Algorithms and Data Structures for Data Science

List Implementations

CS 277

Brad Solomon

February 1, 2023

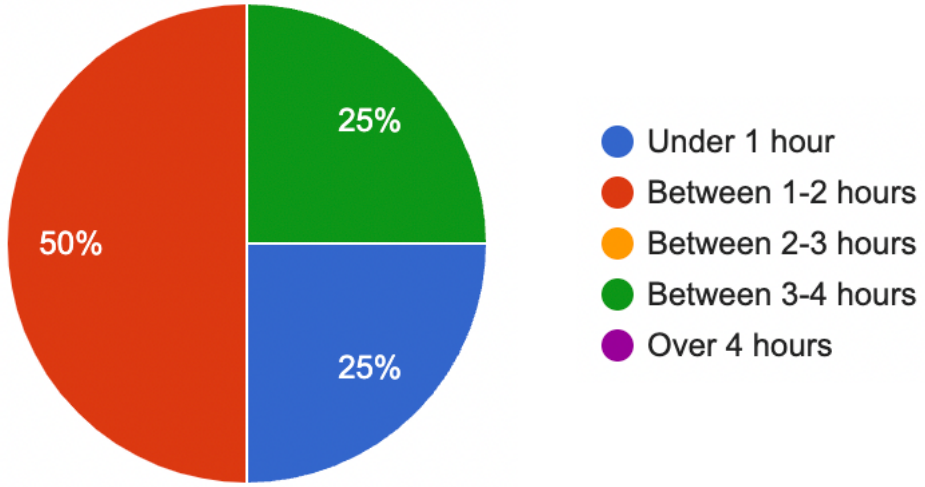
UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Lab_debug Feedback

Average score: 98%

PL average time: 52 minutes



Most but not all found class helpful for completing lab.

Lab generally improved confidence

Lab matched experience levels; request for slightly harder content

Exam 1

Exams will be proctored by the CBTF: https://cbtf.engr.illinois.edu/

(That link will have a link to **Prairietest**, where you can sign up for exam 1)

Reservations open on February 2nd @ 9 AM

You must take the exam sometime between 2/14 and 2/16!

See website for expected content:

https://courses.grainger.illinois.edu/cs277/sp2023/exams/

Learning Objectives

Conceptualize and code a linked list

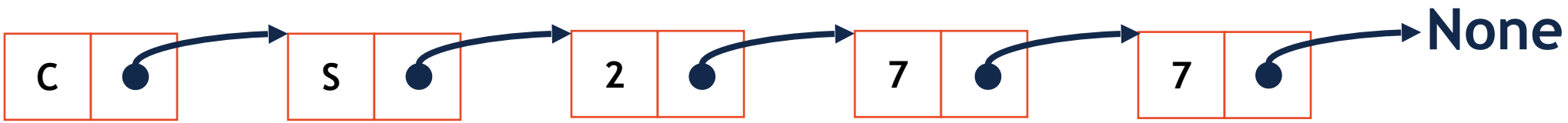
Conceptualize an array list

Compare list implementations

Linked Lists

```
1  class Node:
2      def __init__(self, data, next=None):
3          self.data = data
4          self.next = next
5
```

```
1  class linkedList:
2      def __init__(self, head=None):
3          self.head = head
4
5
```



Linked List Structure (Node vs List)

```
1 n1 = Node(3)
2 n2 = Node(5)
3 n3 = Node(7)
4
5 n1.next = n2
6 n2.next = n3
7
```

```
1 t1 = [3, None]
2 t2 = [5, None]
3 t3 = [7, None]
4
5 t1[1]=t2
6 t2[1]=t3
7
```

```
1 curr = n1
2 print(curr.next.next.data)
3
```

```
1 curr = t1
2 print(t1[1][1][0])
3
```

Class Linked List

```
1  class Node:
2      def __init__(self, data, next=None):
3          self.data = data
4          self.next = next
5
6  class linkedList:
7      def __init__(self, head=None):
8          self.head = head
9
10     def __str__(self):
11
12     def __len__(self):
13
14     def __getitem__(self):
15
16     def add(self):
17
18     def insert(self):
19
20     def delete(self):
21
22     def remove(self):
23
```

Linked List Add

```
1  ll = linkedList()
2
3  for i in range(3):
4      ll.add(i)
5
```

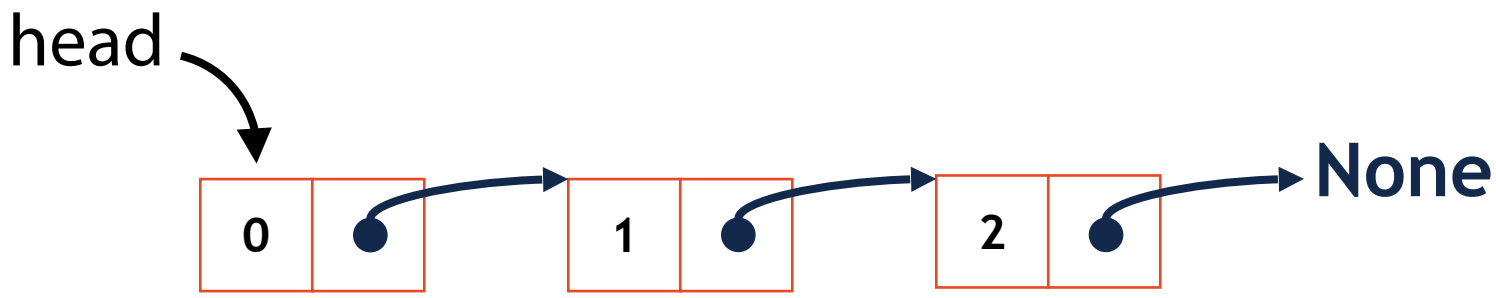
Linked List Add

```
1  ll = linkedList()
2
3  for i in range(3):
4      ll.add(i)
5
```

```
1  class linkedList:
2      def add(self, data):
3          temp = self.head
4          self.head = Node(data, temp)
5
```

Linked List Length

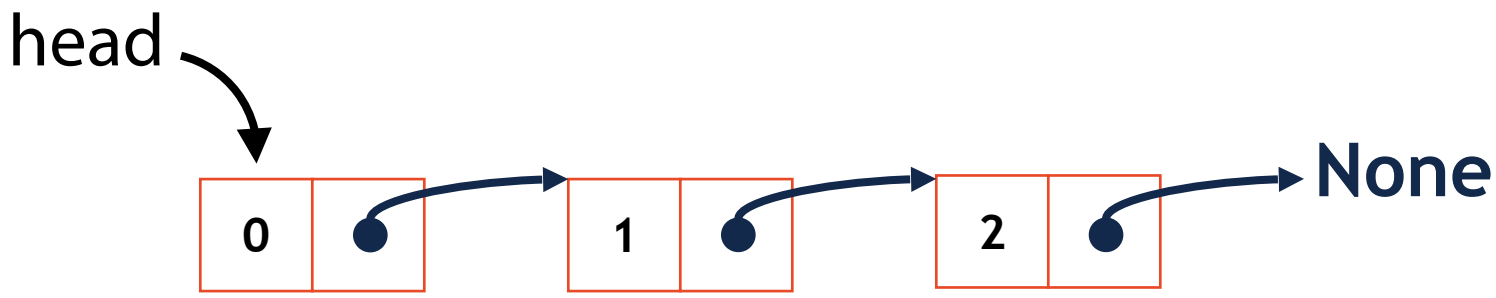
```
1  ll = linkedList()
2  for i in range(3):
3      ll.add(i)
4
5  print(len(ll))
```



Linked List Length

```
1  ll = linkedList()
2  for i in range(3):
3      ll.add(i)
4
5  print(len(ll))
```

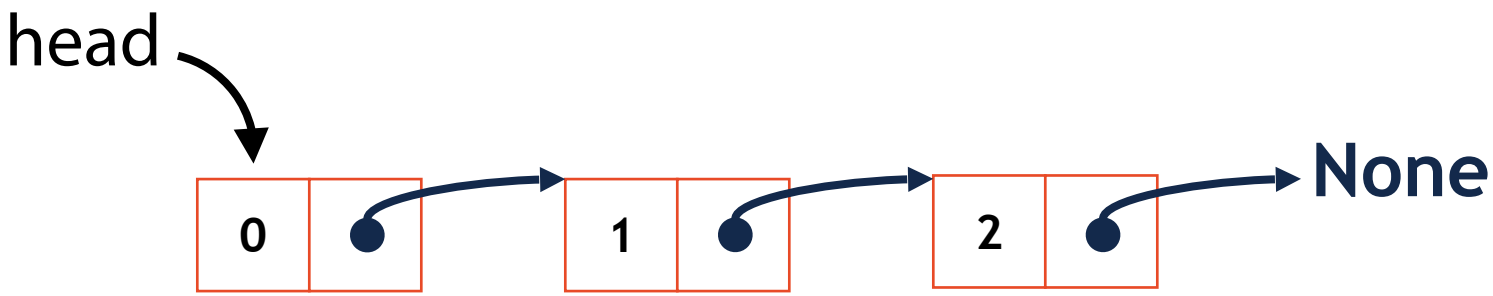
```
1  class linkedList:
2      def __len__(self):
3          i = 0
4          curr = self.head
5          while(curr):
6              curr = curr.next
7              i+=1
8          return i
```



Linked List Print

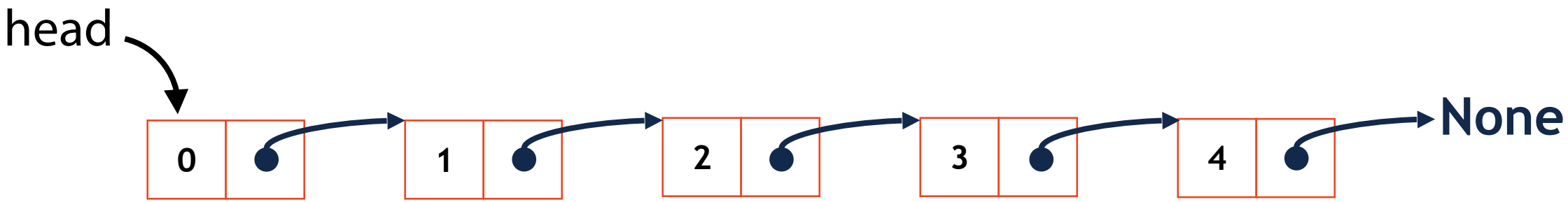
```
1  ll = linkedList()
2  for i in range(3):
3      ll.add(i)
4
5  print(ll)
```

```
1  class linkedList:
2      def __str__(self):
3          curr = self.head
4          out="["
5          while(curr):
6              out+="{},".format(curr.data)
7              curr = curr.next
8
9          if out[-1]==",":
10             out = out[:-1]
11         out = out +"]\n"
12         return out
13
```



Linked List __getitem__()

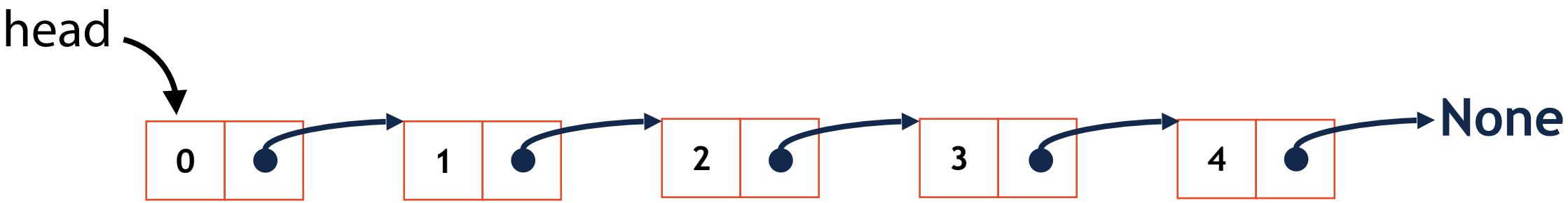
```
1 ll = linkedList()
2 for i in range(5):
3     ll.add(i)
4
5 print(ll[3])
```



Linked List __getitem__()

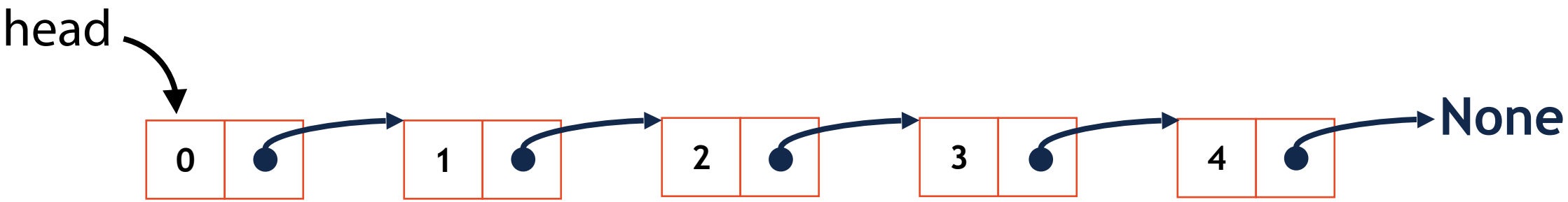
```
1  ll = linkedList()
2  for i in range(5):
3      ll.add(i)
4
5  print(ll[3].data)
```

```
1  def __getitem__(self, pos):
2      curr = self.head
3
4      i = 0
5      while(curr and i < pos):
6          curr = curr.next
7          i+=1
8
9      if i == pos:
10         return curr
11     else:
12         raise ValueError("Out of bounds")
13         return None
```



Linked List insert()

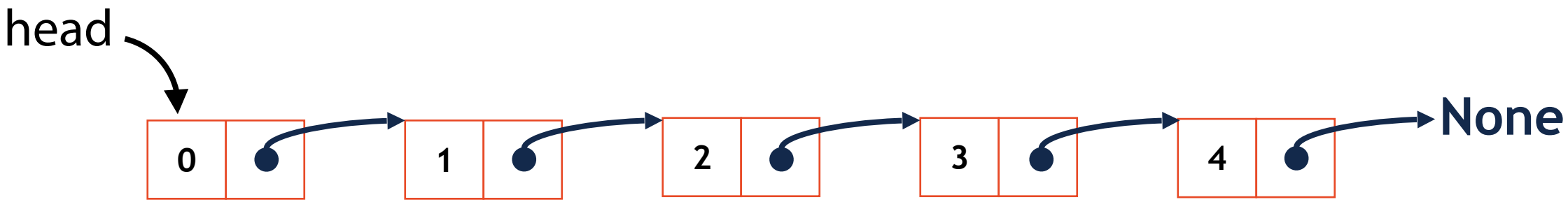
```
1 for i in range(5):
2     ll.add(i)
3
4 ll.insert("Value", 2)
5 print(ll)
```



Linked List insert()

```
1  for i in range(5):
2      ll.add(i)
3
4  ll.insert("Value", 2)
5  print(ll)
```

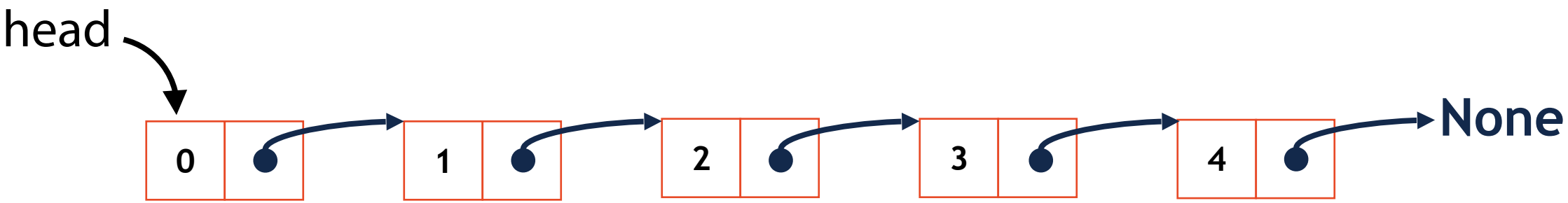
```
1
2
3
4
5
6
7
```



Linked List insert()

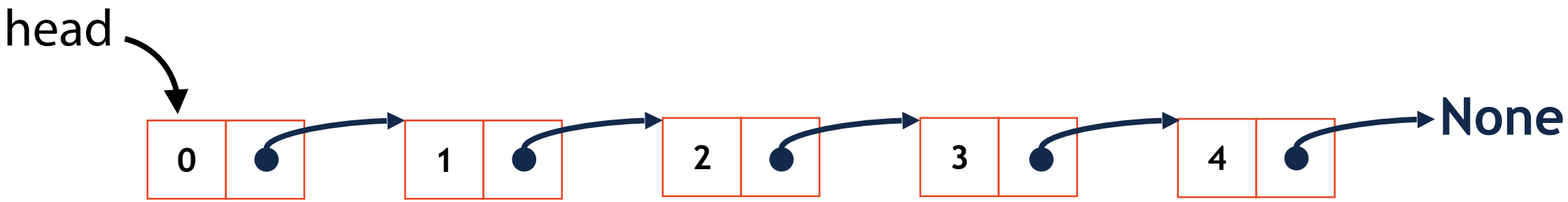
```
1  for i in range(5):
2      ll.add(i)
3
4  ll.insert("Value", 2)
5  print(ll)
```

```
1  def insert(self, data, pos=0):
2      if (pos == 0):
3          self.add(data)
4      else:
5          prev = self.__getitem__(pos-1)
6          temp = prev.next
7          prev.next = Node(data,temp)
```



Linked List delete()

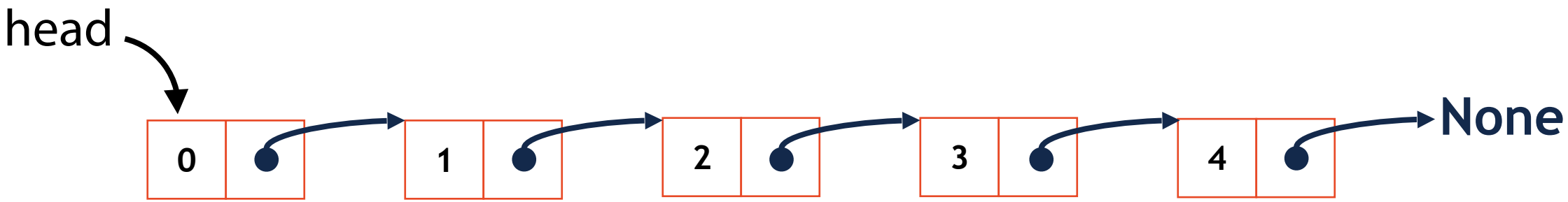
```
1 for i in range(5):
2     ll.add(i)
3
4 ll.delete(1)
5 print(ll)
```



Linked List delete()

```
1  for i in range(5):
2      ll.add(i)
3
4  ll.delete(1)
5  print(ll)
```

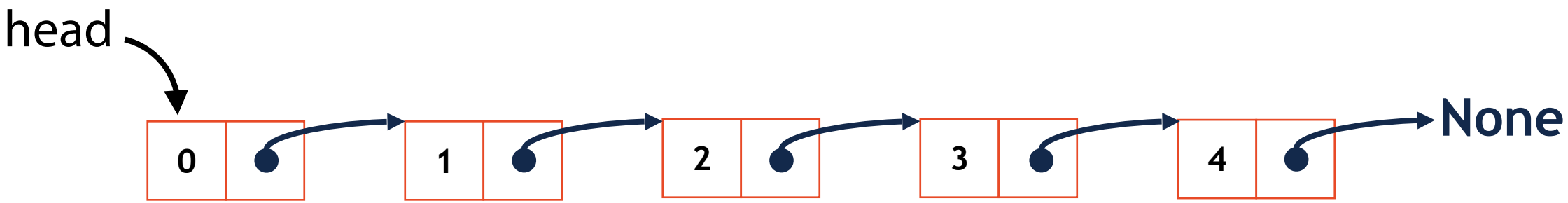
```
1
2
3
4
5
6
7
```



Linked List delete()

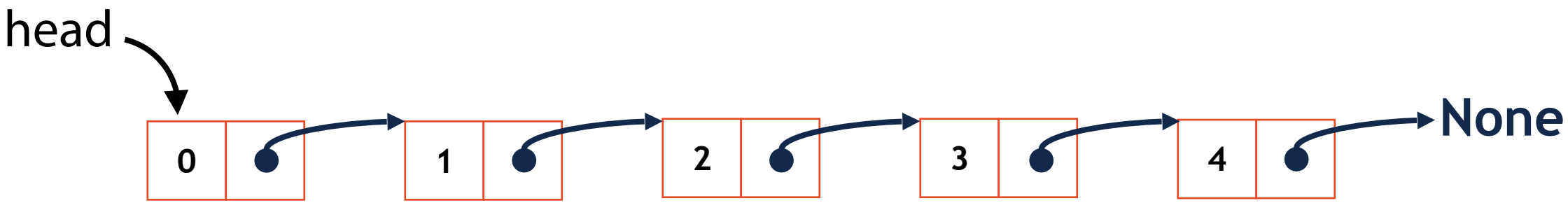
```
1  for i in range(5):
2      ll.add(i)
3
4  ll.delete(1)
5  print(ll)
```

```
1  def delete(self, i):
2      if i == 0:
3          self.head = self.head.next
4      else:
5          prev = self.__getitem__(i-1)
6          prev.next = prev.next.next
7
```



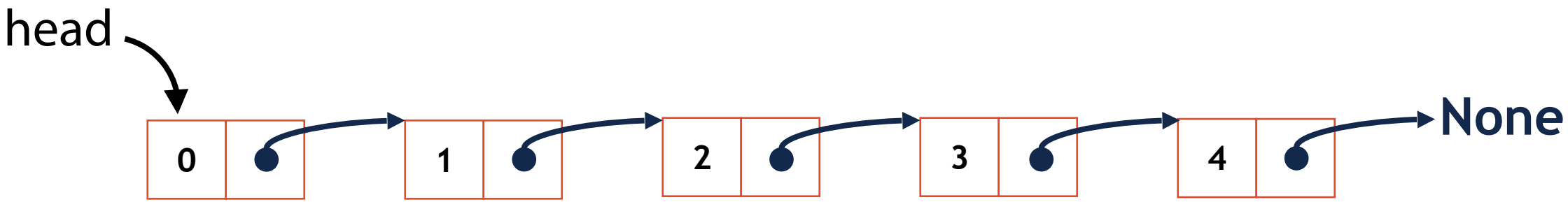
In-Class Exercise: remove()

```
1  ll = linkedList()
2
3  for i in range(5):
4      ll.add(i)
5      ll.add(i)
6
7  ll.remove(4)
8
9
```



In-Class Exercise: find()

```
1  #initialize ll
2
3  node = ll.find("B")
4
5  if(node):
6      print("Exists!")
7  else:
8      print("Doesnt exist!")
9
```



Linked List Pros and Cons

What do we care about when we write code?