Ali Valinejad*

valinejad.ir
valinejad@umz.ac.ir

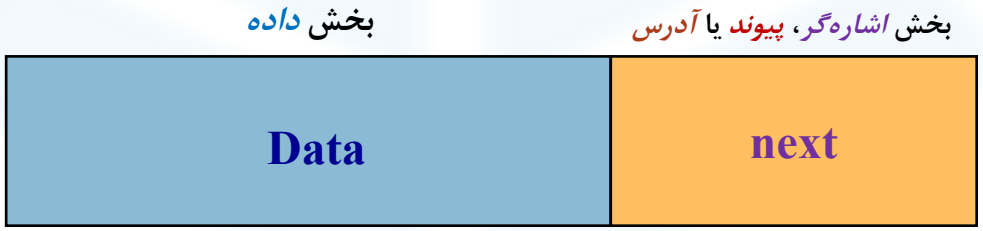
فهرست مطالب

1. تعریف لیست تک پیوندی
2. نمایش لیست تک پیوندی
3. نوع داده انتزاعی لیست تک پیوندی
4. پیاده‌سازی لیست تک پیوندی

تعریف لیست تک پیوندی(Singly Linked List)

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

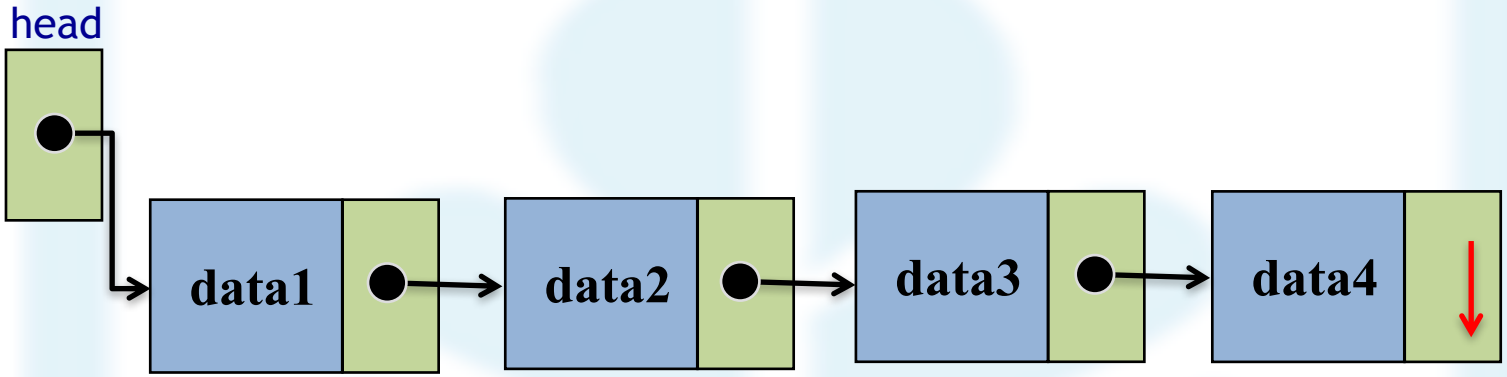
- هر گره از لیست تک پیوندی شامل دو بخش مختلف است:
  ۱- بخش ***داده***
  ۲- بخش ***اشاره‌گر***، ***آدرس***، ***پیوند***



- هرگره خود یک ساختار داده‌ای است.

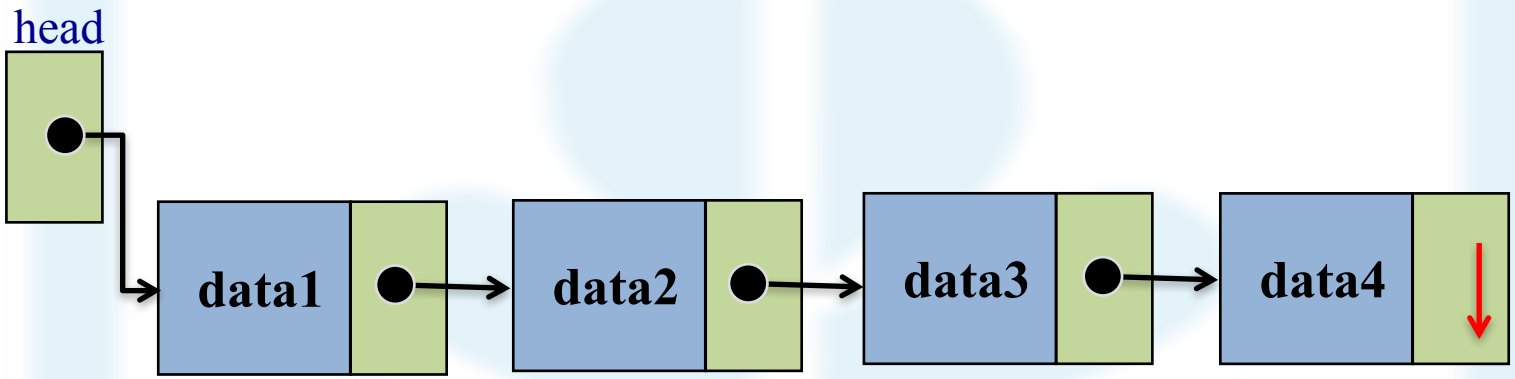
- ترتیب خطی گره‌های یک لیست تک پیوندی، توسط اشارگرها تعیین می‌گردد.

نمایش لیست تک پیوندی(Singly Linked List)



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

نمایش لیست تک پیوندی(Singly 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 تک پیوندی

نوع داده انتزاعی لیست تک پیوندی

تعریف:

لیست تک پیوندی (یا لیست یک طرفه) نوعی ساختار داده‌ای است که در آن یک مجموعه از عناصر به نام **گره** (Node)، طوری قرار می‌گیرند که هر گره آدرس گره 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 تک 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 تک پیوندی می‌توان کلاسی برای آن نوشت.

پیاده‌سازی لیست تک پیوندی

نوع داده انتزاعی یک گره از لیست تک پیوندی

تعریف گره:

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

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

پیاده‌سازی لیست تک پیوندی

نمایش یک گره از لیست تک پیوندی

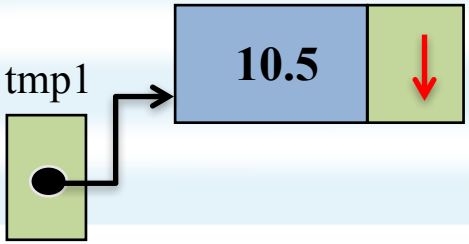
```
class SinglyLinkedListNode:
    def __init__(self,  data=0.0, next=None):
        self._data = data
        self._next = next
```

پیاده‌سازی لیست تک پیوندی

نمایش یک گره از لیست تک پیوندی

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


```
tmp1=SinglyLinkedListNode(10.5, None)
```

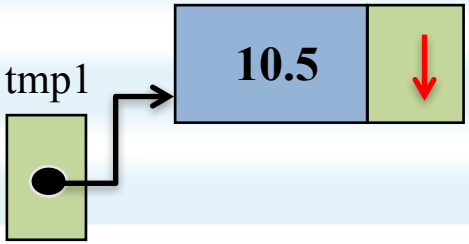


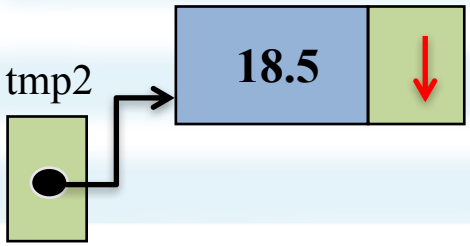
پیاده‌سازی لیست تک پیوندی

نمایش یک گره از لیست تک پیوندی

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


```
tmp1=SinglyLinkedListNode(10.5, None)
tmp2=SinglyLinkedListNode(18.5, None)
```



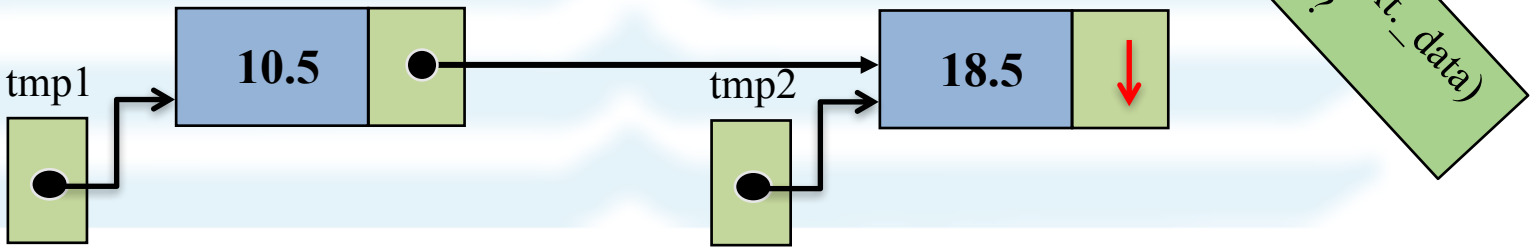


پیاده‌سازی لیست تک پیوندی

نمایش یک گره از لیست تک پیوندی

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


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



پیاده‌سازی لیست تک پیوندی

نمایش لیست تک پیوندی

```
class SinglyLinkedList:
    class SinglyLinkedListNode:
        def __init__(self,  data=0.0, next=None):
            self._data = data
            self._next = next
```

پیاده‌سازی لیست تک پیوندی

نمایش لیست تک پیوندی

```
class SinglyLinkedList:
    class SinglyLinkedListNode:
        def __init__(self,  data=0.0, next=None):
            self._data = data
            self._next = next

    def __init__(self):
        self.head = None
        self._size = 0
```

- تعریف تابع **init**

پیاده‌سازی لیست تک پیوندی

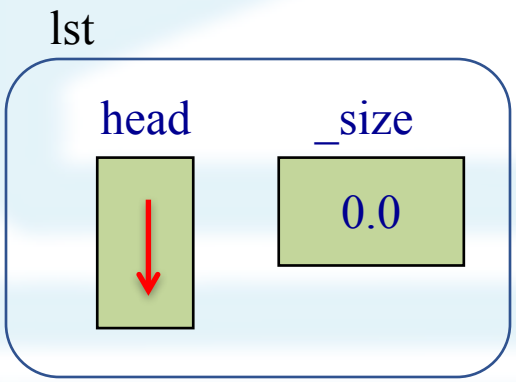
نمایش لیست تک پیوندی

```
class SinglyLinkedList:
    class SinglyLinkedListNode:
        def __init__(self,  data=0.0, next=None):
            self._data = data
            self._next = next

    def __init__(self):
        self.head = None
        self._size = 0
```

- تعریف تابع **init**

```
lst = SinglyLinkedList()
```



پیاده‌سازی لیست تک پیوندی

نمایش لیست تک پیوندی

```
class SinglyLinkedList:
    class SinglyLinkedListNode:
        def __init__(self,  data=0.0, next=None):
            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 تک پیوندی

نمایش لیست تک پیوندی

```
class SinglyLinkedList:
    class SinglyLinkedListNode:
        def __init__(self,  data=0.0, next=None):
            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
```

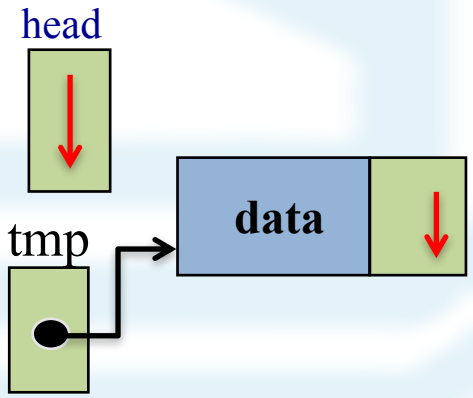
- تعریف تابع init
- تعریف تابع __len__
- تعریف تابع is_empty

پیاده‌سازی لیست تک پیوندی

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

```
def Add2FirstByValue(self, data):
    tmp = self.SinglyLinkedListNode(data, None)
```

گام۱: تعریف گره



پیاده‌سازی لیست تک پیوندی

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

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

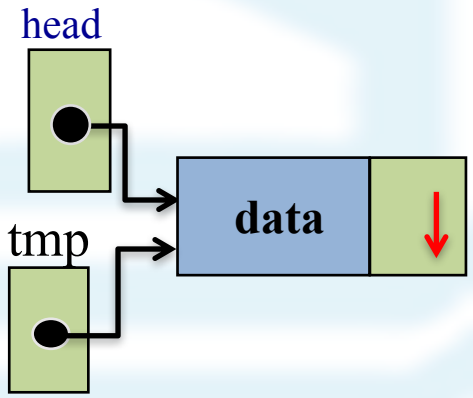
گام۱: تعریف گره

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 تک پیوندی

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

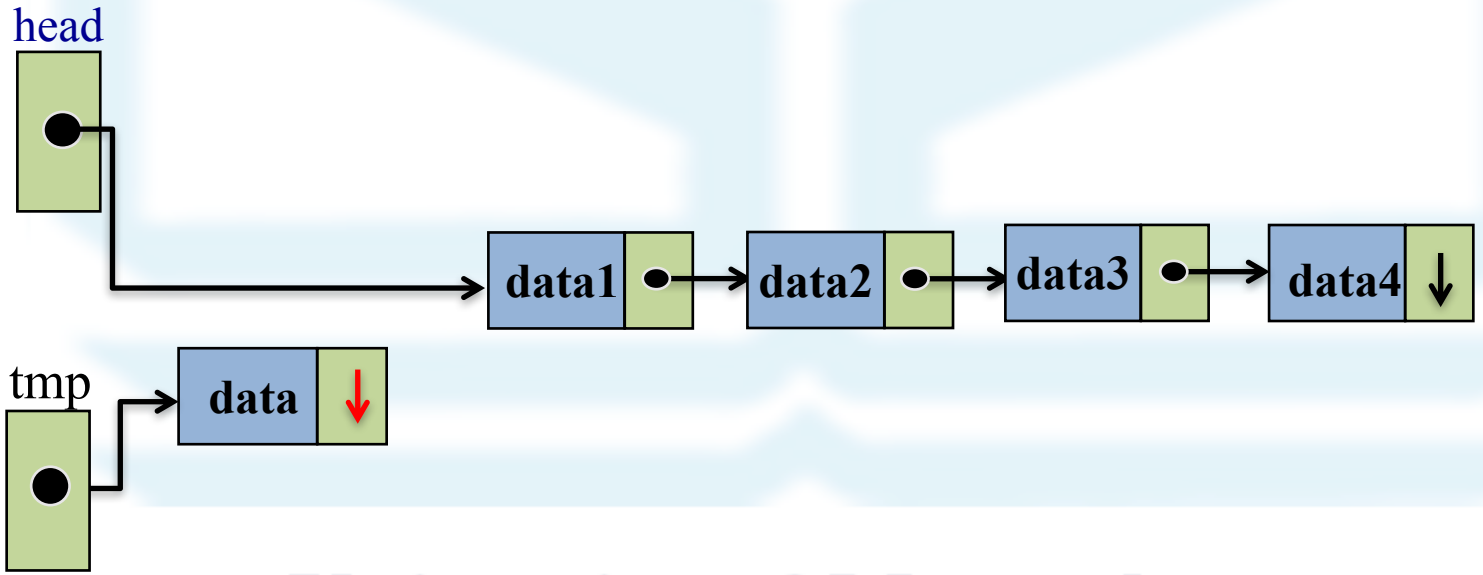
```
def Add2FirstByValue(self, data):
    tmp = self.SinglyLinkedListNode(data, None)
    if self._size == 0:
        self.head = tmp
    else:
        ??????
```

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:



پیاده‌سازی لیست تک پیوندی

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

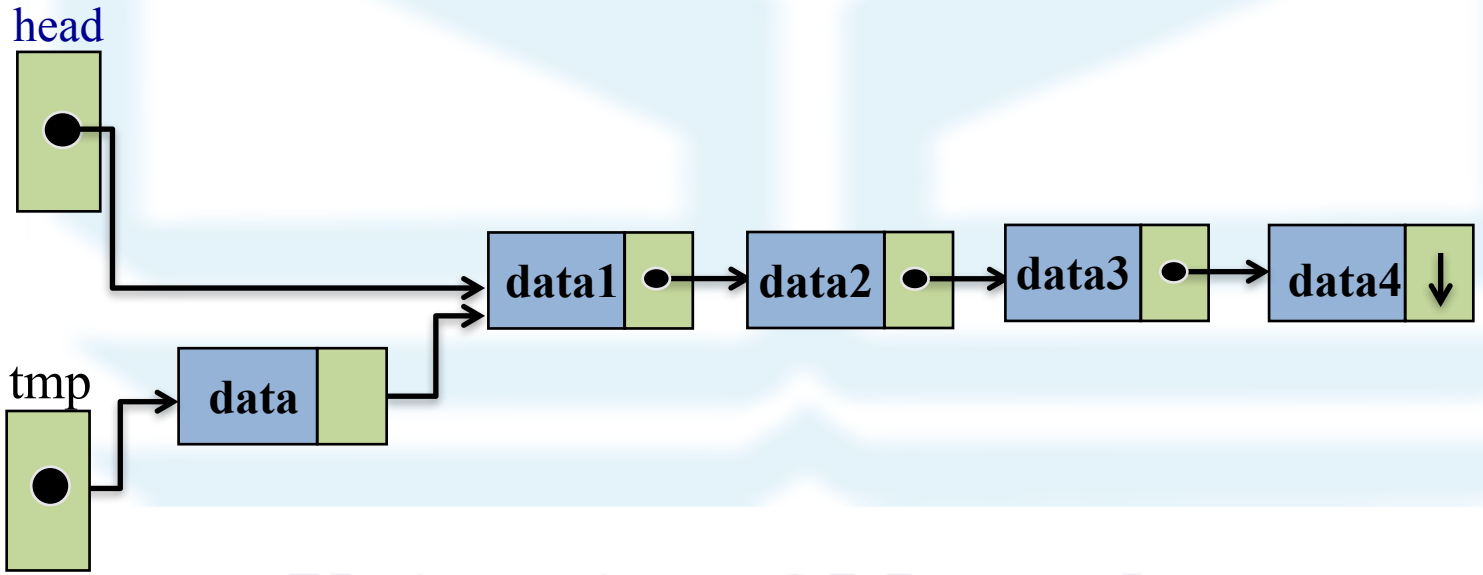
```
def Add2FirstByValue(self, data):
    tmp = self.SinglyLinkedListNode(data, None)
    if self._size == 0:
        self.head = tmp
    else:
        tmp._next = self.head
```

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:



پیاده‌سازی لیست تک پیوندی

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

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

گام۱: تعریف گره

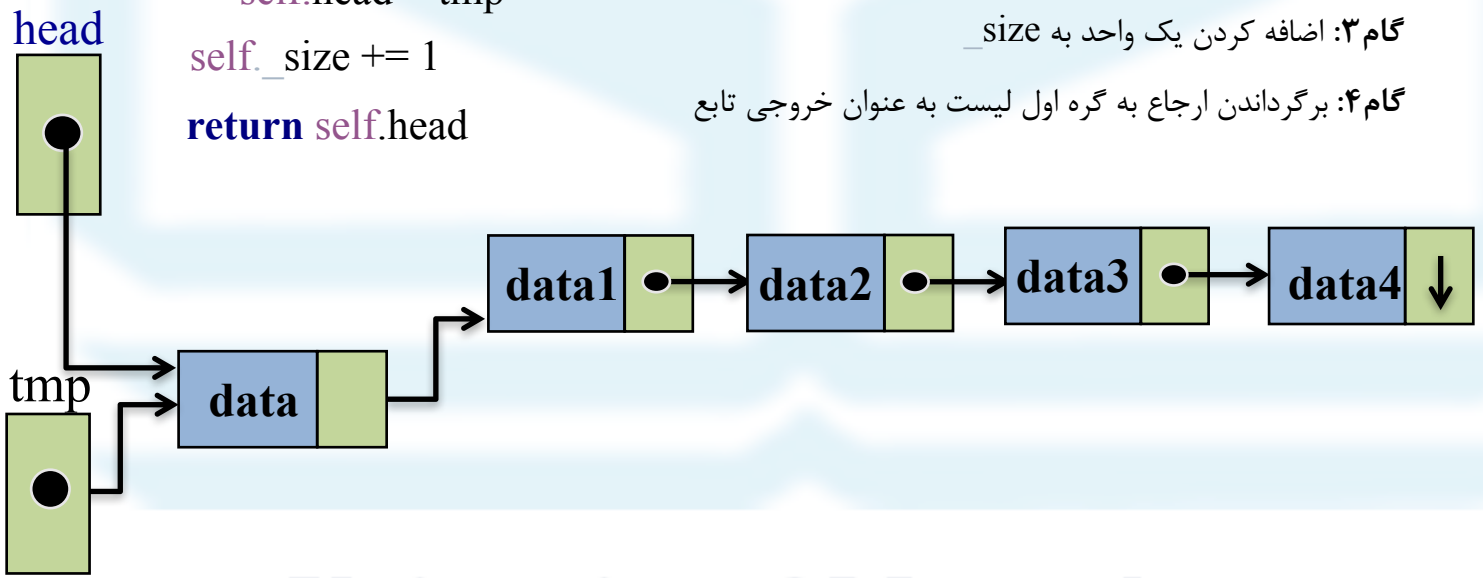
گام۲: اضافه کردن گره به لیست

- 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

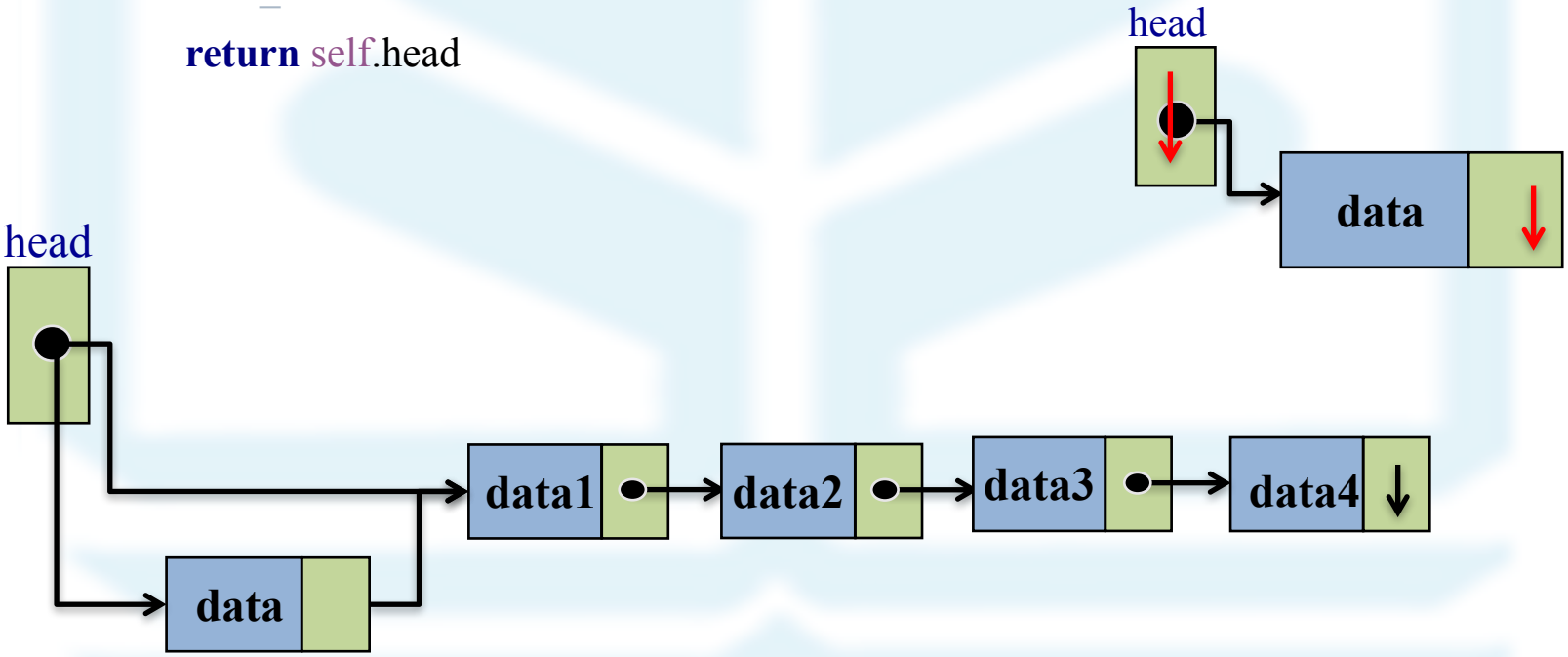


پیاده‌سازی لیست تک پیوندی

قطعه برنامه بهینه

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

```
def Add2FirstByValue(self, data):
    self.head = self.SinglyLinkedListNode(data, self.head )
    self._size += 1
    return self.head
```



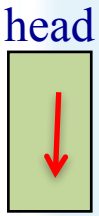
پیاده‌سازی لیست تک پیوندی

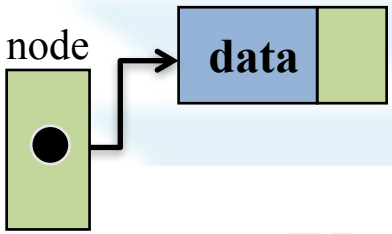
- **تابع درج یک گره در ابتدای لیست (با داشتن آدرس گره):**

```
def Add2FirstByRefrence(self, node):
    if self._size == 0:
        ????
```

گام۱: اضافه کردن گره به لیست

- دستورات لازم در صورتی که لیست خالی باشد:





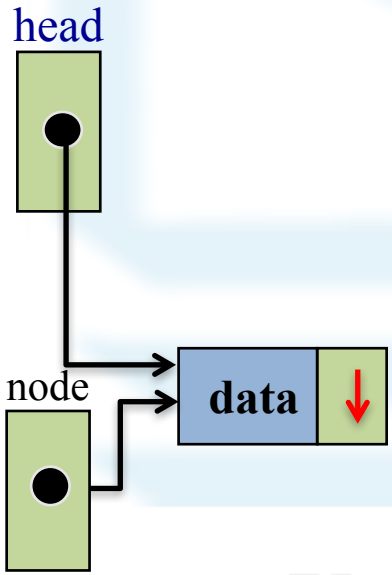
پیاده‌سازی لیست تک پیوندی

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

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

گام۱: اضافه کردن گره به لیست

- دستورات لازم در صورتی که لیست خالی باشد:



پیاده‌سازی لیست تک پیوندی

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

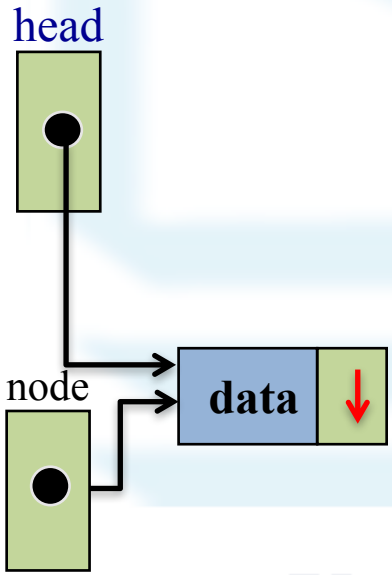
```
def Add2FirstByRefrence(self, node):
    if self._size == 0:
        node._next = None
        self.head = node
    self._size += 1
    return self.head
```

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 تک پیوندی

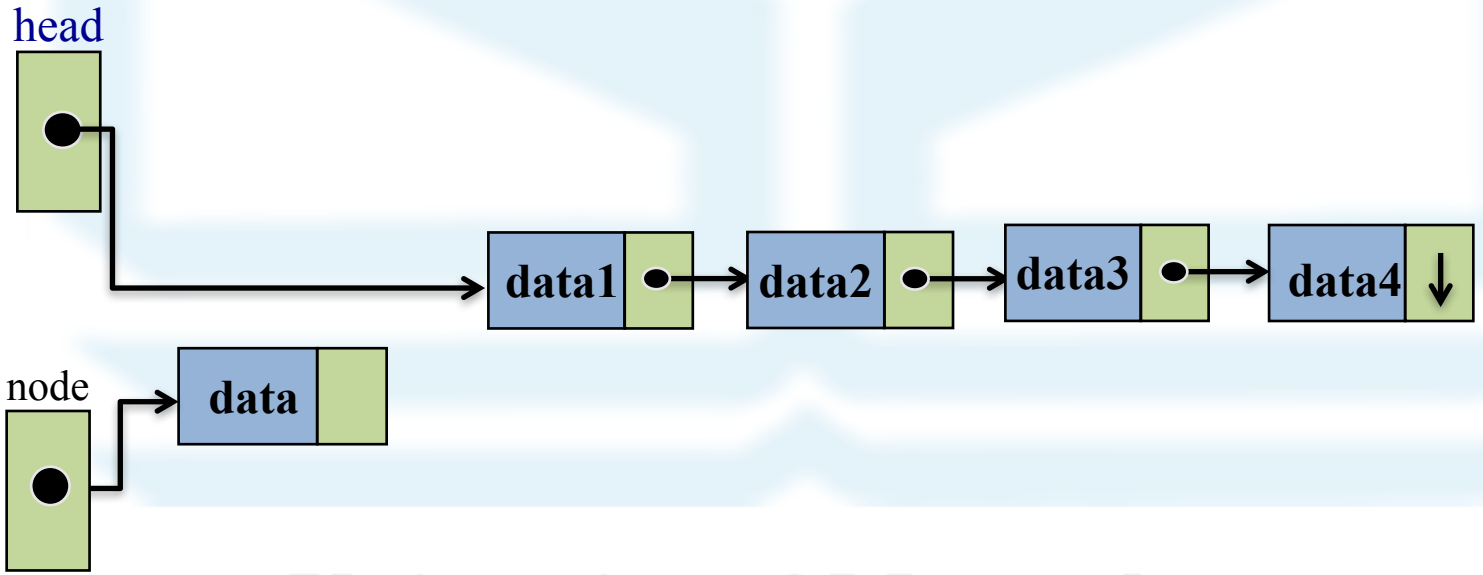
- **تابع درج یک گره در ابتدای لیست (با داشتن آدرس گره):**

```
def Add2FirstByRefrence(self, node):
    if self._size == 0:
        node._next = None
        self.head = node
    else:
        ????
```

گام۱: اضافه کردن گره به لیست

- دستورات لازم در صورتی که لیست خالی باشد:

- دستورات لازم در صورتی که لیست خالی نباشد:



پیاده‌سازی لیست تک پیوندی

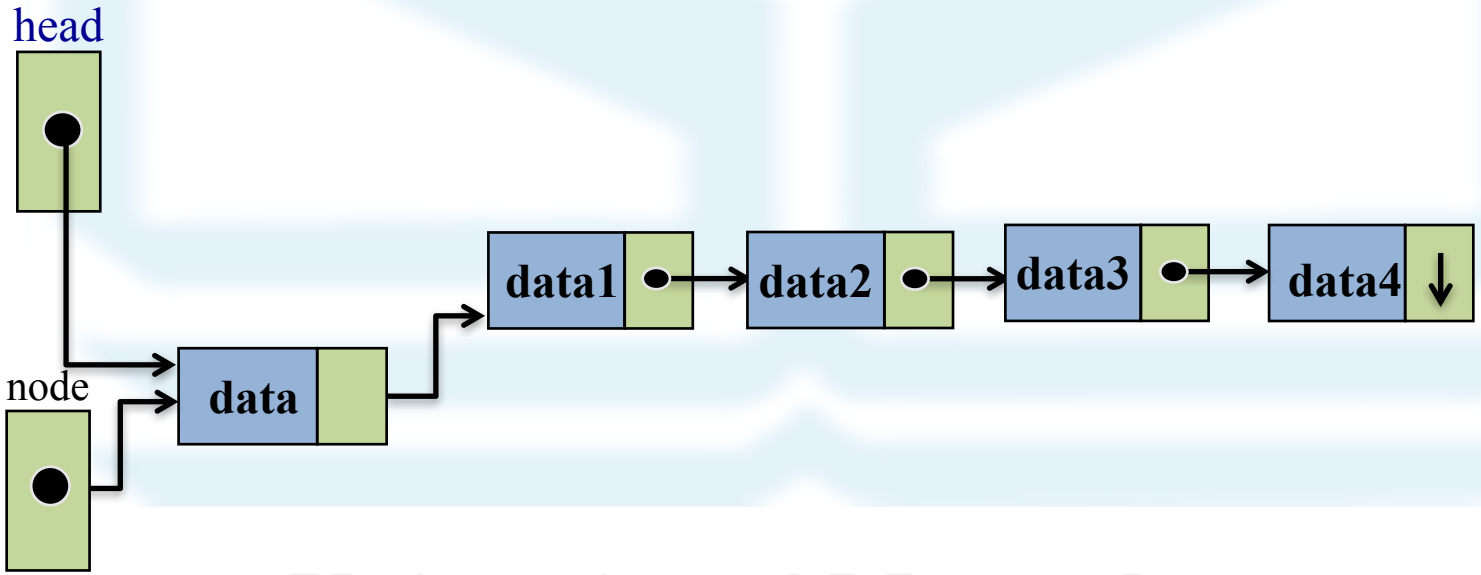
- **تابع درج یک گره در ابتدای لیست (با داشتن آدرس گره):**

```
def Add2FirstByRefrence(self, node):
    if self._size == 0:
        node._next = None
        self.head = node
    else:
        node._next = self.head
        self._head = node
```

گام۱: اضافه کردن گره به لیست

- دستورات لازم در صورتی که لیست خالی باشد:

- دستورات لازم در صورتی که لیست خالی نباشد:



پیاده‌سازی لیست تک پیوندی

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

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

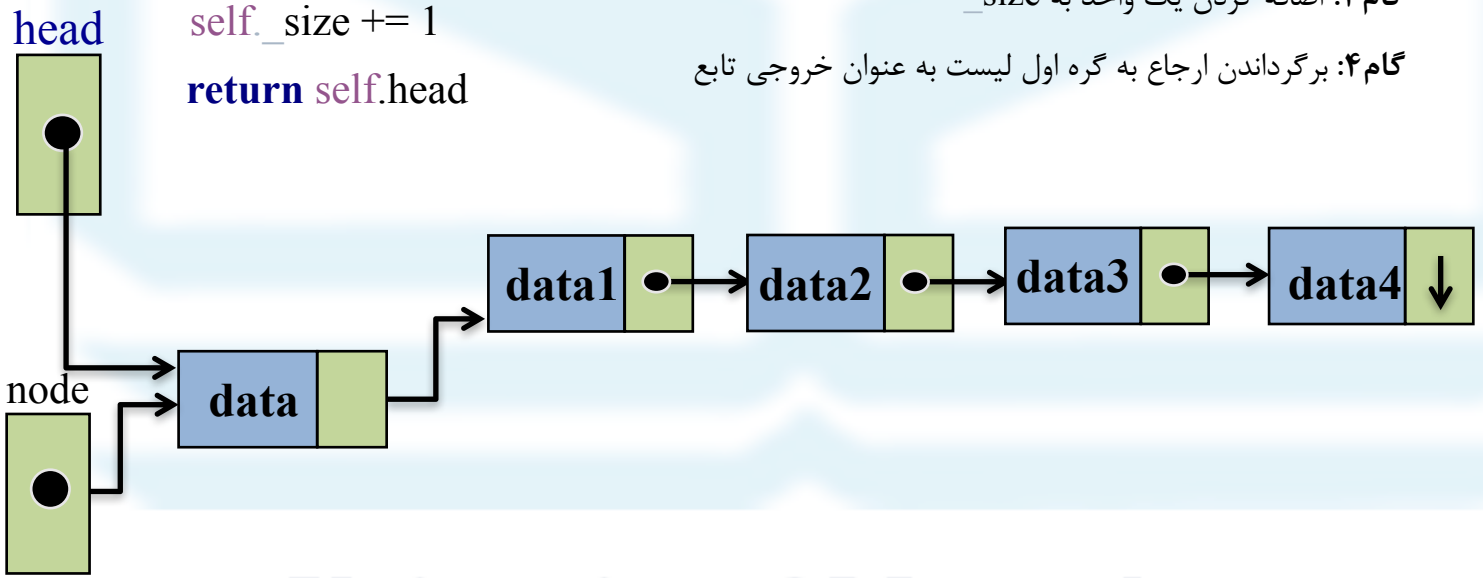
گام۱: اضافه کردن گره به لیست

- 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

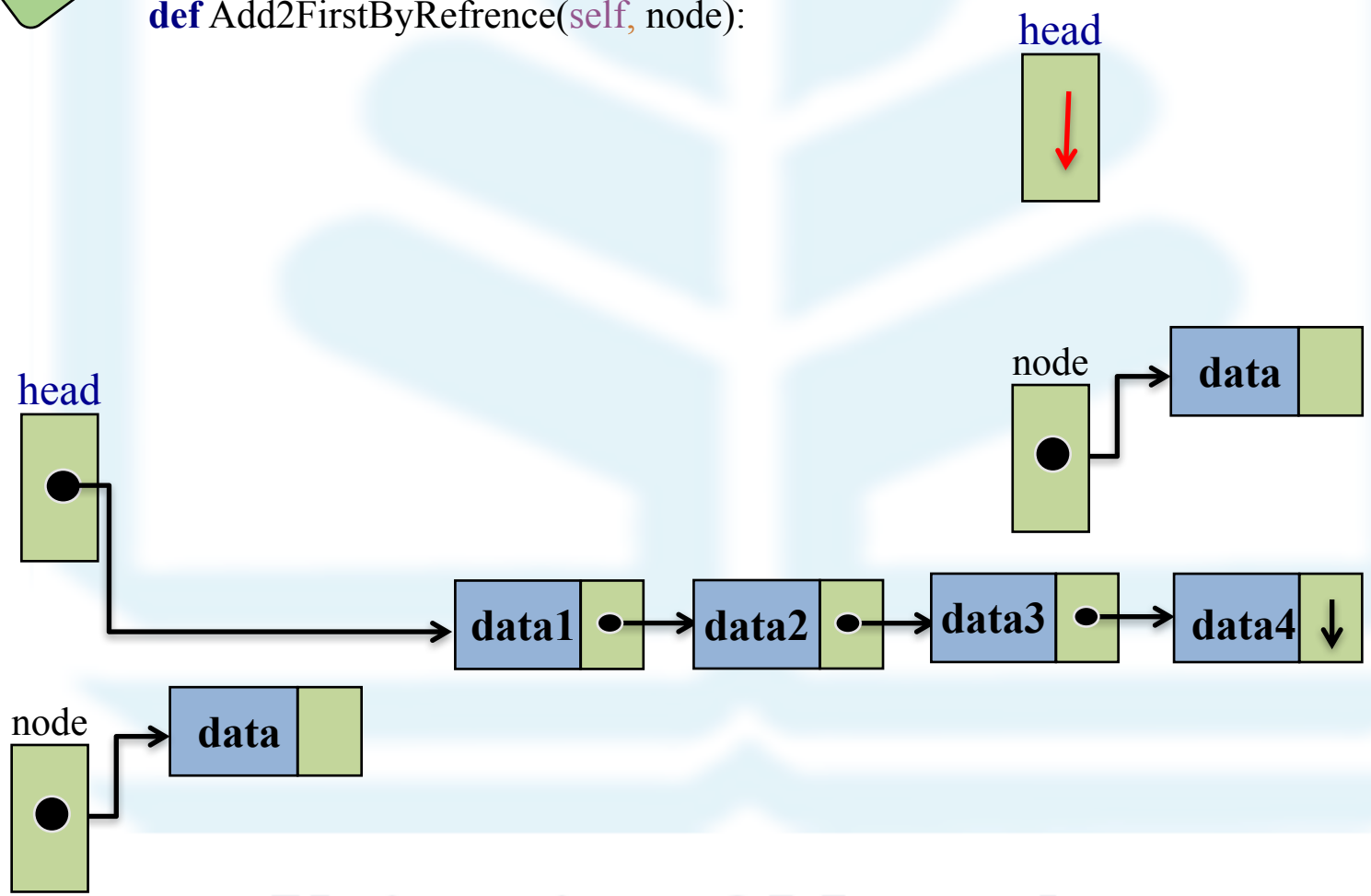


پیاده‌سازی لیست تک پیوندی

قطعه برنامه بهینه

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

```
def Add2FirstByRefrence(self, node):
```

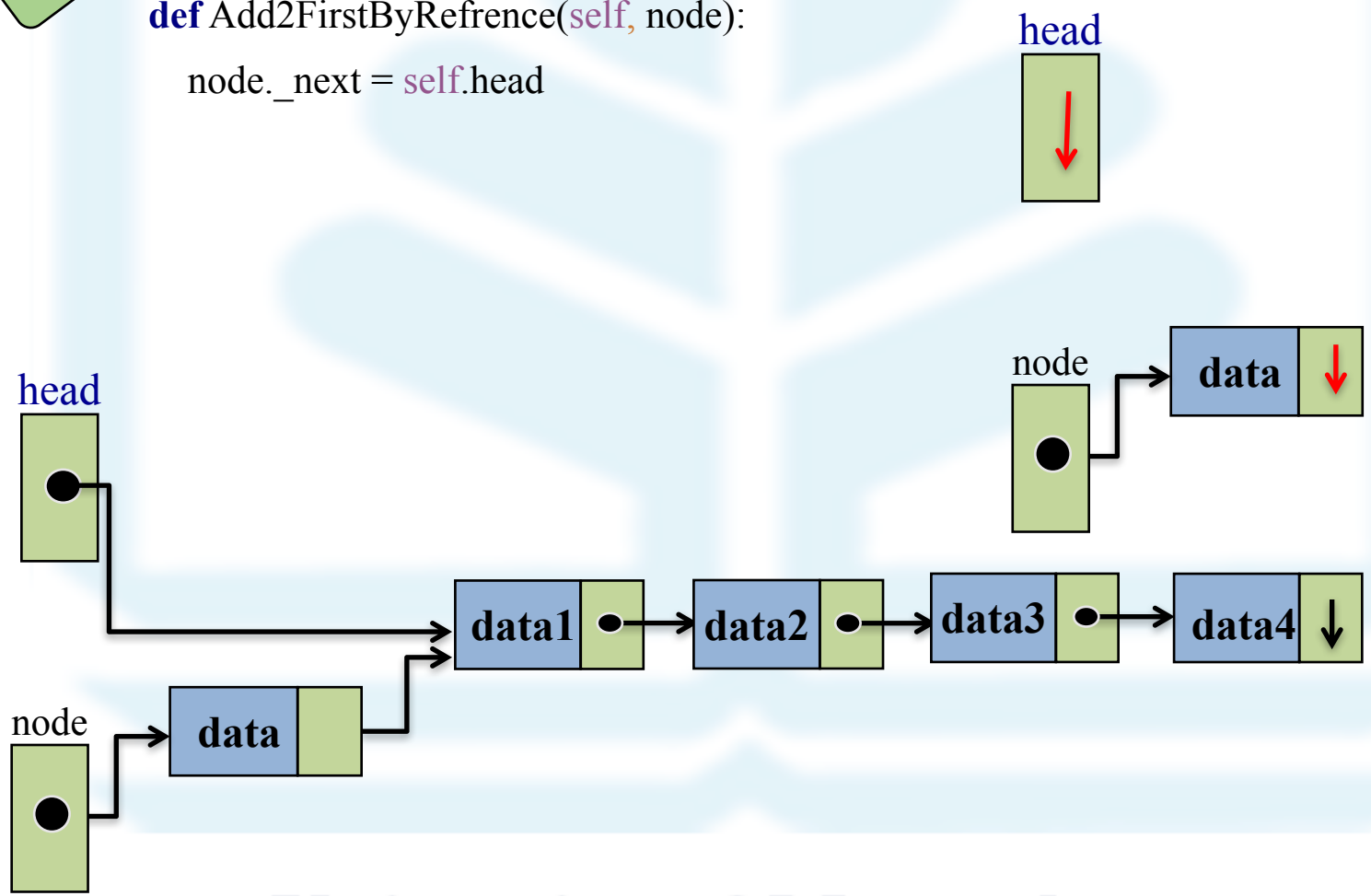


پیاده‌سازی لیست تک پیوندی

قطعه برنامه بهینه

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

```
def Add2FirstByRefrence(self, node):
    node._next = self.head
```

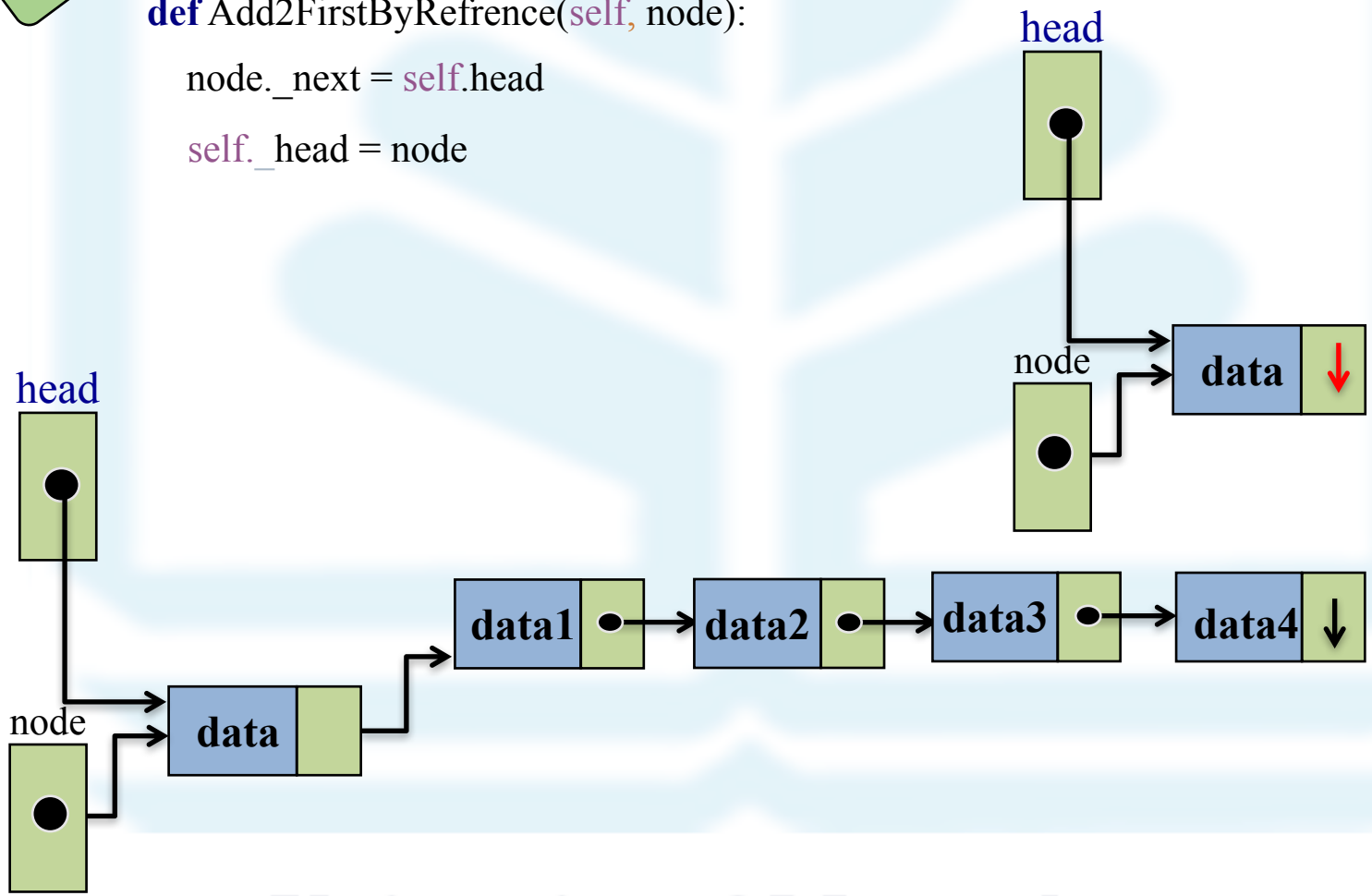


پیاده‌سازی لیست تک پیوندی

قطعه برنامه بهینه

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

```
def Add2FirstByRefrence(self, node):
    node._next = self.head
    self._head = node
```

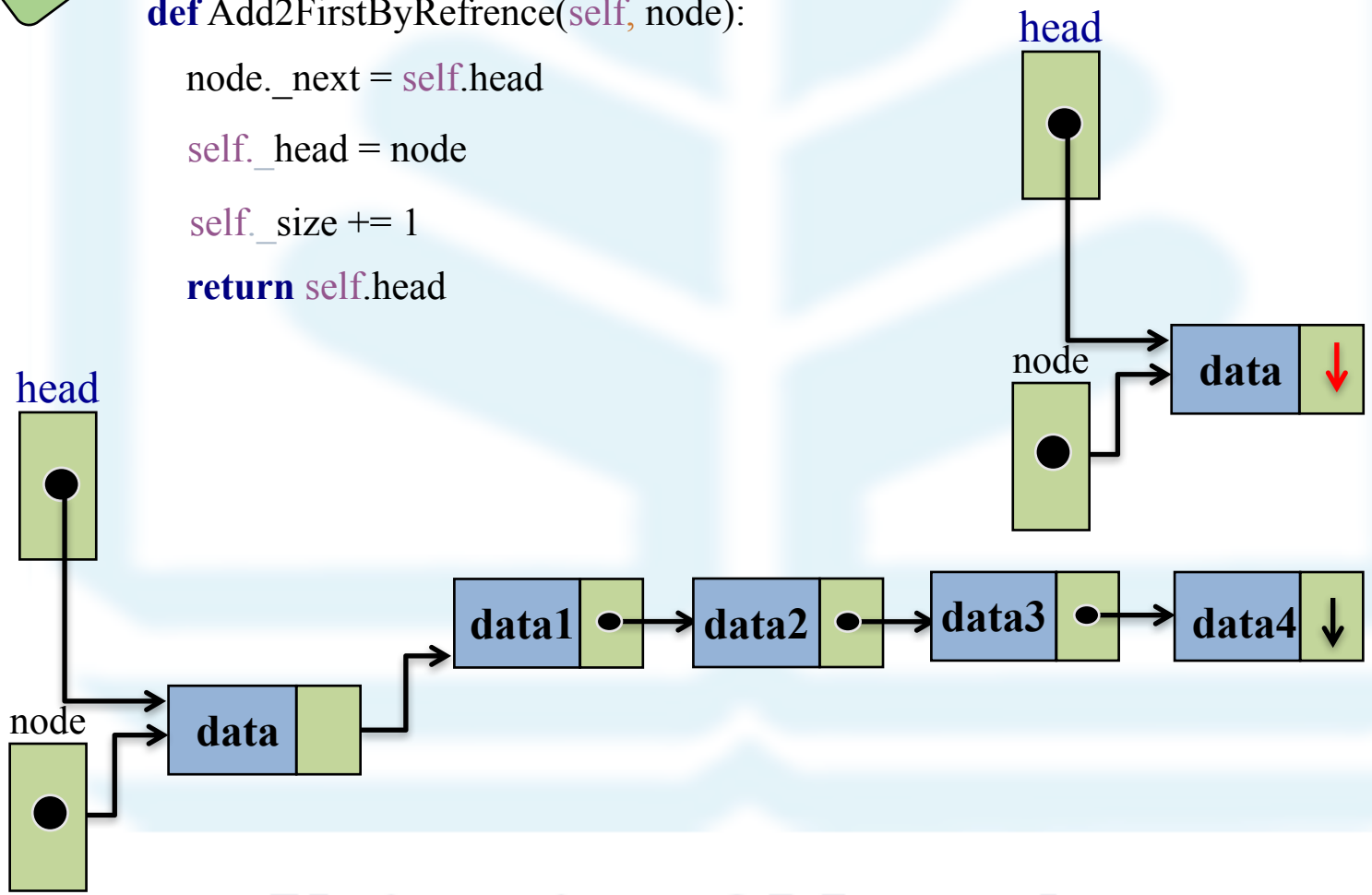


پیاده‌سازی لیست تک پیوندی

قطعه برنامه بهینه

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

```
def Add2FirstByRefrence(self, node):
    node._next = self.head
    self._head = node
    self._size += 1
    return self.head
```



پیاده‌سازی لیست تک پیوندی

قطعه برنامه بهینه‌تر

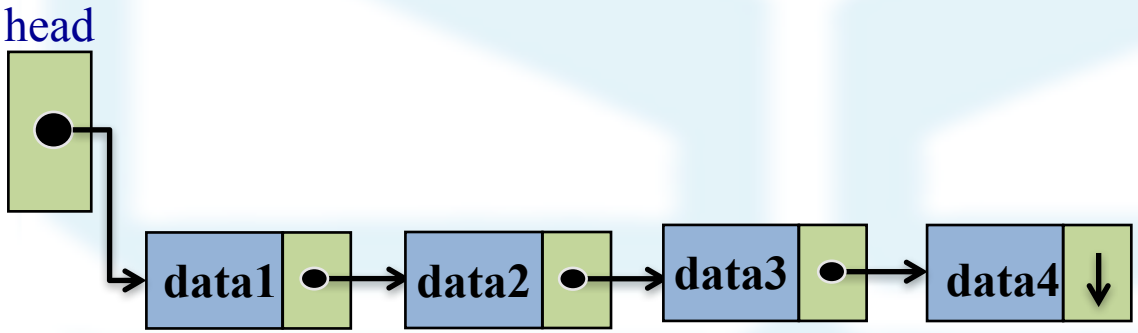
- تابع درج یک گره در ابتدای لیست:

```
def Add2First(self, inp):
    if type(self.head) == type(inp):
        inp._next = self.head
        self.head = inp
    else:
        self.head = self.SinglyLinkedListNode(inp, self.head )
    self._size += 1
    return self.head
```

پیاده‌سازی لیست تک پیوندی

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

```
def ReturnFirstNode(self):
```



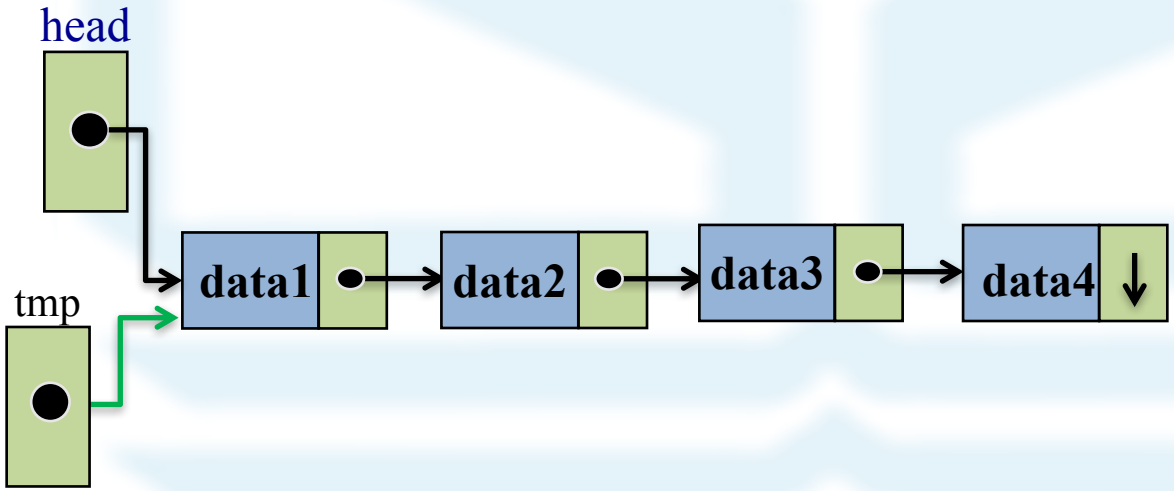
پیاده‌سازی لیست تک پیوندی

- **تابع دسترسی به اولین گره از لیست تک 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

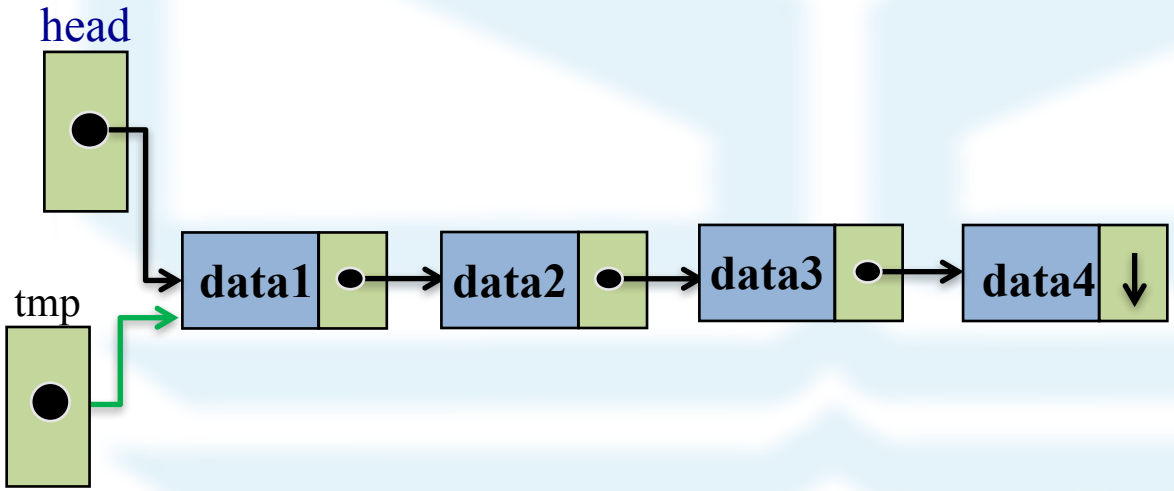


پیاده‌سازی لیست تک پیوندی

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

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

گام۱: تعریف اشاره گری به اولین گره لیست



پیاده‌سازی لیست تک پیوندی

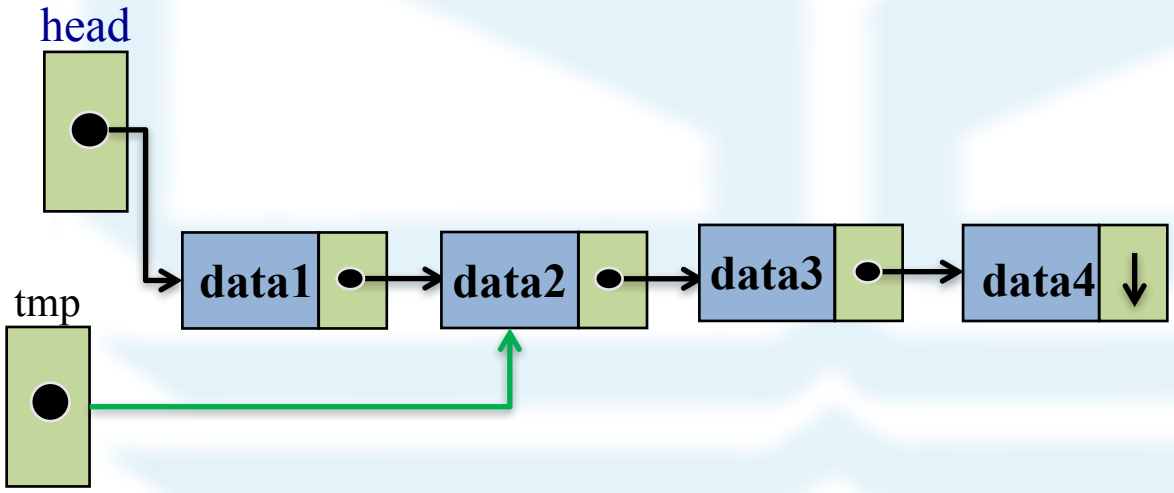
- **تابع دسترسی به دومین گره از لیست تک پیوندی با بیش از یک گره:**

```
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 تک پیوندی

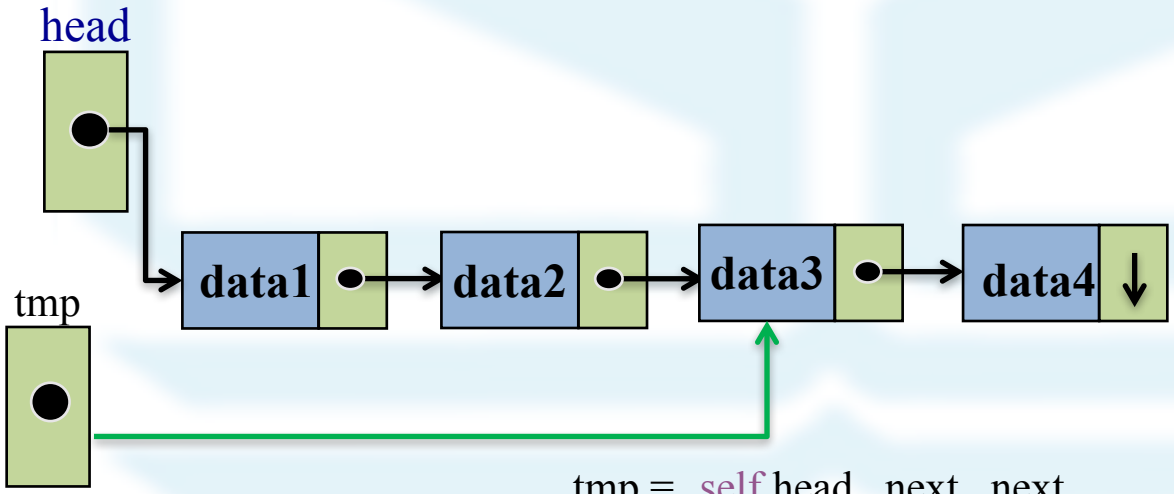
- **تابع دسترسی به سومین گره از لیست تک پیوندی با بیش از دو گره:**

```
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:



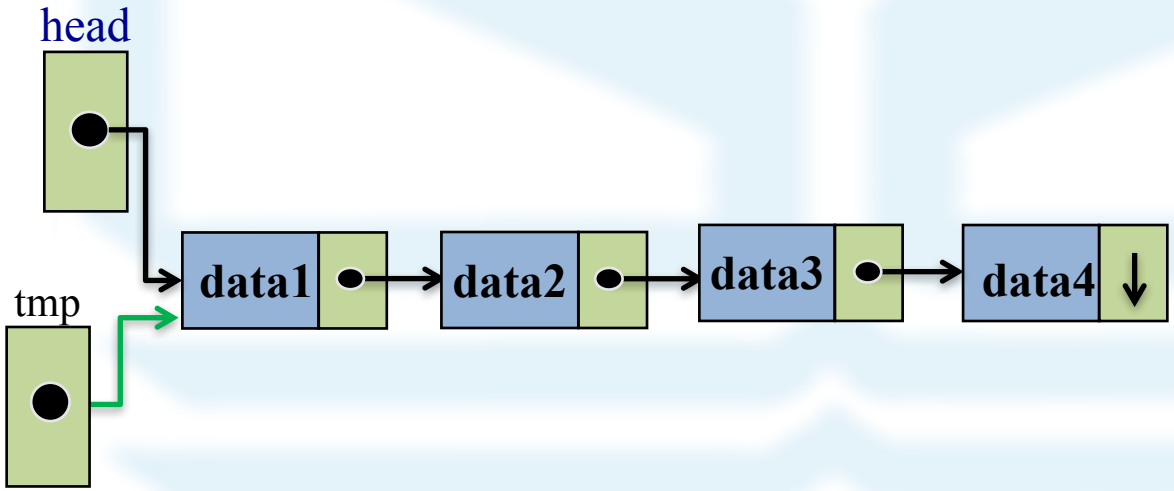
tmp = self.head._next._next

پیاده‌سازی لیست تک پیوندی

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

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

گام۱: تعریف اشاره گری به اولین گره لیست



پیاده‌سازی لیست تک پیوندی

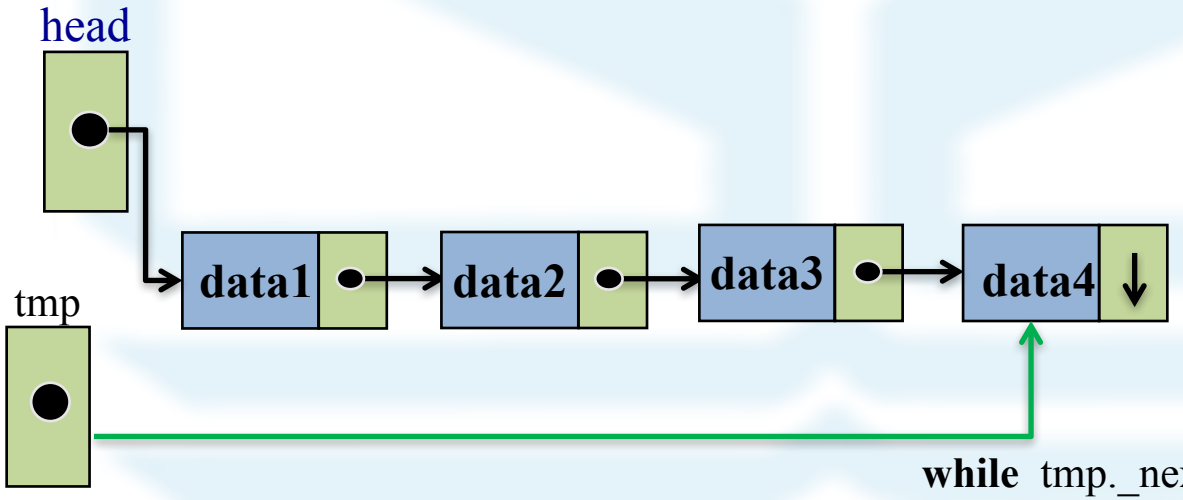
- **تابع دسترسی به آخرین گره از لیست تک پیوندی غیر خالی:**

```
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:



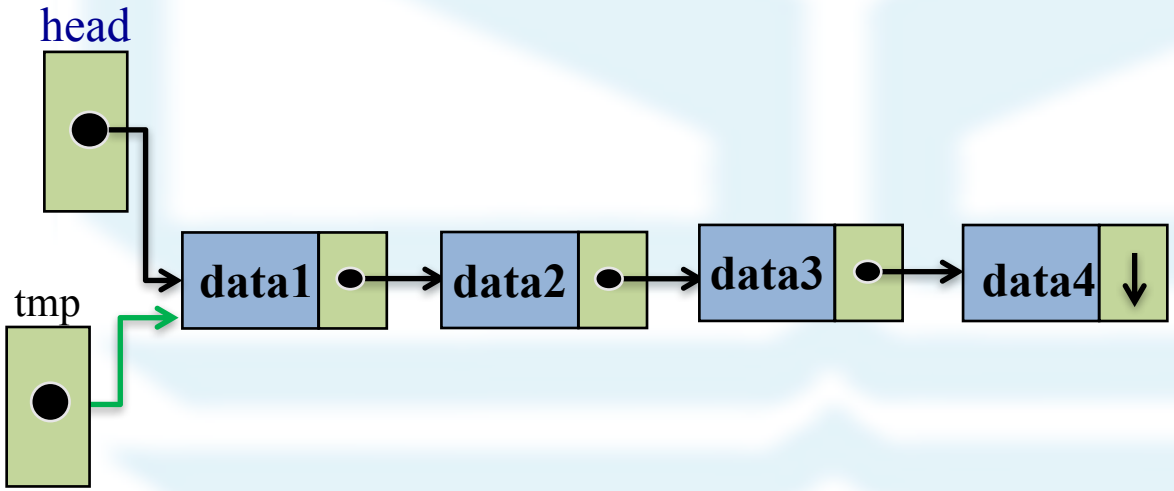

```
while  tmp._next!=None:
    tmp =  tmp._next
```

پیاده‌سازی لیست تک پیوندی

قطعه برنامه بهینه

- تابع دسترسی به آخرین گره از لیست تک پیوندی:

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

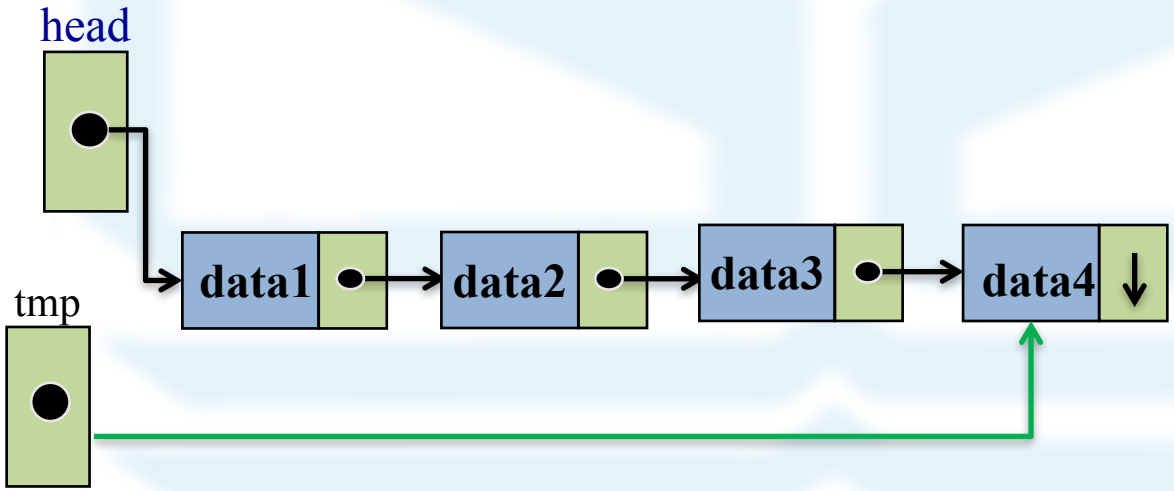


پیاده‌سازی لیست تک پیوندی

قطعه برنامه بهینه

- تابع دسترسی به آخرین گره از لیست تک پیوندی:

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

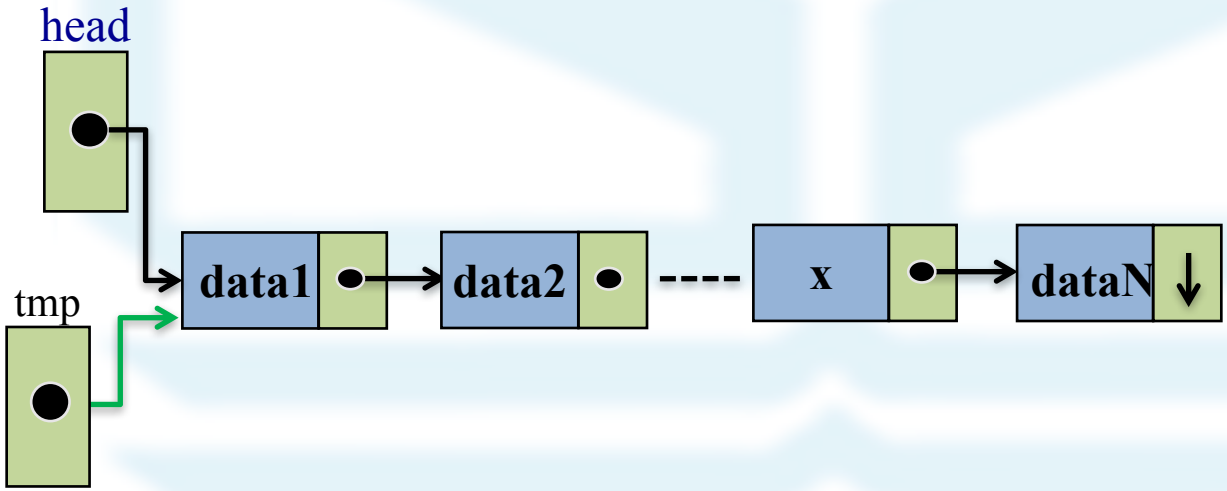


پیاده‌سازی لیست تک پیوندی

- تابع جستجوی یک گره با مقدار داده‌ای x:

گام۱: تعریف اشاره گری به اولین گره لیست

```
def SearchNodeByValue(self,x):
    tmp = self.head
```



پیاده‌سازی لیست تک پیوندی

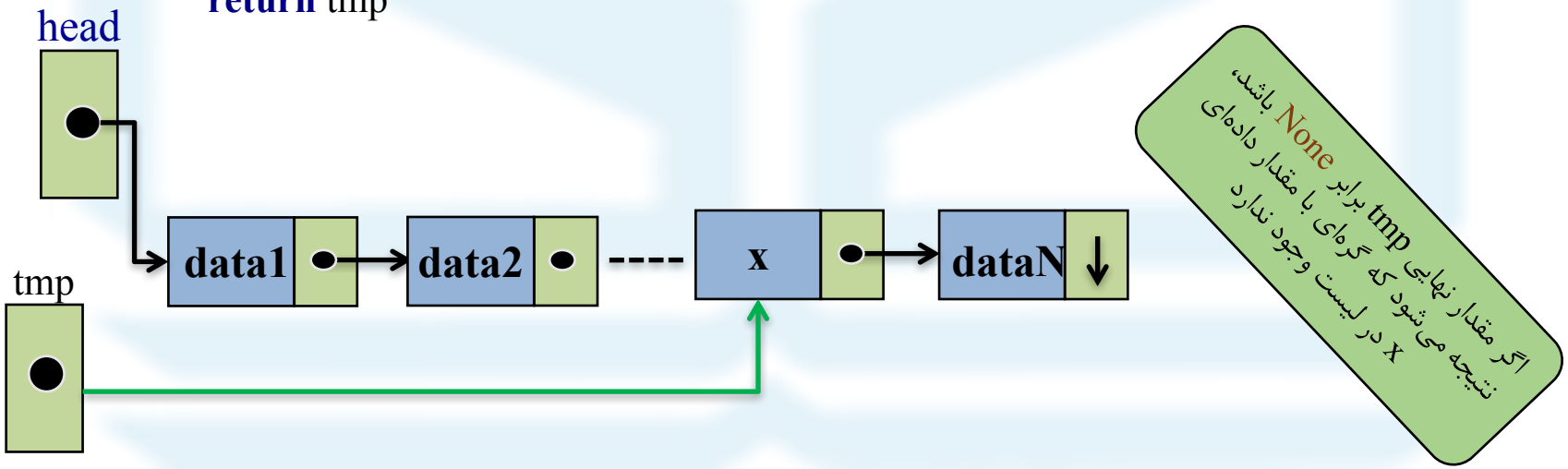
- **تابع جستجوی یک گره با مقدار داده‌ای x:**

```
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:

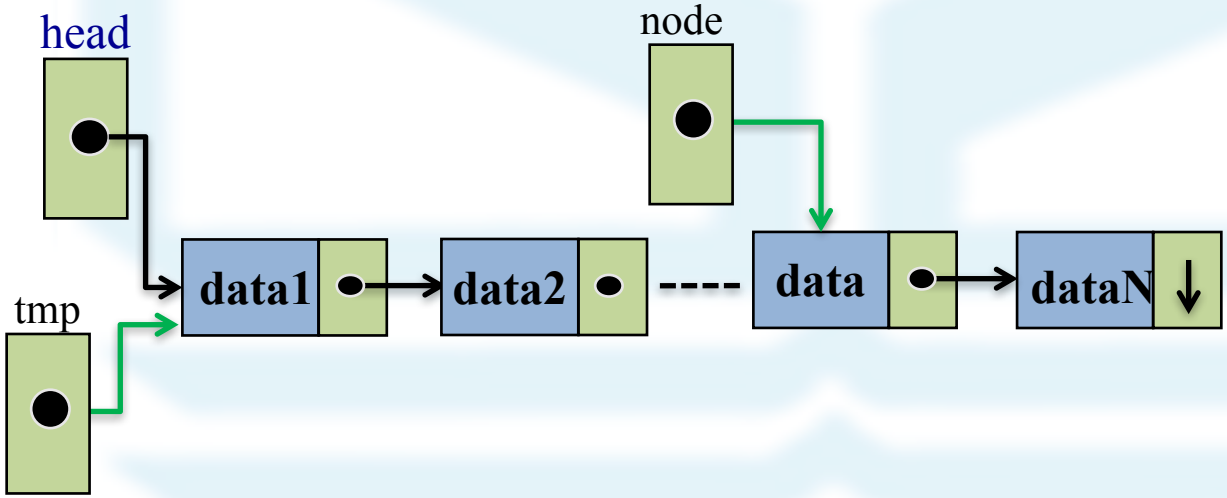


پیاده‌سازی لیست تک پیوندی

- **تابع جستجوی یک گره با ارجاع:**

```
def SearchNodeByReference(self,node):
    tmp = self.head
```

گام۱: تعریف اشاره گری به اولین گره لیست



پیاده‌سازی لیست تک پیوندی

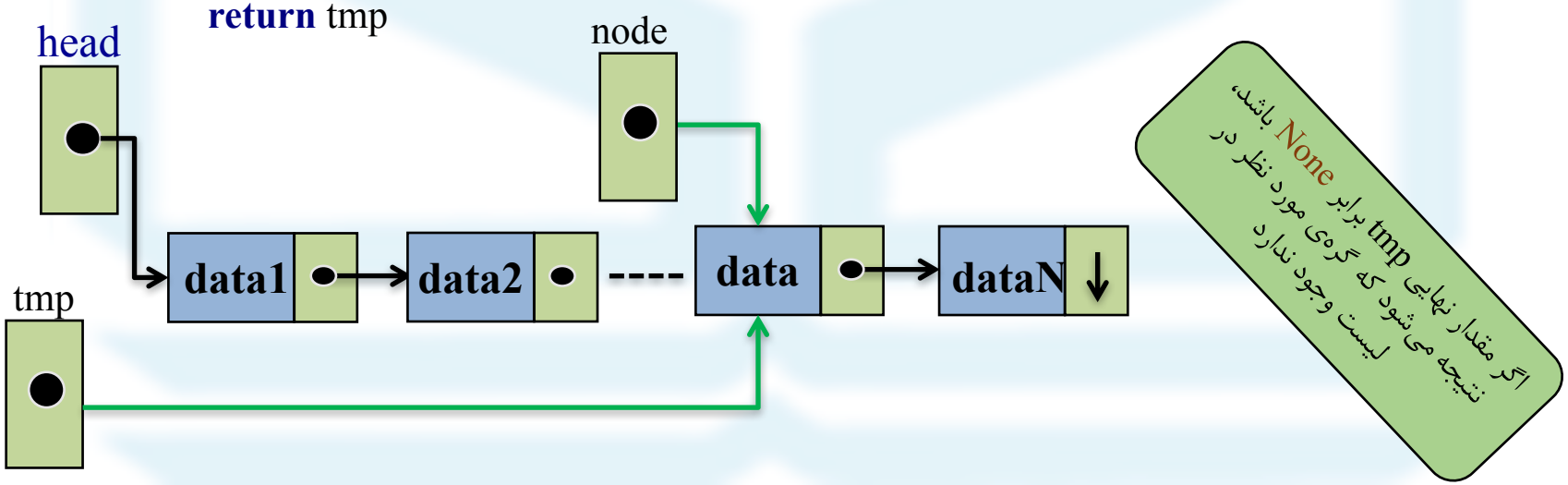
- **تابع جستجوی یک گره با ارجاع:**

```
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.

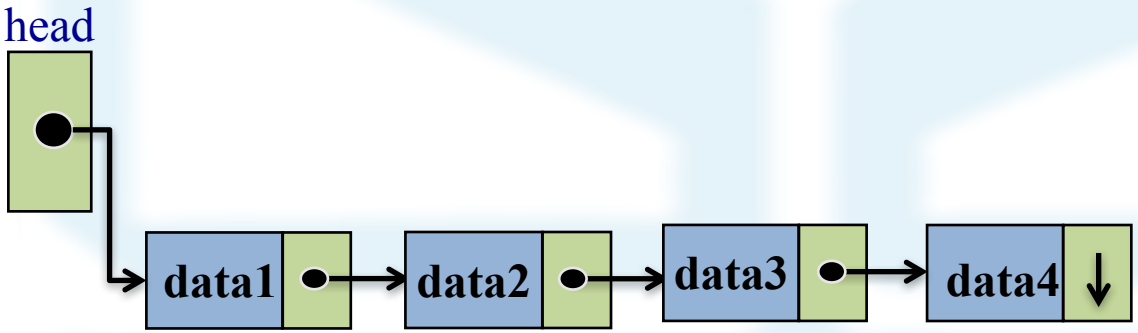
گام۳: برگرداندن ارجاع مناسب:



پیاده‌سازی لیست تک پیوندی

- تابع چاپ داده گره‌های لیست تک پیوندی:

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



data1→data2→data3→data4→