

# Algorithms and Data Structures for Data Science

## Lists Implementation Day 2

CS 277

February 7, 2024

Brad Solomon



UNIVERSITY OF  
**ILLINOIS**  
URBANA - CHAMPAIGN

Department of Computer Science

# Exam 0: 2/13 — 2/15

Two coding questions in 50 minutes

Practice exam has four total questions

Content is Python fundamentals and lists

# Office Hours Time Change

Not available Thursday @ 11 AM

For this week only: Friday @ 10 - 11 AM

# MP Generate: Part 3

```
1 for qlen in [5, 10, 15, 20, 25]:
2     print("For query length: {}".format(qlen))
3     time_list = []
4     for pow in [15, 16, 17, 18, 19, 20]:
5         rowcount = 2**pow
6
7         query =
8 random.sample(string.ascii_uppercase, qlen)
9         random.shuffle(query)
10
11         start = timeit.default_timer()
12         checkPathExists("data/graph_r{
13 _c8.txt".format(rowcount), query)
14         end = timeit.default_timer()
15
16         time = end-start
17
18         time_list.append(time)
19
20     print("Runtime with increasing edge
21 count:")
22     print(time_list)
23
24
25
```

1. The x-dimension should be the number of rows in the file. (Each plotted line should be run on the same files)
2. The y-dimension should be the timing information for a randomly generated query on a particular file.
3. Each query length should have its own line, which should be labeled.

# Learning Objectives

Discuss list implementation strategies in the context of Big O

Build a linked list implementation in Python

Extend lists to multi-dimensions using built-in and NumPy

# Big O Practice: Reading code

```
1 def convert_1D_to_2D(inList, rowSize):
2     listLen = len(inList)
3     numRows = math.ceil(listLen/rowSize)
4
5     outList = []
6     count = 0
7
8     ops = 0
9     for i in range(numRows):
10         tempList = []
11
12         for j in range(rowSize):
13
14             if count >= listLen:
15                 tempList.append(-1)
16             else:
17                 tempList.append(inList[count])
18
19             ops+=1
20             count+=1
21
22         outList.append(tempList)
23
24     print(ops)
25     return outList
```

# (Theoretical) List Implementations

1. Array List

2. Linked List

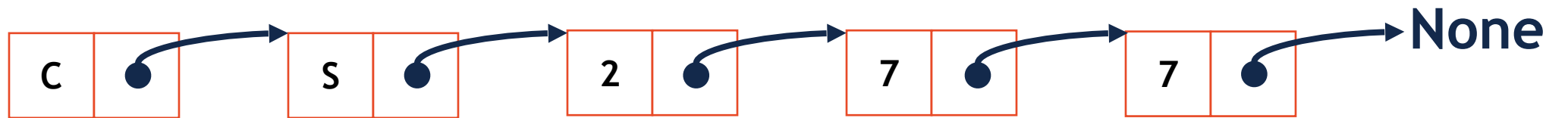
# Array Implementation

	Array	Singly Linked List
Look up given an input <b>position</b>		
Search given an input <b>value</b>		
Insert/Remove at <b>front</b>		
Insert/Remove at <b>arbitrary</b> location		



# Linked List Node

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



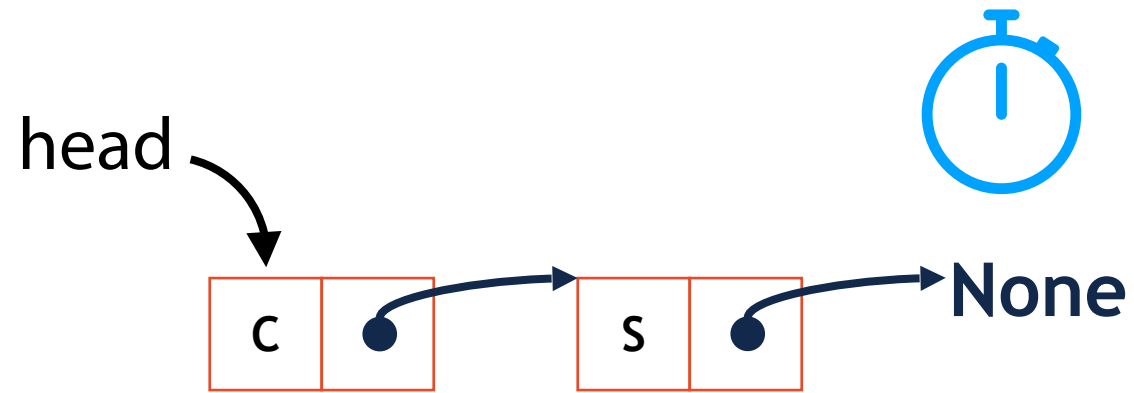
# Linked List Node

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

```
1 n1 = Node(3)  
2 n2 = Node(5)  
3 n3 = Node(7)  
4  
5 n1.next = n2  
6 n2.next = n3  
7  
8 curr = n1  
9 print(curr.next.next.data)
```

# 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  
4  
5  
6  
7
```

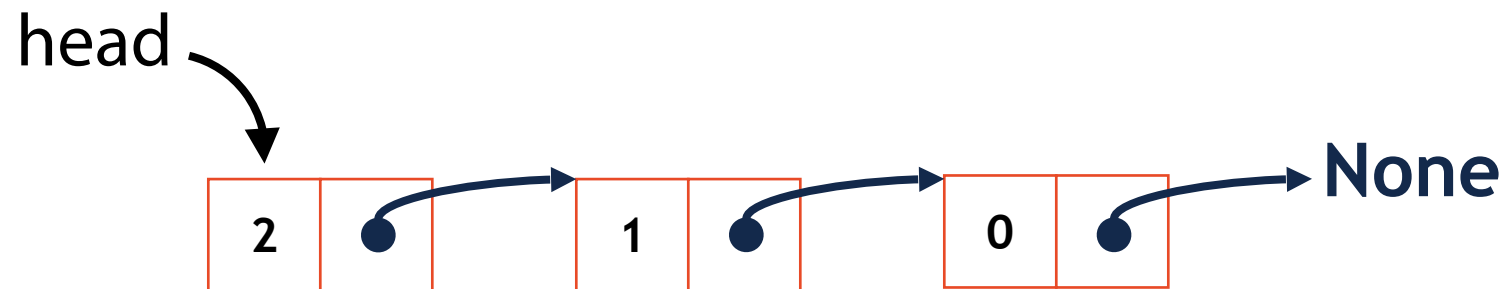
# Linked List: add()

```
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: `__len__()`

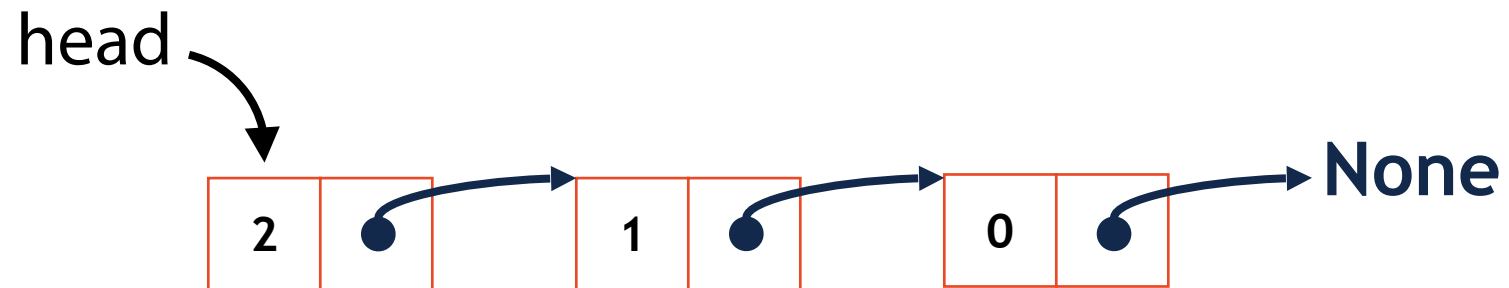
```
1 ll = linkedList()  
2 for i in range(3):  
3     ll.add(i)  
4  
5 print(len(ll))
```



# Linked List: `__len__()`

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

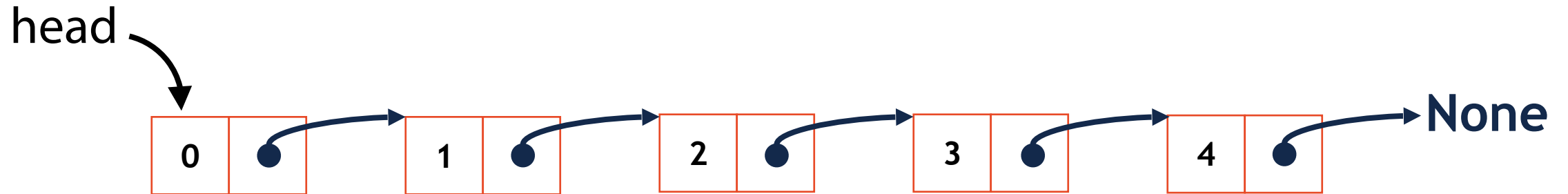
```
1 class linkedList:  
2     def __len__(self):  
3         i = 0  
4         curr = self.head  
5         while(curr):  
6             curr = curr.next  
7             i+=1  
8         return i
```





# In-Class Exercise: Linked List `__getitem__()`

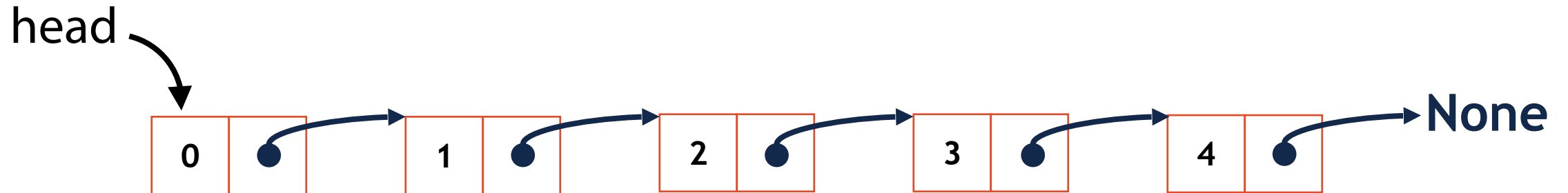
```
1 ll = linkedList()  
2 for i in range(5):  
3     ll.add(i)  
4  
5 print(ll[3])
```



# In-Class Exercise: Linked List `__getitem__()`

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

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13
```

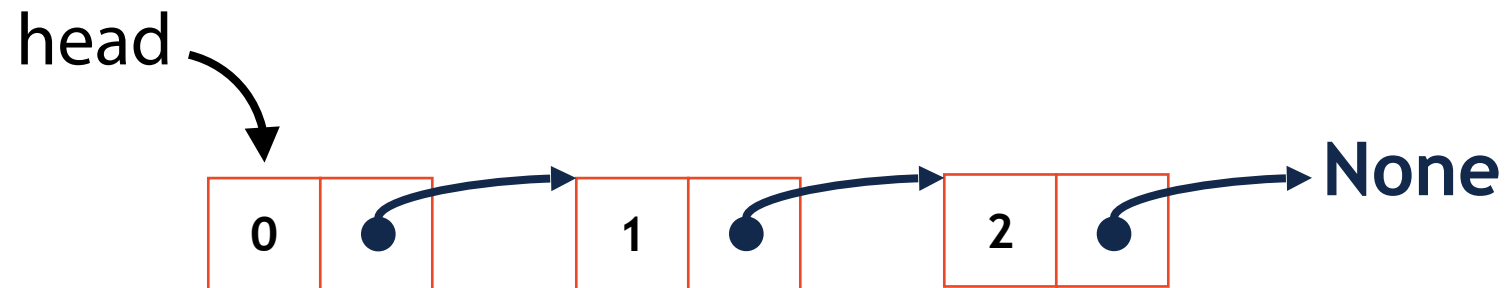


# Linked List: `__str__()`



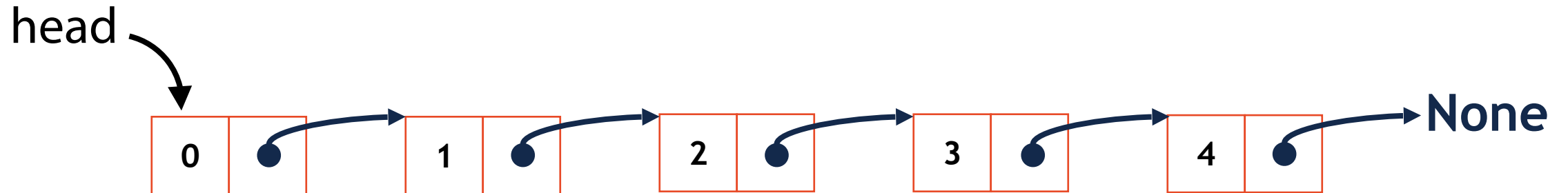
```
1 ll = linkedList()
2 for i in range(3):
3     ll.add(i)
4
5 print(ll)
```

```
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
```



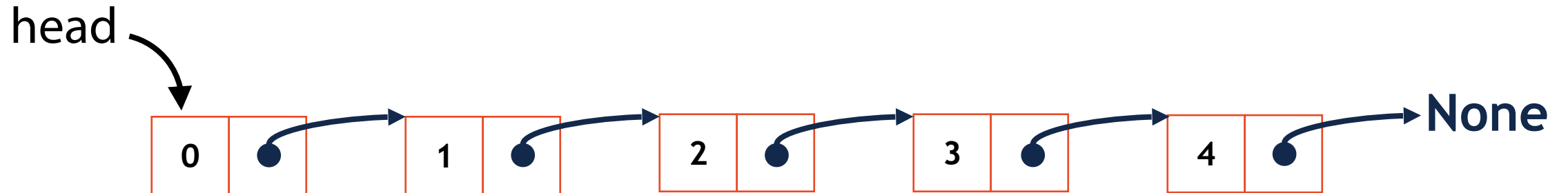
# On your own: Linked List find()

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



# Linked List: insert()

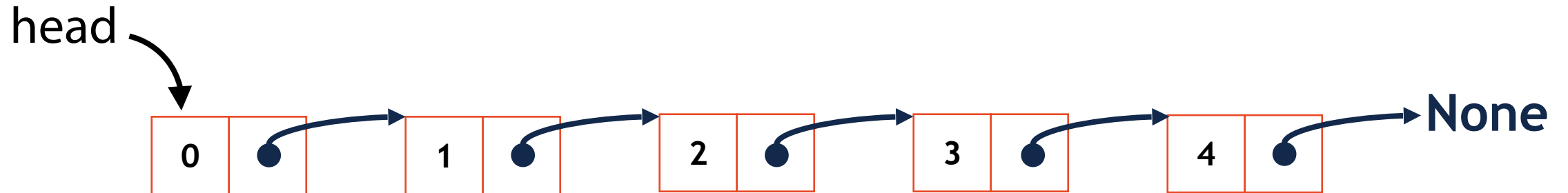
```
1 for i in range(5):  
2     ll.add(i)  
3  
4 ll.insert("Value", 2)  
5 print(ll)
```



# In-Class Exercise: Linked List insert()

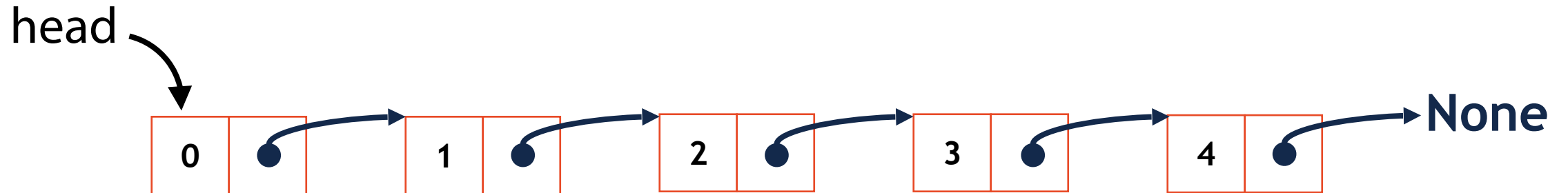
```
1 for i in range(5):  
2     ll.add(i)  
3  
4 ll.insert("Value", 2)  
5 print(ll)
```

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



# Linked List: delete()

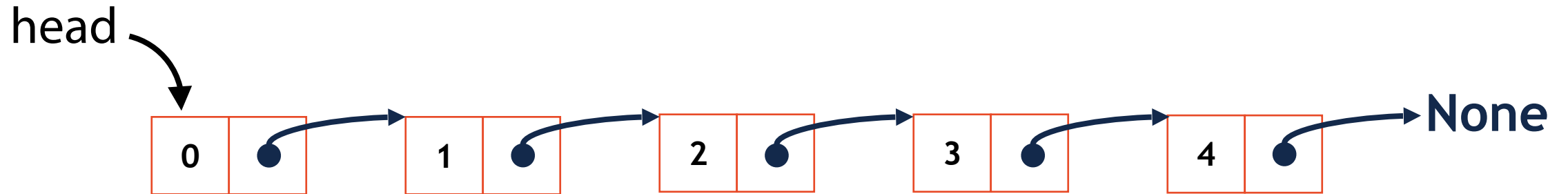
```
1 for i in range(5):  
2     ll.add(i)  
3  
4 ll.delete(1)  
5 print(ll)
```



# Linked List: delete()



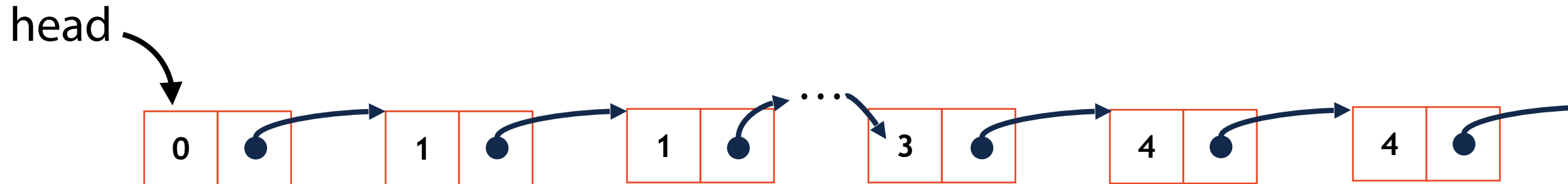
```
1 for i in range(5):  
2     ll.add(i)  
3  
4 ll.delete(1)  
5 print(ll)
```





# Linked List: remove()

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

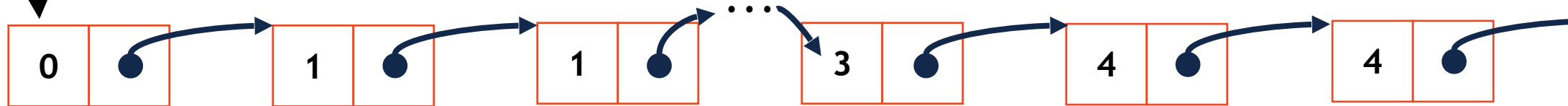


# Optional Exercise: remove()

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

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

head



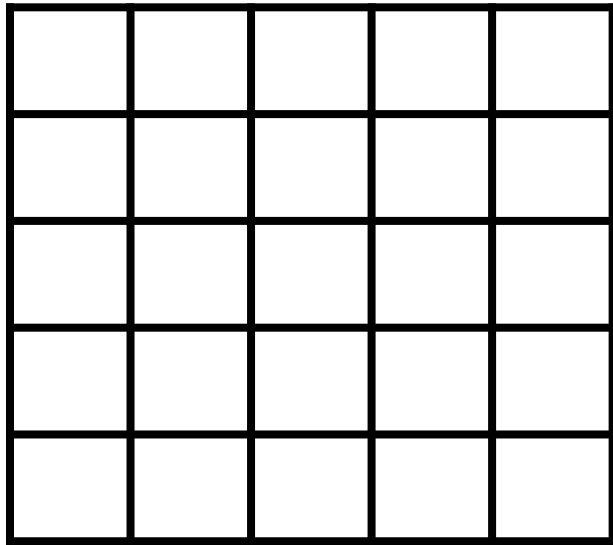
# Array Implementation



	Singly Linked List	Array
Look up given an input <b>position</b>		
Search given an input <b>value</b>		
Insert/Remove at <b>front</b>		
Insert/Remove at <b>arbitrary</b> location		

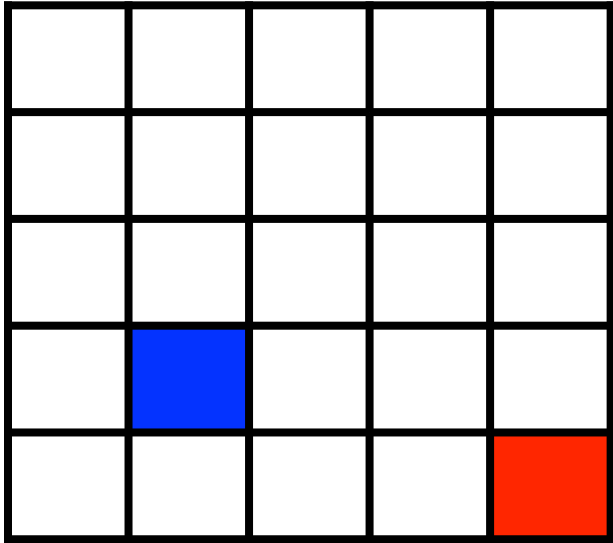
# Programming Toolbox: Multidimensional Lists

How can we make a matrix in Python?



# Programming Toolbox: Multidimensional Lists

How is a matrix in Python indexed?



# Python 2D lists


```
1 outerList = []
2
3 for i in range(5):
4     innerList = []
5
6     for j in range(5):
7         innerList.append(i+j)
8
9     outerList.append(innerList)
10
11 print(outerList)
12
13 print(outerList[3][1])
14
15
16
17
18
```

# Python 2D lists

0	1	2	3	4
1	2	3	4	5
2	3	4	5	6
3	4	5	6	7
4	5	6	7	8

```
1 outerList = []
2
3 for i in range(5):
4     innerList = []
5
6     for j in range(5):
7         innerList.append(i+j)
8
9     outerList.append(innerList)
10
11 print(outerList)
12
13 print(outerList[3][1])
14
15
16
17
18
```

# Programming Toolbox: NumPy

NumPy is optimized for multidimensional arrays of numbers

```
1 import numpy as np
2
3 # Convert list to np list
4 n1 = np.array([1, 2, 3, 4, 5, 6])
5 print(n1)
6
7 print(n1.shape)
8
9 # Modify list shape
10 n12 = n1.reshape(3, 2)
11
12 print(n1)
13 print(n12)
14
15 # Create a new list
16 n13 = np.arange(15).reshape(5, 3)
17 n14 = np.zeros((2, 5))
18
19 print(n13)
20 print(n14)
21
22
23
```



# Programming Toolbox: NumPy

Basic operations are applied **elementwise** (to each item of a list)

```
1 n1 = np.arange(4).reshape(2, 2)
2
3 print(n1)
4
5 n12 = n1 * 2
6
7 print(n12)
8
9 # Matrix multiplication
10 # 0*0+1*4      0*0+1*6
11 # 2*0+3*4      2*2+3*6
12 print(n1.dot(n12))
```

Explore on your own: <https://numpy.org/devdocs/>