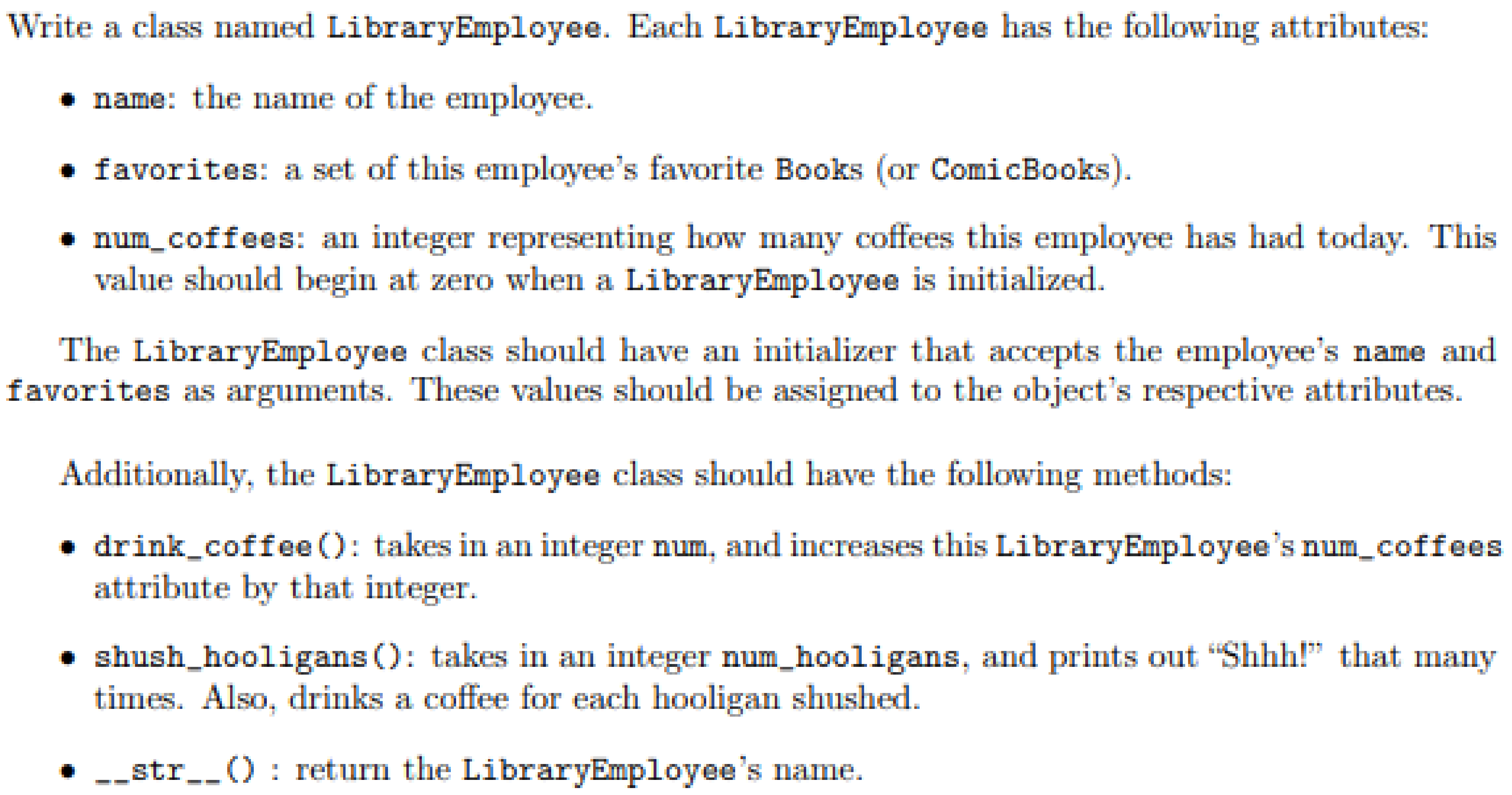
APS106 – Final Exam

Thursday April 14th, 2022

Long Answer Questions

Disclaimer: There may be other ways to solve these problems! There is never one single solution.

Question 1 – Classes (not from a midterm)



Write a class named **LibraryEmployee**. Each **LibraryEmployee** has the following attributes:

- **name**: the name of the employee.
- **favorites**: a set of this employee's favorite **Books** (or **ComicBooks**).
- **num_coffees**: an integer representing how many coffees this employee has had today. This value should begin at zero when a **LibraryEmployee** is initialized.

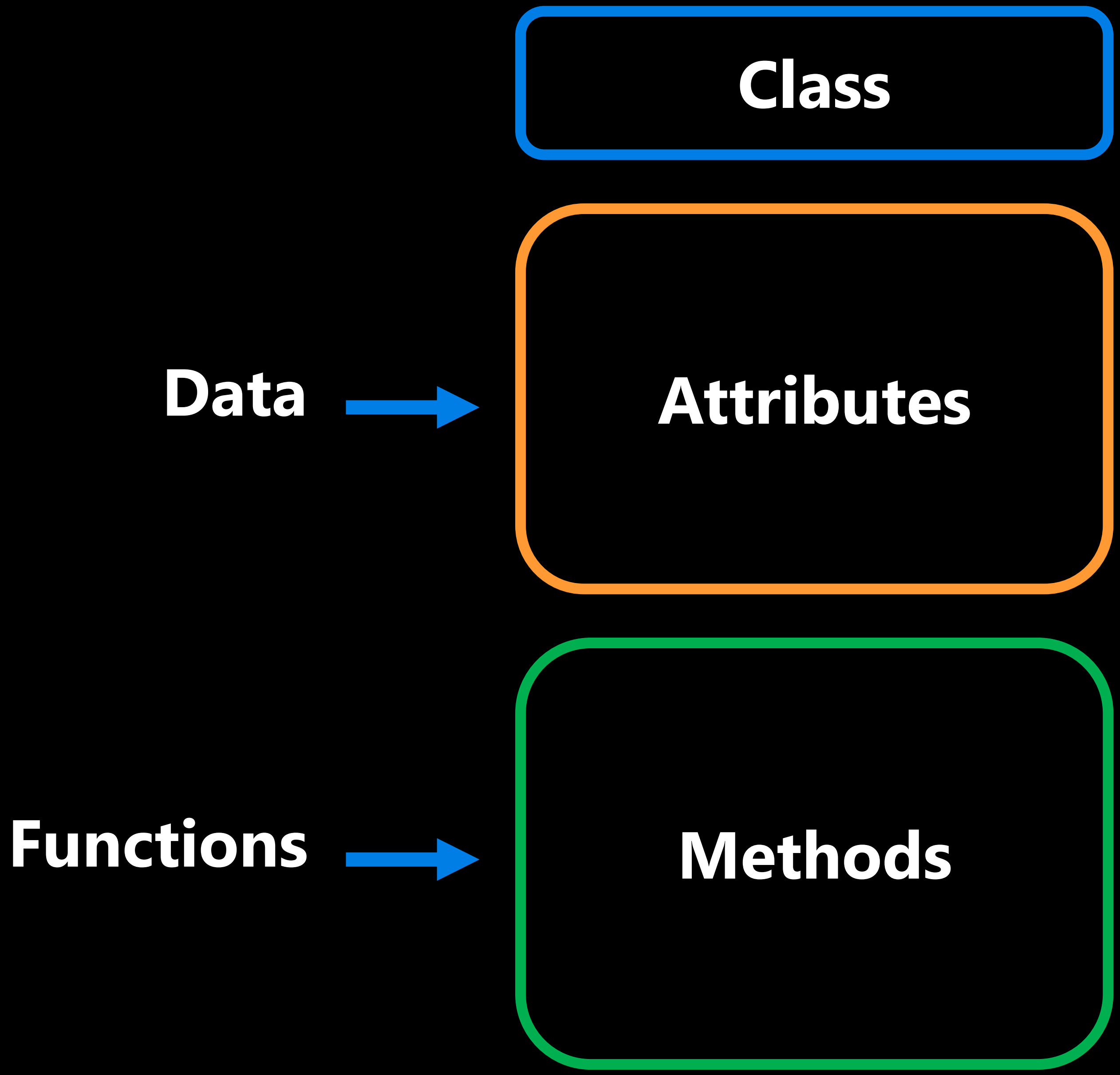
The **LibraryEmployee** class should have an initializer that accepts the employee's **name** and **favorites** as arguments. These values should be assigned to the object's respective attributes.

Additionally, the **LibraryEmployee** class should have the following methods:

- **drink_coffee()**: takes in an integer **num**, and increases this **LibraryEmployee**'s **num_coffees** attribute by that integer.
- **shush_hooligans()**: takes in an integer **num_hooligans**, and prints out "Shhh!" that many times. Also, drinks a coffee for each hooligan shushed.
- **__str__()** : return the **LibraryEmployee**'s name.

OOP Recap

- A class can be thought of as a template for the objects that are instances of it.



Class

Data → Attributes

Functions → Methods

OOP Recap

Instances (objects) of the Turtle class.

name: Susmit
x location: 134
y location: 45

name: Lucy
x location: 24
y location: 35

name: Brian
x location: 92
y location: 62

Turtle

name
x location
y location

move up
move down
move left
move right
go to

OOP Recap

- General form of a Class:
  - Class Name
    - **CamelCase**
    - CourseGrades
    - BankAccount
    - FlightStatus
    - XRayImage
  - Constructor
  - Methods

```
class Name:

    def __init__(self, param1, param2, …):
        self.param1 = param1
        self.param2 = param2
        …
        body

    def method1(self, parameters):
        body

    def method2(self, parameters):
        body

    def method3(self, parameters):
        body
```

OOP Recap

- self
- Reference to the instance of the class.

```
class Turtle:

    def __init__(self, x, y):
        self.x = x
        self.y = y

    def up(self):
        self.y += 1

    def goto(self, x, y):
        self.x = x
        self.y = y

    def get_position(self):
        return self.x, self.y
```

OOP Recap

- Because at the time of designing the class we don't know what these instance names will be, we just chose one.
    - self

```
class Turtle:

    def __init__(self, x, y):
        self.x = x
        self.y = y

    def up(self):
        self.y += 1

    def goto(self, x, y):
        self.x = x
        self.y = y

    def get_position(self):
        return self.x, self.y
```

OOP Recap

- Accessing attributes (Data) and methods (Functions) is different.

```
ben = Turtle(0, 0)
```

```
ben.x
```

Attribute.

```
ben.up()
```

A **Method** is a function, and we call functions using parentheses.

```
def my_func():
    print(“Hello”)
```

```
my_func
```

This function has not been called.

OOP Recap

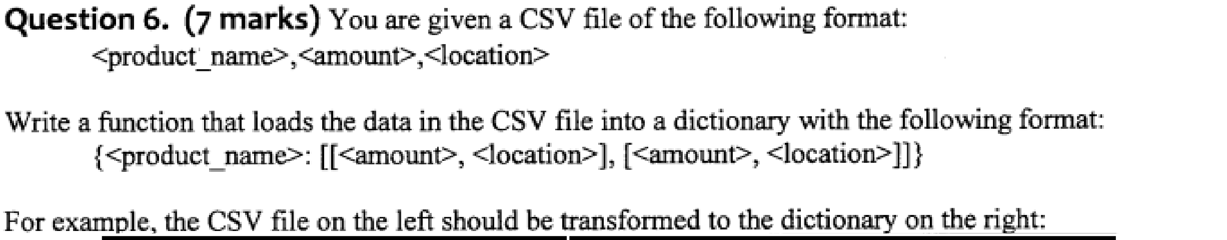
```
seb = Turtle(0, 0)
```

These parameters are passed to the constructor (the `__init__` method).

```
class Turtle:

    def __init__(self, x, y):
        self.x = x
        self.y = y

    def up(self):
        self.y += 1

    def goto(self, x, y):
        self.x = x
        self.y = y

    def get_position(self):
        return self.x, self.y
```

See Jupyter Notebook for the solution!

Question 2 – String manipulation & CSVs (modified from exam!)



Question 6. (7 marks) You are given a CSV file of the following format:

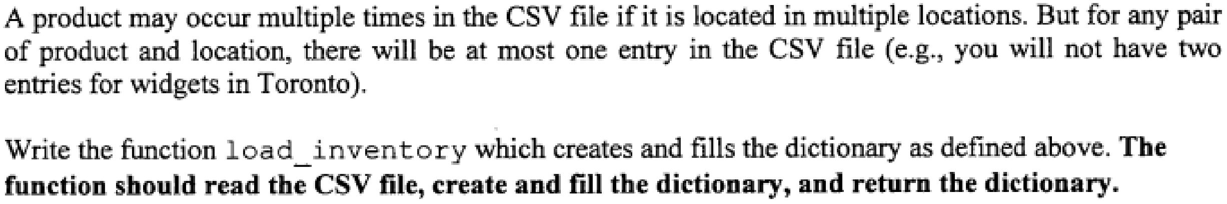
<product_name>,<amount>,<location>

Write a function that loads the data in the CSV file into a dictionary with the following format:

{<product_name>: [[<amount>, <location>], [<amount>, <location>]]}

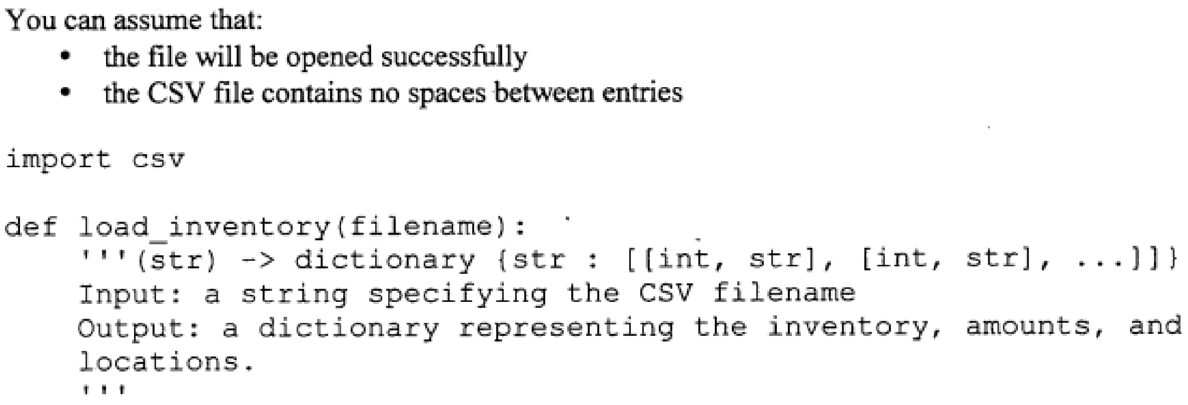
For example, the CSV file on the left should be transformed to the dictionary on the right:

CSV	Dictionary
widget,230,Toronto-Ontario gadget,113,Montreal-Quebec bucket,200,Toronto-Ontario widget,200,Vancouver-BC	{'widget': [[230, ['Toronto', 'Ontario']], [200, ['Vancouver', 'BC']]], 'bucket':[[200, ['Toronto', 'Ontario']]], 'gadget': [[113, ['Montreal', 'Quebec']]



A product may occur multiple times in the CSV file if it is located in multiple locations. But for any pair of product and location, there will be at most one entry in the CSV file (e.g., you will not have two entries for widgets in Toronto).

Write the function `load_inventory` which creates and fills the dictionary as defined above. **The function should read the CSV file, create and fill the dictionary, and return the dictionary.**



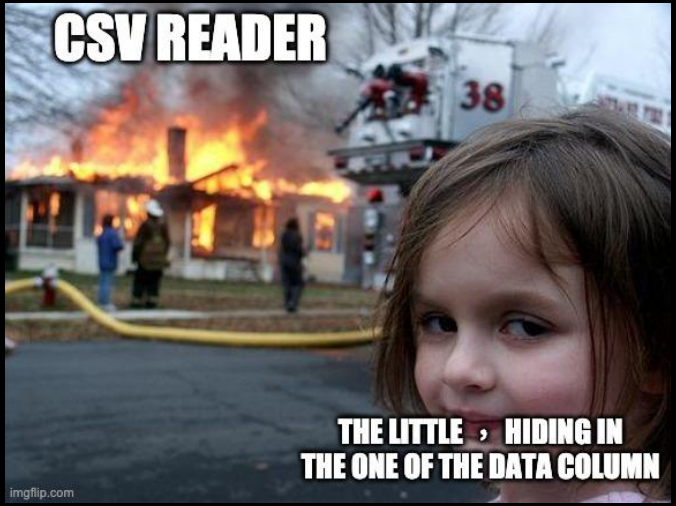
You can assume that:

- the file will be opened successfully
- the CSV file contains no spaces between entries

```
import csv

def load_inventory(filename):
    '''(str) -> dictionary {str : [[int, str], [int, str], ...]]}
    Input: a string specifying the CSV filename
    Output: a dictionary representing the inventory, amounts, and
    locations.
    '''
```

Reading CSV Files

- The CSV module is a powerful solution developed for working with CSV files.
- Reading of CSV files is done using the CSV reader. You can construct a reader object using `csv.reader()` which takes the file object as input.
- The reader object can be used to iterate through the contents of the CSV file, similarly to how a file object was used to iterate through the contents in a text file.

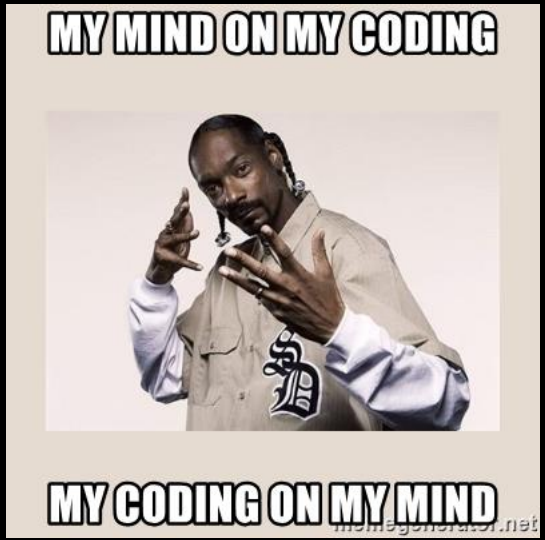


Example: Reading a CSV File (open)

- Read each row of a CSV file using open

```
import csv
csvfile = open("grades.csv", "r")
grades_reader = csv.reader(csvfile)

row_num = 1
for row in grades_reader:
    print('Row #', row_num, ':', row)
    row_num += 1

csvfile.close()
```

```
Row # 1 : ['Name', 'Test1', 'Test2', 'Final']
Row # 2 : [‘Kendrick', '100', '50', '29']
Row # 3 : [‘Dre', '76', '32', '33']
Row # 4 : [‘Snoop', '25', '75', '95']
```

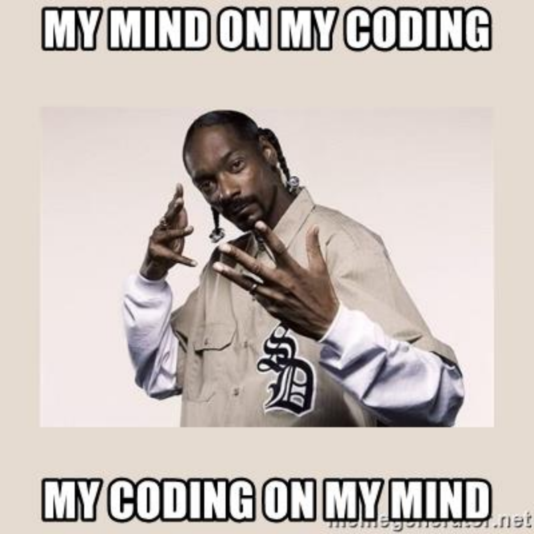


Example: Reading a CSV File (with)

- Read each row of a CSV file using with

```
import csv
with open('grades.csv', 'r') as csvfile:
    grades_reader = csv.reader(csvfile)

    row_num = 1
    for row in grades_reader:
        print('Row #', row_num, ':', row)
        row_num += 1
```

```
Row # 1 : ['Name', 'Test1', 'Test2', 'Final']
Row # 2 : [‘Kendrick', '100', '50', '29']
Row # 3 : [‘Dre', '76', '32', '33']
Row # 4 : [‘Snoop', '25', '75', '95']
```



Writing CSV Files

- To write to the file we would first need to create a CSV writer object, `csv.writer()`, which similar to how we made a, CSV reader object.
- Once the CSV writer object is created, we can use the `writerow()` method to populate it with data.
- The `writerow()` method can only write a single row to the file at a time.

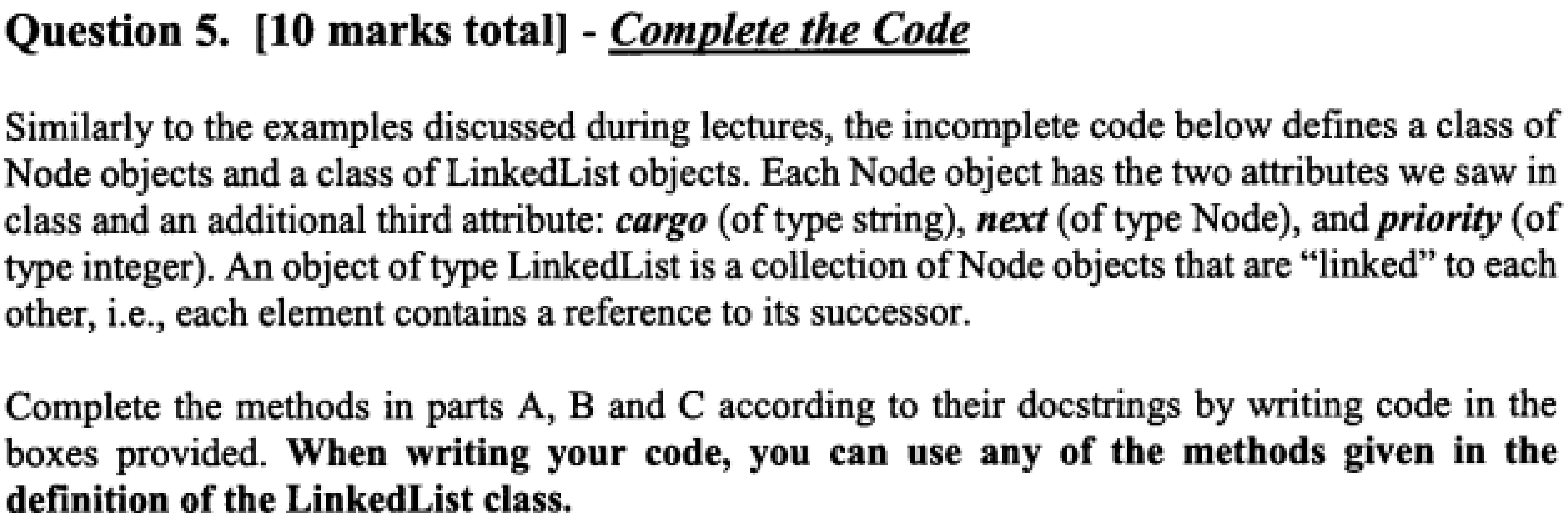
Example: CSV Files

- In the previous grade example there were a few marking errors on the final exam and both John and Mark should have received a higher grade. Update the grades using the CSV `writerow()` method.

```
import csv

grades = [['Name', 'Test1', 'Test2', 'Final'],
          [‘Kendrick', '100', '50', '69'],
          [‘Dre', '76', '32', '53'],
          [‘Snoop', '25', '75', '95']]

with open('grades_new.csv', 'w') as csvfile:
    grades_writer = csv.writer(csvfile)

    for row in grades:
        grades_writer.writerow(row)
```

See Jupyter Notebook for the solution!

Question 3 - LinkedLists

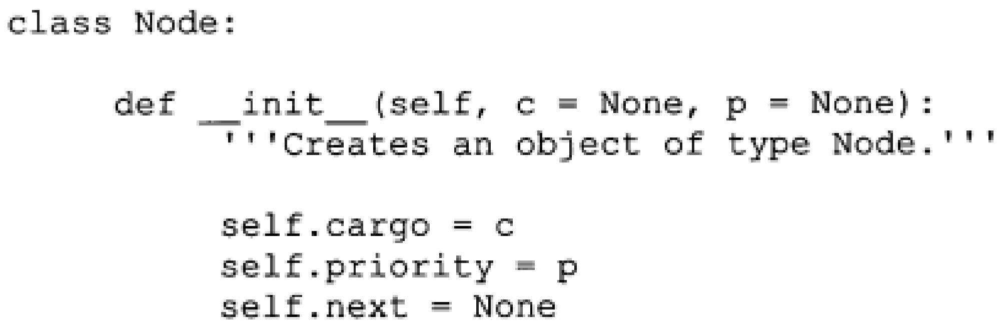


Question 5. [10 marks total] - *Complete the Code*

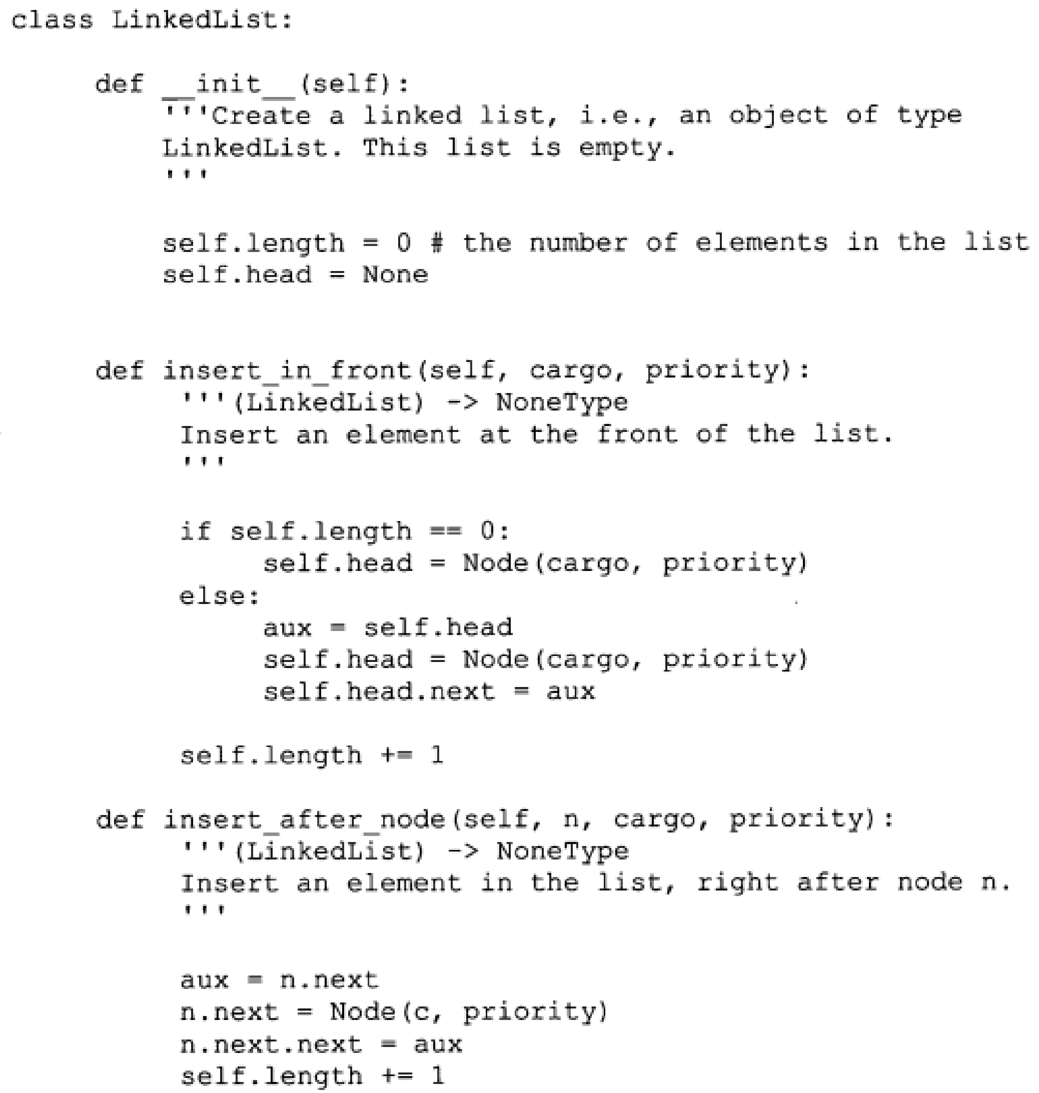
Similarly to the examples discussed during lectures, the incomplete code below defines a class of Node objects and a class of LinkedList objects. Each Node object has the two attributes we saw in class and an additional third attribute: ***cargo*** (of type string), ***next*** (of type Node), and ***priority*** (of type integer). An object of type LinkedList is a collection of Node objects that are "linked" to each other, i.e., each element contains a reference to its successor.

Complete the methods in parts A, B and C according to their docstrings by writing code in the boxes provided. **When writing your code, you can use any of the methods given in the definition of the LinkedList class.**

Question 3 - LinkedLists

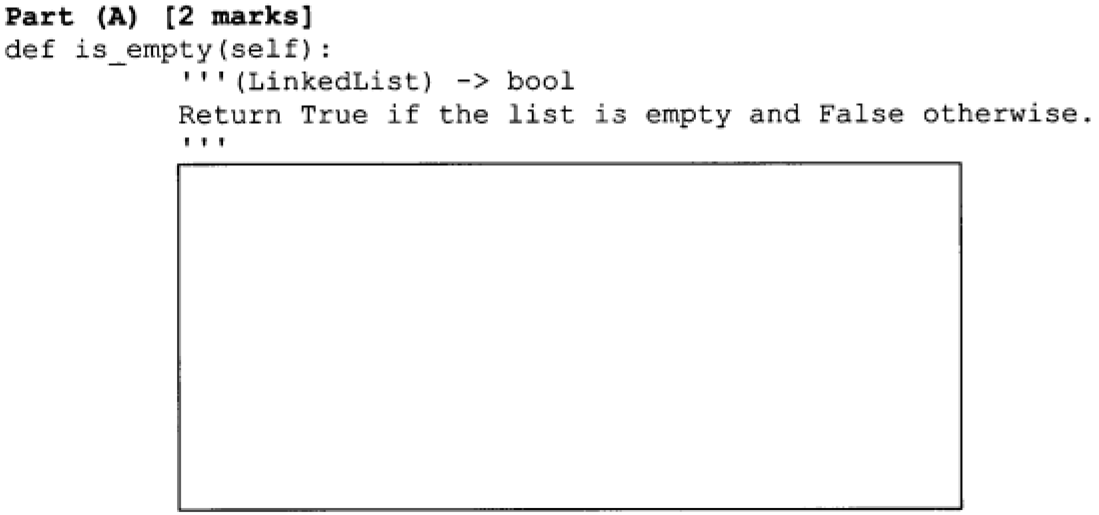


```
class Node:

    def __init__(self, c = None, p = None):
        '''Creates an object of type Node.'''

        self.cargo = c
        self.priority = p
        self.next = None
```



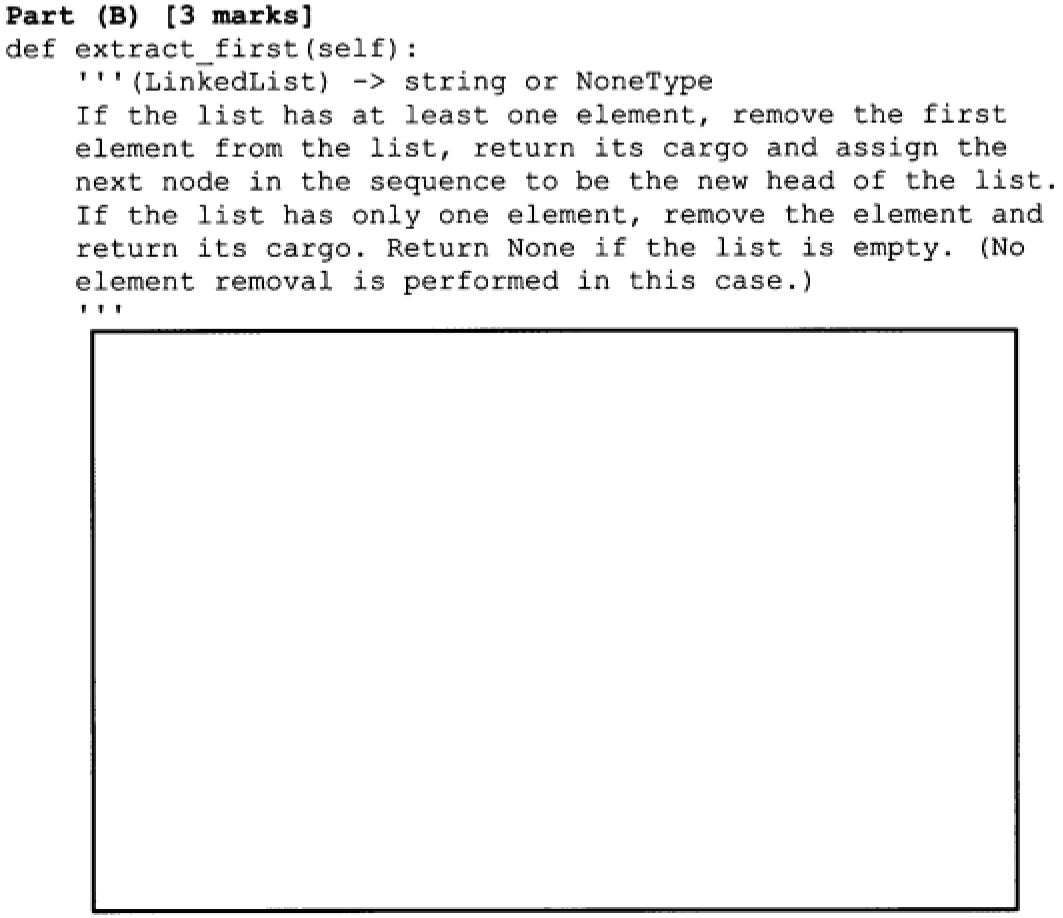
```
class LinkedList:

    def __init__(self):
        '''Create a linked list, i.e., an object of type
        LinkedList. This list is empty.
        '''

        self.length = 0 # the number of elements in the list
        self.head = None


    def insert_in_front(self, cargo, priority):
         '''(LinkedList) -> NoneType
         Insert an element at the front of the list.
         '''

         if self.length == 0:
              self.head = Node(cargo, priority)
         else:
              aux = self.head
              self.head = Node(cargo, priority)
              self.head.next = aux

         self.length += 1

    def insert_after_node(self, n, cargo, priority):
         '''(LinkedList) -> NoneType
         Insert an element in the list, right after node n.
         '''

         aux = n.next
         n.next = Node(c, priority)
         n.next.next = aux
         self.length += 1
```

Parts of the question



Part (A) [2 marks]

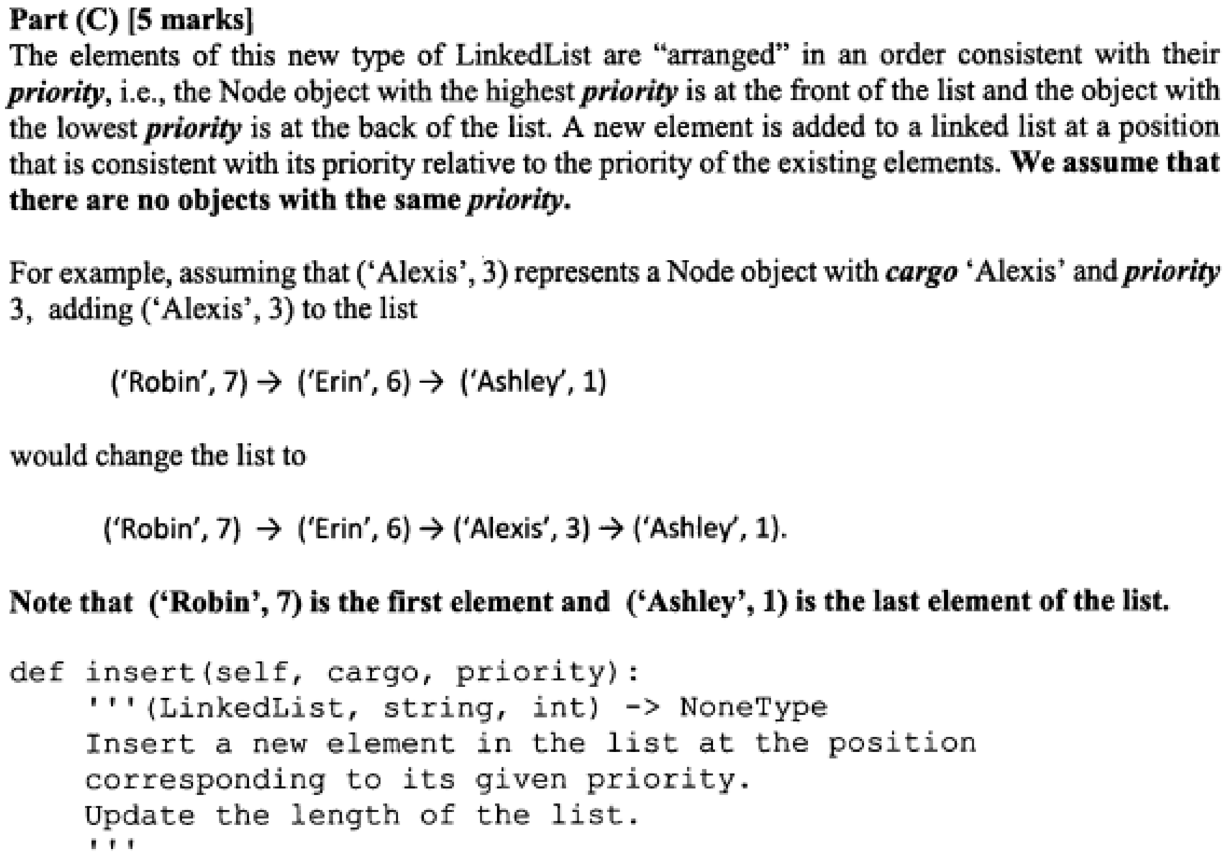
```
def is_empty(self):
        '''(LinkedList) -> bool
        Return True if the list is empty and False otherwise.
        '''
```



Part (B) [3 marks]

```
def extract_first(self):
    '''(LinkedList) -> string or NoneType
    If the list has at least one element, remove the first
    element from the list, return its cargo and assign the
    next node in the sequence to be the new head of the list.
    If the list has only one element, remove the element and
    return its cargo. Return None if the list is empty. (No
    element removal is performed in this case.)
    '''
```

Parts of the question



Part (C) [5 marks]

The elements of this new type of LinkedList are “arranged” in an order consistent with their ***priority***, i.e., the Node object with the highest ***priority*** is at the front of the list and the object with the lowest ***priority*** is at the back of the list. A new element is added to a linked list at a position that is consistent with its priority relative to the priority of the existing elements. **We assume that there are no objects with the same *priority*.**

For example, assuming that (‘Alexis’, 3) represents a Node object with ***cargo*** ‘Alexis’ and ***priority*** 3, adding (‘Alexis’, 3) to the list

(‘Robin’, 7) → (‘Erin’, 6) → (‘Ashley’, 1)

would change the list to

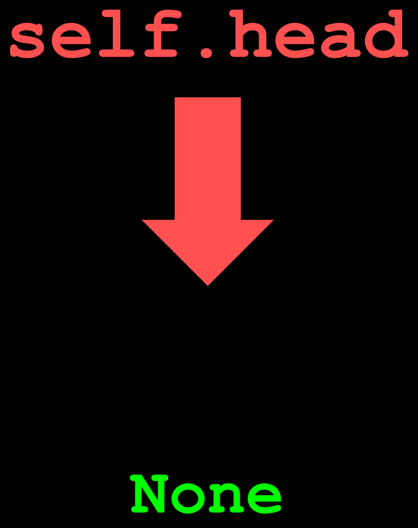
(‘Robin’, 7) → (‘Erin’, 6) → (‘Alexis’, 3) → (‘Ashley’, 1).

Note that (‘Robin’, 7) is the first element and (‘Ashley’, 1) is the last element of the list.

```
def insert(self, cargo, priority):
    '''(LinkedList, string, int) -> NoneType
    Insert a new element in the list at the position
    corresponding to its given priority.
    Update the length of the list.
    '''
```

```python
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

```
>>> linked_list = LinkedList()
>>> linked_list.__str__()
'empty list'
```



self.head

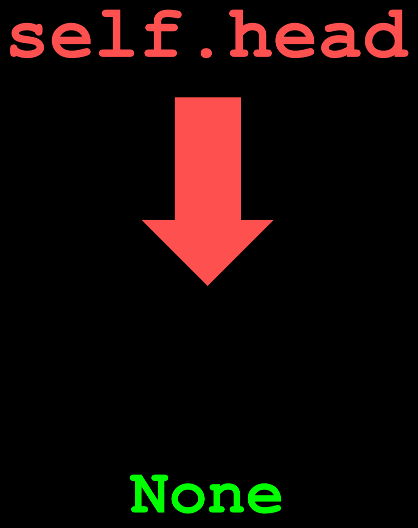
None

```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

add_to_head method.

```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the front of the list.
        """
        node = Node(cargo)
        node.next = self.head
        self.head = node
        self.length += 1

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

```
>>> linked_list = LinkedList()
>>> linked_list.__str__()
'empty list'
```

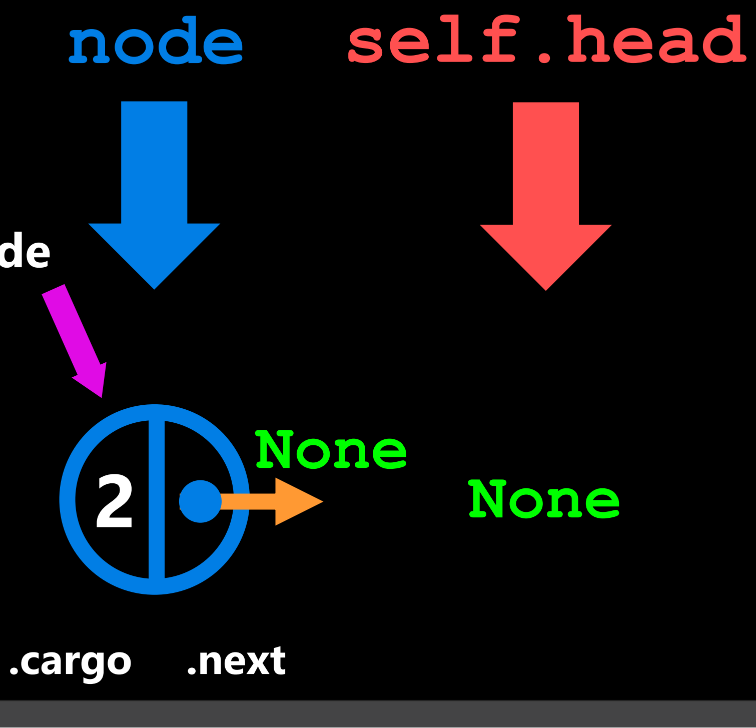


```python
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the front of the list.
        """
        node = Node(cargo)
        node.next = self.head
        self.head = node
        self.length += 1

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

Create Node

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> None'
```

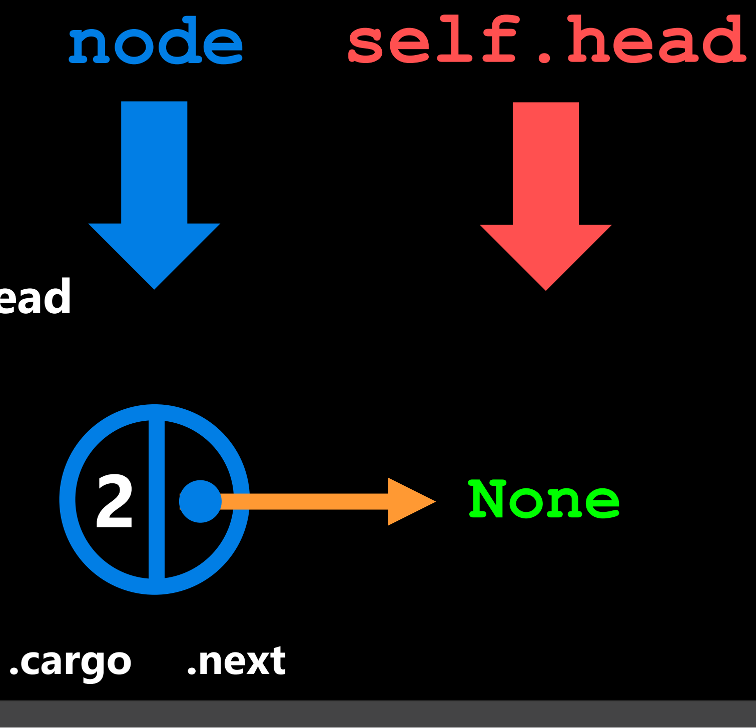
Add Node



```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the front of the list.
        """
        node = Node(cargo)
        node.next = self.head
        self.head = node
        self.length += 1

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

← **Point to head**

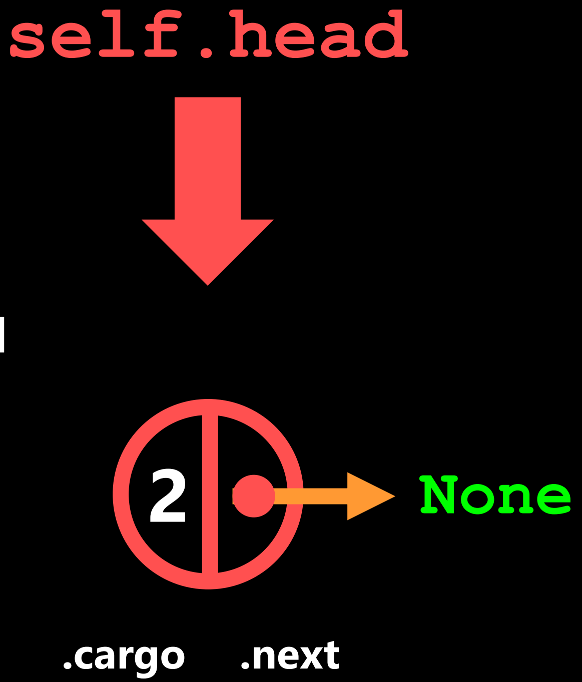
```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> None'
```



```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the front of the list.
        """
        node = Node(cargo)
        node.next = self.head
        self.head = node
        self.length += 1

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

Assign new Node to head

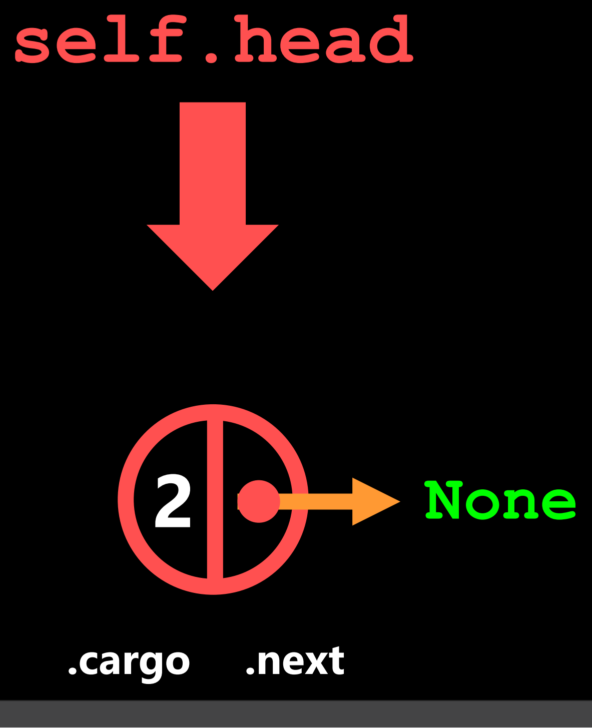
```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> None'
```



```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the front of the list.
        """
        node = Node(cargo)
        node.next = self.head
        self.head = node
        self.length += 1

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

Increase length

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> None'
```

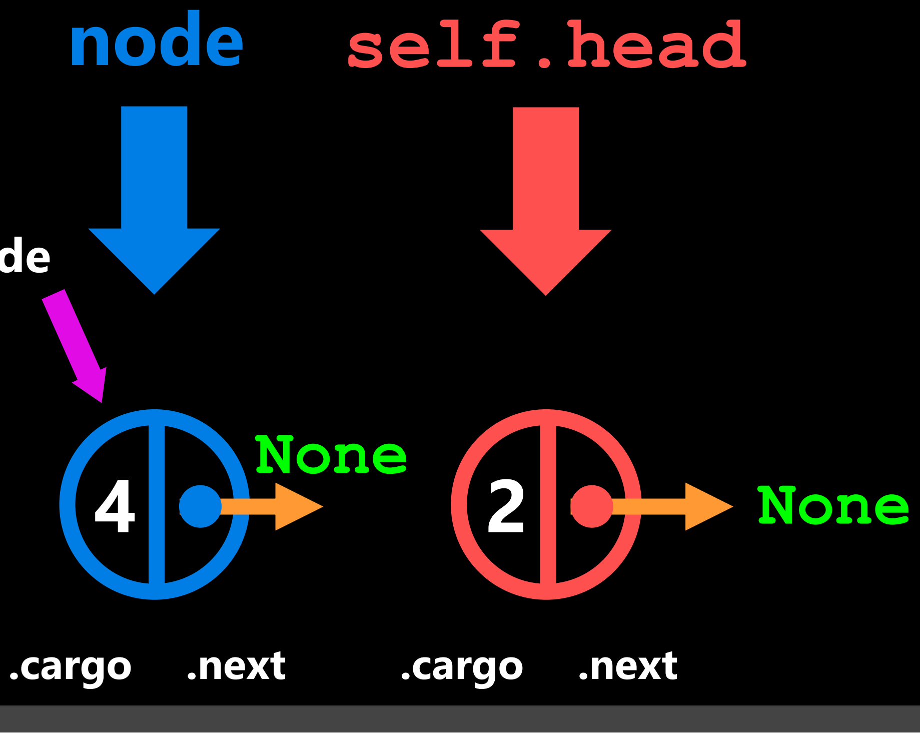


```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the front of the list.
        """
        node = Node(cargo)
        node.next = self.head
        self.head = node
        self.length += 1

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

Create Node

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(2)
>>> linked_list.add_to_head(4)
>>> linked_list.__str__()
'(4) --> (2) --> None'
```

Add Node

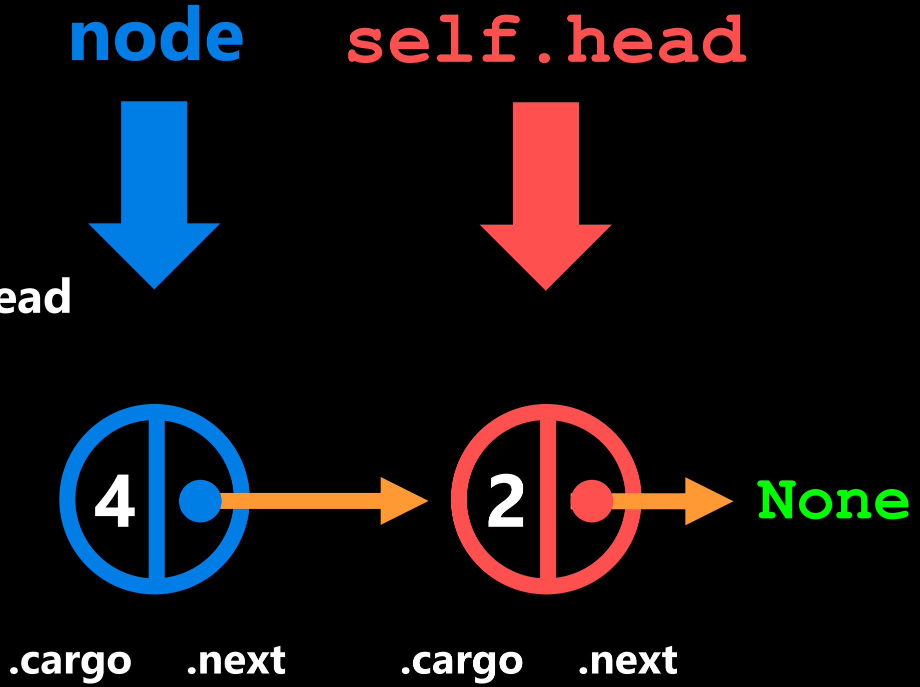


```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the front of the list.
        """
        node = Node(cargo)
        node.next = self.head
        self.head = node
        self.length += 1

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

Point to head

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(2)
>>> linked_list.add_to_head(4)
>>> linked_list.__str__()
'(4) --> (2) --> None'
```

Add Node

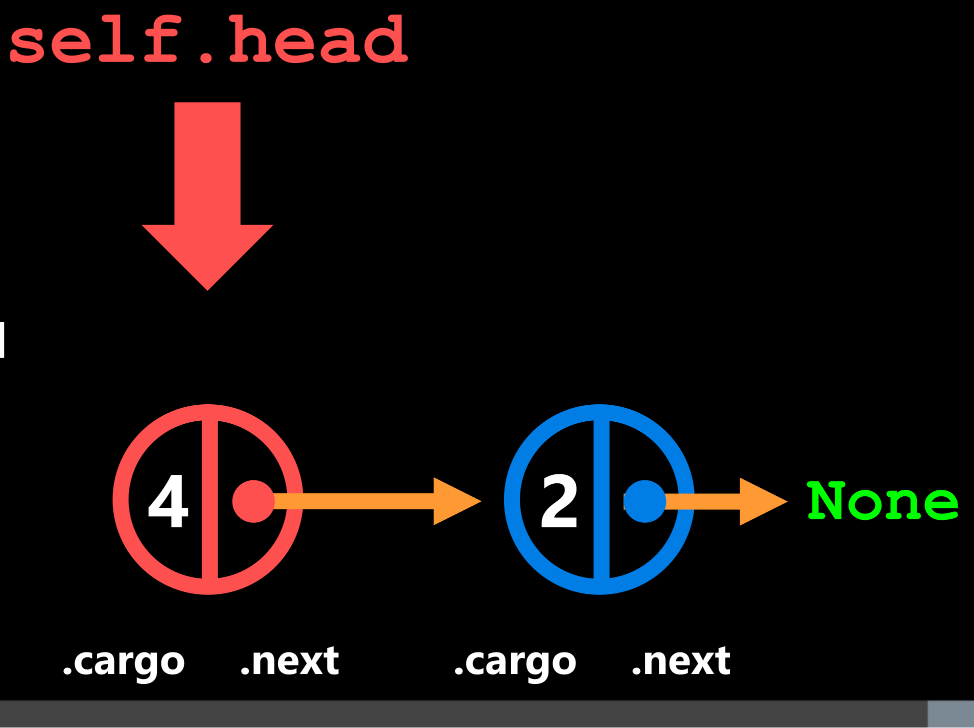


```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the front of the list.
        """
        node = Node(cargo)
        node.next = self.head
        self.head = node
        self.length += 1

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

Assign new Node to head

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(2)
>>> linked_list.add_to_head(4)
>>> linked_list.__str__()
'(4) --> (2) --> None'
```

Add Node

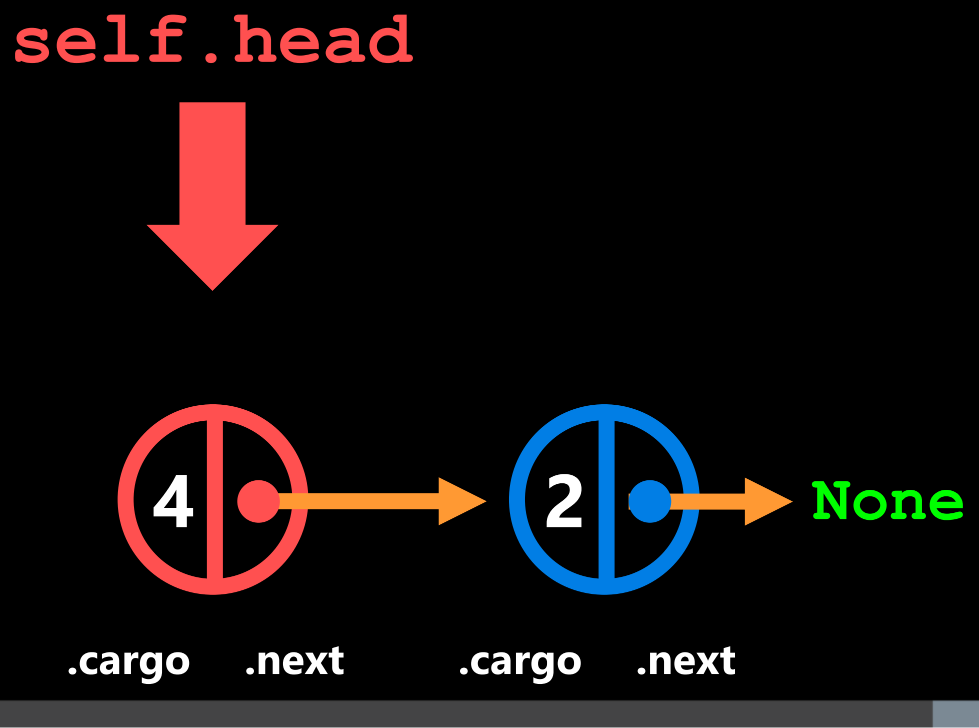


```python
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the front of the list.
        """
        node = Node(cargo)
        node.next = self.head
        self.head = node
        self.length += 1

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

Increase length

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(2)
>>> linked_list.add_to_head(4)
>>> linked_list.__str__()
'(4) --> (2) --> None'
```

Add Node

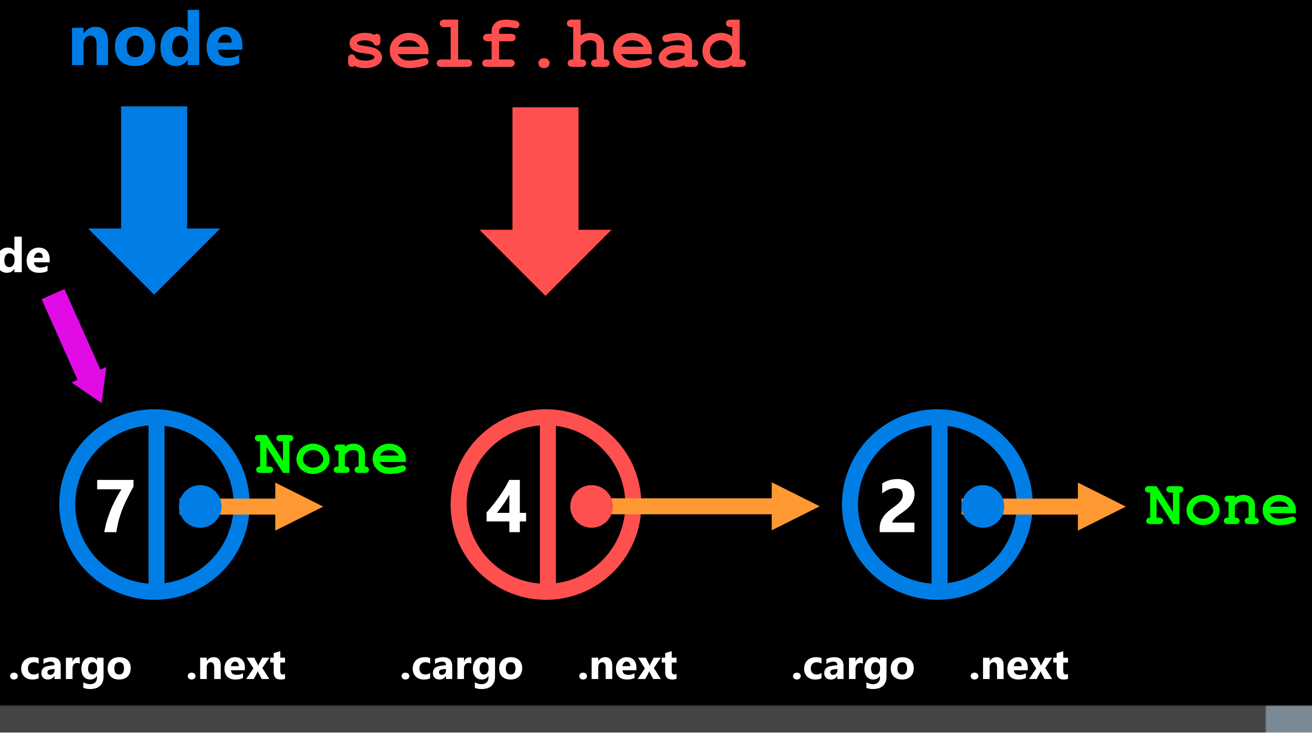


```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the front of the list.
        """
        node = Node(cargo)
        node.next = self.head
        self.head = node
        self.length += 1

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

Create Node

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(2)
>>> linked_list.add_to_head(4)
>>> linked_list.add_to_head(7)
>>> linked_list.__str__()
'(7) --> (4) --> (2) --> None'
```

Add Node

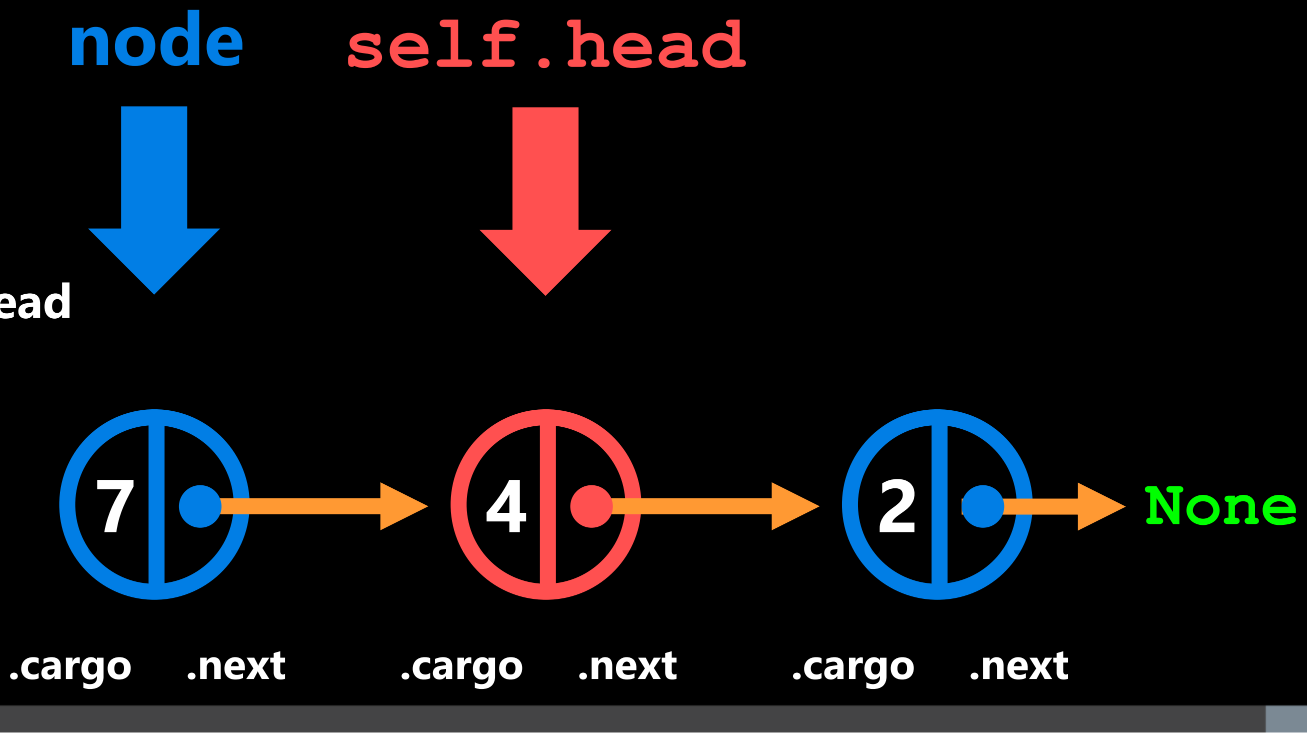


```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the front of the list.
        """
        node = Node(cargo)
        node.next = self.head
        self.head = node
        self.length += 1

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

Point to head

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(2)
>>> linked_list.add_to_head(4)
>>> linked_list.add_to_head(7)
>>> linked_list.__str__()
'(7) --> (4) --> (2) --> None'
```

Add Node

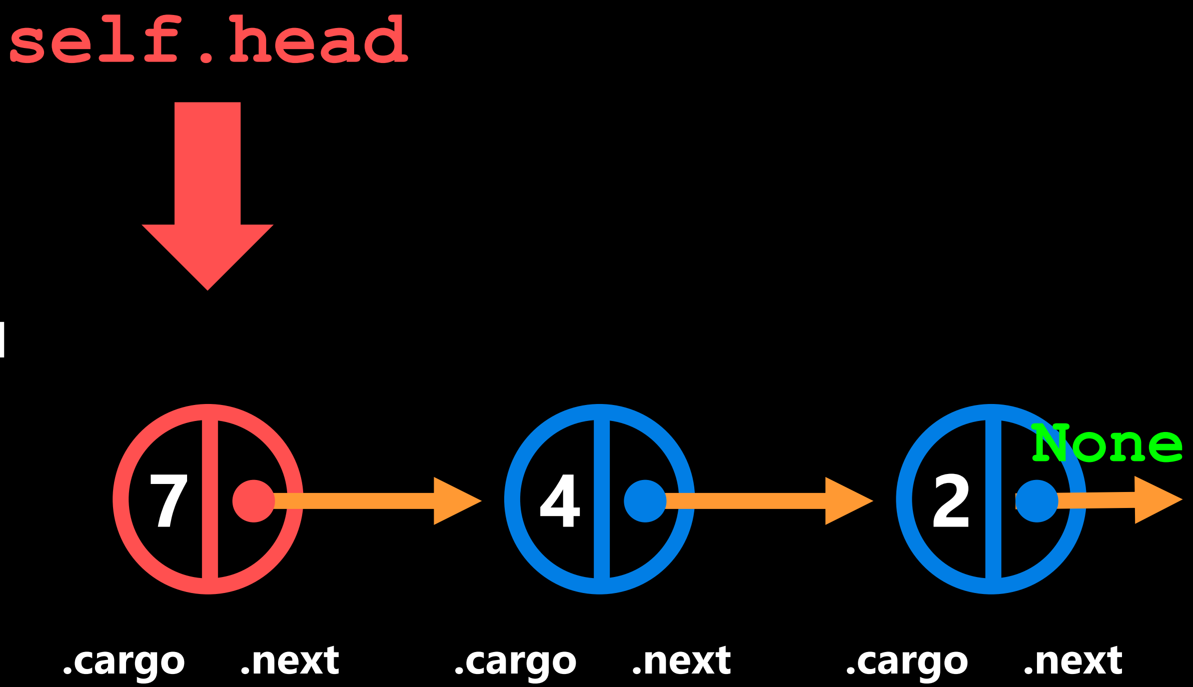


```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the front of the list.
        """
        node = Node(cargo)
        node.next = self.head
        self.head = node
        self.length += 1

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

Assign new Node to head

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(2)
>>> linked_list.add_to_head(4)
>>> linked_list.add_to_head(7)
>>> linked_list.__str__()
'(7) --> (4) --> (2) --> None'
```

Add Node

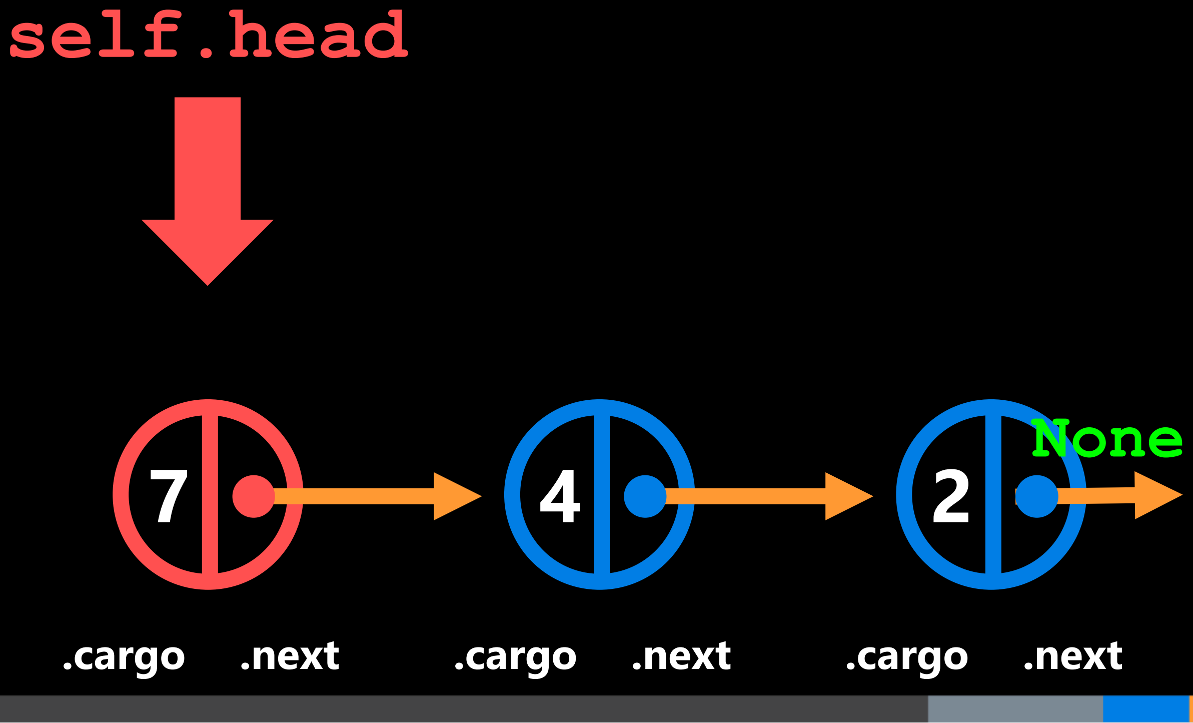


```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the front of the list.
        """
        node = Node(cargo)
        node.next = self.head
        self.head = node
        self.length += 1

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

Increase length

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(2)
>>> linked_list.add_to_head(4)
>>> linked_list.add_to_head(7)
>>> linked_list.__str__()
'(7) --> (4) --> (2) --> None'
```

Add Node

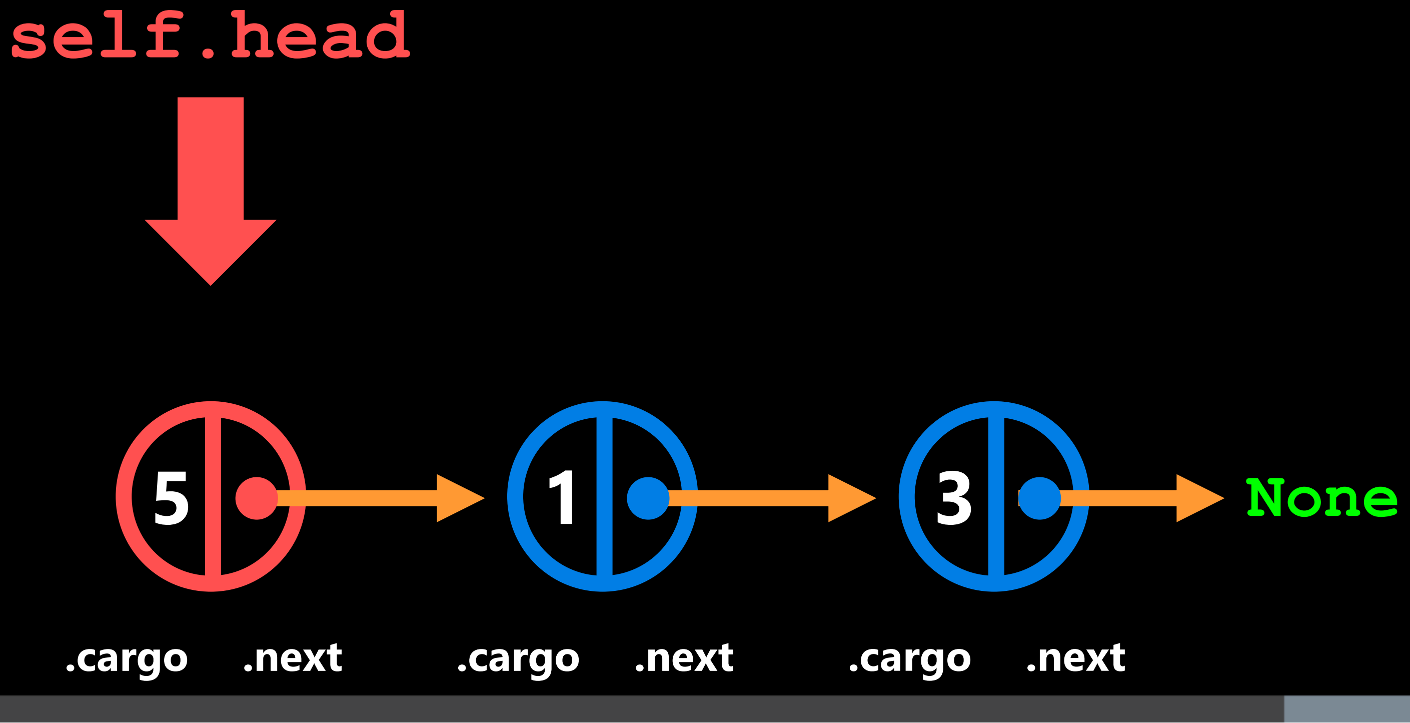


```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

add_to_tail method.

```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the tail of the list.
        """
        on = self.head

        while on.next is not None:
            on = on.next

        on.next = Node(cargo)

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

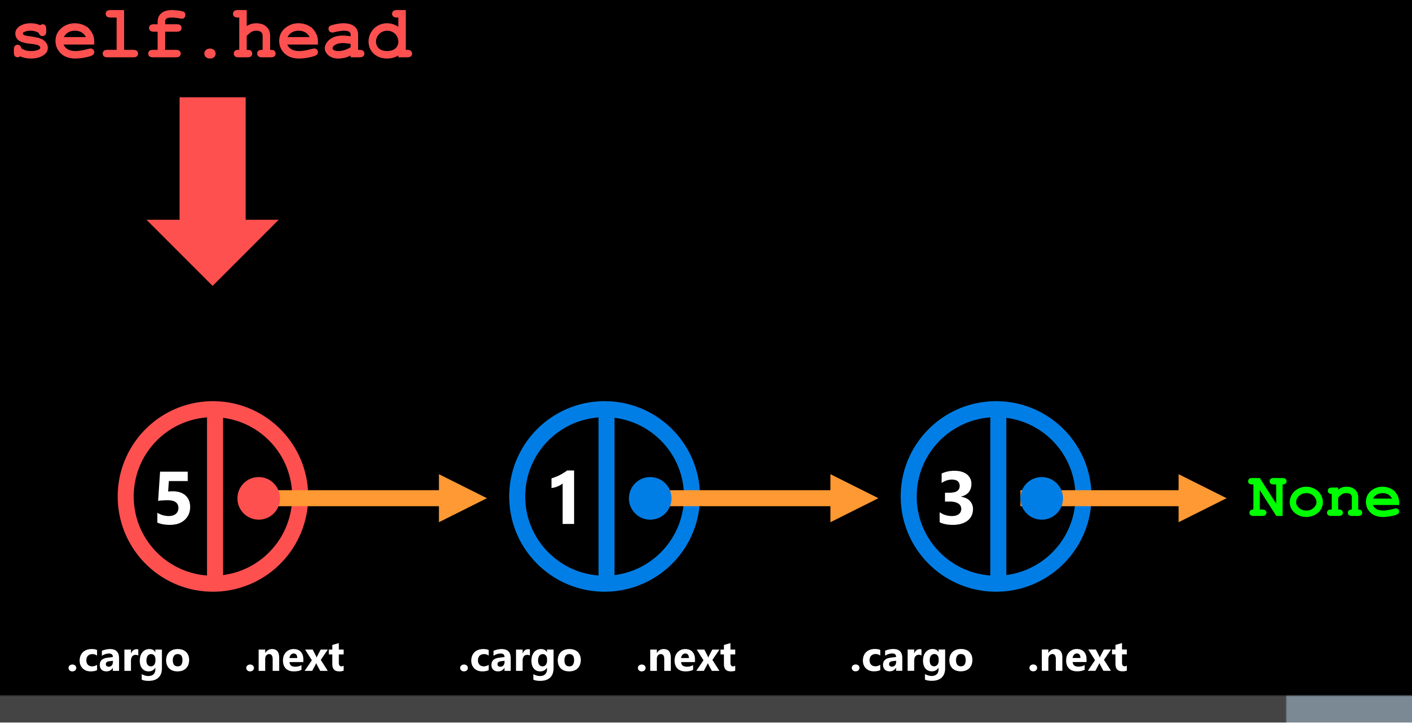
```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(1)
>>> linked_list.add_to_head(5)
>>> linked_list.__str__()
'(5) --> (1) --> (3) --> None'
```



```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the tail of the list.
        """
        on = self.head

        while on.next is not None:
            on = on.next

        on.next = Node(cargo)

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(1)
>>> linked_list.add_to_head(5)
>>> linked_list.add_to_tail(9)
>>> linked_list.__str__()
'(5) --> (1) --> (3) --> (9) --> None'
```

Add to tail. →

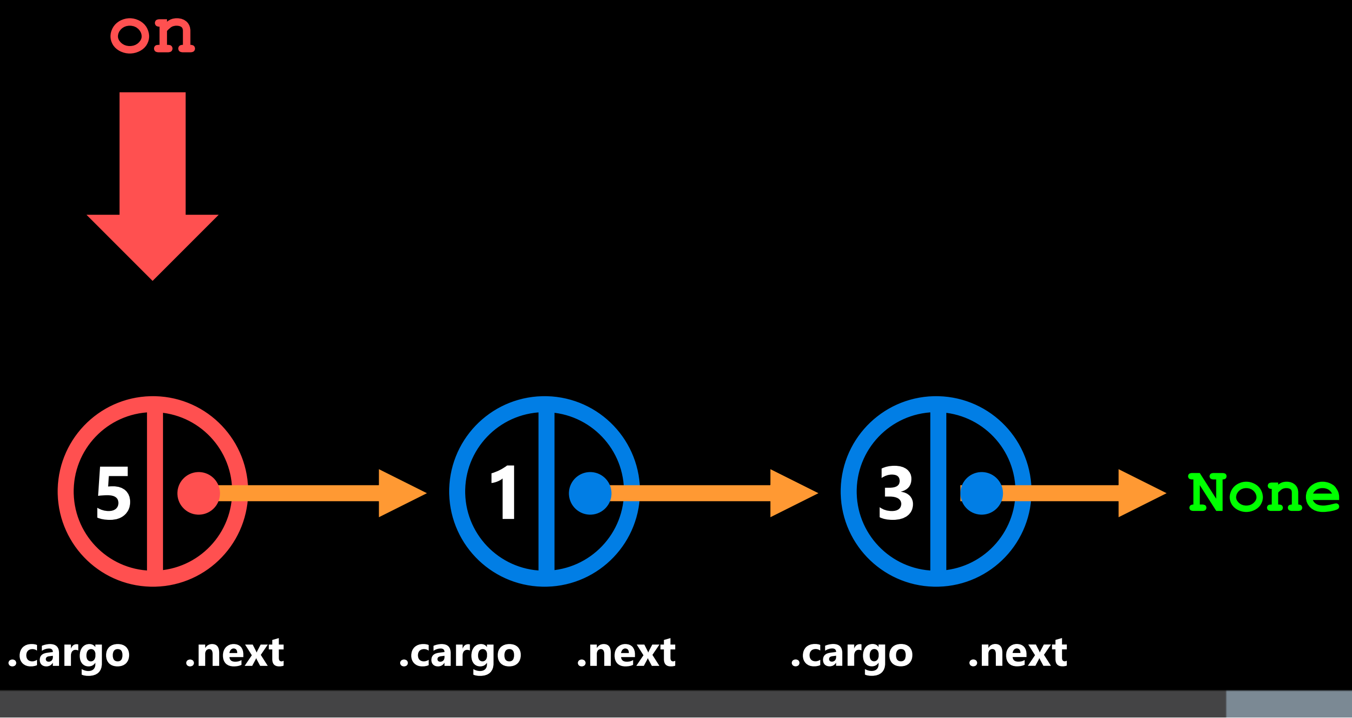


```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the tail of the list.
        """
        on = self.head

        while on.next is not None:
            on = on.next

        on.next = Node(cargo)

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

Set **on** position ← `on = self.head`

Add to tail. → `>>> linked_list.add_to_tail(9)`

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(1)
>>> linked_list.add_to_head(5)
>>> linked_list.add_to_tail(9)
>>> linked_list.__str__()
'(5) --> (1) --> (3) --> (9) --> None'
```



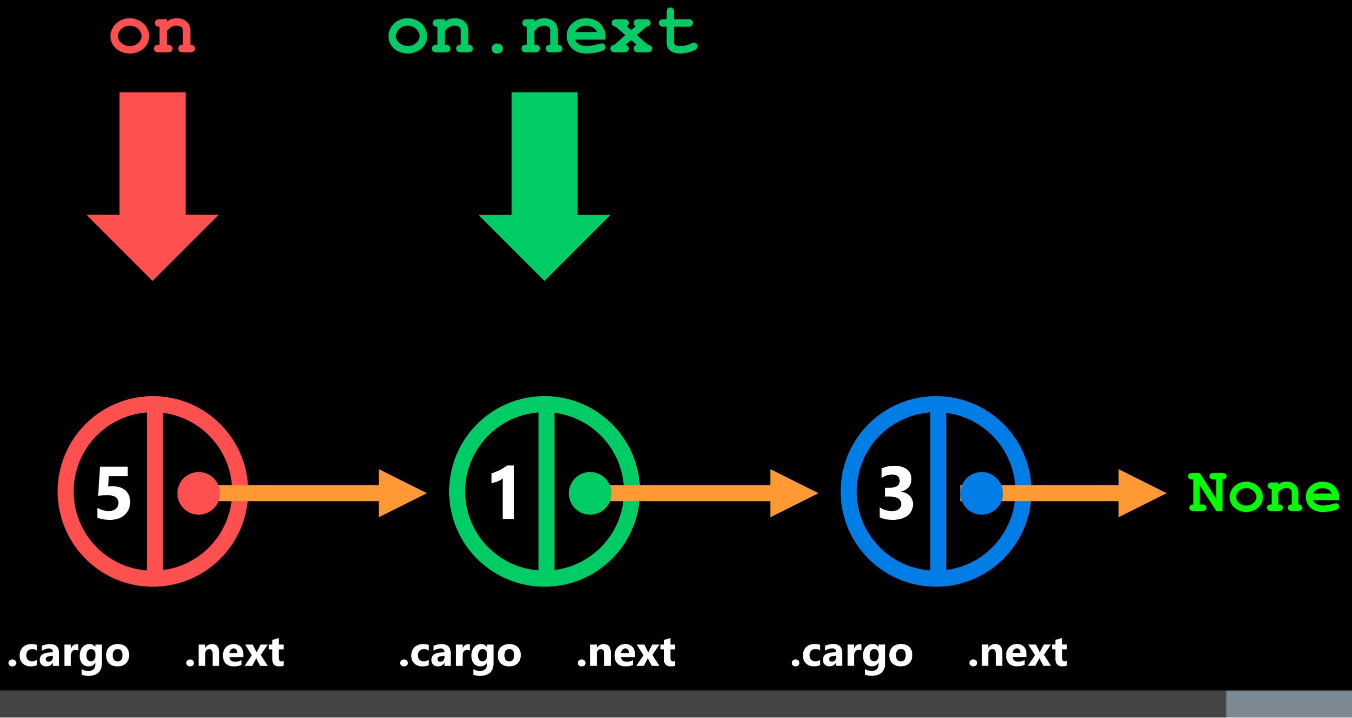
```python
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the tail of the list.
        """
        on = self.head

        while on.next is not None:
            on = on.next

        on.next = Node(cargo)

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

True

Add to tail.

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(1)
>>> linked_list.add_to_head(5)
>>> linked_list.add_to_tail(9)
>>> linked_list.__str__()
'(5) --> (1) --> (3) --> (9) --> None'
```

`on.next` is `None` when `on` is at the last Node.



```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the tail of the list.
        """
        on = self.head

        while on.next is not None:
            on = on.next

        on.next = Node(cargo)

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

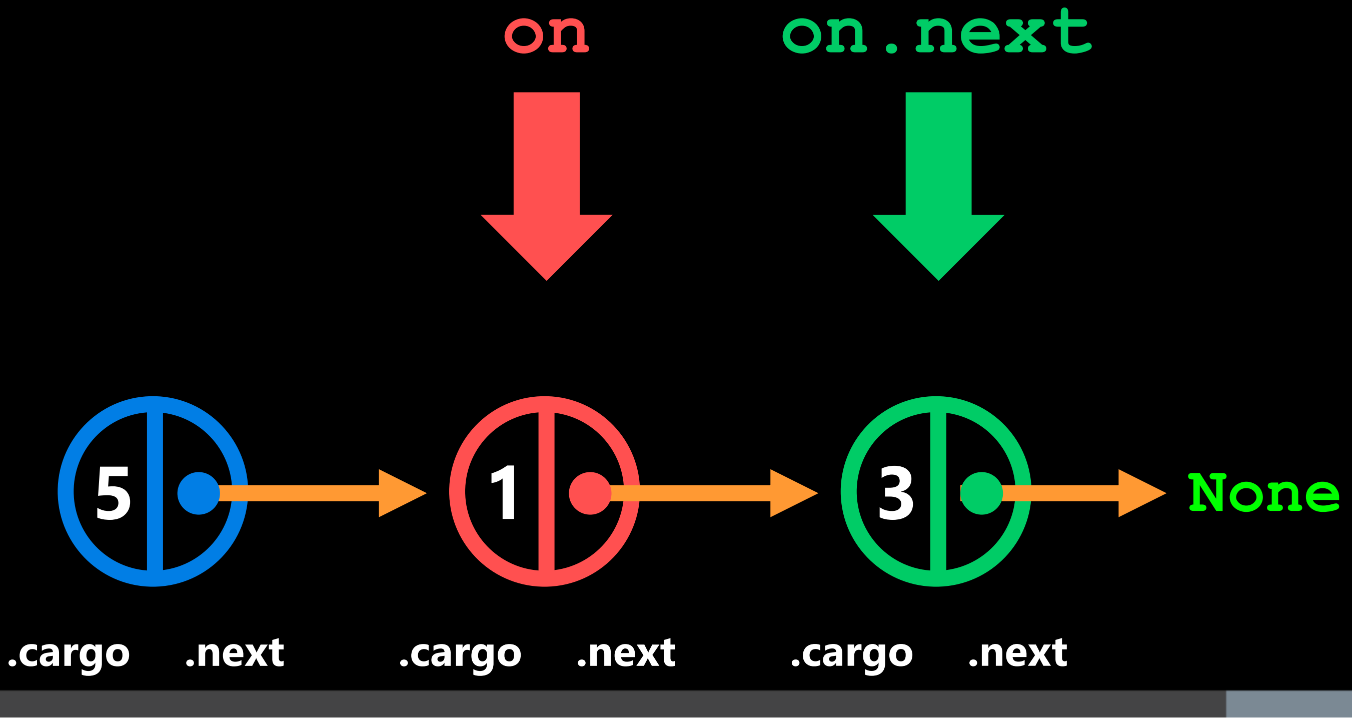
True

Move on to next position.

Add to tail.

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(1)
>>> linked_list.add_to_head(5)
>>> linked_list.add_to_tail(9)
>>> linked_list.__str__()
'(5) --> (1) --> (3) --> (9) --> None'
```

on.next is None when on is at the last Node.



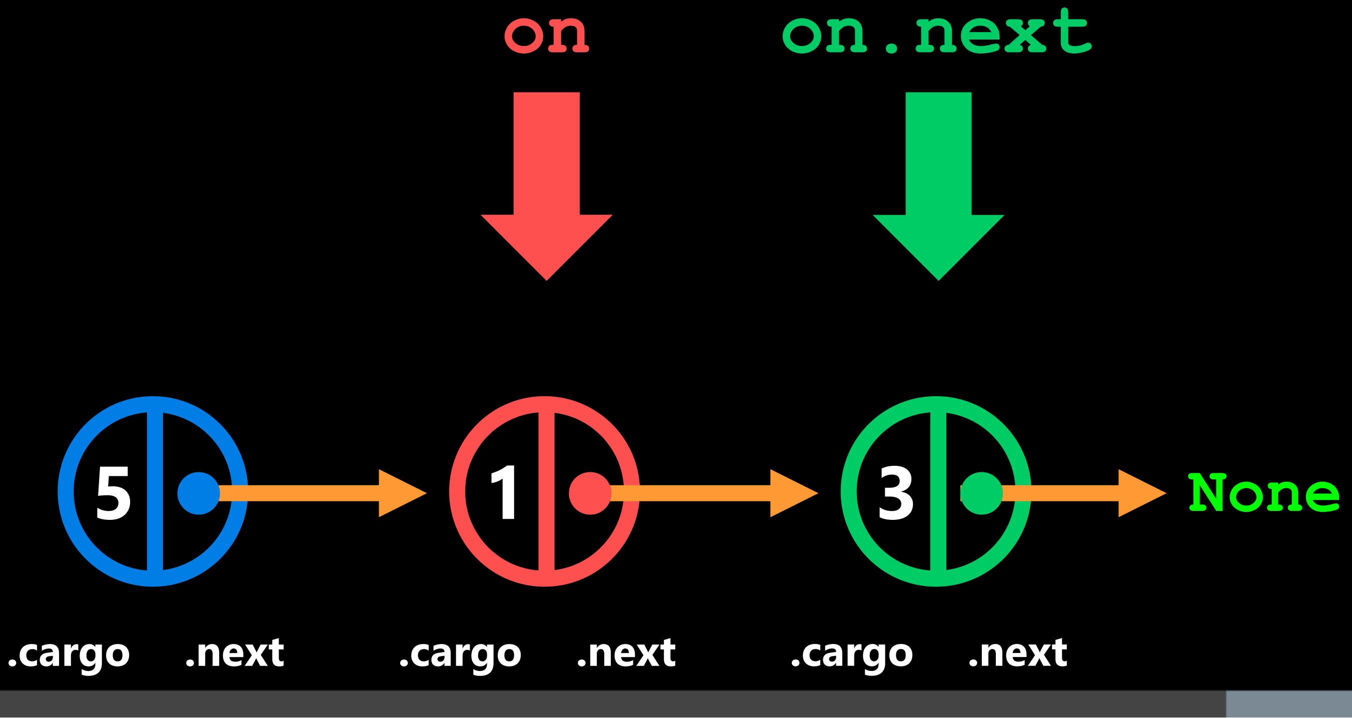
```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the tail of the list.
        """
        on = self.head

        while on.next is not None:
            on = on.next

        on.next = Node(cargo)

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

True

Add to tail.

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(1)
>>> linked_list.add_to_head(5)
>>> linked_list.add_to_tail(9)
>>> linked_list.__str__()
'(5) --> (1) --> (3) --> (9) --> None'
```

`on.next` is `None` when `on` is at the last Node.



```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the tail of the list.
        """
        on = self.head

        while on.next is not None:
            on = on.next

        on.next = Node(cargo)

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

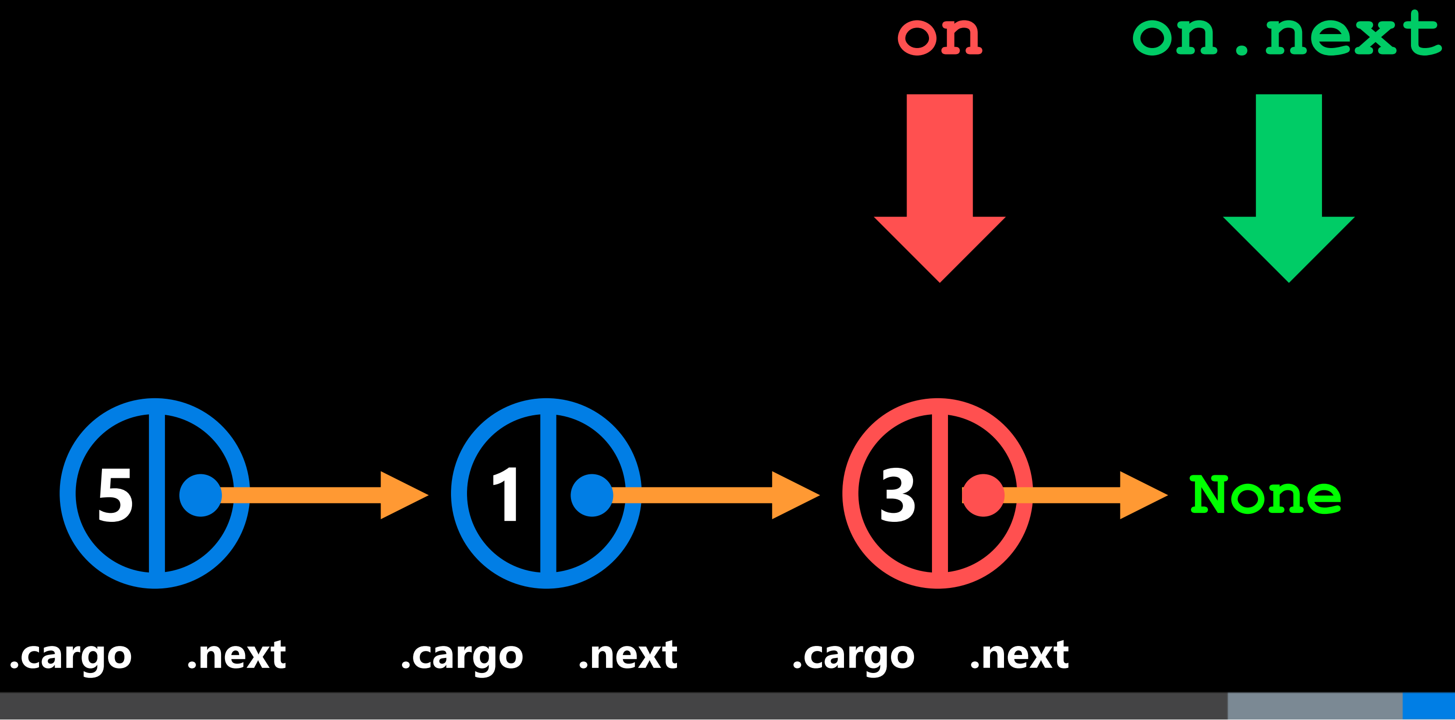
True

Move on to next position.

Add to tail.

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(1)
>>> linked_list.add_to_head(5)
>>> linked_list.add_to_tail(9)
>>> linked_list.__str__()
'(5) --> (1) --> (3) --> (9) --> None'
```

`on.next` is `None` when `on` is at the last Node.



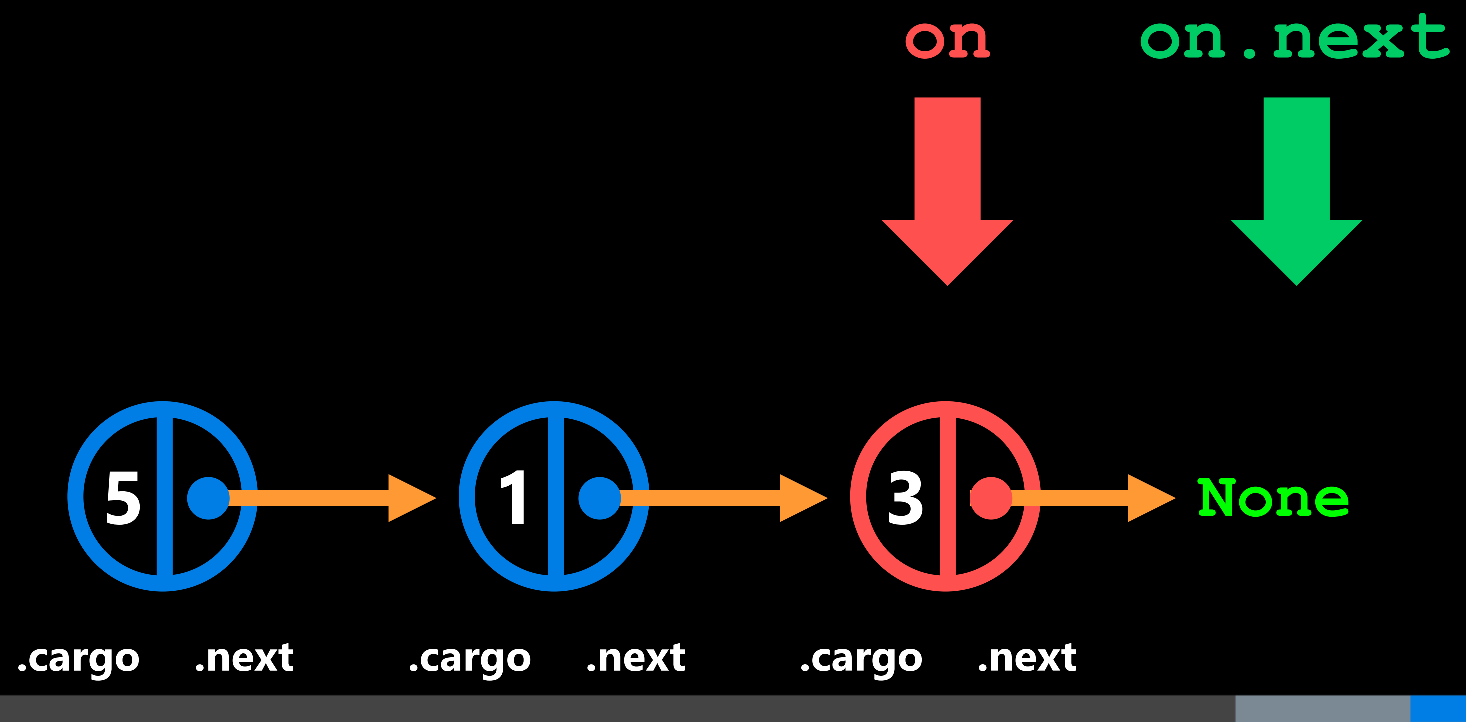
```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the tail of the list.
        """
        on = self.head

        while on.next is not None:
            on = on.next

        on.next = Node(cargo)

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

False

Add to tail.

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(1)
>>> linked_list.add_to_head(5)
>>> linked_list.add_to_tail(9)
>>> linked_list.__str__()
'(5) --> (1) --> (3) --> (9) --> None'
```

`on.next` is `None` when `on` is at the last Node.



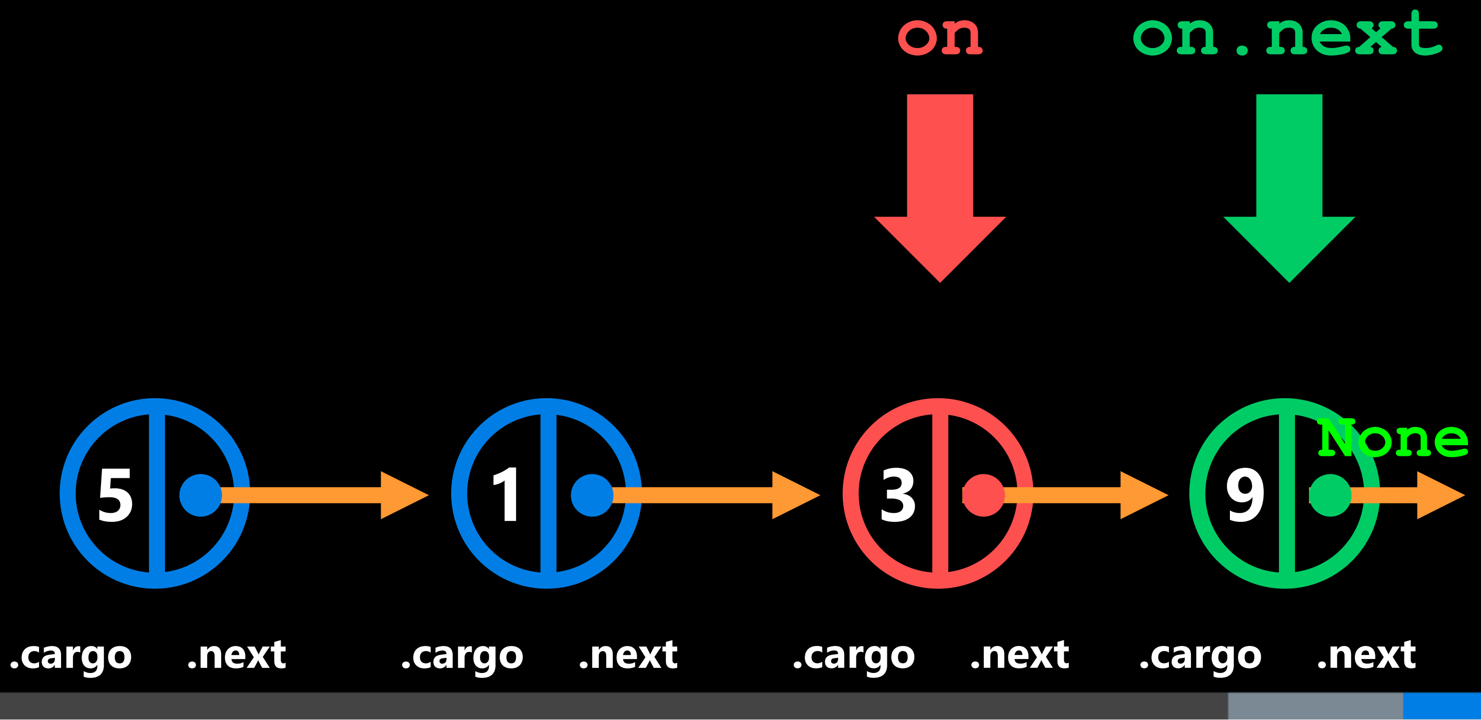
```python
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo):
        """
        (self, object) -> NoneType
        Add cargo to the tail of the list.
        """
        on = self.head

        while on.next is not None:
            on = on.next

        on.next = Node(cargo)

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

Add new node to tail

Add to tail.

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(1)
>>> linked_list.add_to_head(5)
>>> linked_list.add_to_tail(9)
>>> linked_list.__str__()
'(5) --> (1) --> (3) --> (9) --> None'
```

`on.next` is `None` when `on` is at the last Node.

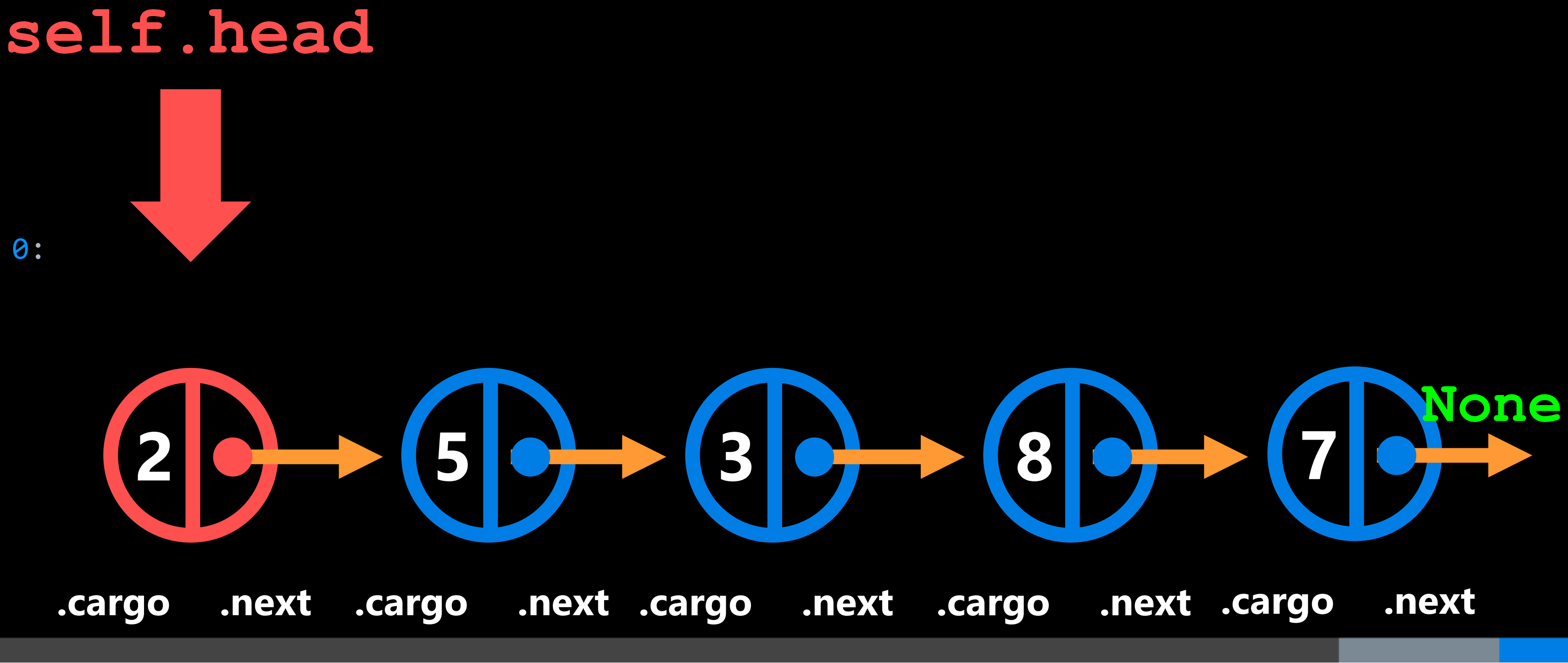


```python
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

get_at_index method.

```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index):
        """
        (self, object) -> NoneType
        Add a new node at certain index.
        """
        on = self.head

        while on is not None and index != 0:
            on = on.next
            index -= 1

        if on is not None:
            return on.cargo
        else:
            return False

    def delete_by_cargo(self, cargo): ...
```

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'
```



```python
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index):
        """
        (self, object) -> NoneType
        Add a new node at certain index.
        """
        on = self.head

        while on is not None and index != 0:
            on = on.next
            index -= 1

        if on is not None:
            return on.cargo
        else:
            return False

    def delete_by_cargo(self, cargo): ...
```

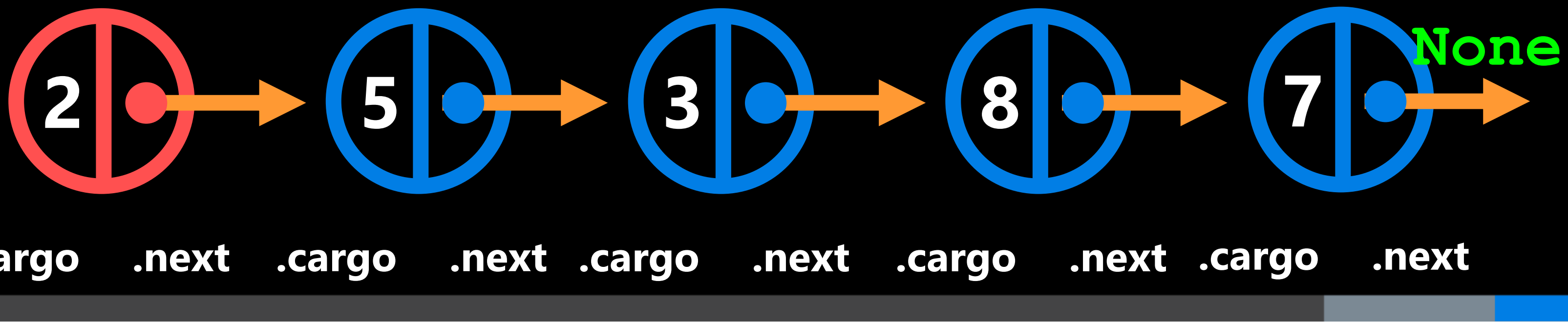
Set on position

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'

>>> linked_list.get_at_index(3)
8
```

Get node at index = 3.

index = 3

on

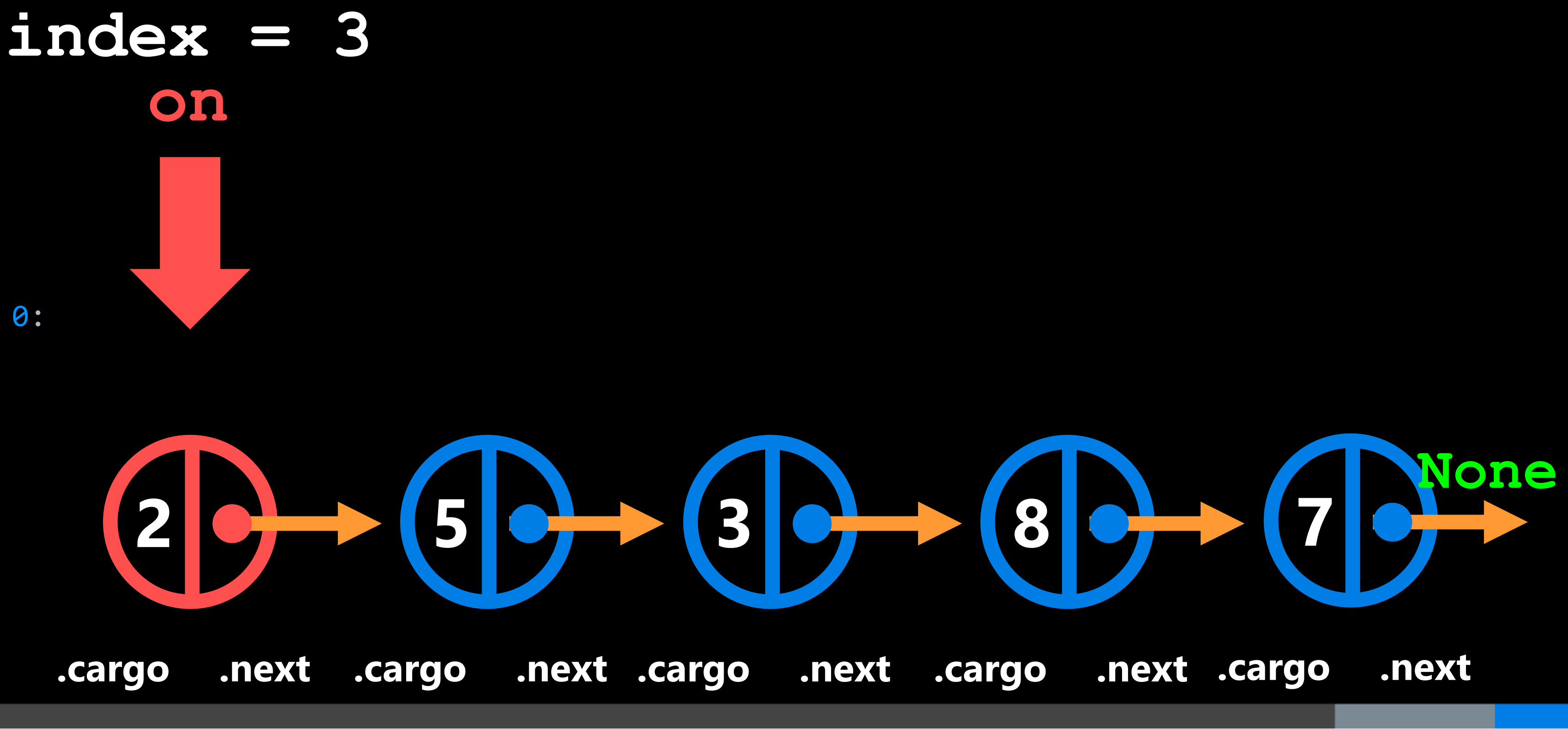


```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index):
        """
        (self, object) -> NoneType
        Add a new node at certain index.
        """
        on = self.head

True ➡  while on is not None and index != 0:
            on = on.next
            index -= 1

        if on is not None:
            return on.cargo
        else:
            return False

    def delete_by_cargo(self, cargo): ...
```

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'
```

Get node at index = 3. ➡

```
>>> linked_list.get_at_index(3)
8
```



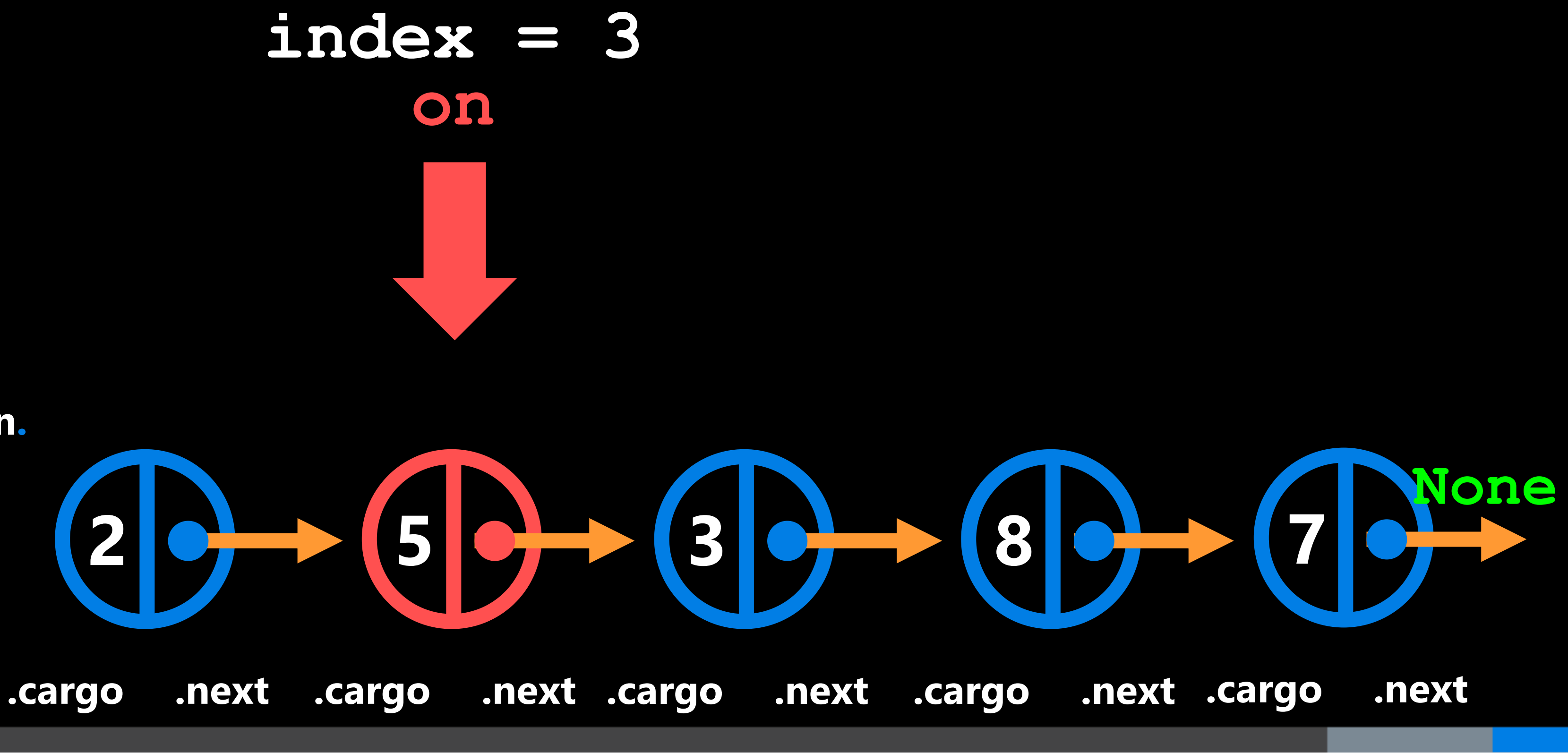
```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index):
        """
        (self, object) -> NoneType
        Add a new node at certain index.
        """
        on = self.head

True    while on is not None and index != 0:
            on = on.next
            index -= 1

        if on is not None:
            return on.cargo
        else:
            return False

    def delete_by_cargo(self, cargo): ...
```

Move on to next position.

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'
```

Get node at index = 3.

```
>>> linked_list.get_at_index(3)
8
```



```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index):
        """
        (self, object) -> NoneType
        Add a new node at certain index.
        """
        on = self.head

        while on is not None and index != 0:
            on = on.next
            index -= 1

        if on is not None:
            return on.cargo
        else:
            return False

    def delete_by_cargo(self, cargo): ...
```

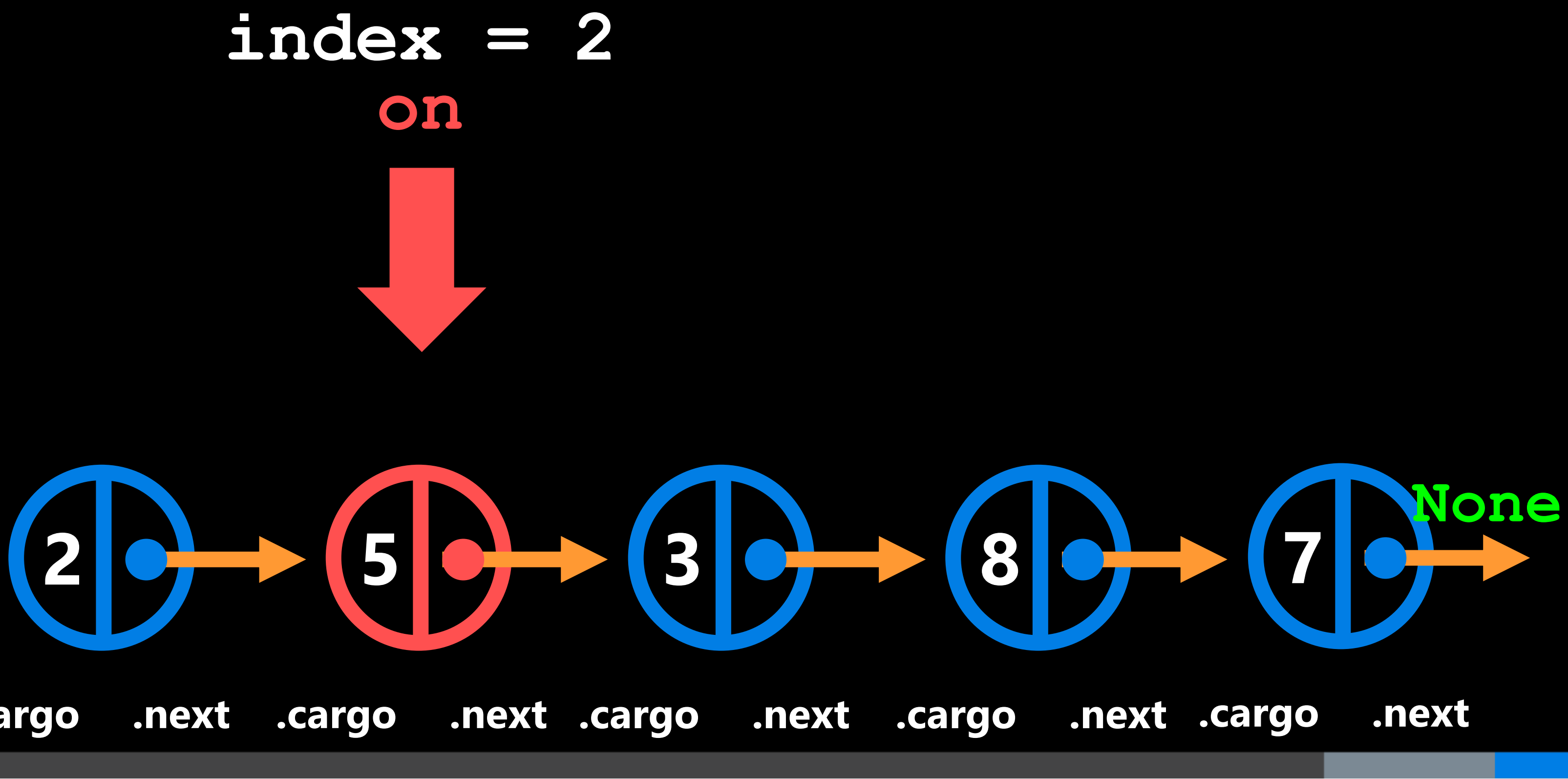
True → while on is not None and index != 0:

Update index. ← index -= 1

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'
```

Get node at index = 3. →

```
>>> linked_list.get_at_index(3)
8
```

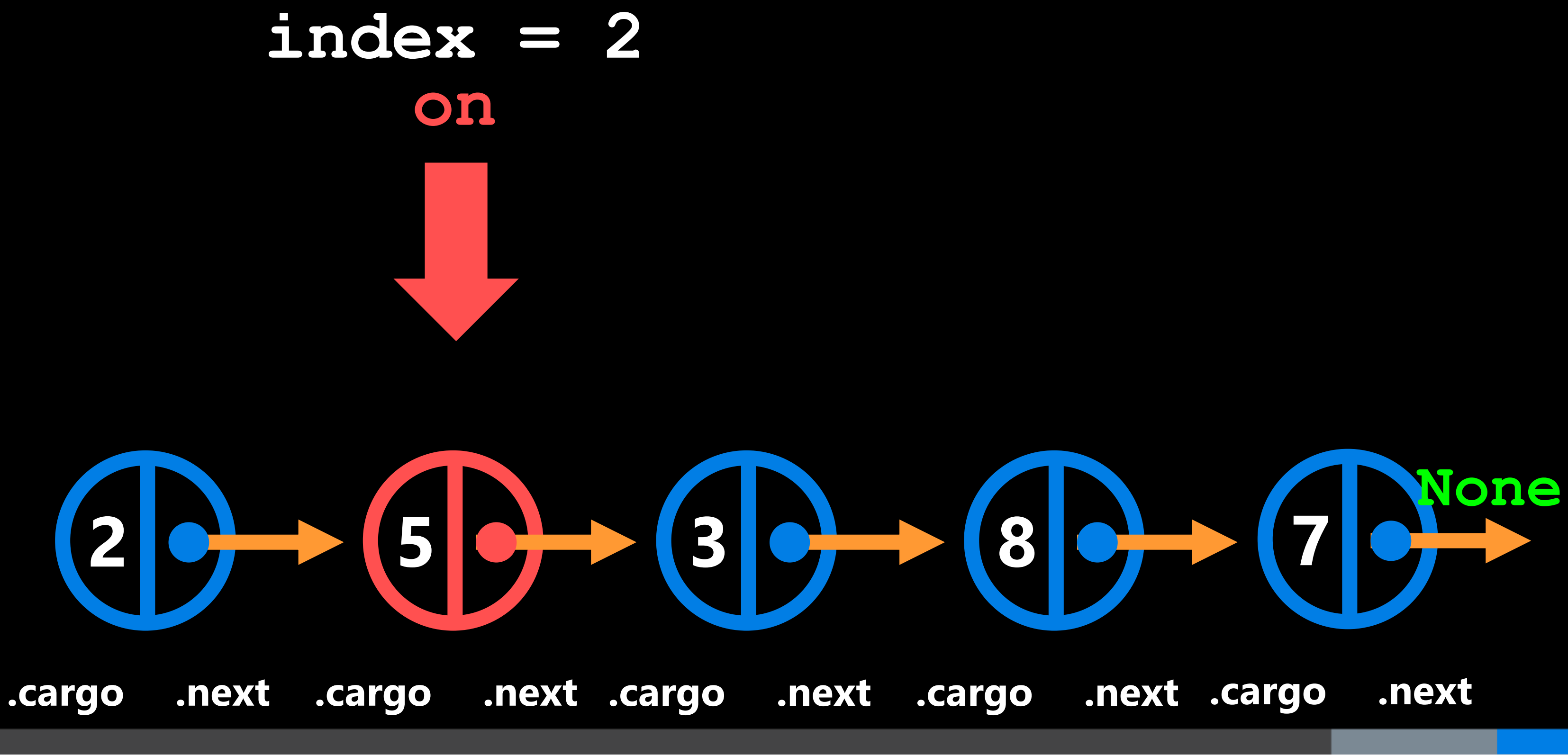


```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index):
        """
        (self, object) -> NoneType
        Add a new node at certain index.
        """
        on = self.head

        while on is not None and index != 0:
            on = on.next
            index -= 1

        if on is not None:
            return on.cargo
        else:
            return False

    def delete_by_cargo(self, cargo): ...
```

True → while on is not None and index != 0:

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'

>>> linked_list.get_at_index(3)
8
```

Get node at index = 3. →



```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index):
        """
        (self, object) -> NoneType
        Add a new node at certain index.
        """
        on = self.head

        while on is not None and index != 0:
            on = on.next
            index -= 1

        if on is not None:
            return on.cargo
        else:
            return False

    def delete_by_cargo(self, cargo): ...
```

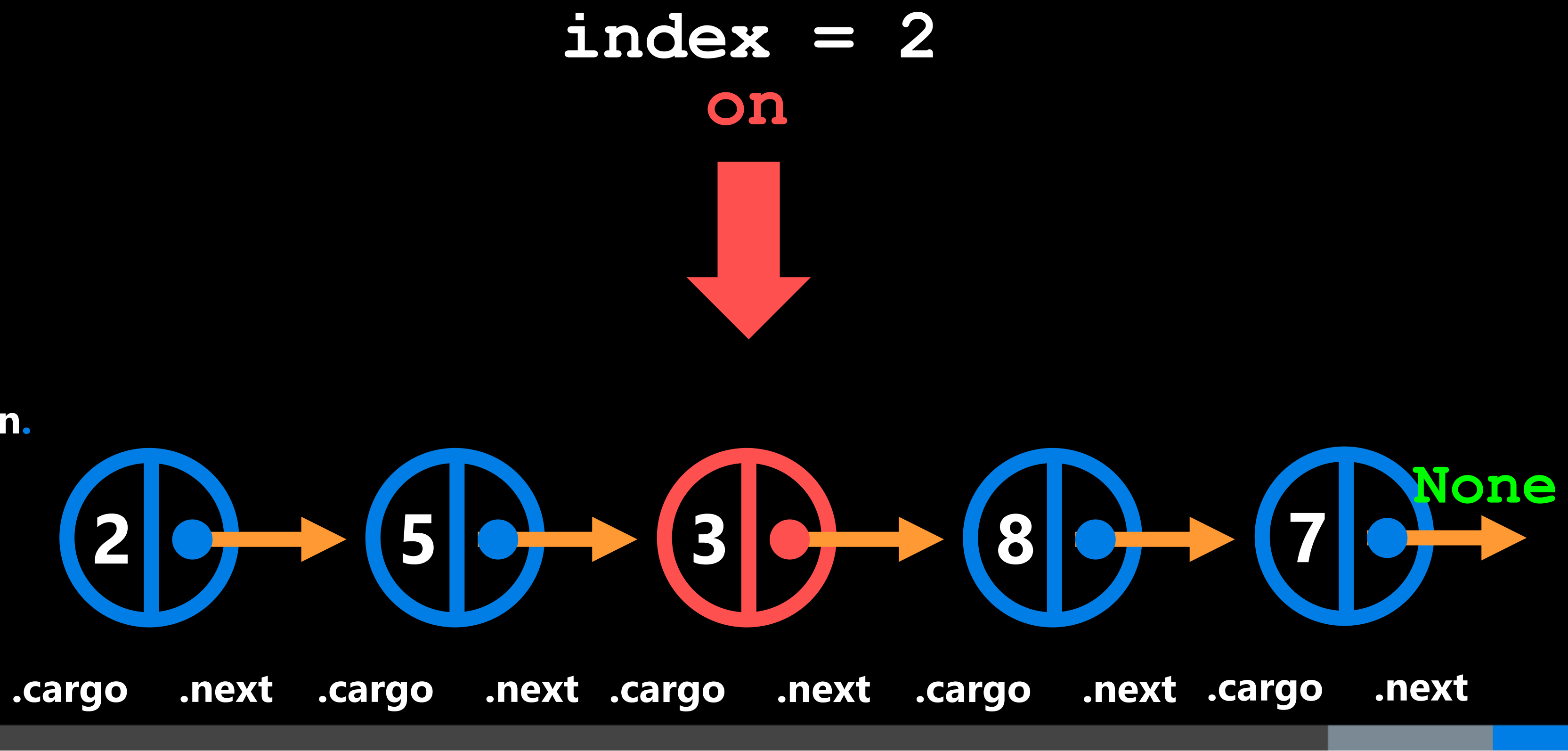
True → `while on is not None and index != 0:`

`on = on.next` ← **Move on to next position.**

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'
```

Get node at index = 3. →

```
>>> linked_list.get_at_index(3)
8
```



```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index):
        """
        (self, object) -> NoneType
        Add a new node at certain index.
        """
        on = self.head

        while on is not None and index != 0:
            on = on.next
            index -= 1

        if on is not None:
            return on.cargo
        else:
            return False

    def delete_by_cargo(self, cargo): ...
```

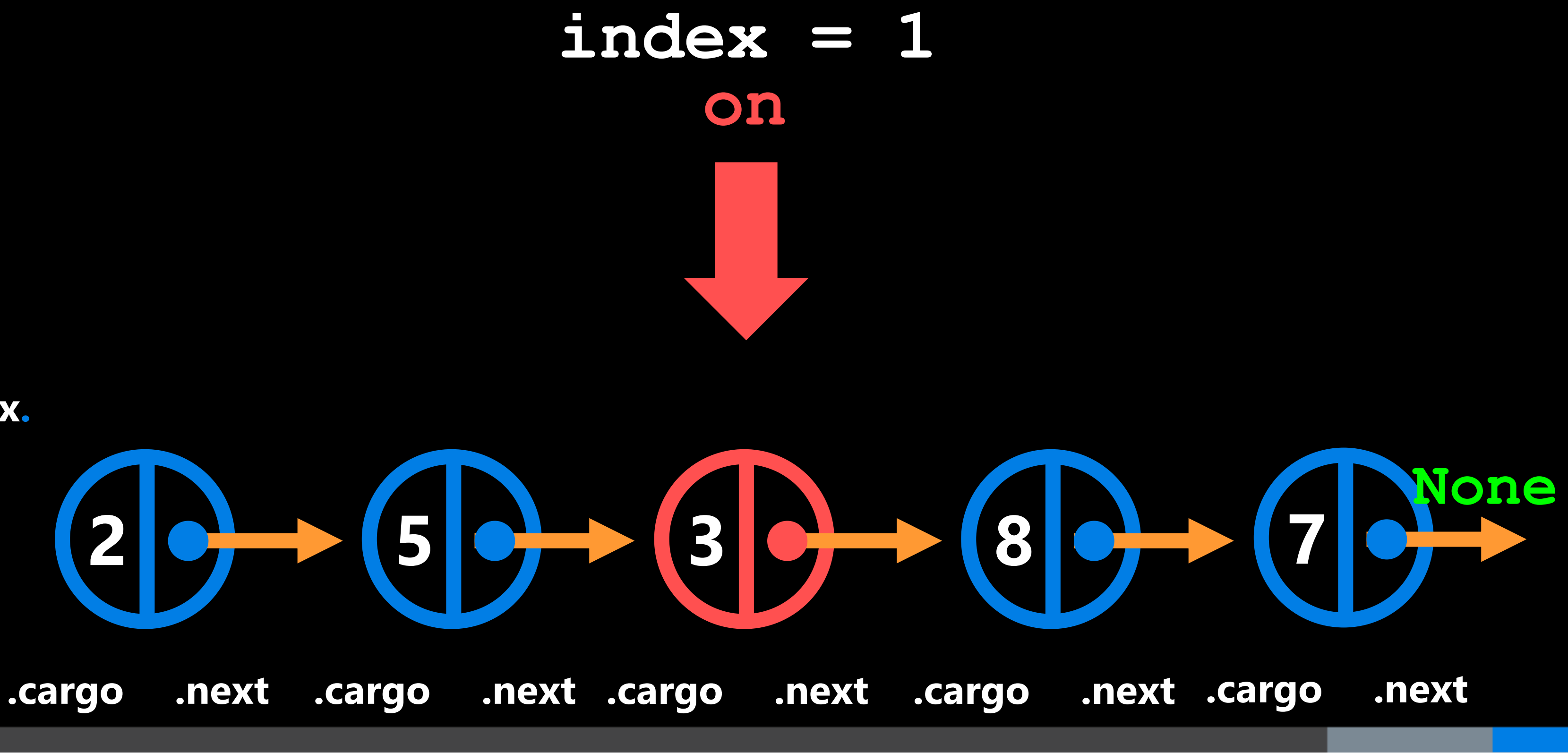
True → while on is not None and index != 0:

Update index. ← index -= 1

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'
```

Get node at index = 3. →

```
>>> linked_list.get_at_index(3)
8
```



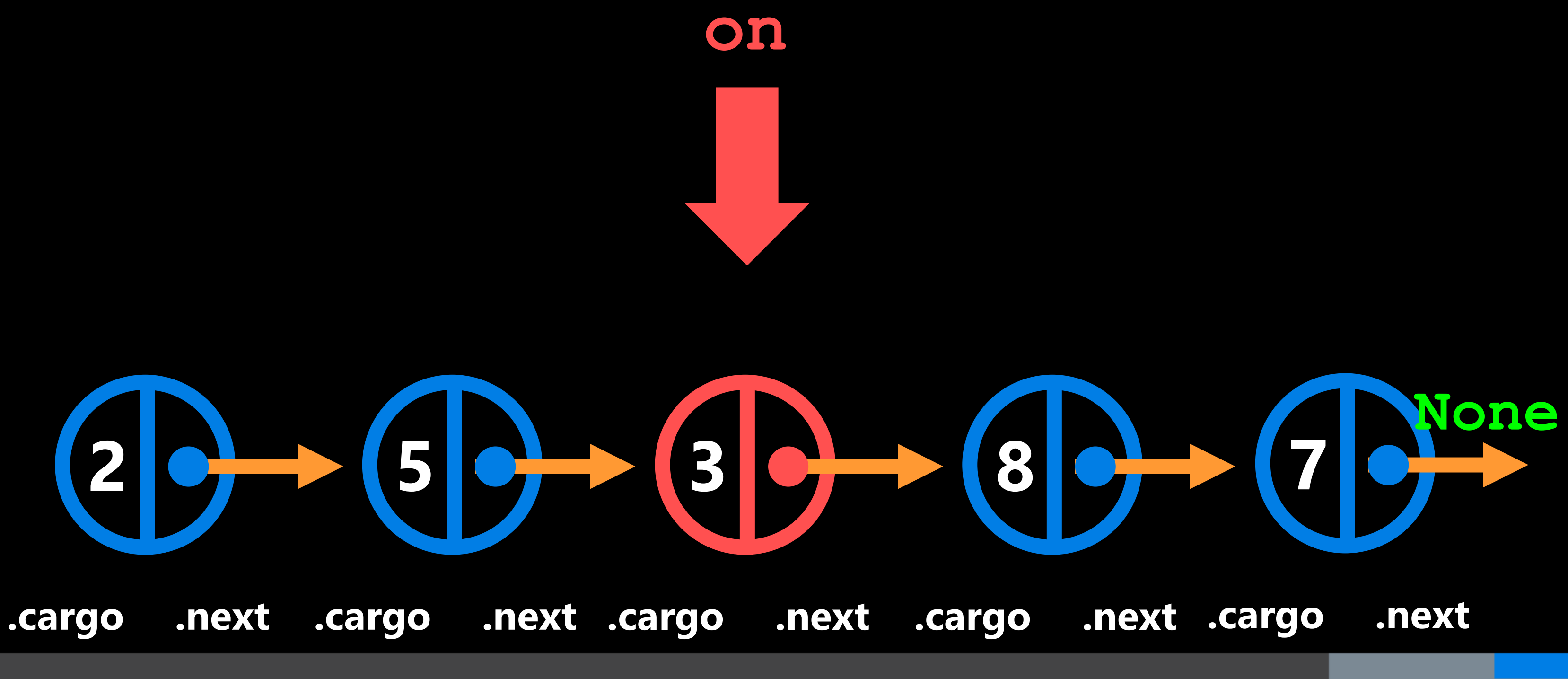
```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index):
        """
        (self, object) -> NoneType
        Add a new node at certain index.
        """
        on = self.head

True    while on is not None and index != 0:
            on = on.next
            index -= 1

        if on is not None:
            return on.cargo
        else:
            return False

    def delete_by_cargo(self, cargo): ...
```

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'
```

Get node at index = 3.

```
>>> linked_list.get_at_index(3)
8
```

index = 1



```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index):
        """
        (self, object) -> NoneType
        Add a new node at certain index.
        """
        on = self.head

        while on is not None and index != 0:
            on = on.next
            index -= 1

        if on is not None:
            return on.cargo
        else:
            return False

    def delete_by_cargo(self, cargo): ...
```

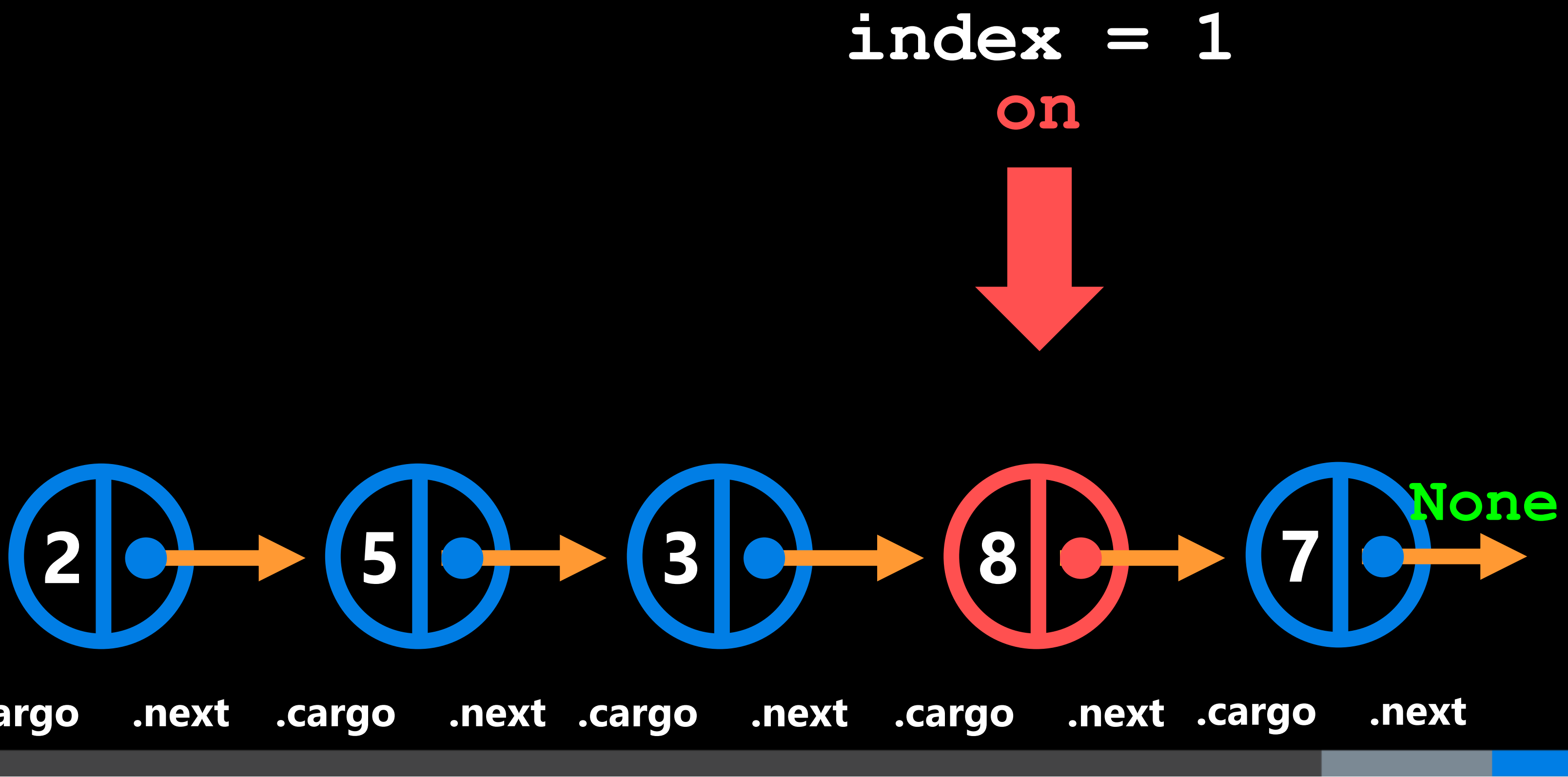
True

Move on to next position.

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'
```

Get node at index = 3.

```
>>> linked_list.get_at_index(3)
8
```



```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index):
        """
        (self, object) -> NoneType
        Add a new node at certain index.
        """
        on = self.head

        while on is not None and index != 0:
            on = on.next
            index -= 1

        if on is not None:
            return on.cargo
        else:
            return False

    def delete_by_cargo(self, cargo): ...
```

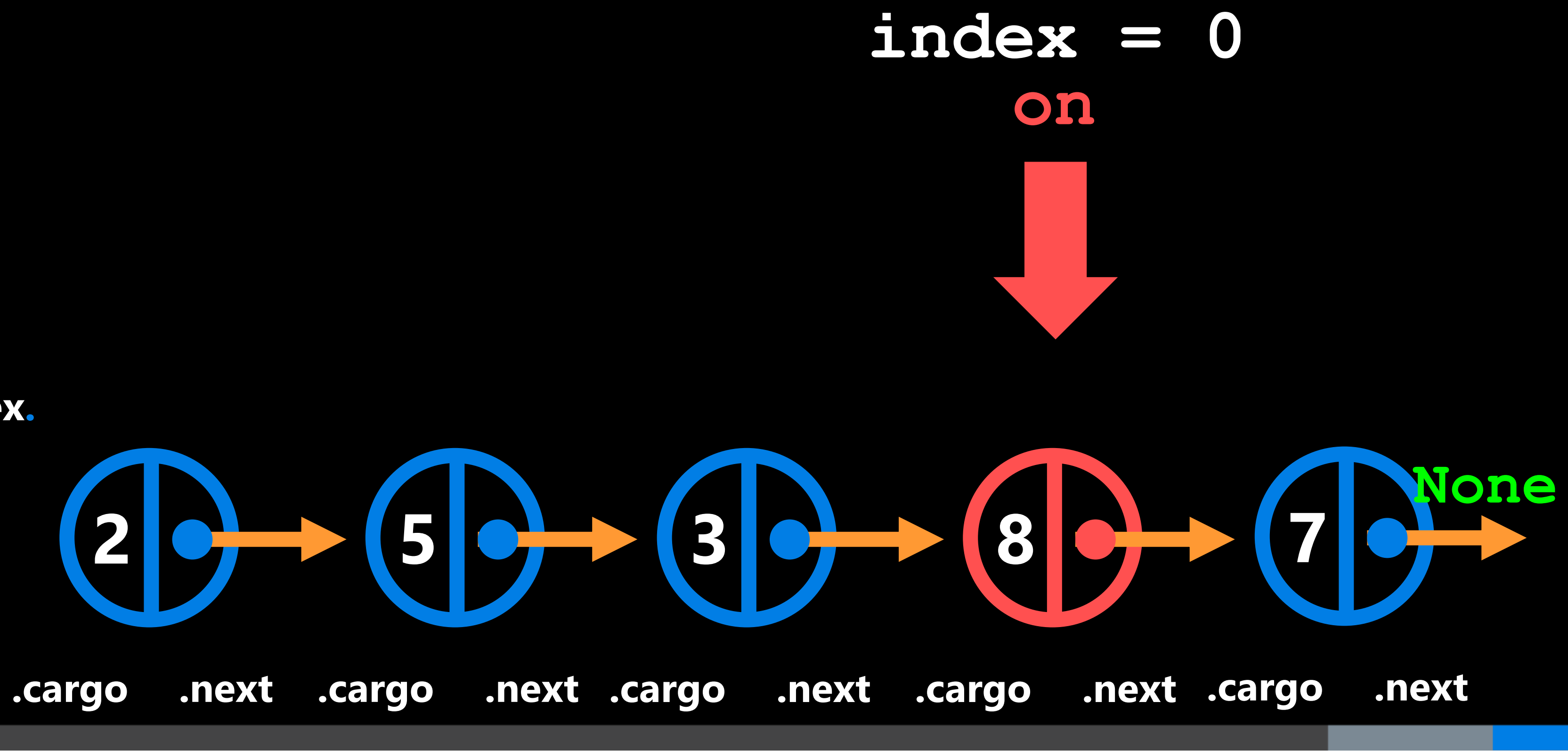
True

Update index.

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'
```

Get node at index = 3.

```
>>> linked_list.get_at_index(3)
8
```

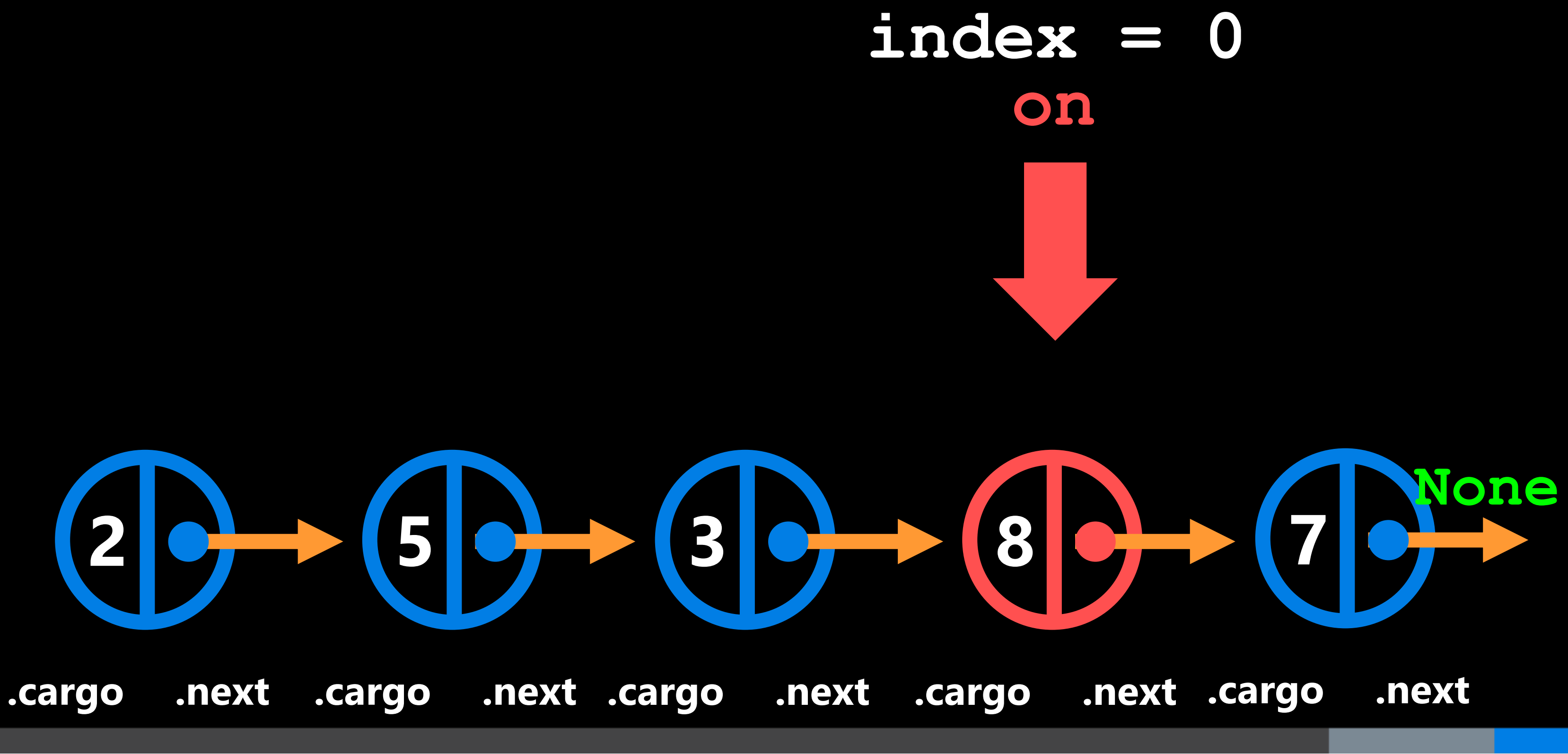


```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index):
        """
        (self, object) -> NoneType
        Add a new node at certain index.
        """
        on = self.head

False   while on is not None and index != 0:
            on = on.next
            index -= 1

        if on is not None:
            return on.cargo
        else:
            return False

    def delete_by_cargo(self, cargo): ...
```

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'
```

Get node at index = 3.

```
>>> linked_list.get_at_index(3)
8
```



```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index):
        """
        (self, object) -> NoneType
        Add a new node at certain index.
        """
        on = self.head

        while on is not None and index != 0:
            on = on.next
            index -= 1

        if on is not None:
            return on.cargo
        else:
            return False

    def delete_by_cargo(self, cargo): ...
```

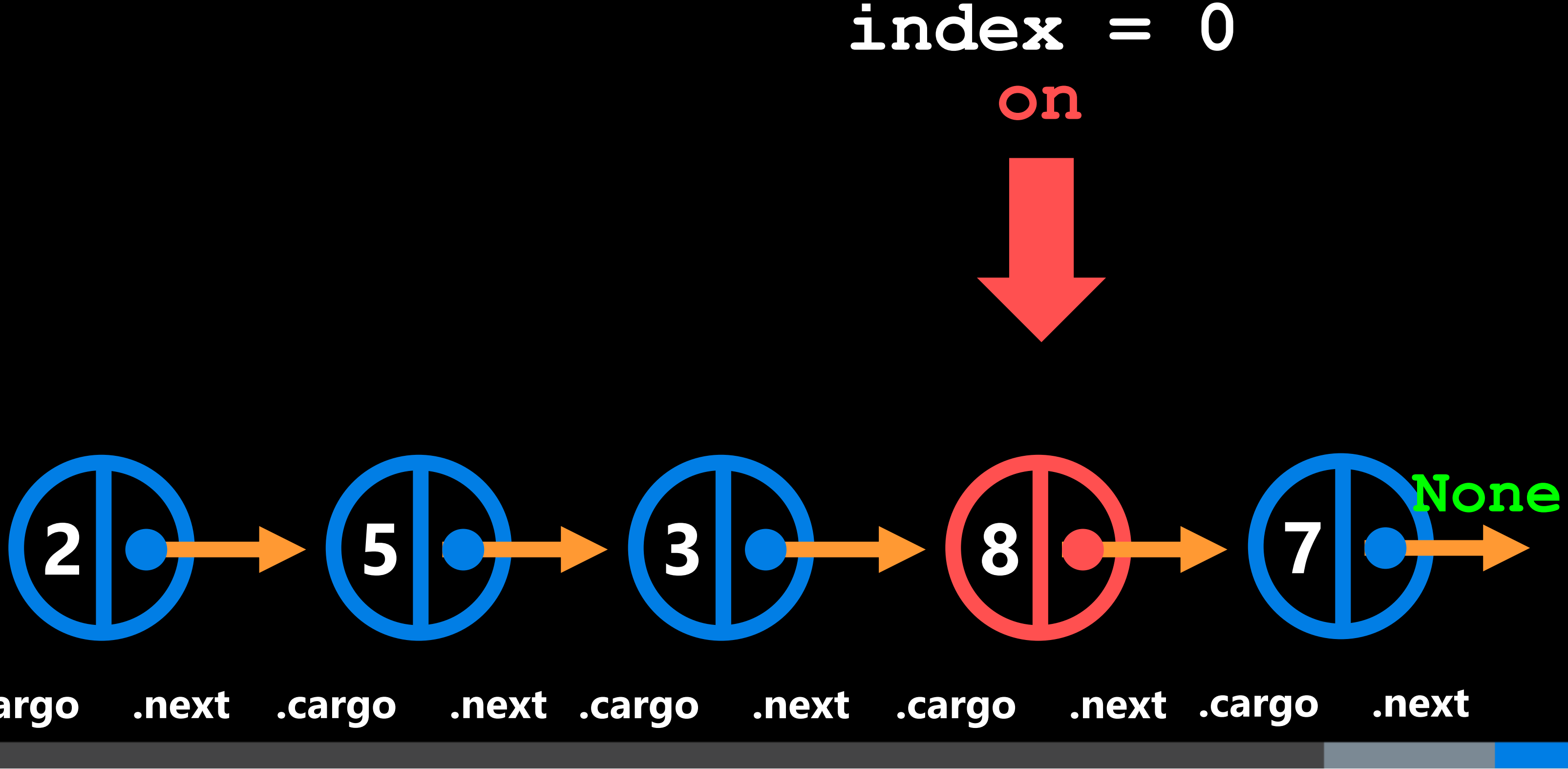
True ➡ if on is not None:

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'
```

Get node at index = 3. ➡

```
>>> linked_list.get_at_index(3)
8
```

index = 0



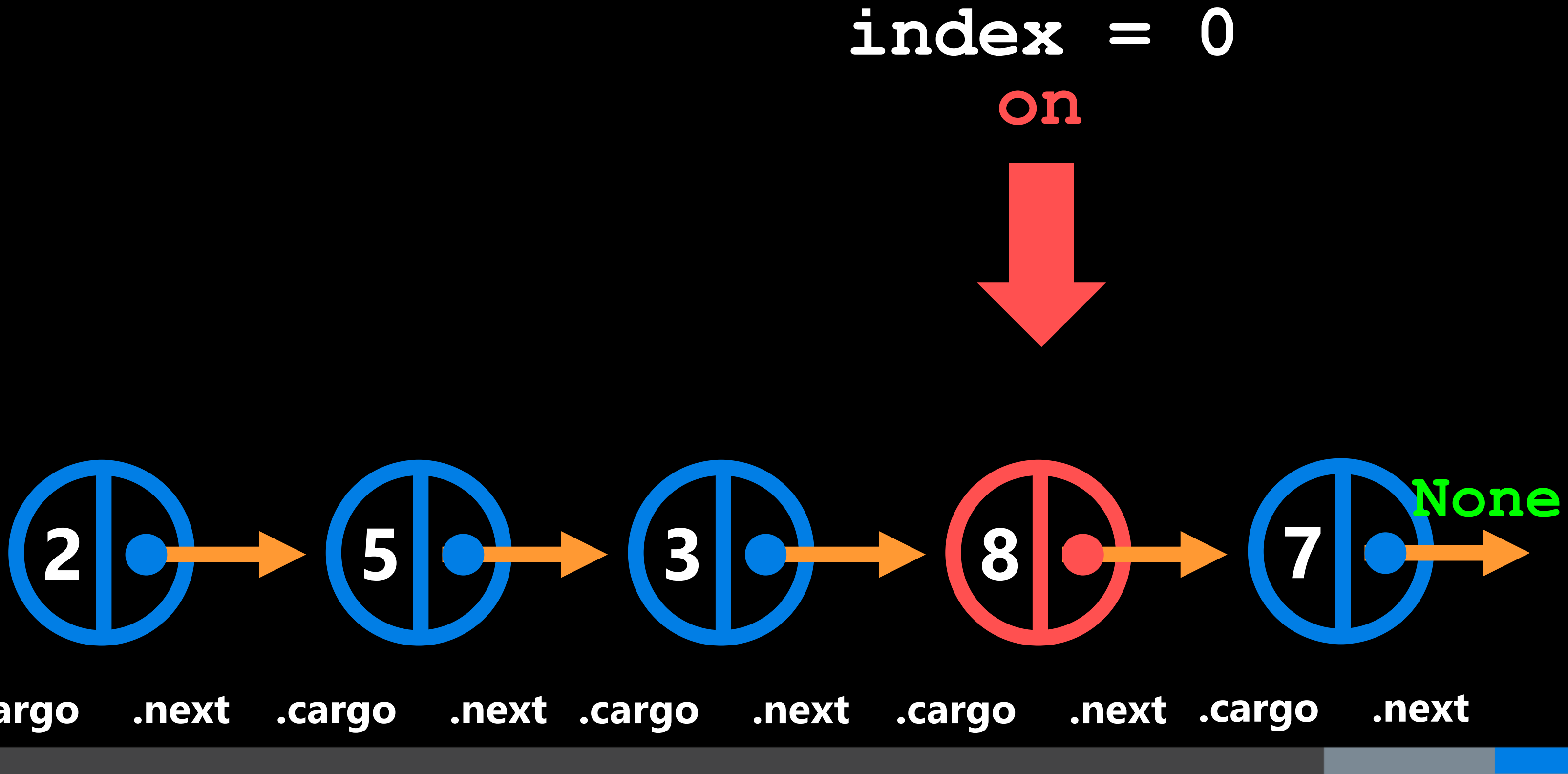
```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index):
        """
        (self, object) -> NoneType
        Add a new node at certain index.
        """
        on = self.head

        while on is not None and index != 0:
            on = on.next
            index -= 1

True    if on is not None:
            return on.cargo
        else:
            return False

    def delete_by_cargo(self, cargo): ...
```

Return cargo at on.

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'
```

Get node at index = 3.

```
>>> linked_list.get_at_index(3)
8
```

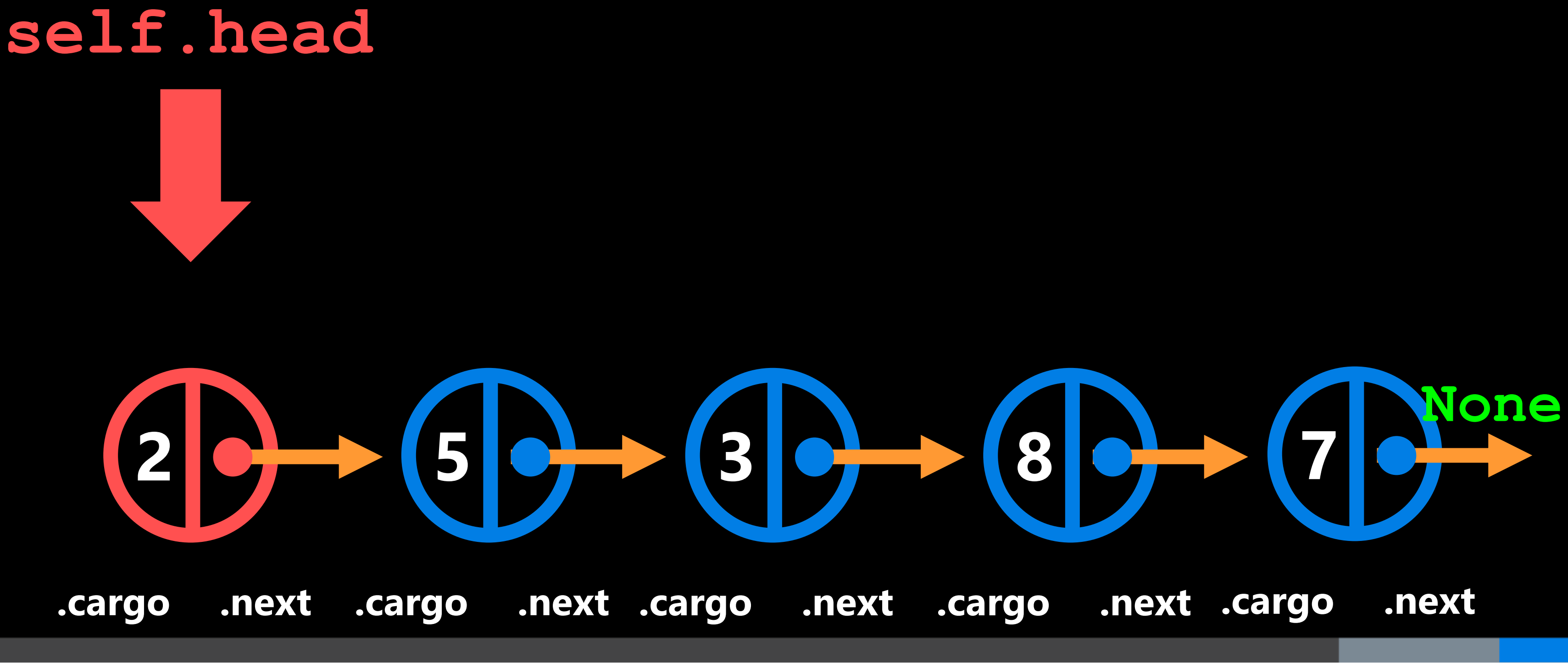


delete_by_cargo method.

```python
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        """
        (self) -> NoneType
        Create an empty linked list.
        """
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo): ...
```

```python
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo):
        """
        (self, object) -> NoneType
        Remove all nodes with certain
        cargo value.
        """
        on = self.head

        while on and on.next:

            if on.next.cargo == cargo:
                on.next = on.next.next

            on = on.next
```

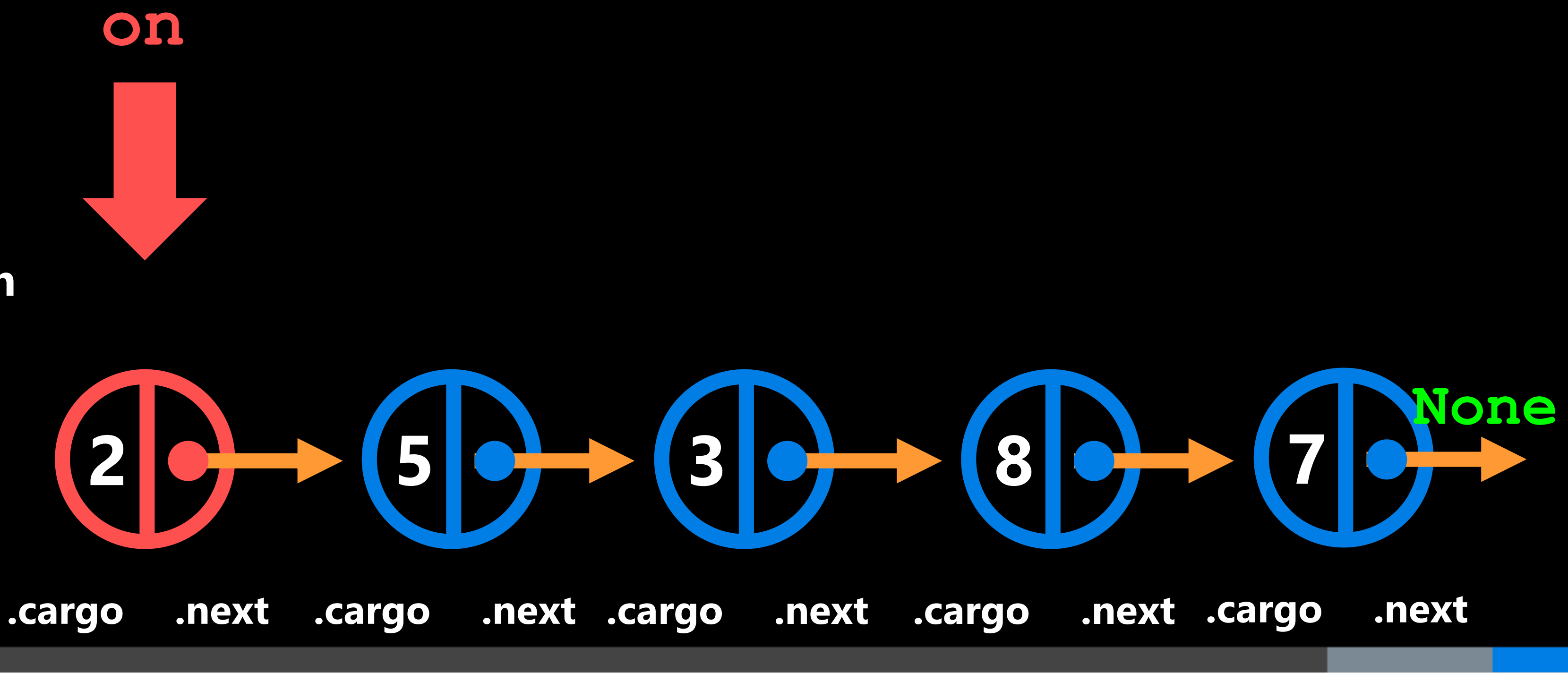
```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'
```



```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo):
        """
        (self, object) -> NoneType
        Remove all nodes with certain
        cargo value.
        """
        on = self.head

        while on and on.next:

            if on.next.cargo == cargo:
                on.next = on.next.next

            on = on.next
```

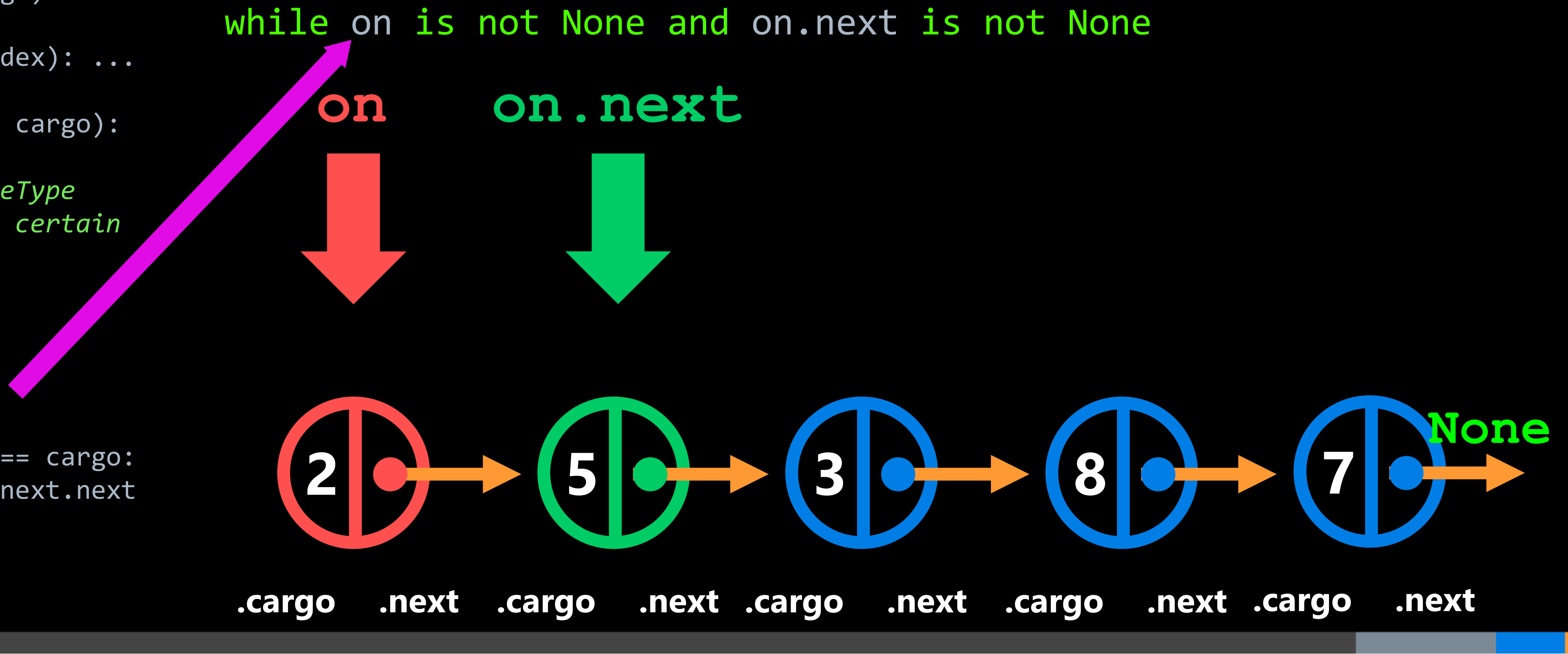
Set on position

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'

>>> linked_list.delete_by_cargo(3)
```



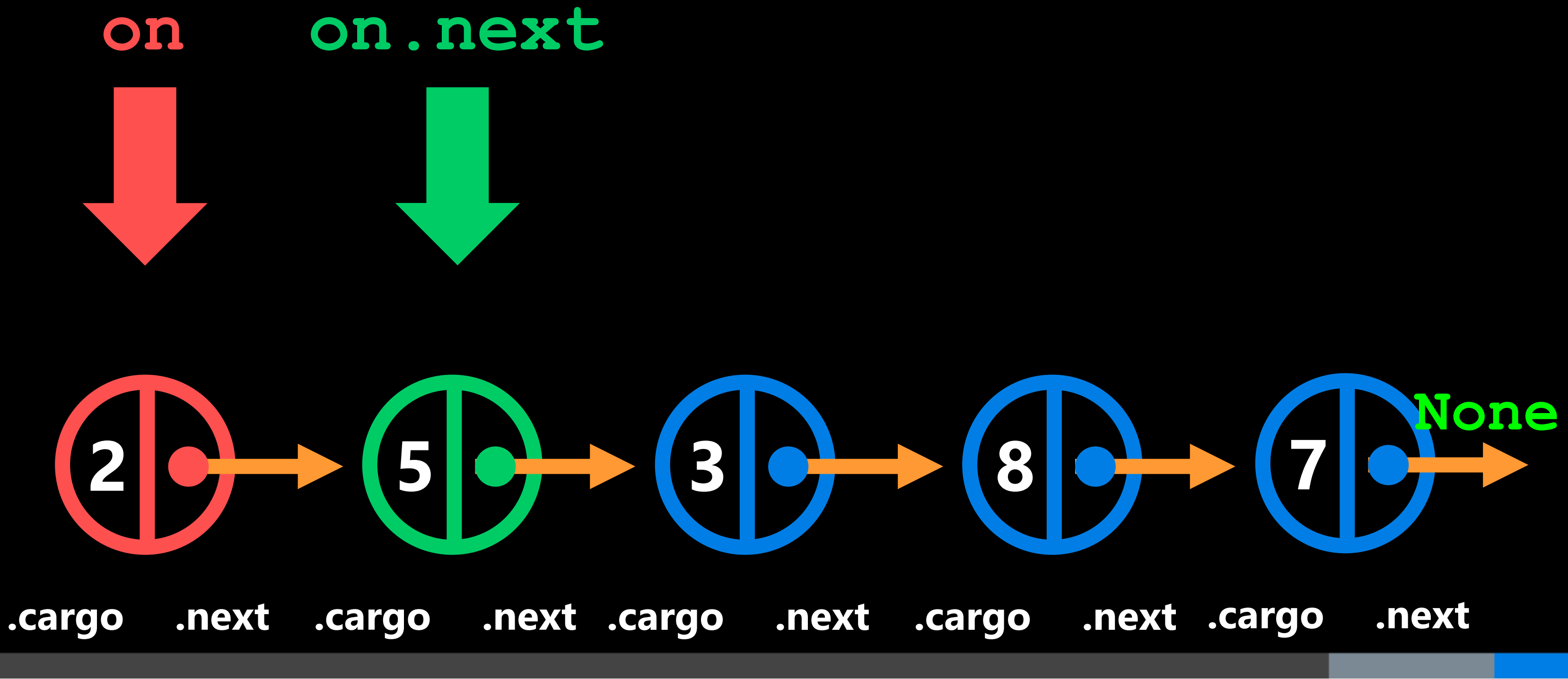
```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo):
        """
        (self, object) -> NoneType
        Remove all nodes with certain
        cargo value.
        """
        on = self.head

        while on and on.next:

            if on.next.cargo == cargo:
                on.next = on.next.next

            on = on.next
```

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'

>>> linked_list.delete_by_cargo(3)
```



```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo):
        """
        (self, object) -> NoneType
        Remove all nodes with certain
        cargo value.
        """
        on = self.head

True ➡  while on and on.next:

  False ➡  if on.next.cargo == cargo:
                on.next = on.next.next

            on = on.next
```

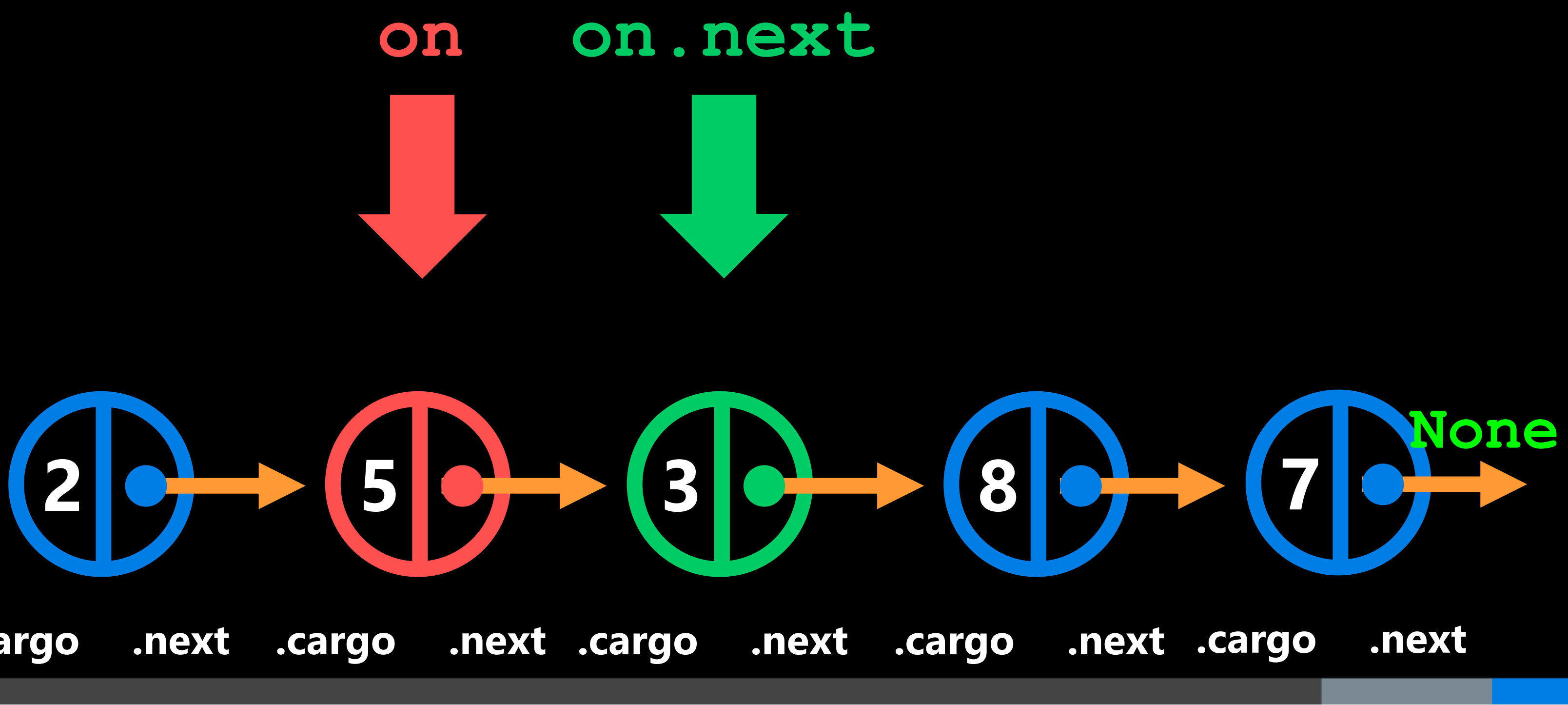
```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'

>>> linked_list.delete_by_cargo(3)
```



```python
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo):
        """
        (self, object) -> NoneType
        Remove all nodes with certain
        cargo value.
        """
        on = self.head

True ➡  while on and on.next:

  False ➡  if on.next.cargo == cargo:
                on.next = on.next.next

            on = on.next ⬅
```

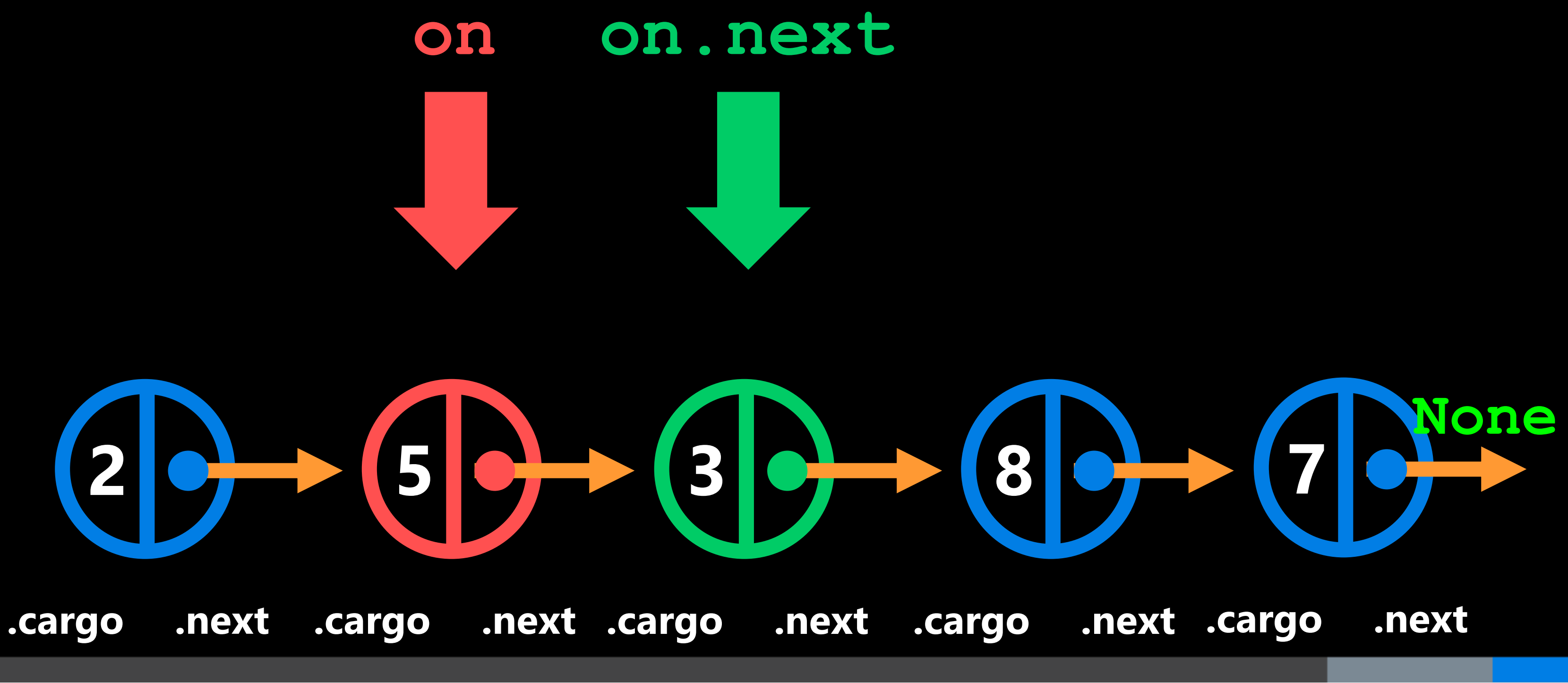
Move on to next position.

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'

>>> linked_list.delete_by_cargo(3)
```



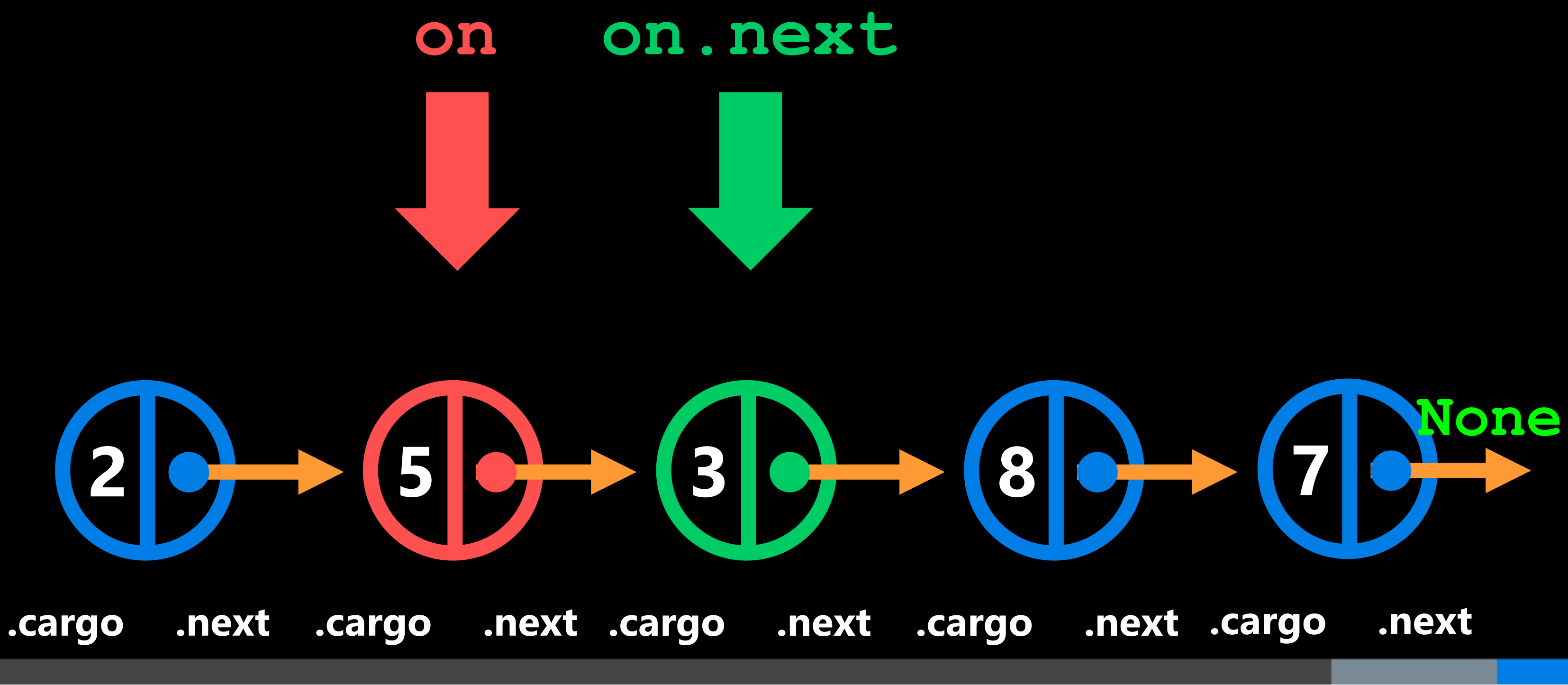
```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo):
        """
        (self, object) -> NoneType
        Remove all nodes with certain
        cargo value.
        """
        on = self.head

True ➡  while on and on.next:

            if on.next.cargo == cargo:
                on.next = on.next.next

            on = on.next
```

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'

>>> linked_list.delete_by_cargo(3)
```



```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo):
        """
        (self, object) -> NoneType
        Remove all nodes with certain
        cargo value.
        """
        on = self.head

True ➡  while on and on.next:

    True ➡  if on.next.cargo == cargo:
                on.next = on.next.next

            on = on.next
```

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'

>>> linked_list.delete_by_cargo(3)
```



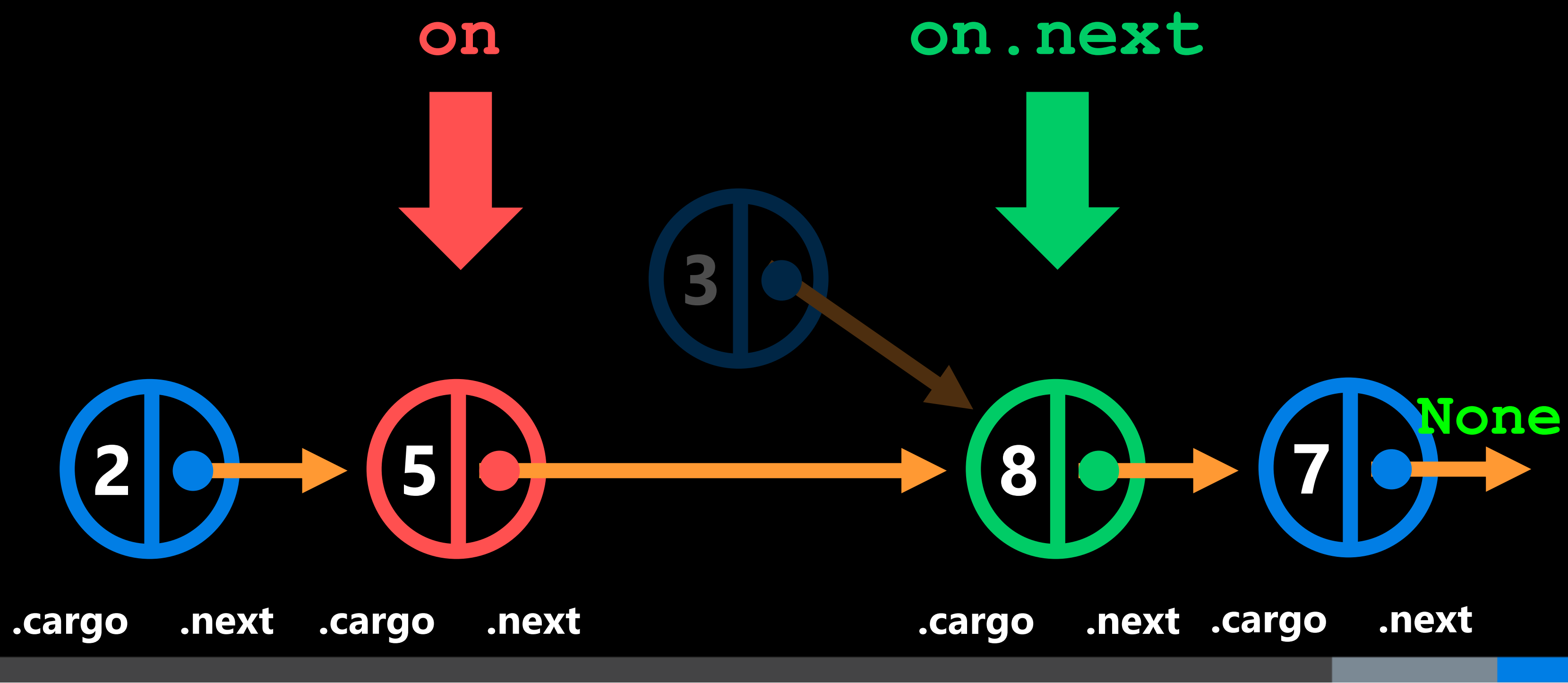
```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo):
        """
        (self, object) -> NoneType
        Remove all nodes with certain
        cargo value.
        """
        on = self.head

        while on and on.next:

            if on.next.cargo == cargo:
                on.next = on.next.next

            on = on.next
```

True → while on and on.next:

True → if on.next.cargo == cargo:

Update pointer. → on.next = on.next.next

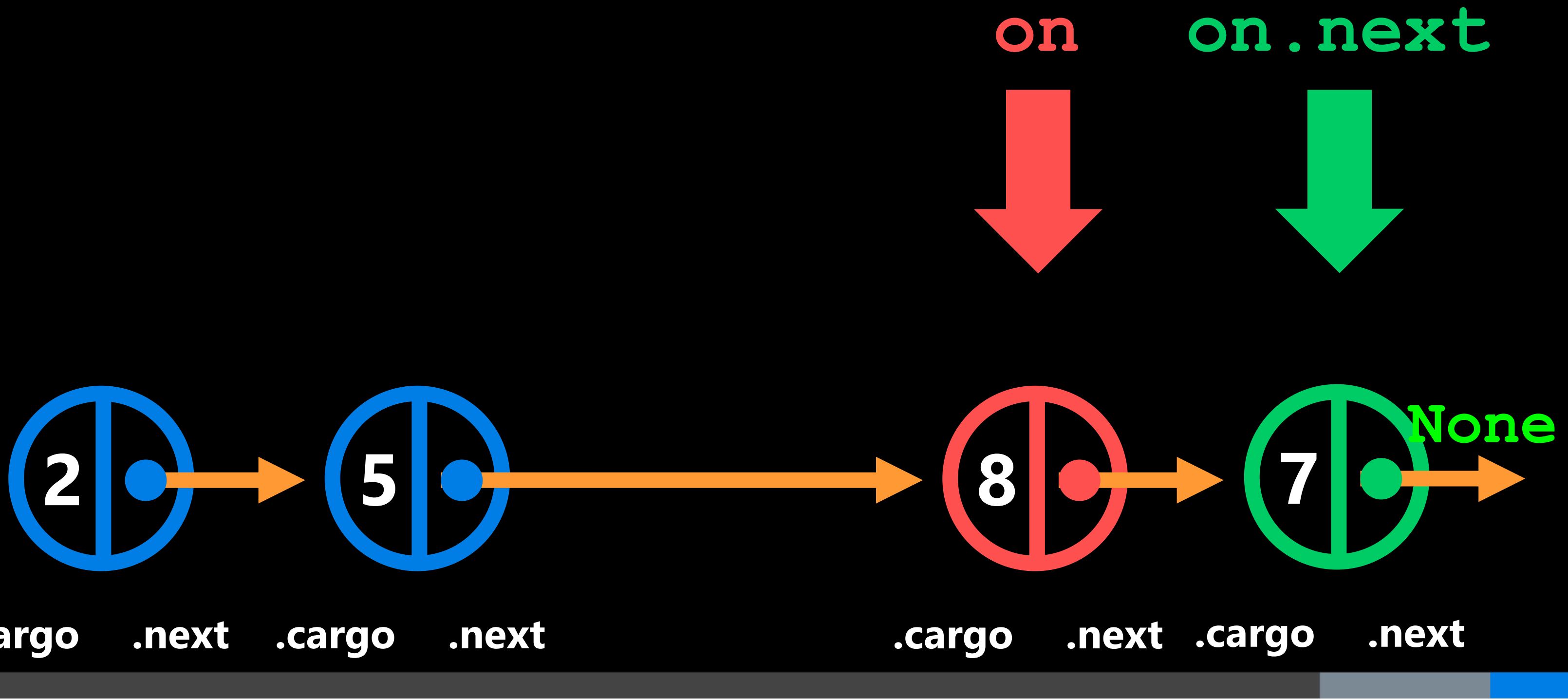
```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'

>>> linked_list.delete_by_cargo(3)
```



```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo):
        """
        (self, object) -> NoneType
        Remove all nodes with certain
        cargo value.
        """
        on = self.head

True ➡  while on and on.next:

    True ➡  if on.next.cargo == cargo:
                on.next = on.next.next

            on = on.next ⬅
```

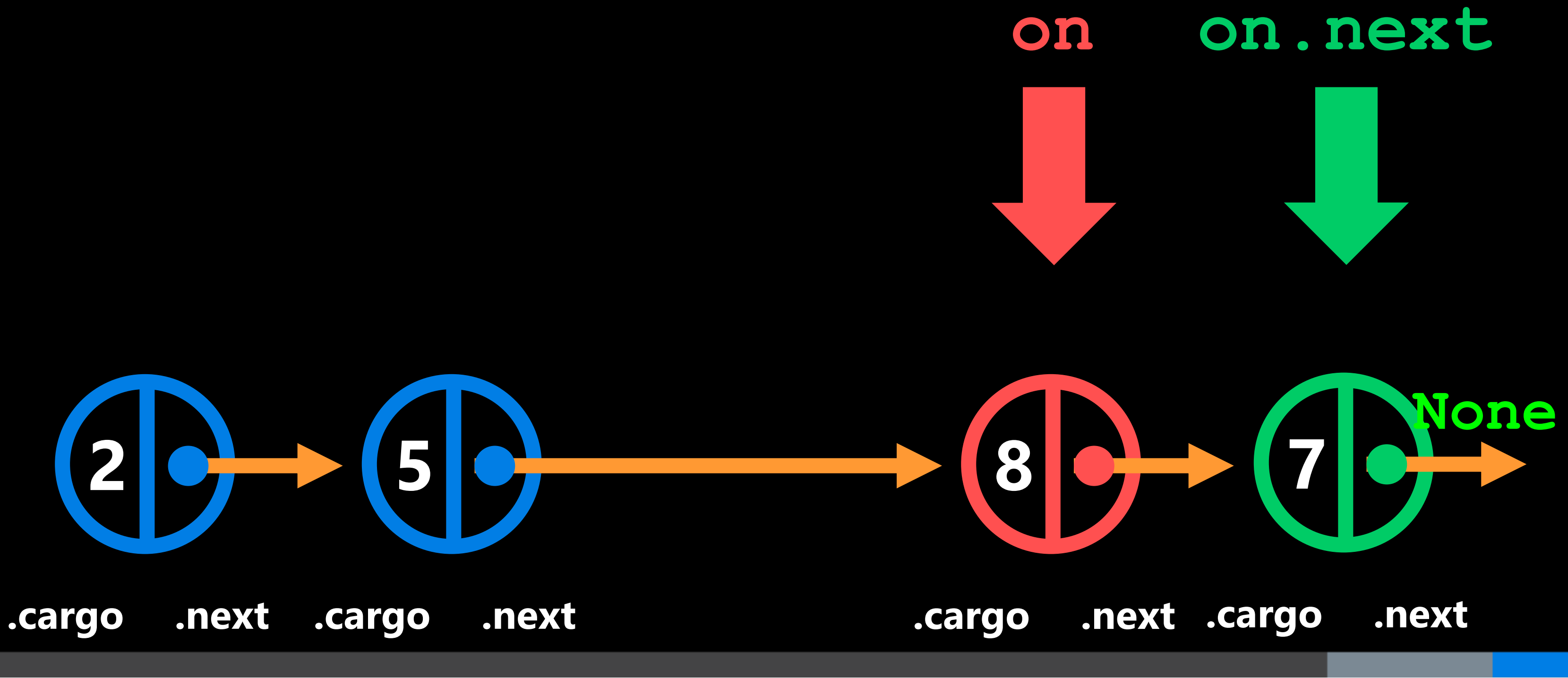
Move on to next position.

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'

>>> linked_list.delete_by_cargo(3)
```



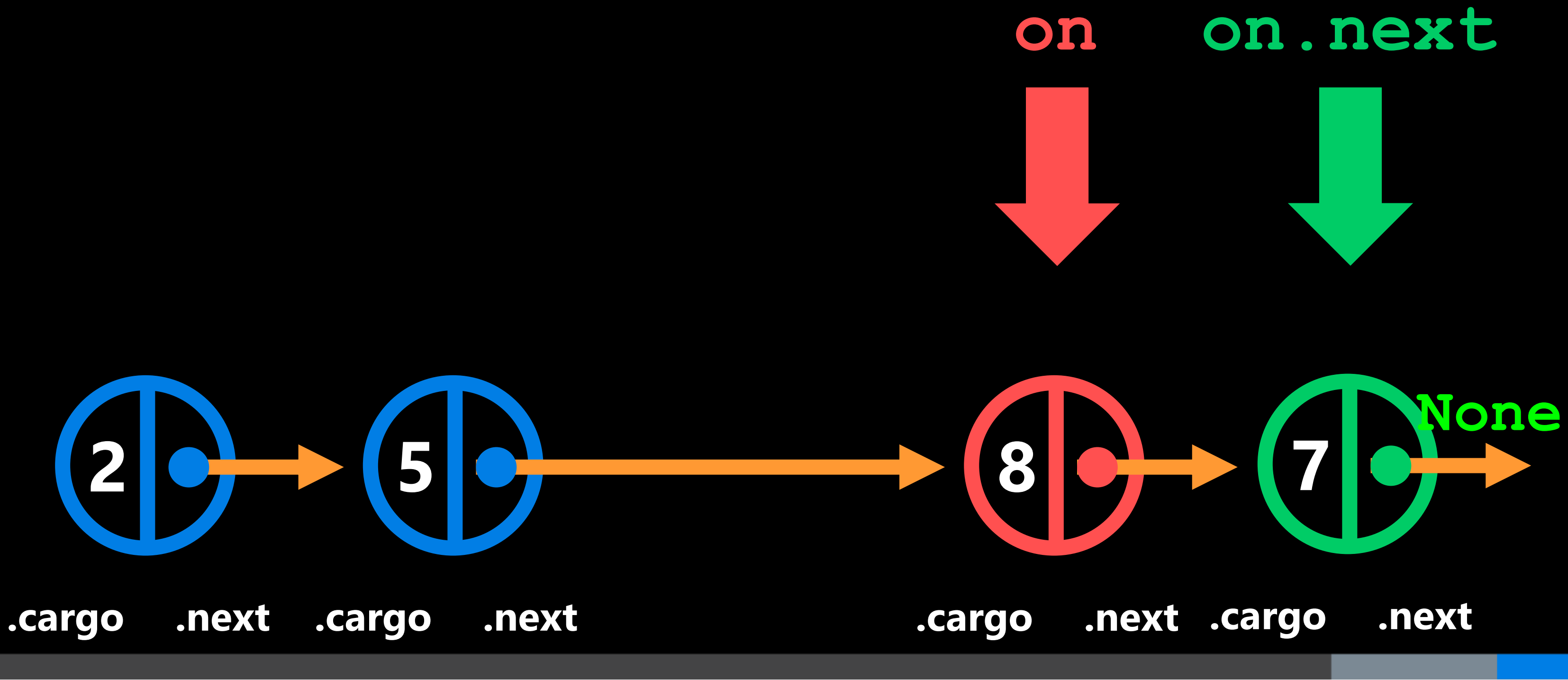
```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo):
        """
        (self, object) -> NoneType
        Remove all nodes with certain
        cargo value.
        """
        on = self.head

True ➡  while on and on.next:

            if on.next.cargo == cargo:
                on.next = on.next.next

            on = on.next
```

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'

>>> linked_list.delete_by_cargo(3)
```



```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo):
        """
        (self, object) -> NoneType
        Remove all nodes with certain
        cargo value.
        """
        on = self.head

True ▶  while on and on.next:

 False ▶  if on.next.cargo == cargo:
              on.next = on.next.next

          on = on.next
```

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'

>>> linked_list.delete_by_cargo(3)
```



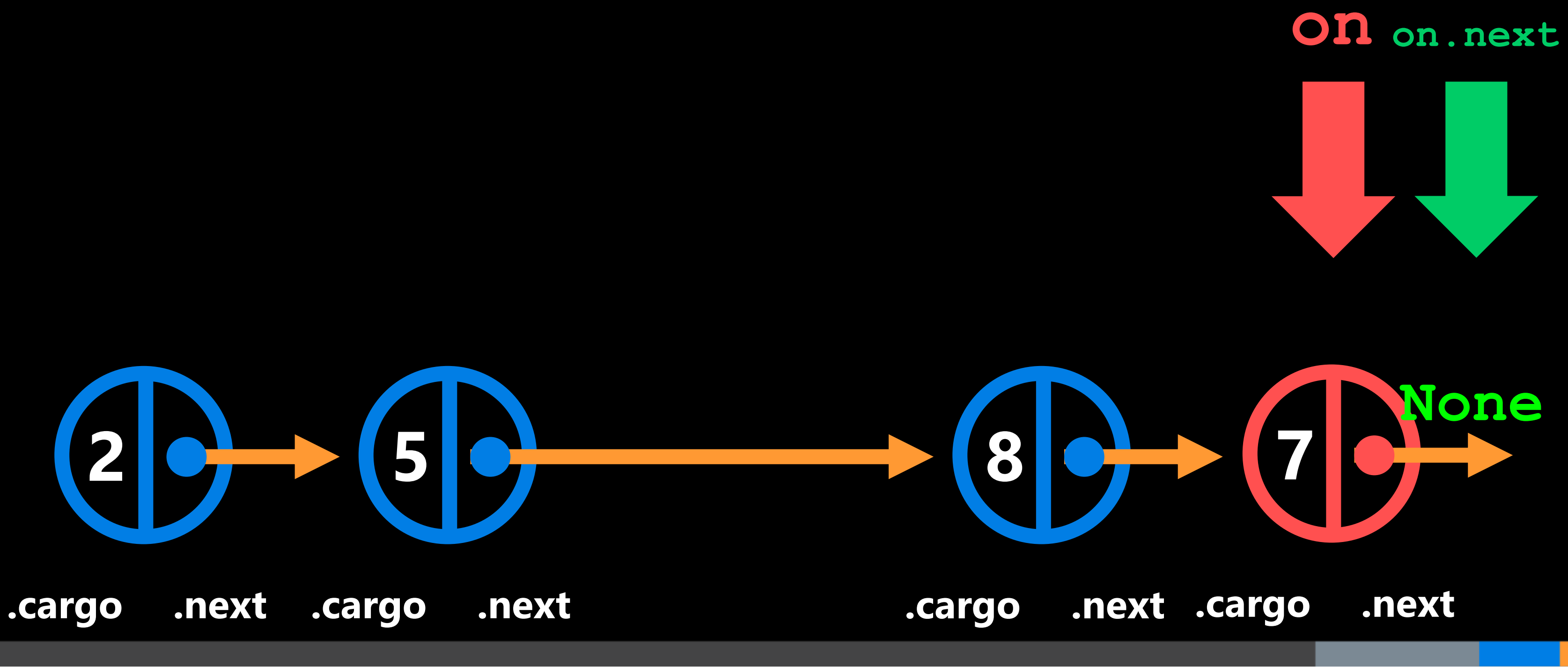
```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo):
        """
        (self, object) -> NoneType
        Remove all nodes with certain
        cargo value.
        """
        on = self.head

        while on and on.next:

            if on.next.cargo == cargo:
                on.next = on.next.next

            on = on.next
```

True ➡ while on and on.next:

False ➡ if on.next.cargo == cargo:

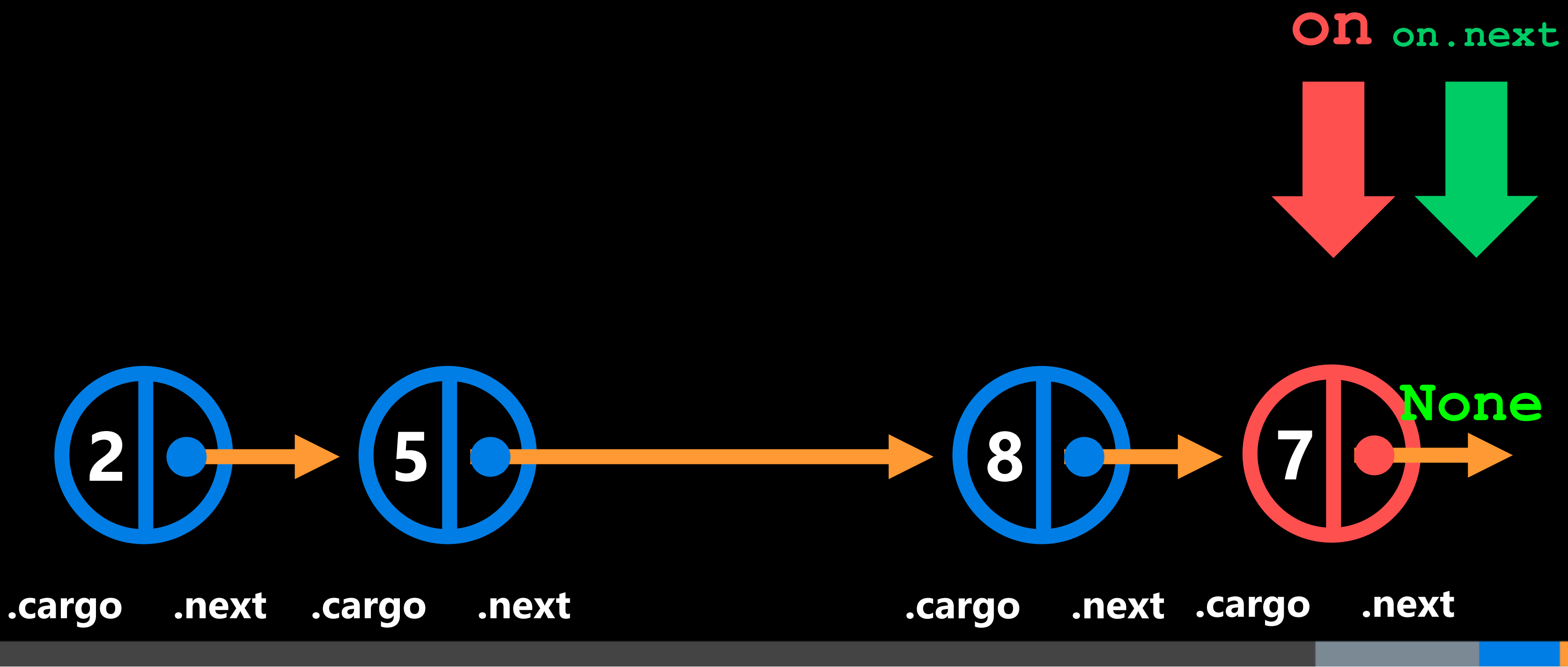
on = on.next ⬅ Move on to next position.

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'

>>> linked_list.delete_by_cargo(3)
```



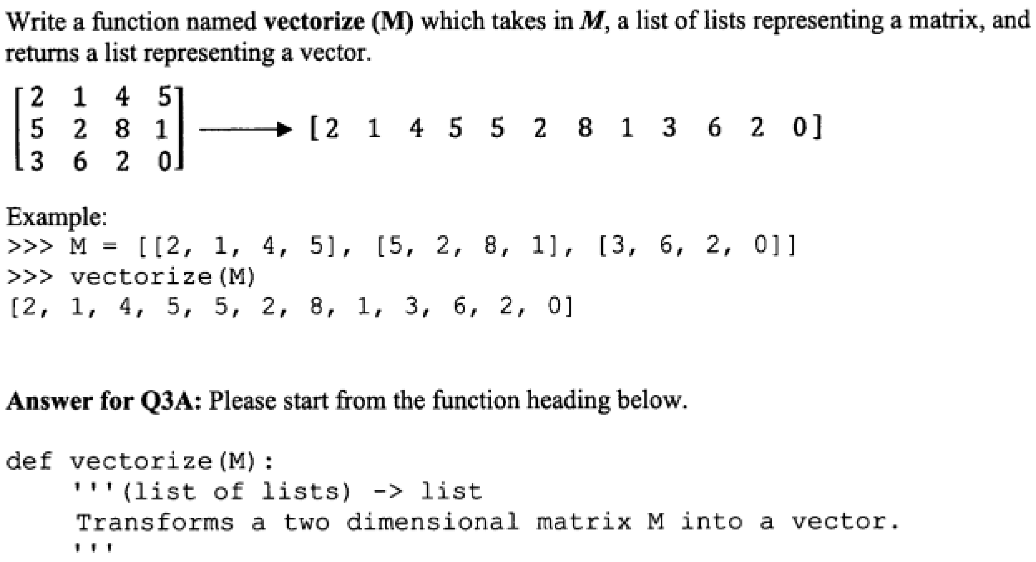
```
class LinkedList:
    """A class that implements a linked list."""

    def __init__(self):
        self.length = 0
        self.head = None

    def __str__(self): ...

    def add_to_head(self, cargo): ...

    def add_to_tail(self, cargo): ...

    def get_at_index(self, index): ...

    def delete_by_cargo(self, cargo):
        """
        (self, object) -> NoneType
        Remove all nodes with certain
        cargo value.
        """
        on = self.head

False ➡ while on and on.next:

            if on.next.cargo == cargo:
                on.next = on.next.next

            on = on.next
```

```
>>> linked_list = LinkedList()
>>> linked_list.add_to_head(7)
>>> linked_list.add_to_head(8)
>>> linked_list.add_to_head(3)
>>> linked_list.add_to_head(6)
>>> linked_list.add_to_head(2)
>>> linked_list.__str__()
'(2) --> (5) --> (3) --> (8) --> (7) --> None'

>>> linked_list.delete_by_cargo(3)
```



See Jupyter Notebook for the solution!

Question 4 (if time)



Write a function named **vectorize (M)** which takes in ***M***, a list of lists representing a matrix, and returns a list representing a vector.

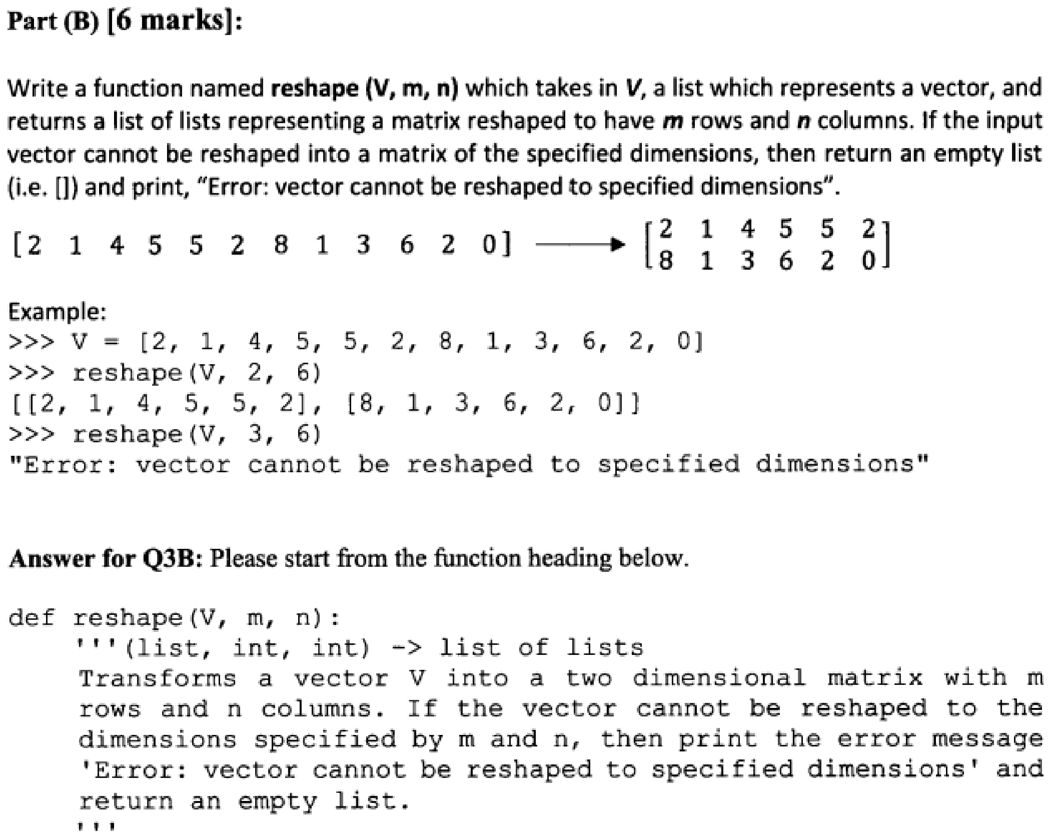
$$\begin{bmatrix}2&1&4&5\\5&2&8&1\\3&6&2&0\end{bmatrix} \longrightarrow [2\ \ 1\ \ 4\ \ 5\ \ 5\ \ 2\ \ 8\ \ 1\ \ 3\ \ 6\ \ 2\ \ 0]$$

Example:

```
>>> M = [[2, 1, 4, 5], [5, 2, 8, 1], [3, 6, 2, 0]]
>>> vectorize(M)
[2, 1, 4, 5, 5, 2, 8, 1, 3, 6, 2, 0]
```

Answer for Q3A: Please start from the function heading below.

```
def vectorize(M):
    '''(list of lists) -> list
    Transforms a two dimensional matrix M into a vector.
    '''
```



Part (B) [6 marks]:

Write a function named **reshape (V, m, n)** which takes in ***V***, a list which represents a vector, and returns a list of lists representing a matrix reshaped to have ***m*** rows and ***n*** columns. If the input vector cannot be reshaped into a matrix of the specified dimensions, then return an empty list (i.e. []) and print, "Error: vector cannot be reshaped to specified dimensions".

$$[2\ \ 1\ \ 4\ \ 5\ \ 5\ \ 2\ \ 8\ \ 1\ \ 3\ \ 6\ \ 2\ \ 0] \longrightarrow \begin{bmatrix}2&1&4&5&5&2\\8&1&3&6&2&0\end{bmatrix}$$

Example:

```
>>> V = [2, 1, 4, 5, 5, 2, 8, 1, 3, 6, 2, 0]
>>> reshape(V, 2, 6)
[[2, 1, 4, 5, 5, 2], [8, 1, 3, 6, 2, 0]]
>>> reshape(V, 3, 6)
"Error: vector cannot be reshaped to specified dimensions"
```

Answer for Q3B: Please start from the function heading below.

```
def reshape(V, m, n):
    '''(list, int, int) -> list of lists
    Transforms a vector V into a two dimensional matrix with m
    rows and n columns. If the vector cannot be reshaped to the
    dimensions specified by m and n, then print the error message
    'Error: vector cannot be reshaped to specified dimensions' and
    return an empty list.
    '''
```

See Jupyter Notebook for the solution!