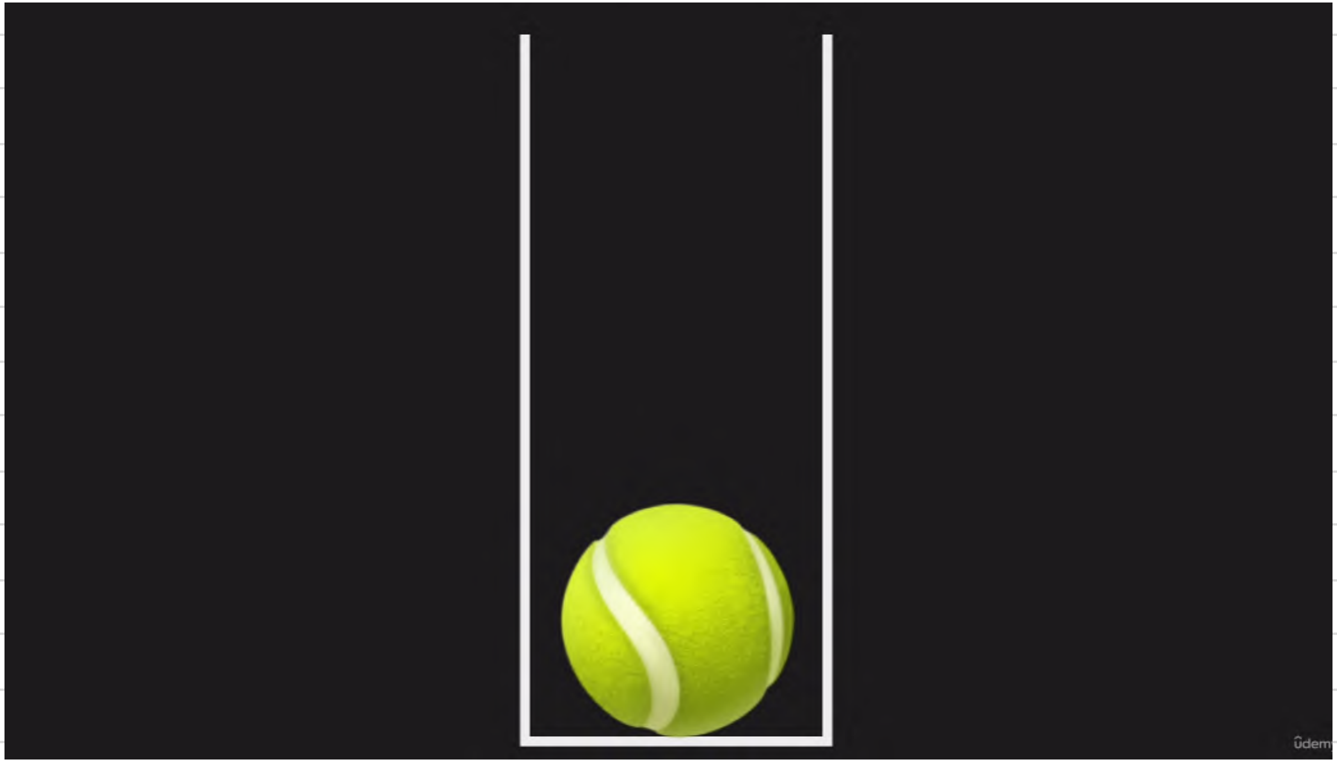
Struktur Data

Meet 07

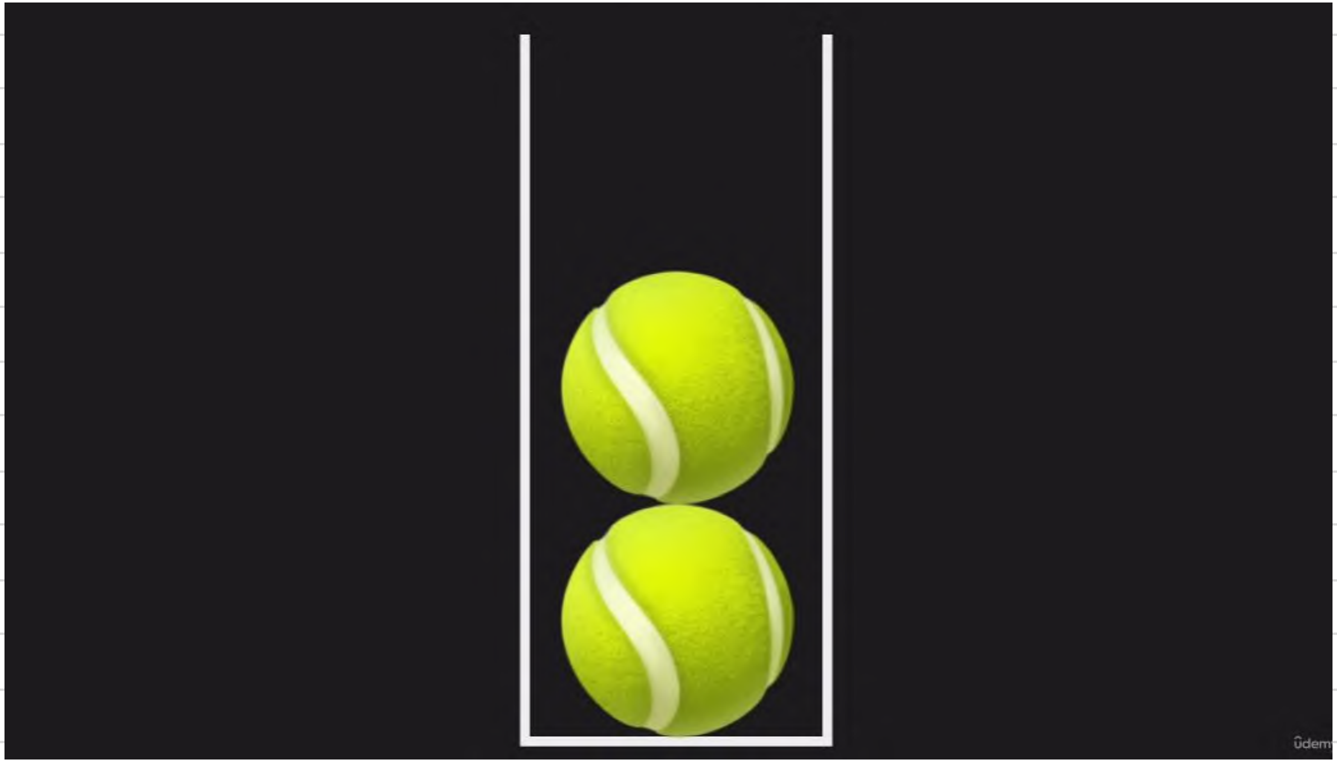
01

Stack Intro

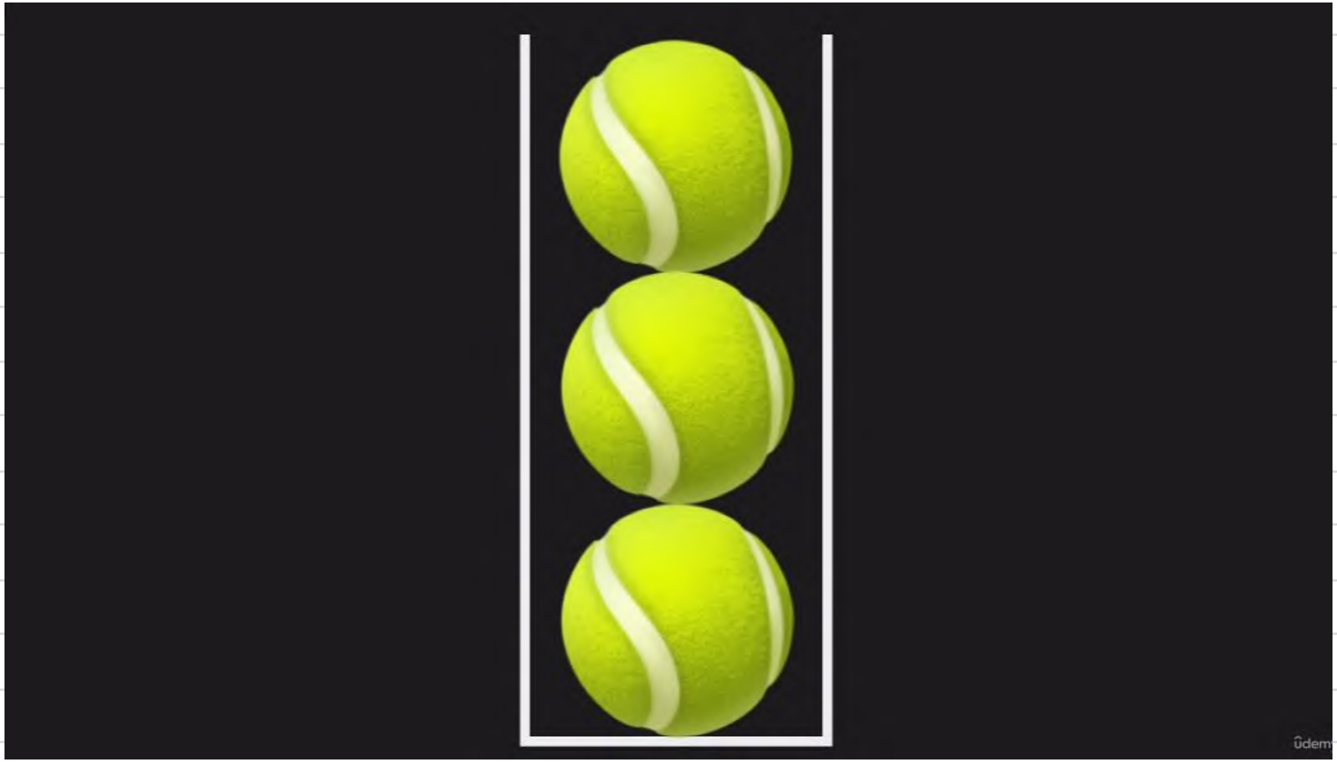
Stack



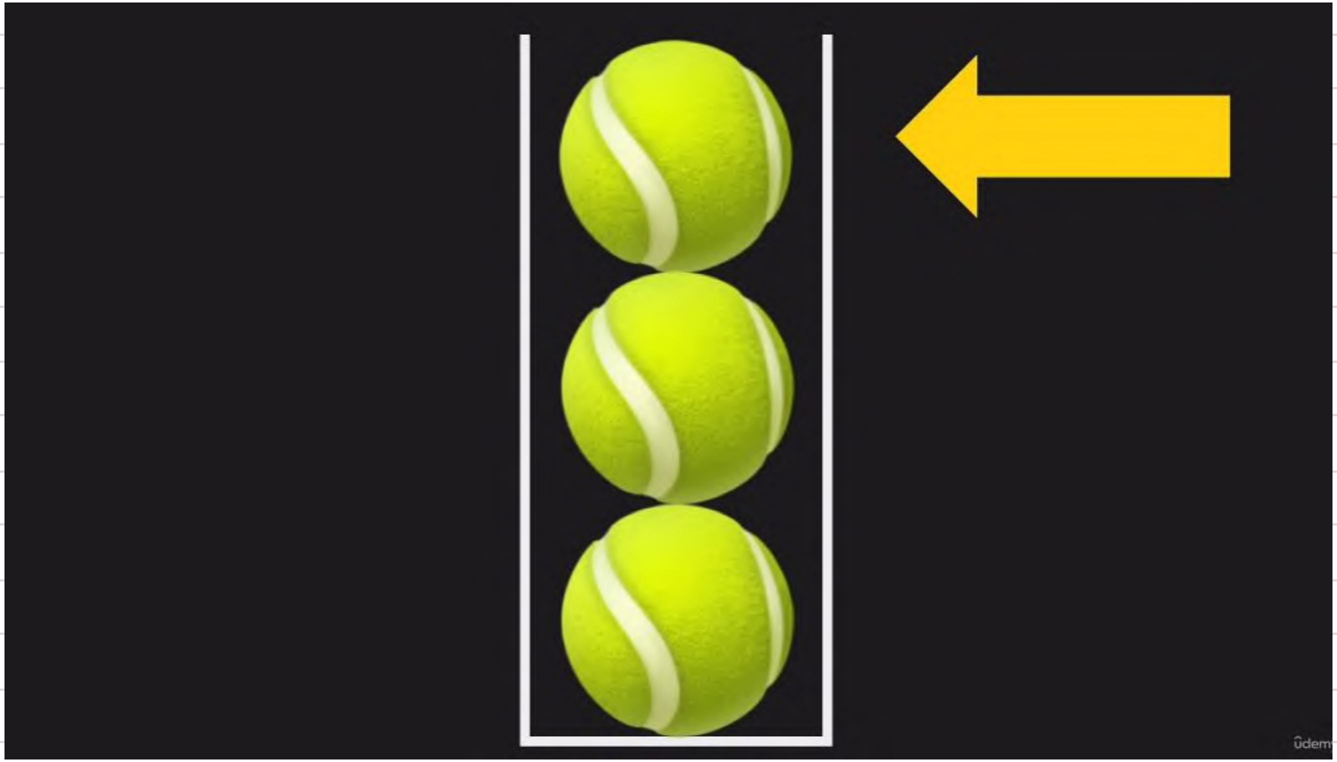
Stack



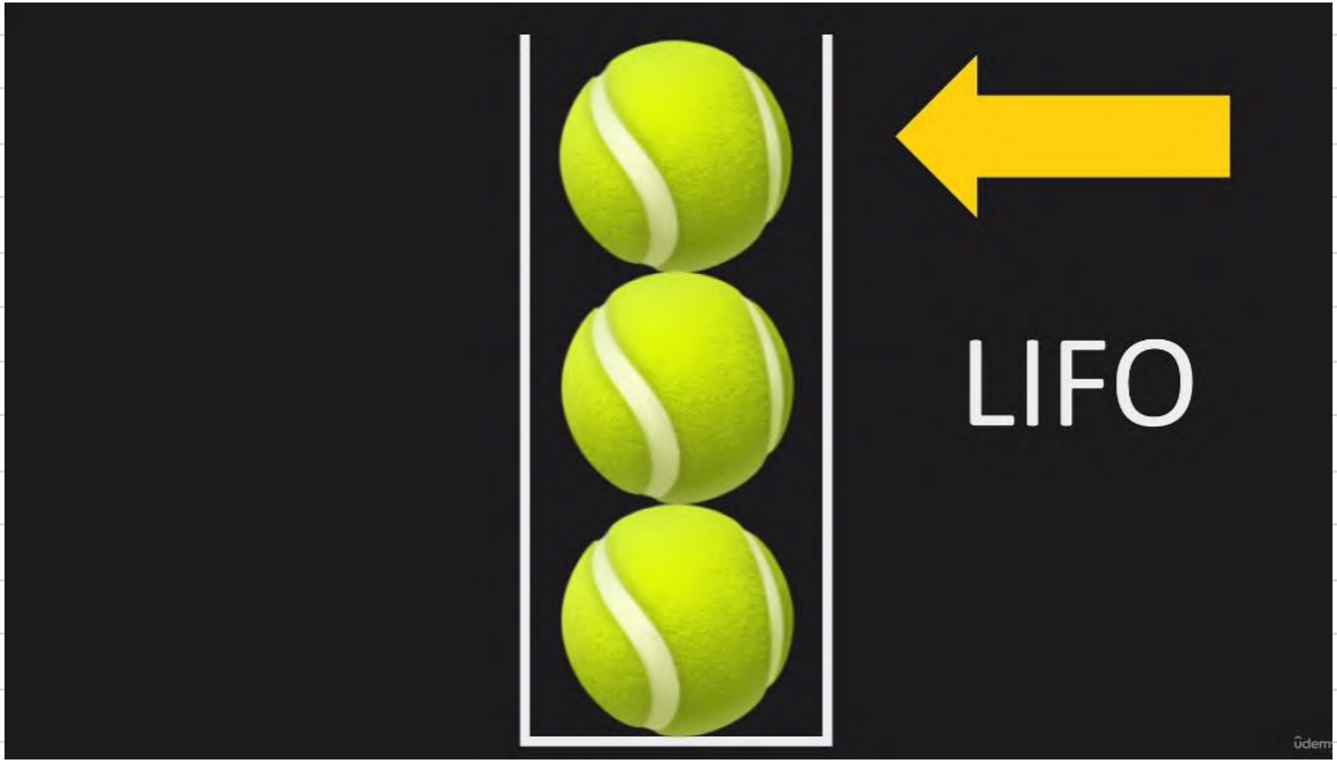
Stack



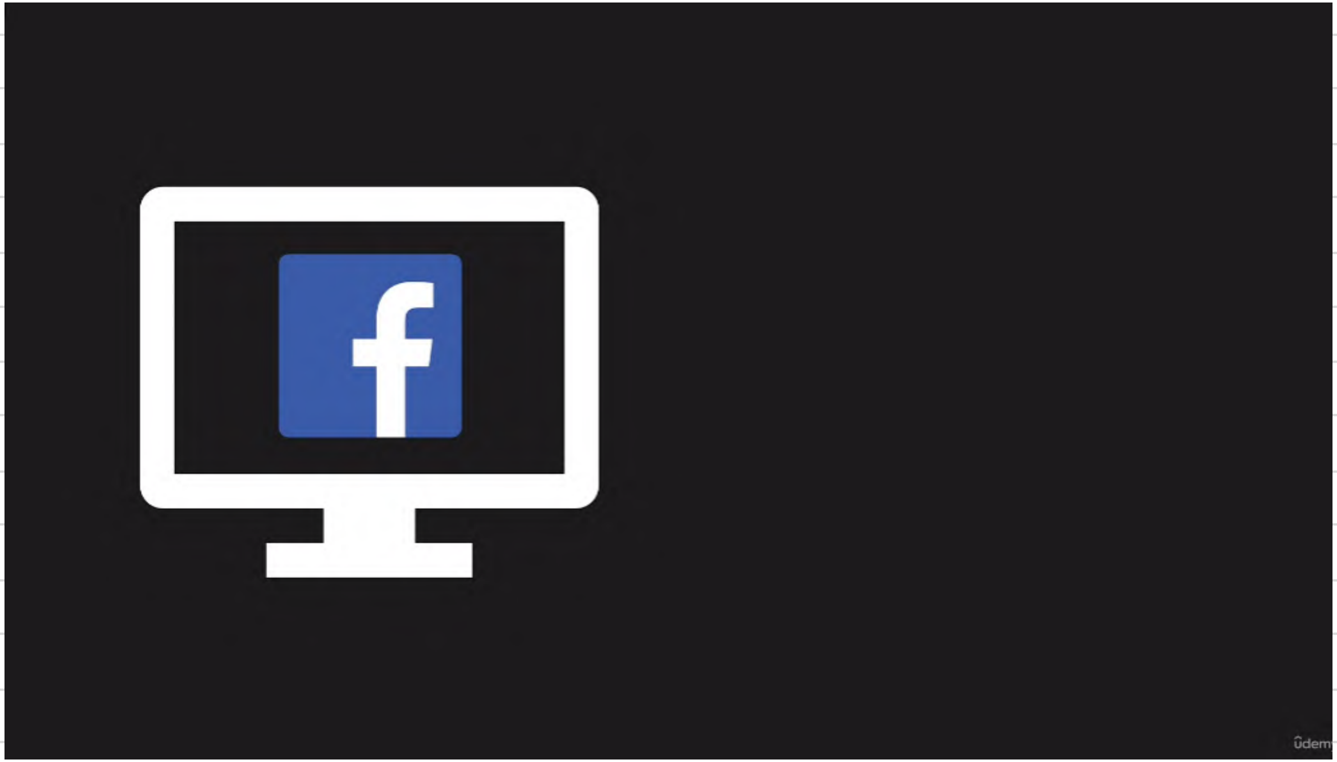
Stack



Stack



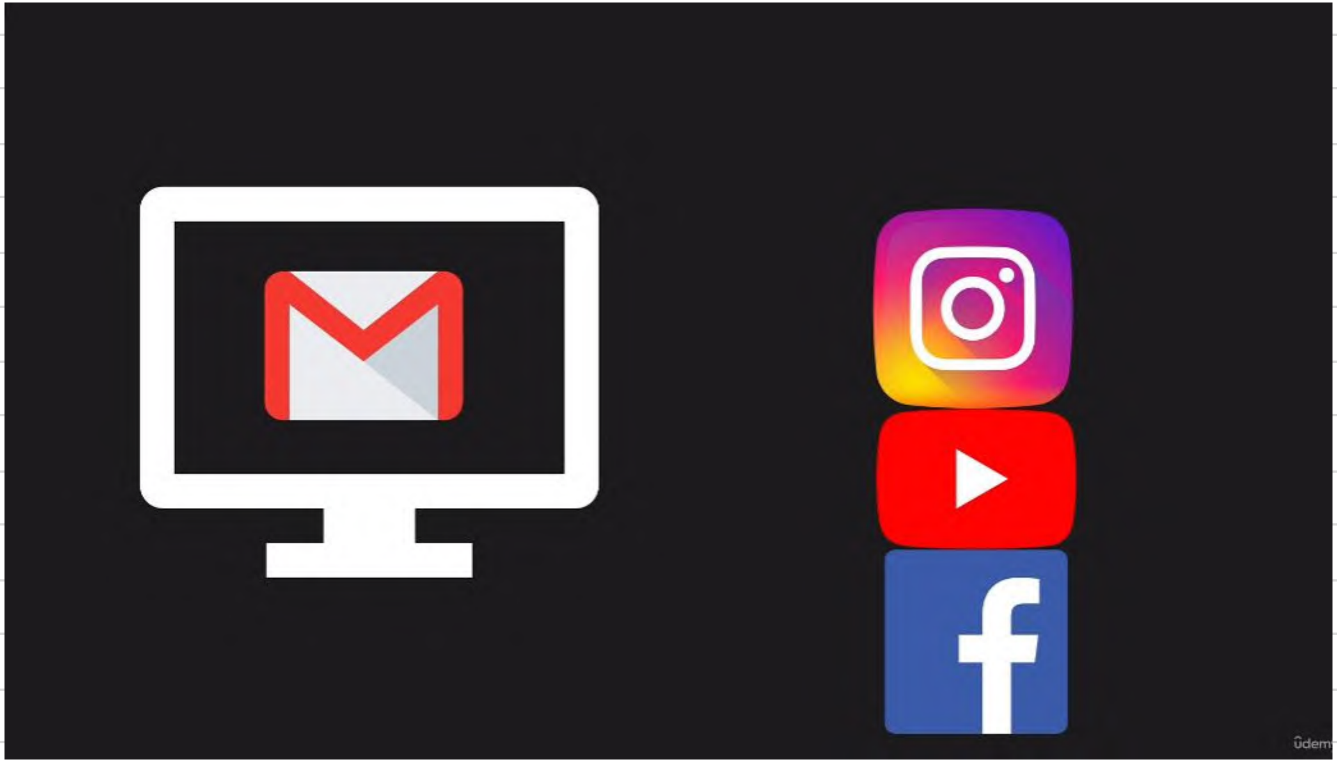
Stack



Stack

Stack

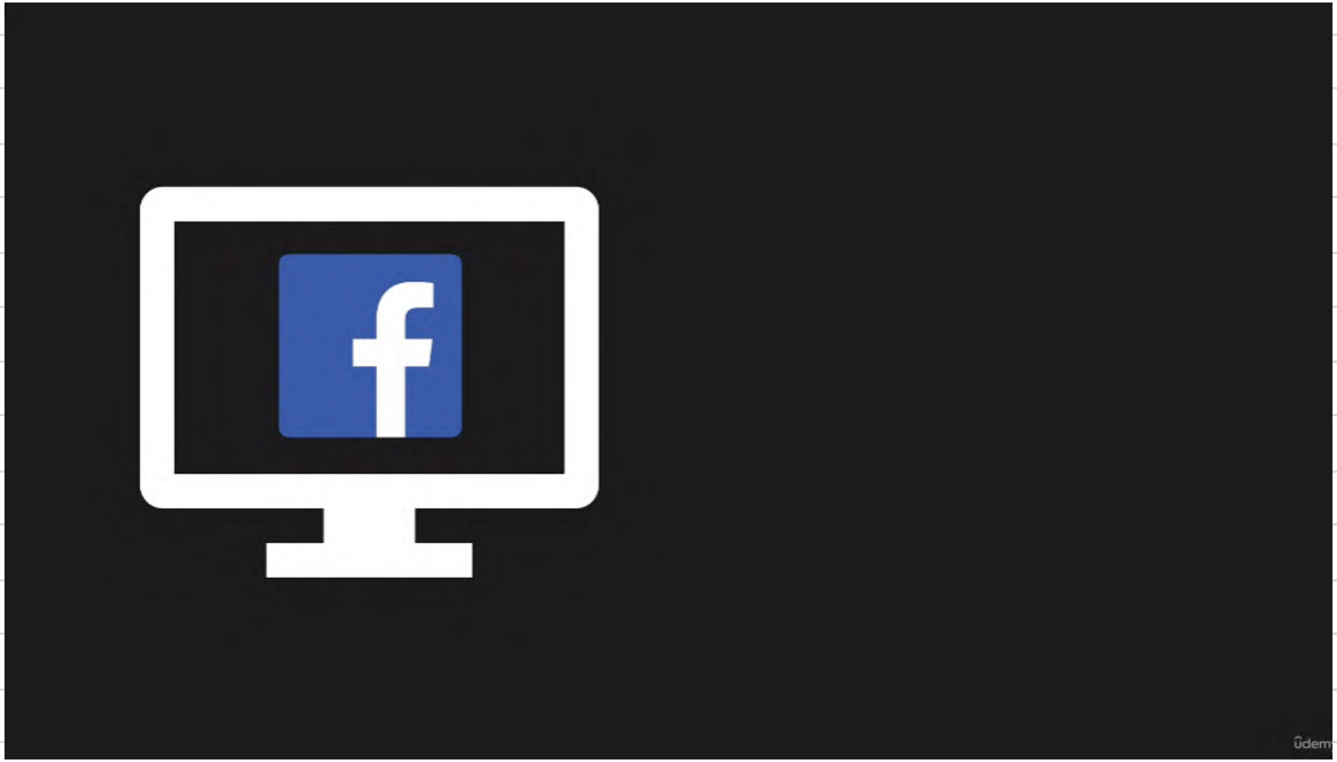
Stack



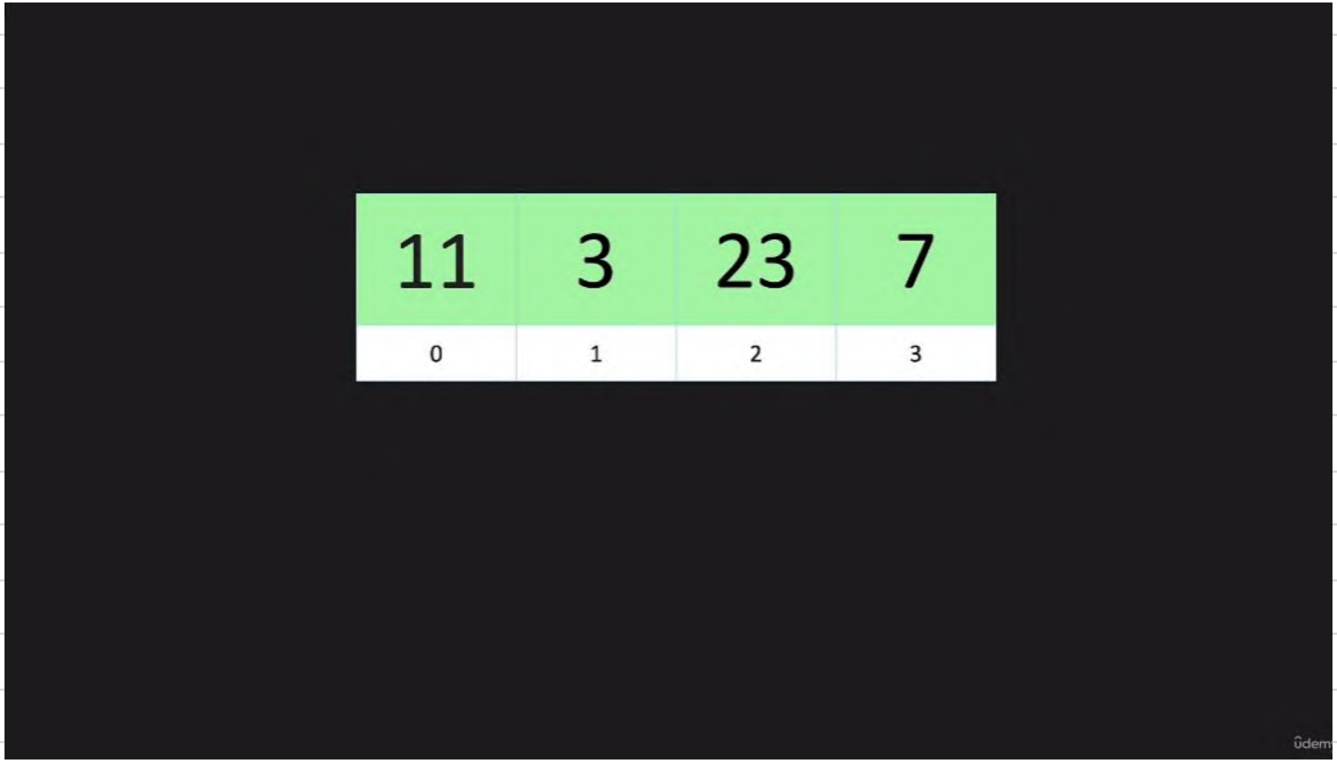
Stack

Stack

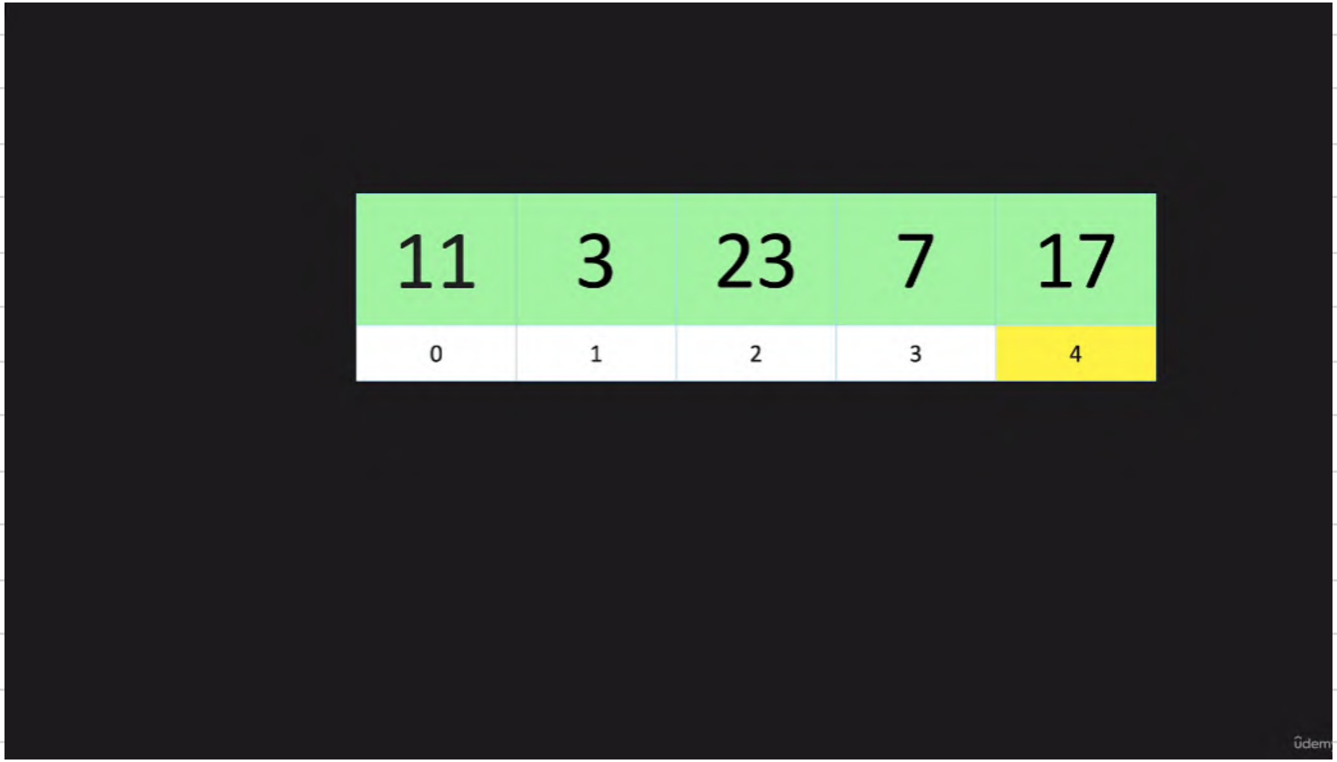
Stack



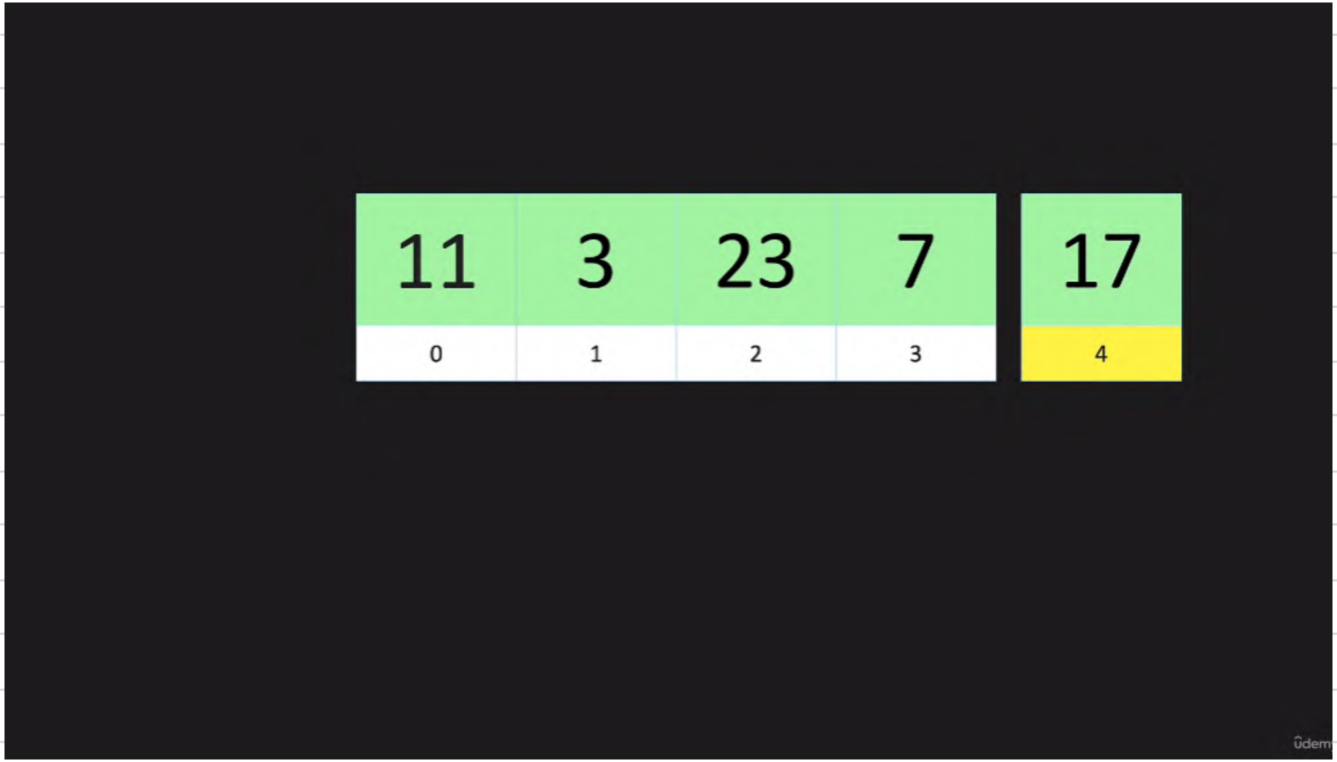
Stack



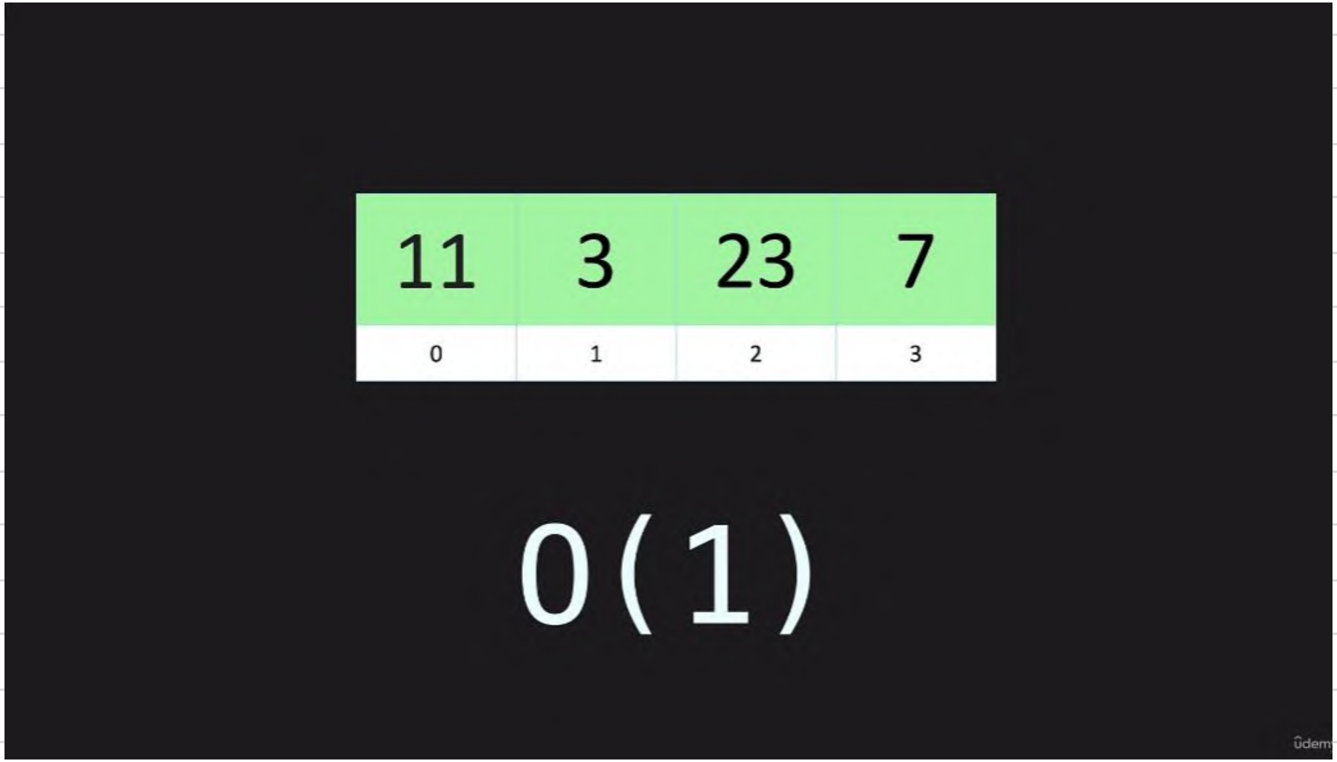
Stack



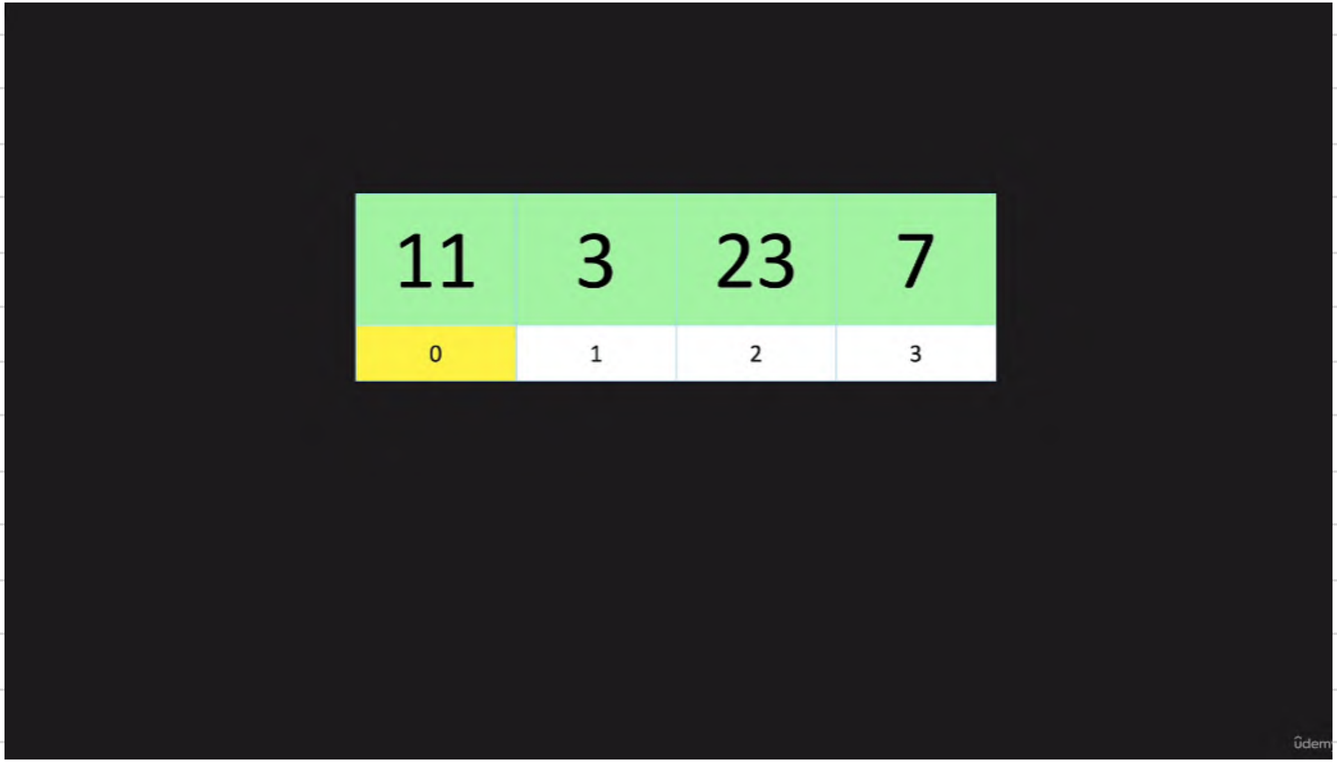
Stack



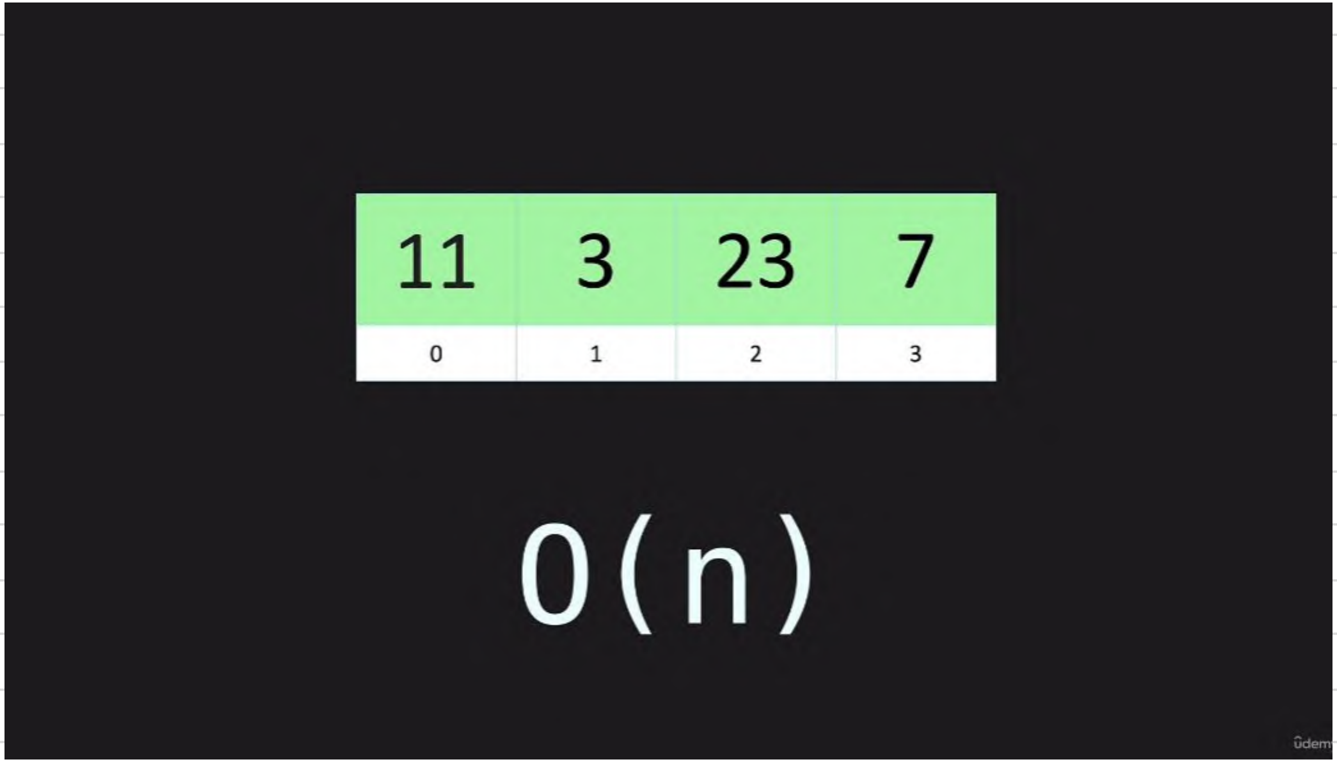
Stack



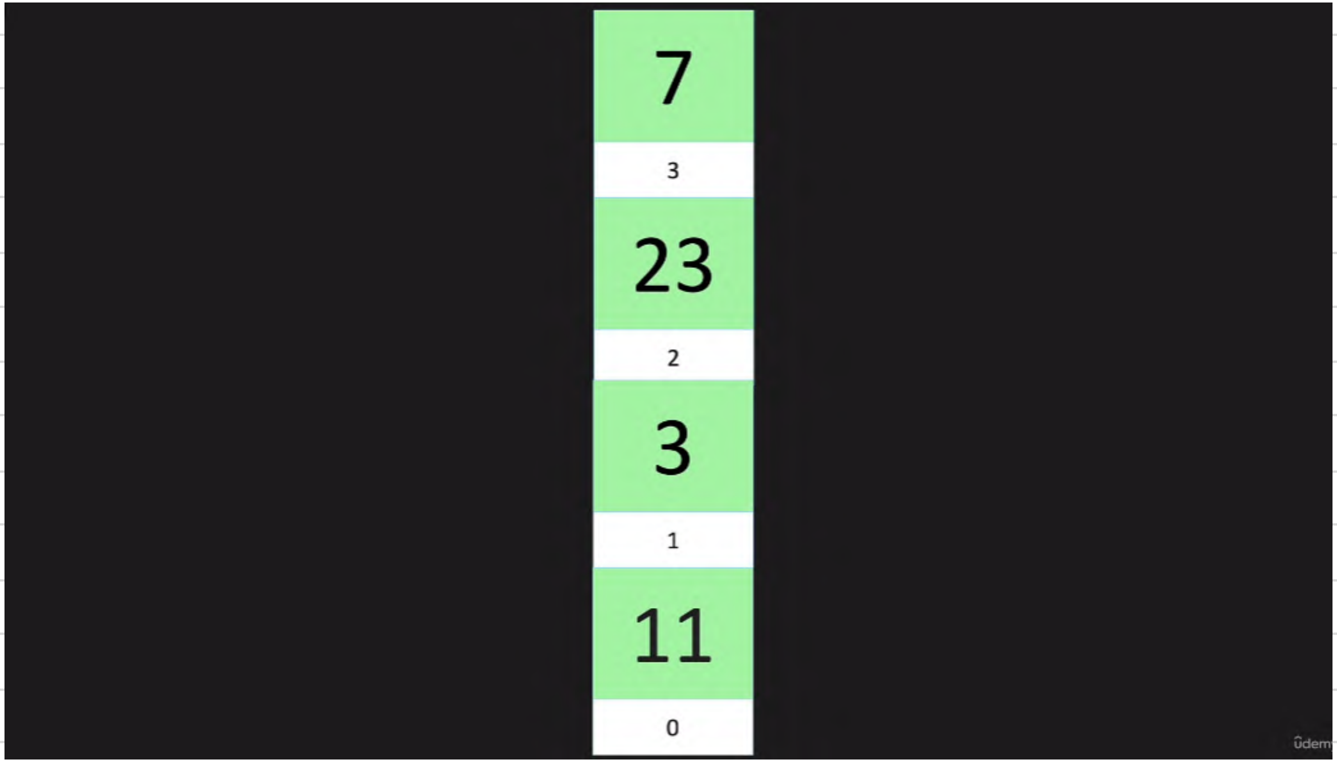
Stack



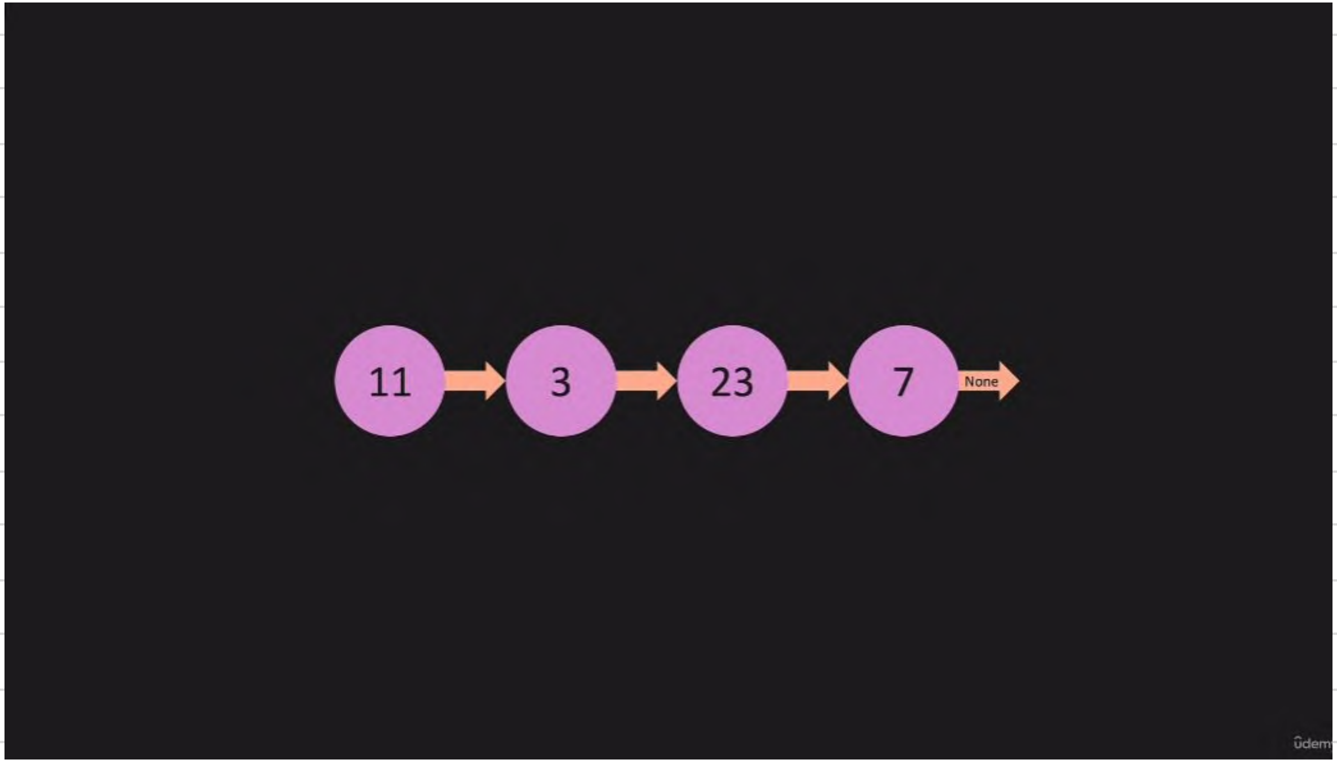
Stack



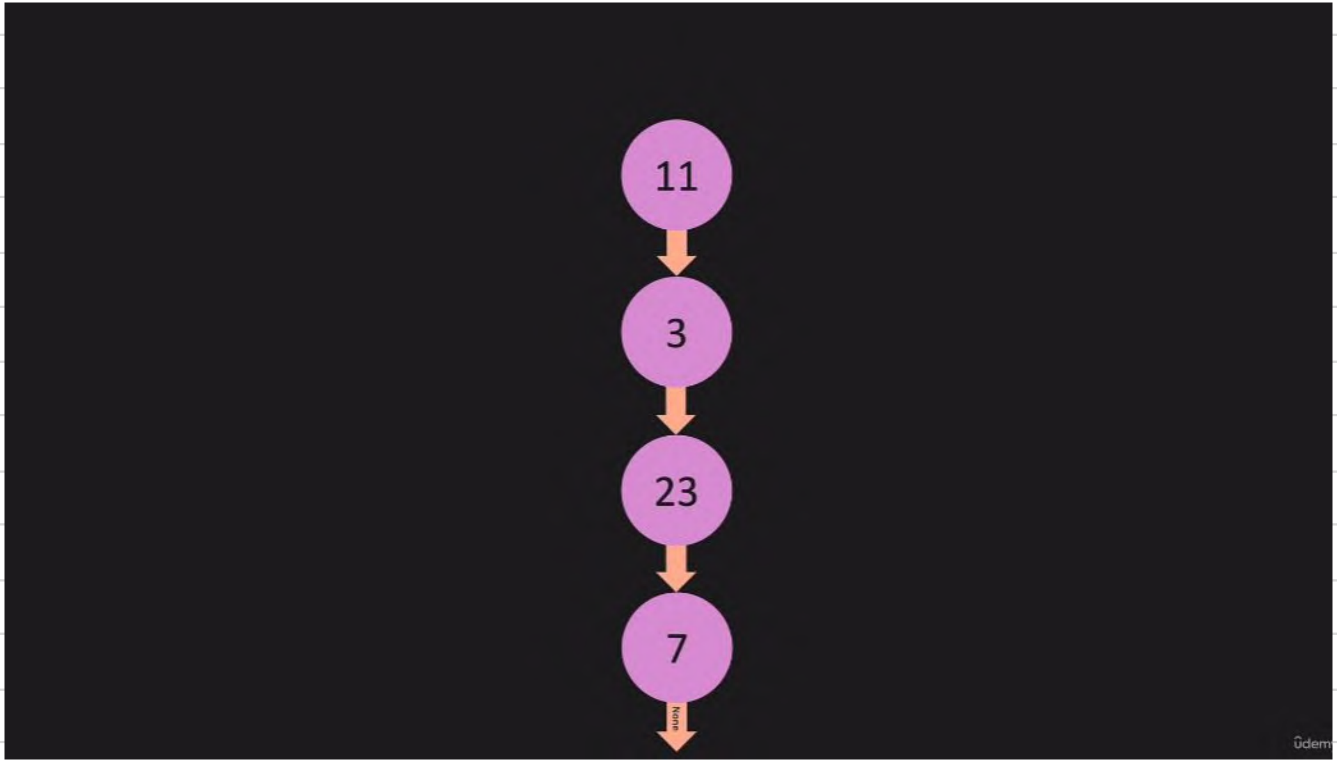
Stack



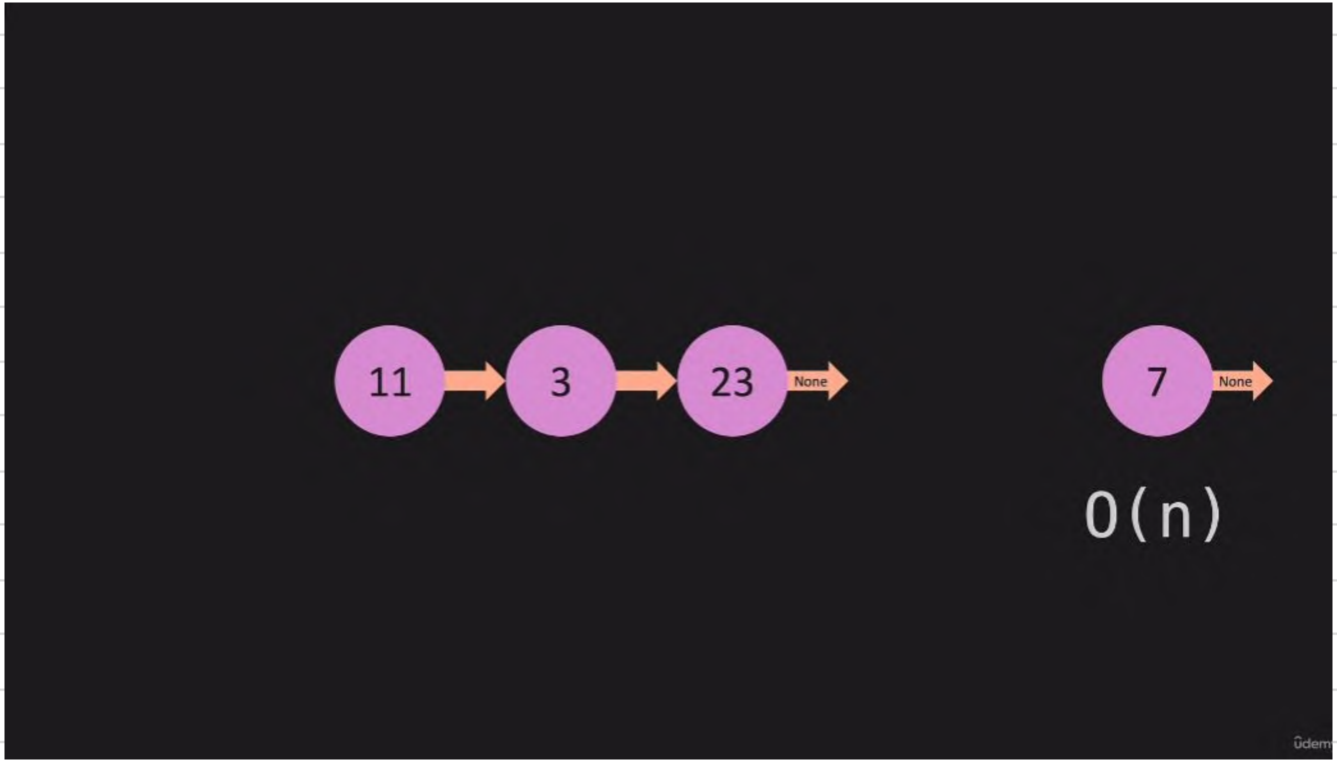
Stack



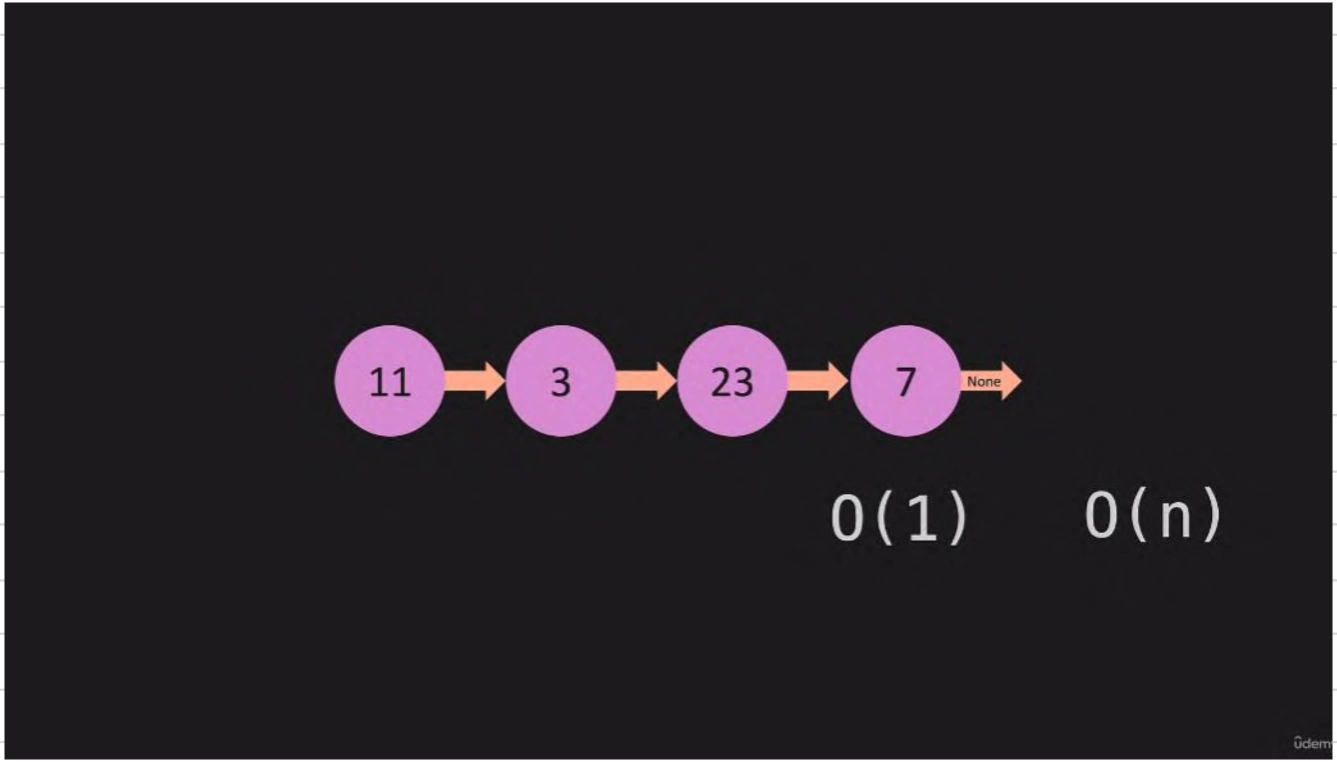
Stack



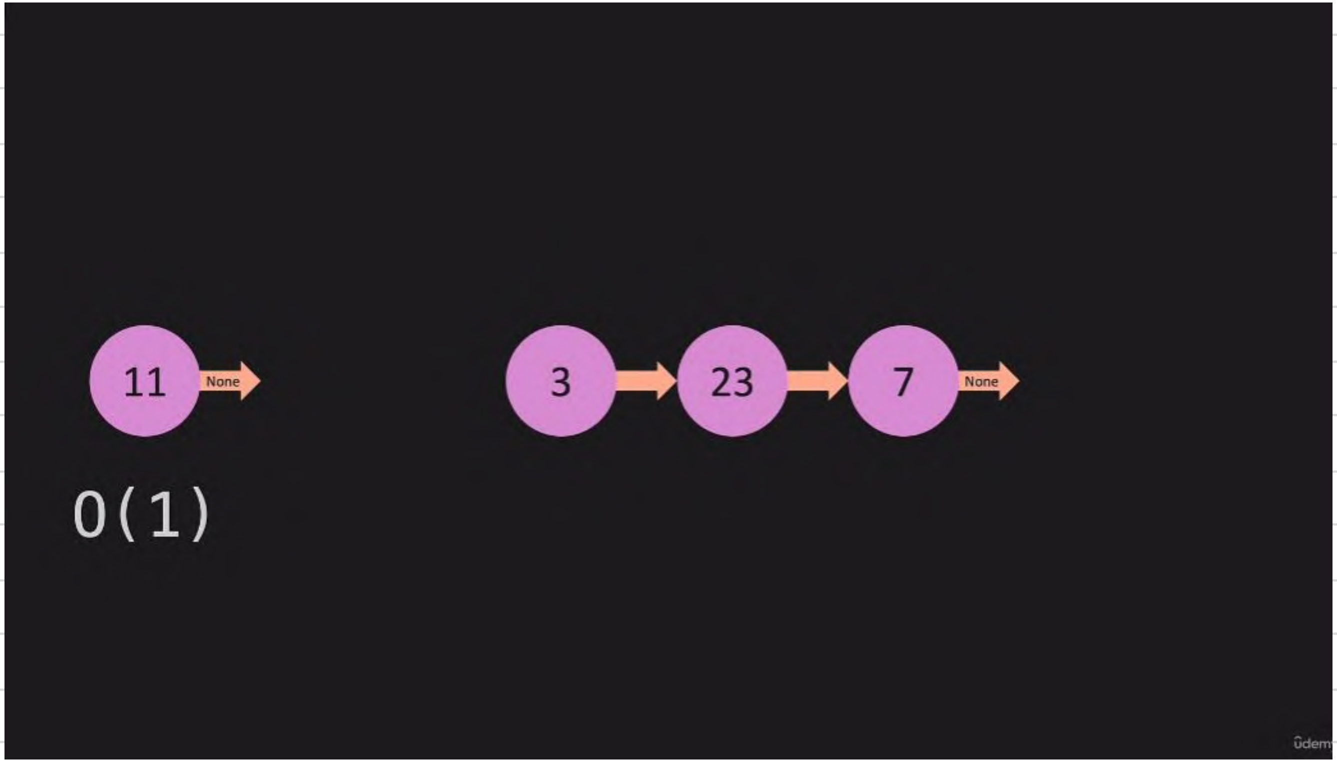
Stack



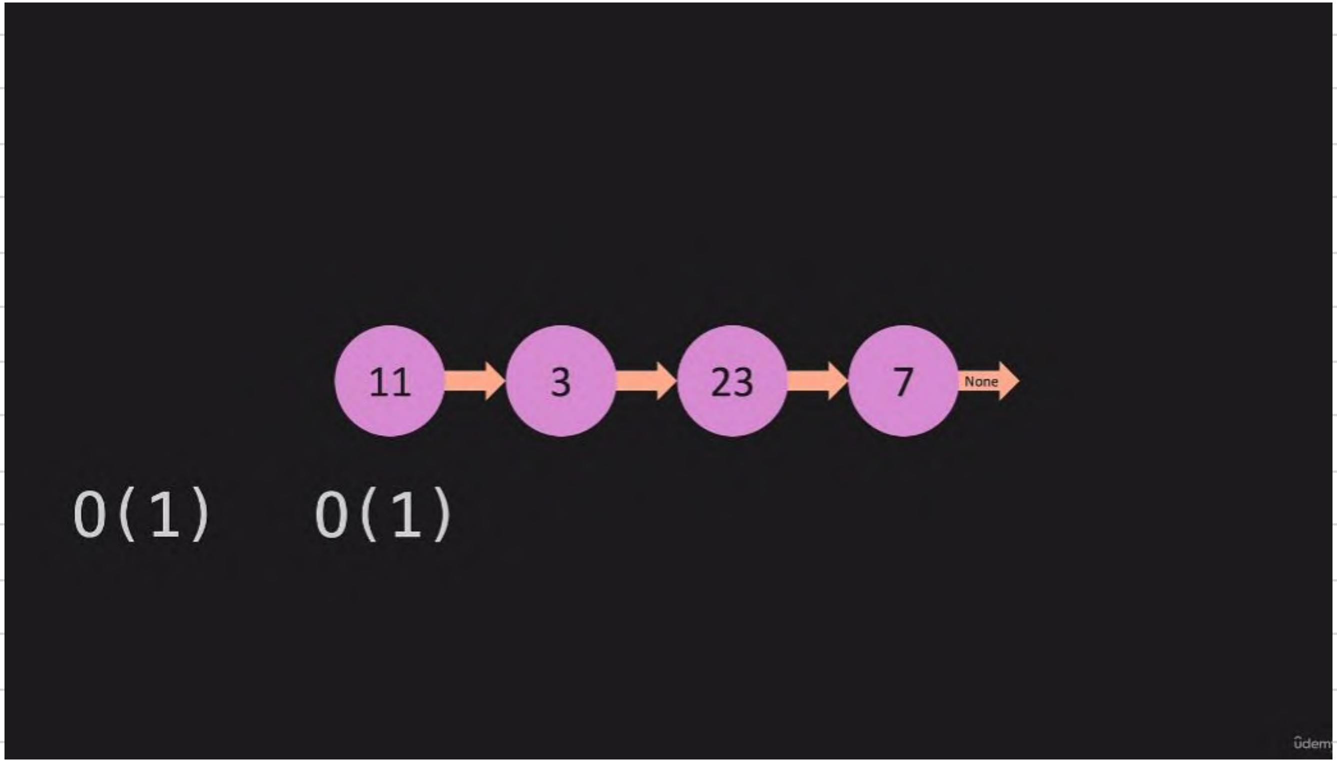
Stack



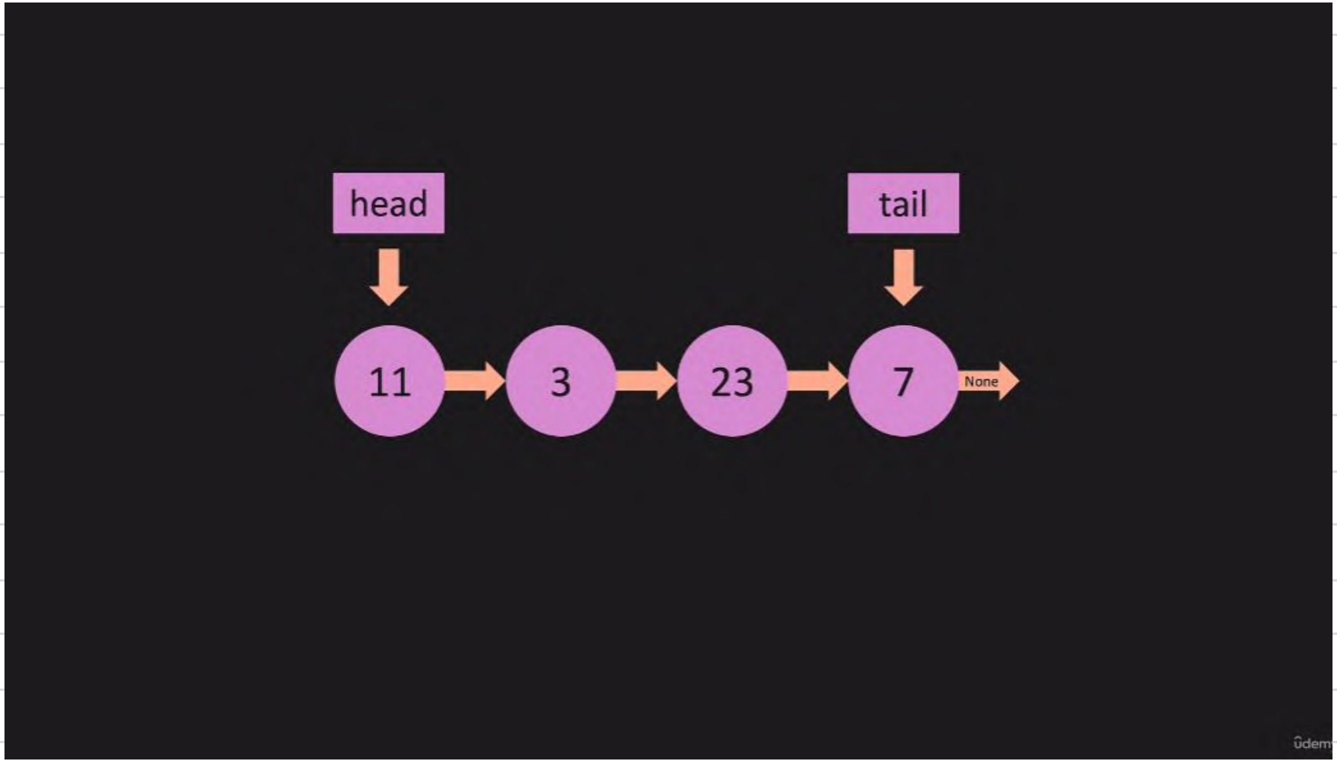
Stack



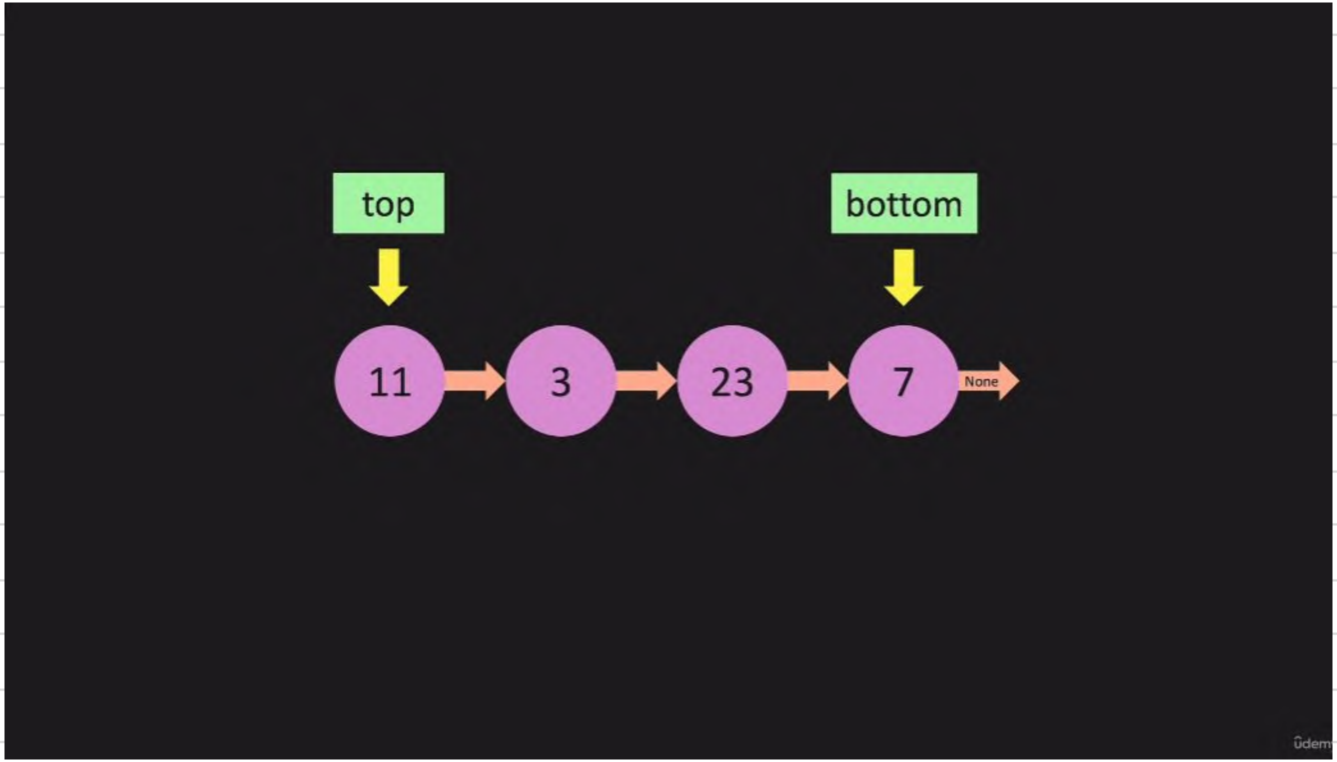
Stack



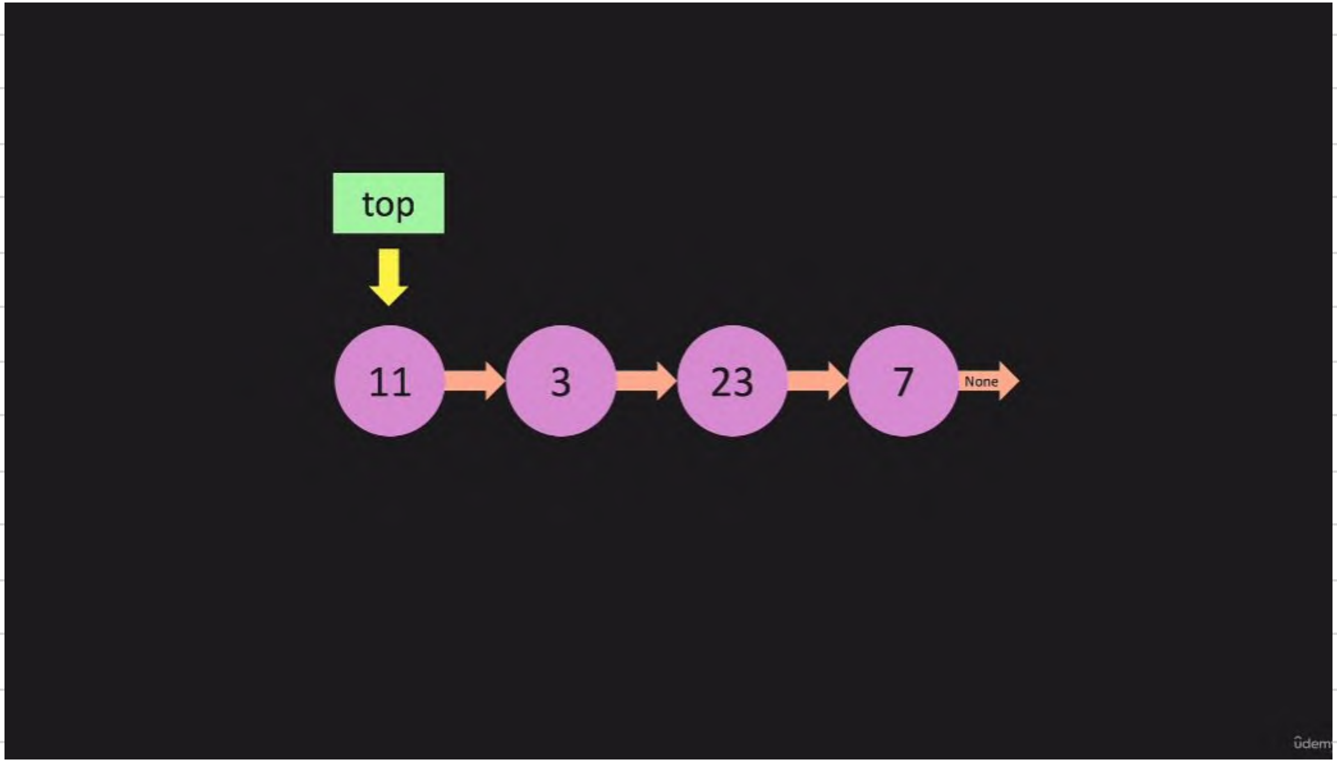
Stack



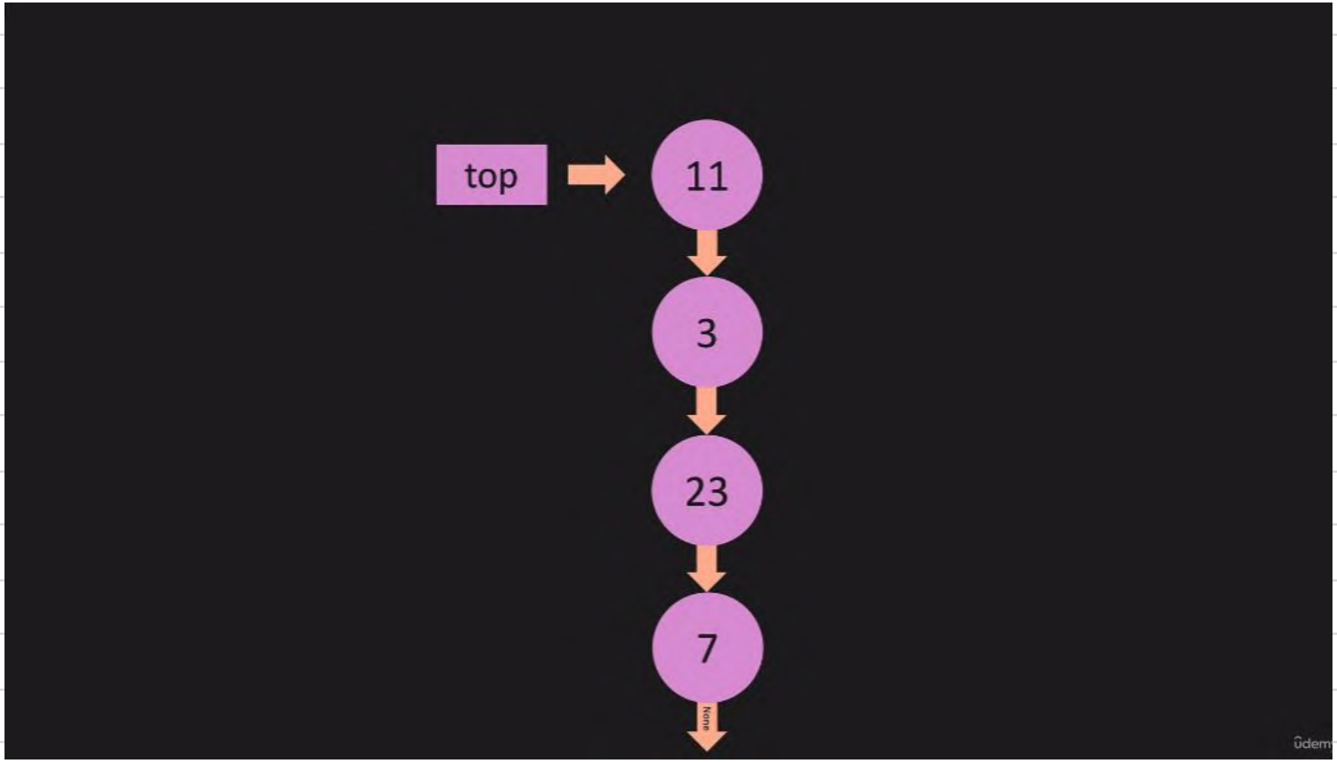
Stack



Stack



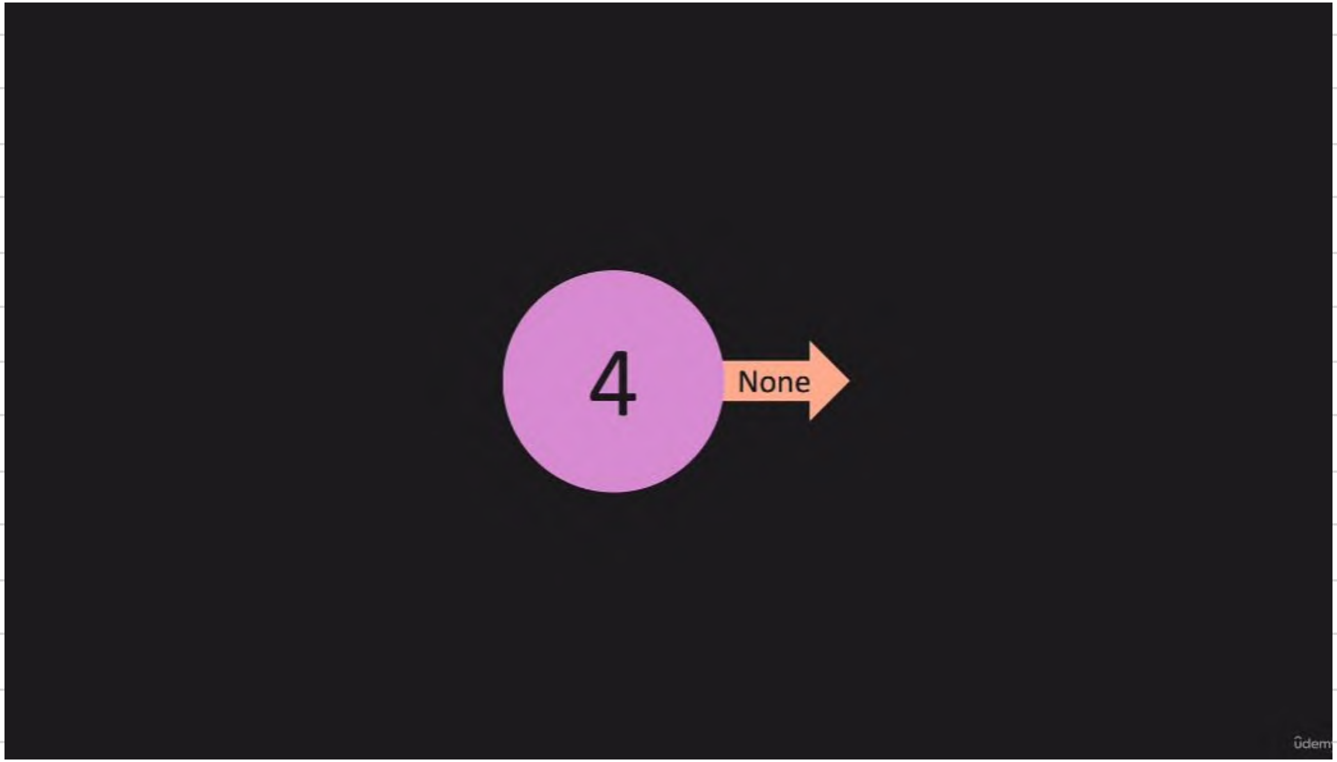
Stack



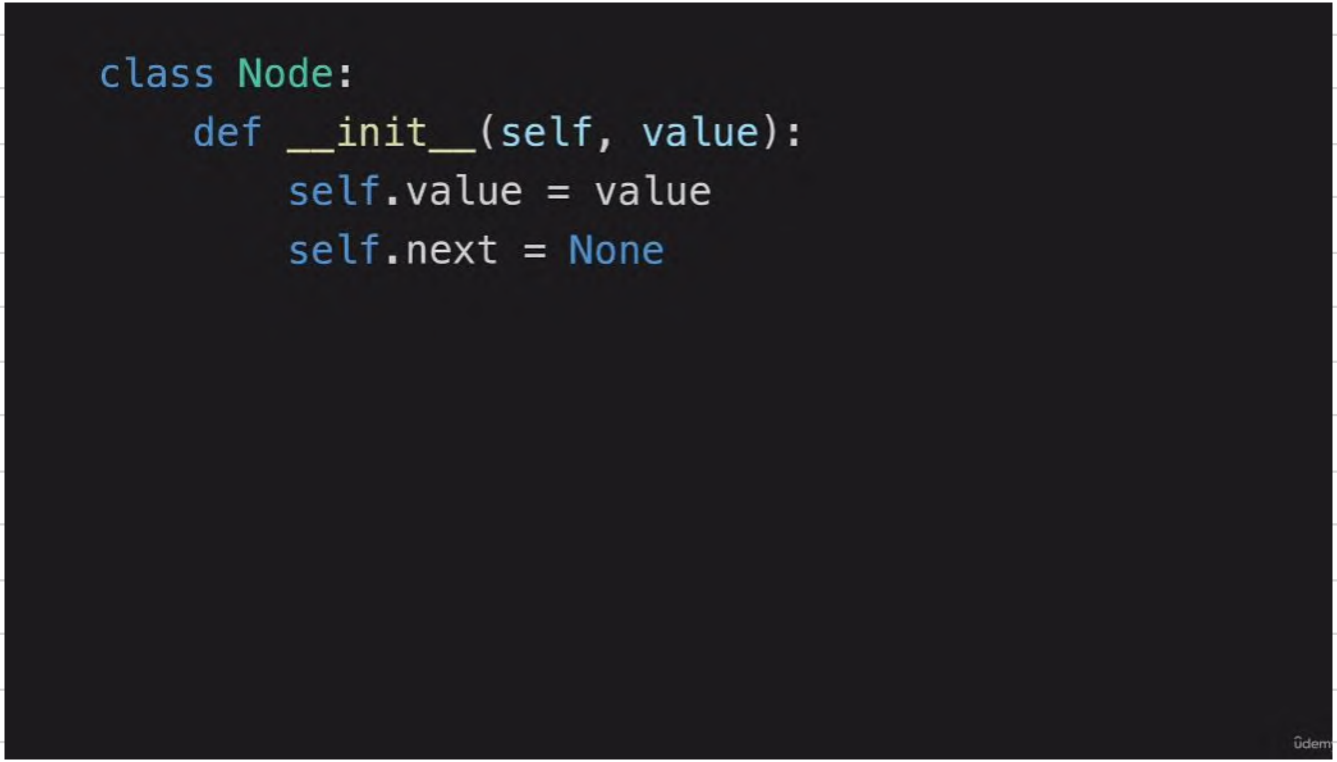
02

Stack Constructor

Stack : Constructor

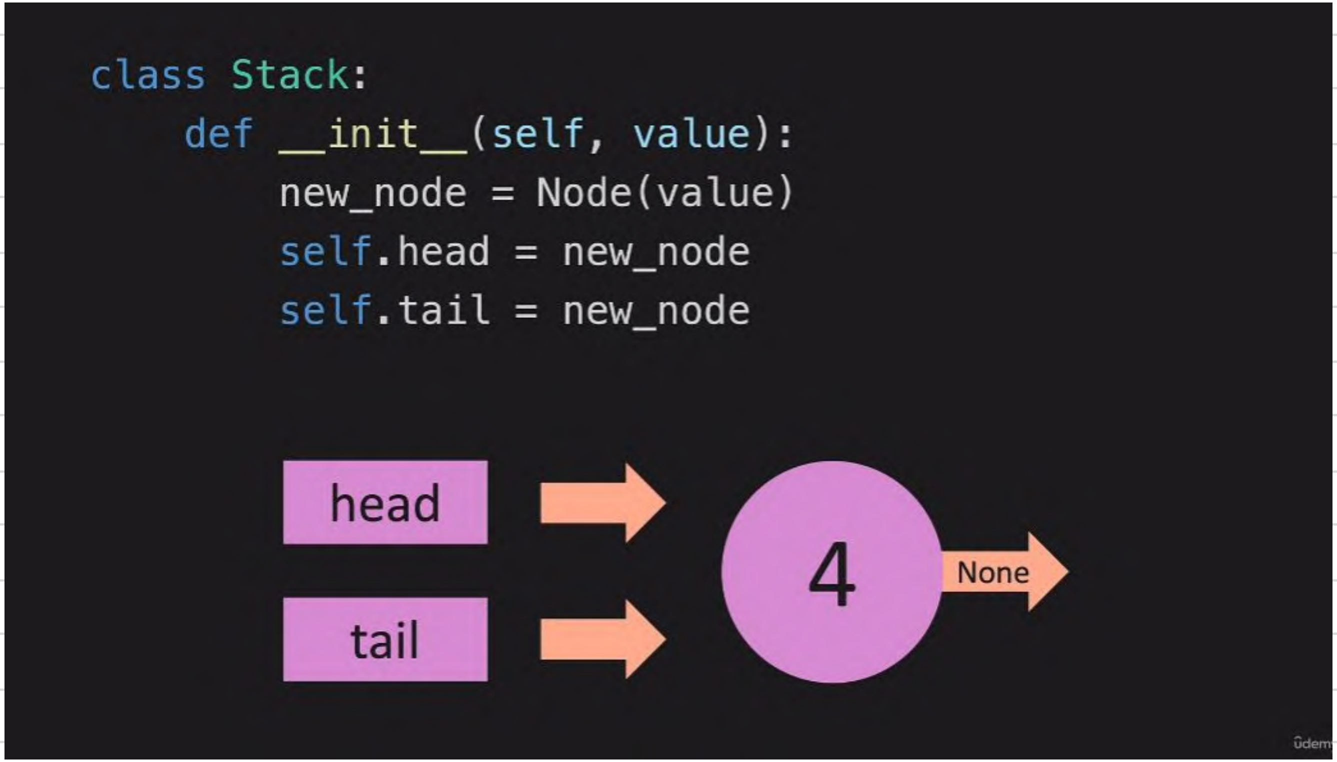


Stack : Constructor



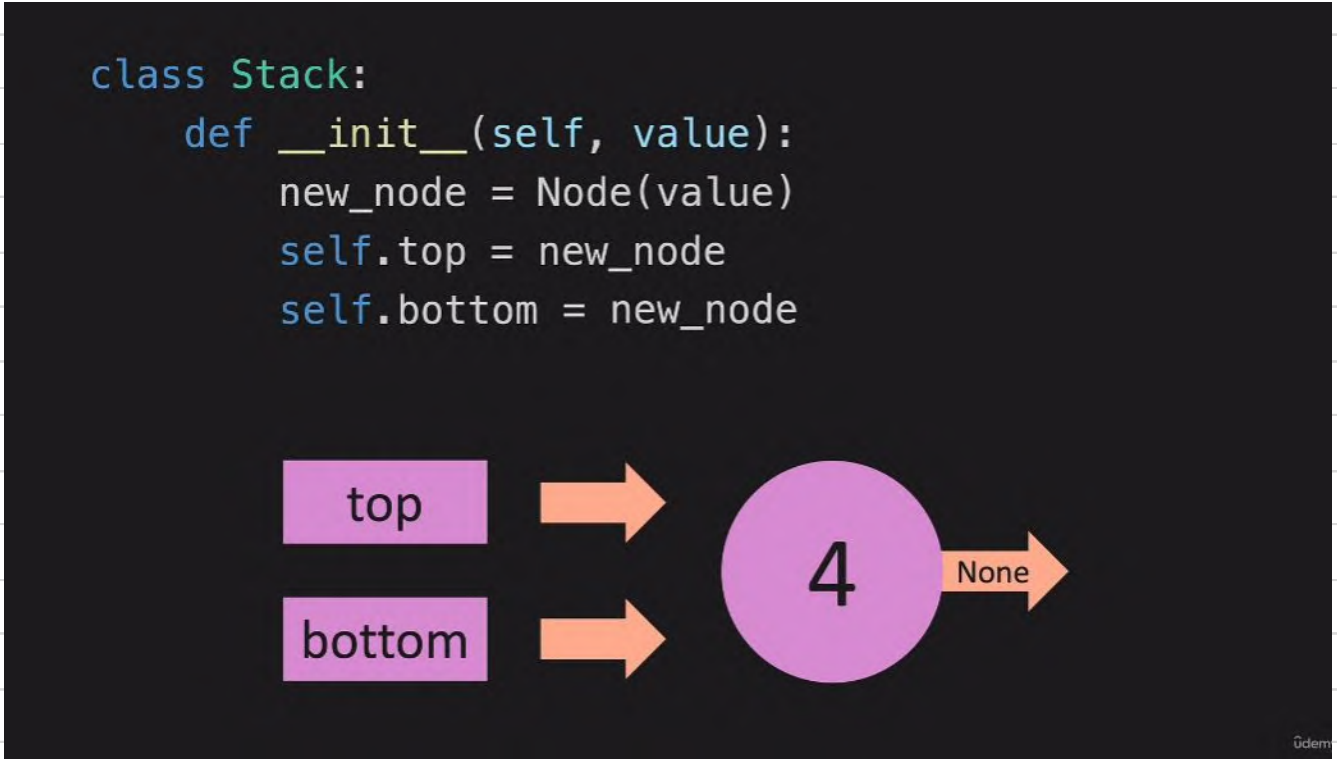
```python
class Node:
    def __init__(self, value):
        self.value = value
        self.next = None
```

Stack : Constructor



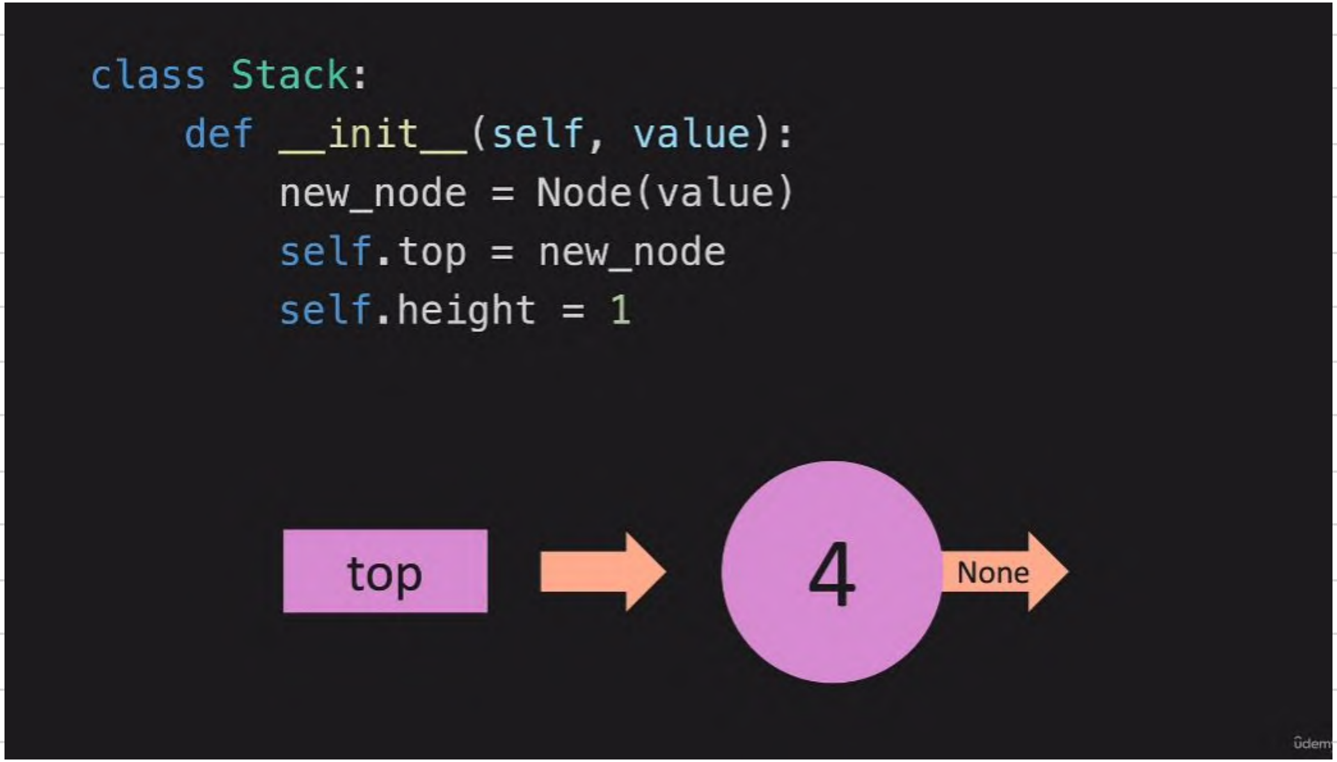
```
class Stack:
    def __init__(self, value):
        new_node = Node(value)
        self.head = new_node
        self.tail = new_node
```

Stack : Constructor



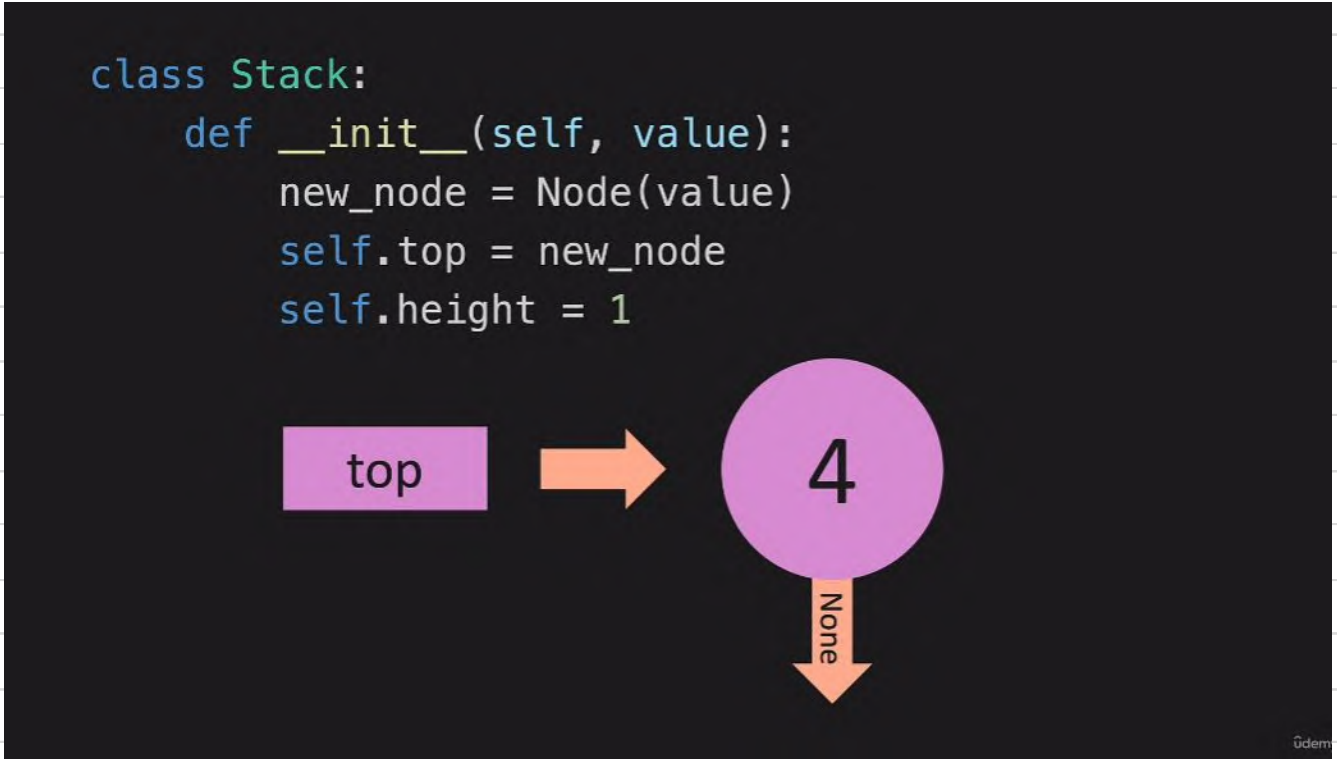
```
class Stack:
    def __init__(self, value):
        new_node = Node(value)
        self.top = new_node
        self.bottom = new_node
```

Stack : Constructor



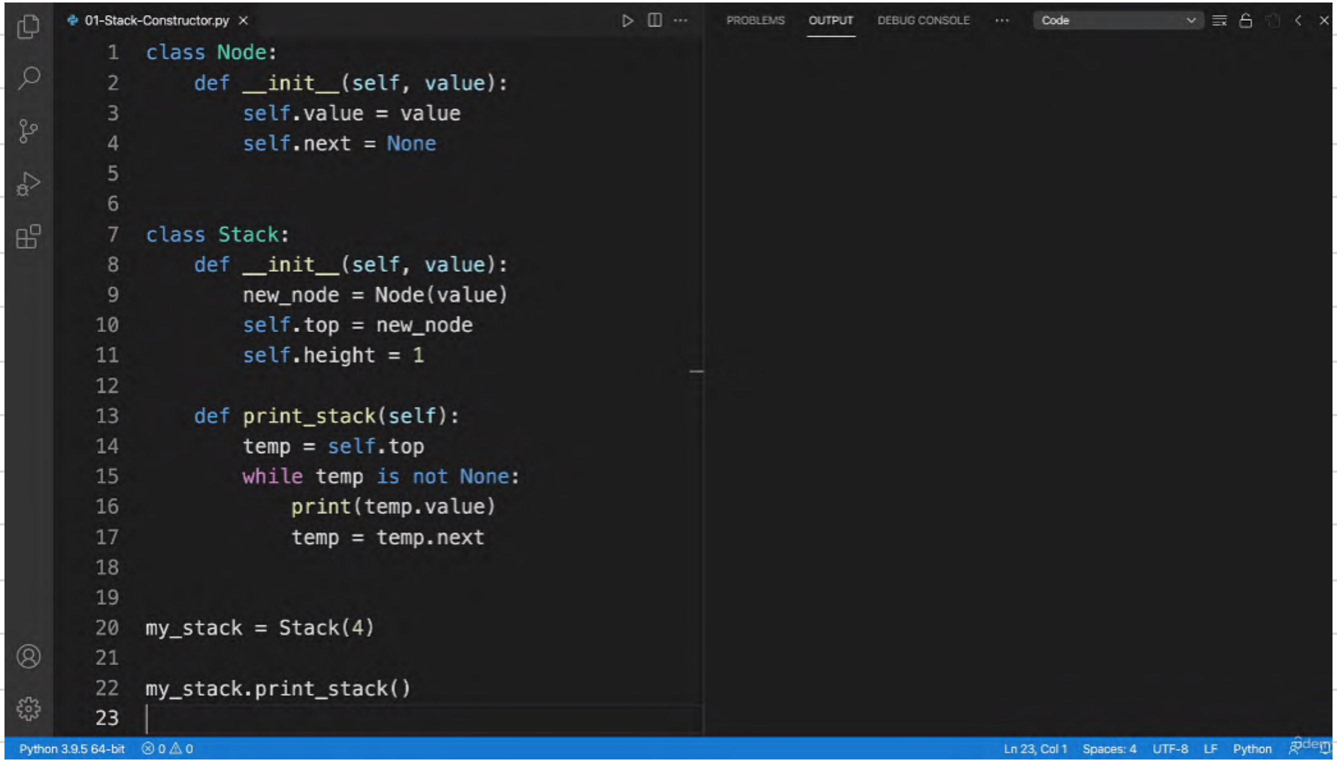
```
class Stack:
    def __init__(self, value):
        new_node = Node(value)
        self.top = new_node
        self.height = 1
```

Stack : Constructor



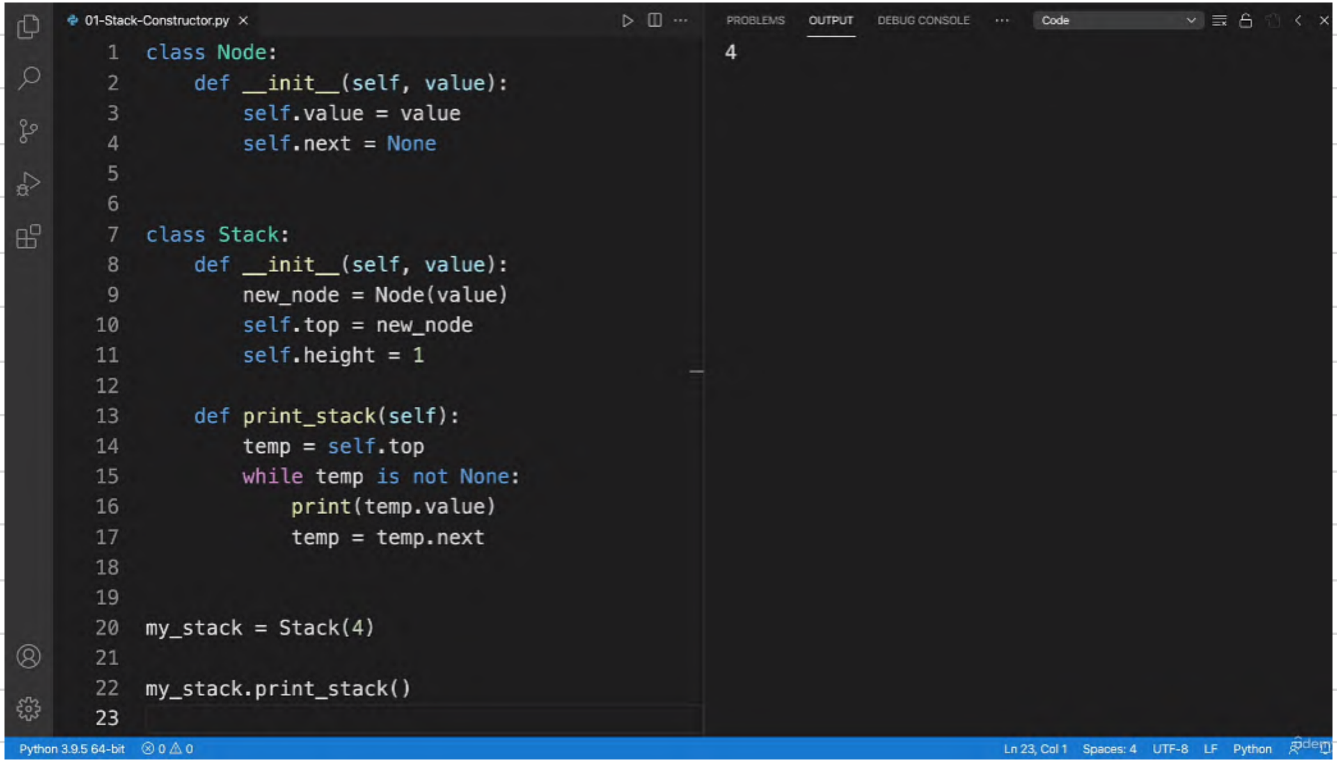
```
class Stack:
    def __init__(self, value):
        new_node = Node(value)
        self.top = new_node
        self.height = 1
```

Stack : Constructor


01-Stack-Constructor.py

```python
class Node:
    def __init__(self, value):
        self.value = value
        self.next = None


class Stack:
    def __init__(self, value):
        new_node = Node(value)
        self.top = new_node
        self.height = 1

    def print_stack(self):
        temp = self.top
        while temp is not None:
            print(temp.value)
            temp = temp.next


my_stack = Stack(4)

my_stack.print_stack()

```

Stack : Constructor



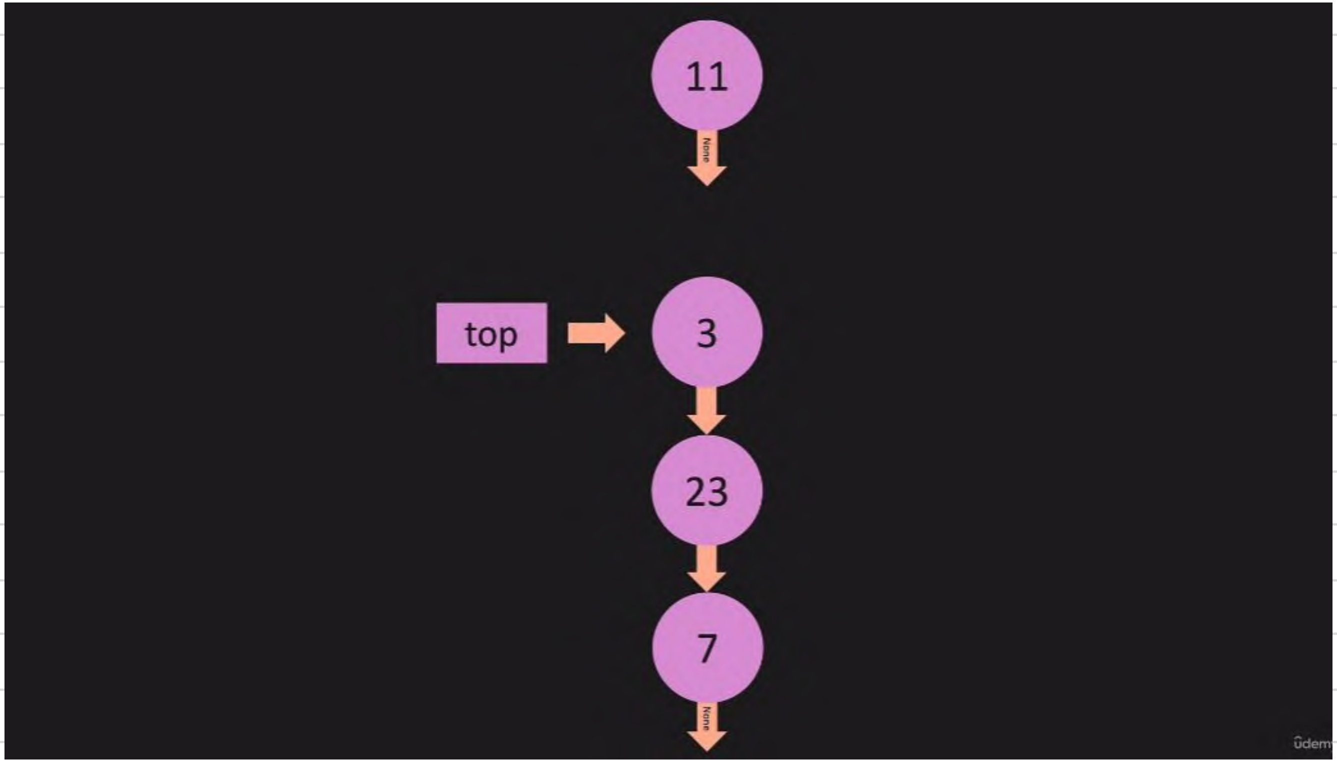
```python
class Node:
    def __init__(self, value):
        self.value = value
        self.next = None


class Stack:
    def __init__(self, value):
        new_node = Node(value)
        self.top = new_node
        self.height = 1

    def print_stack(self):
        temp = self.top
        while temp is not None:
            print(temp.value)
            temp = temp.next


my_stack = Stack(4)

my_stack.print_stack()

```

```
4
```

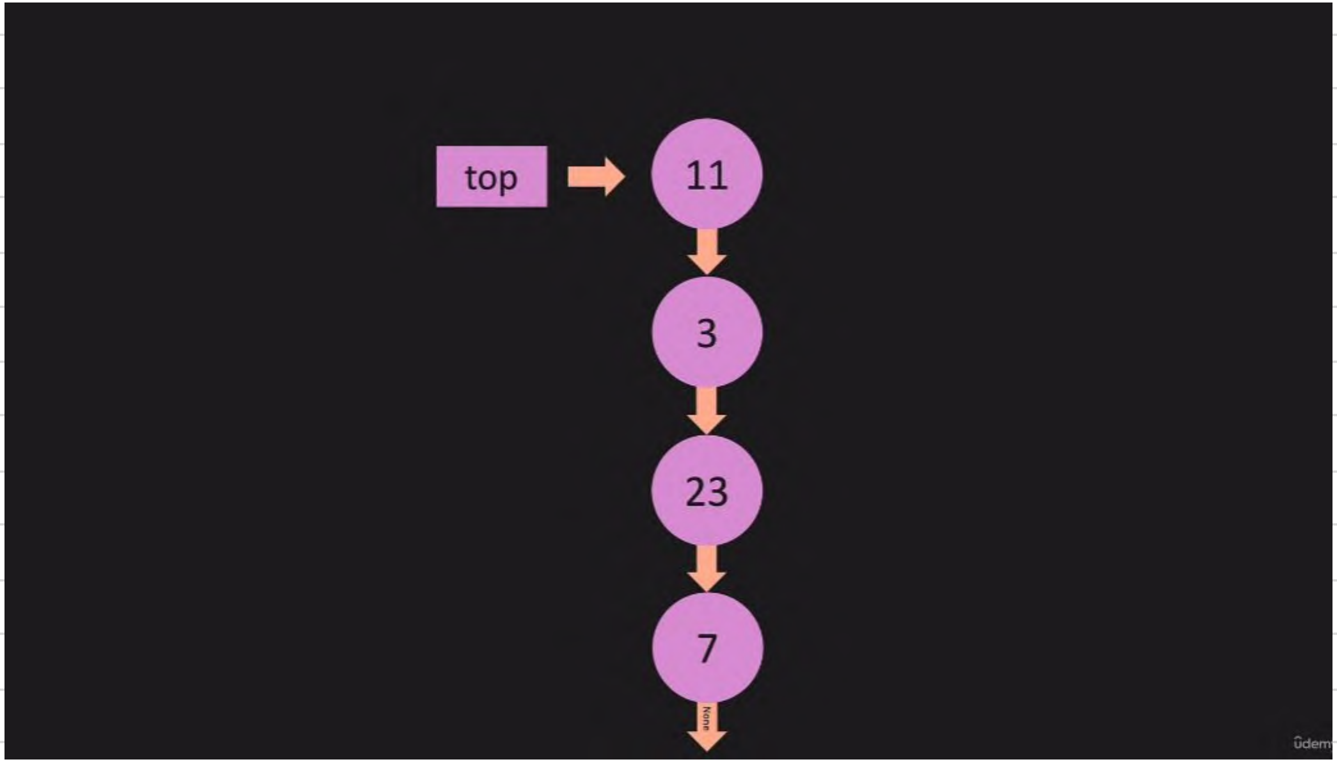
03

Stack Push

Stack : Push

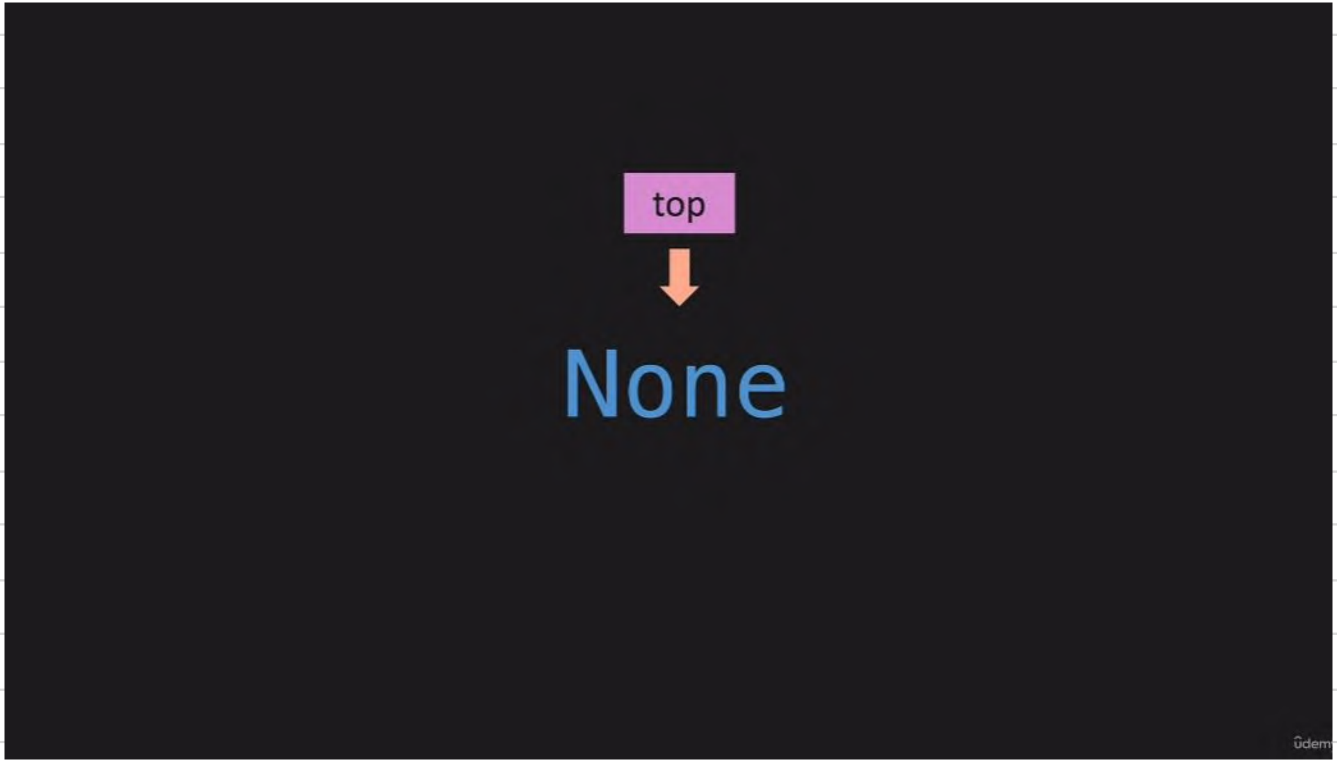


Stack : Push

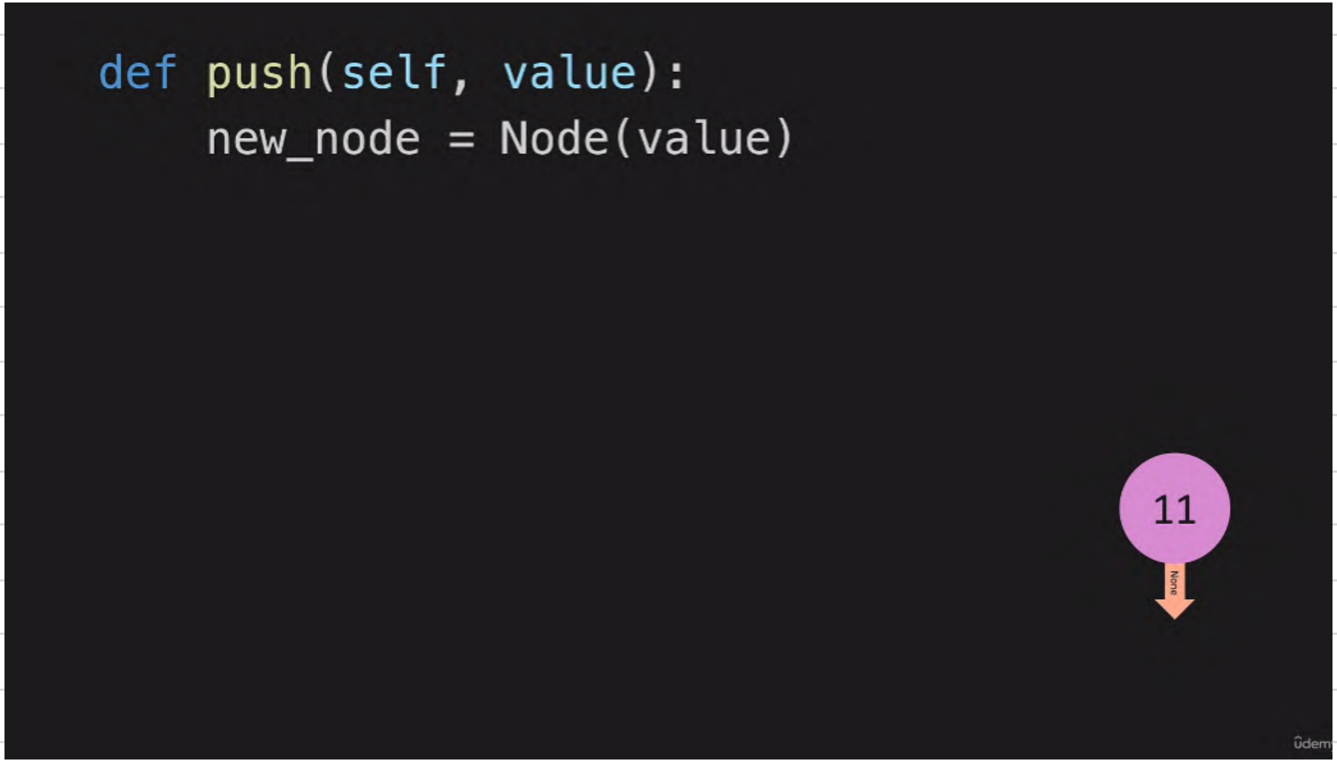


Edge Condition #1

Stack : Push

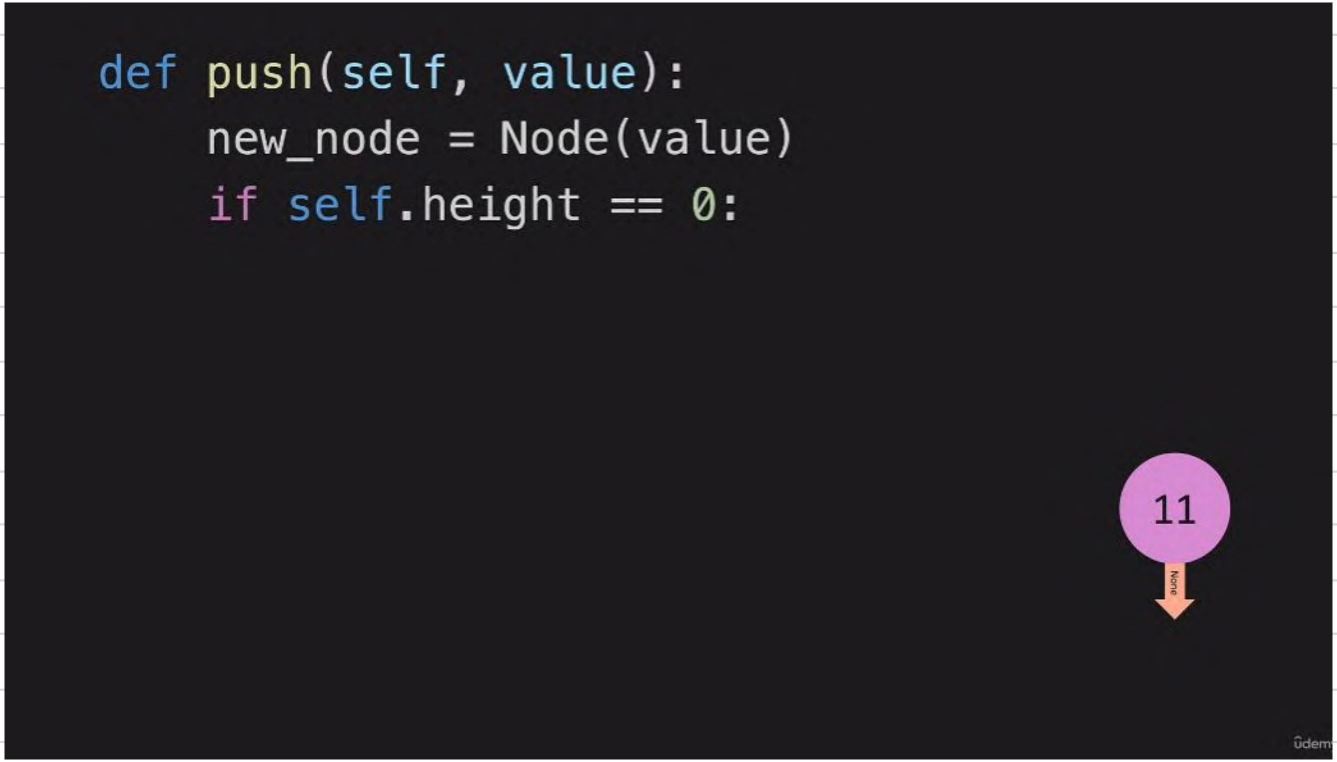


Stack : Push Code



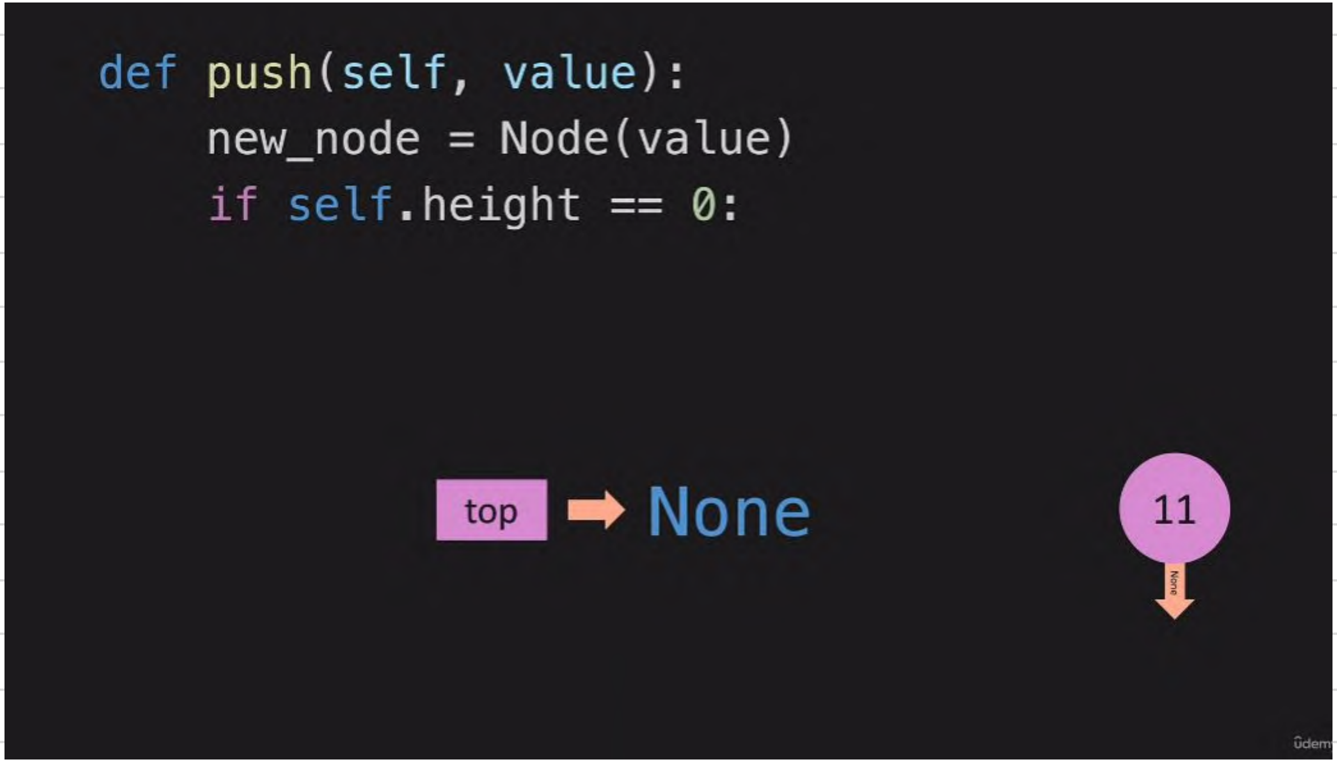
```
def push(self, value):
    new_node = Node(value)
```

Stack : Push Code



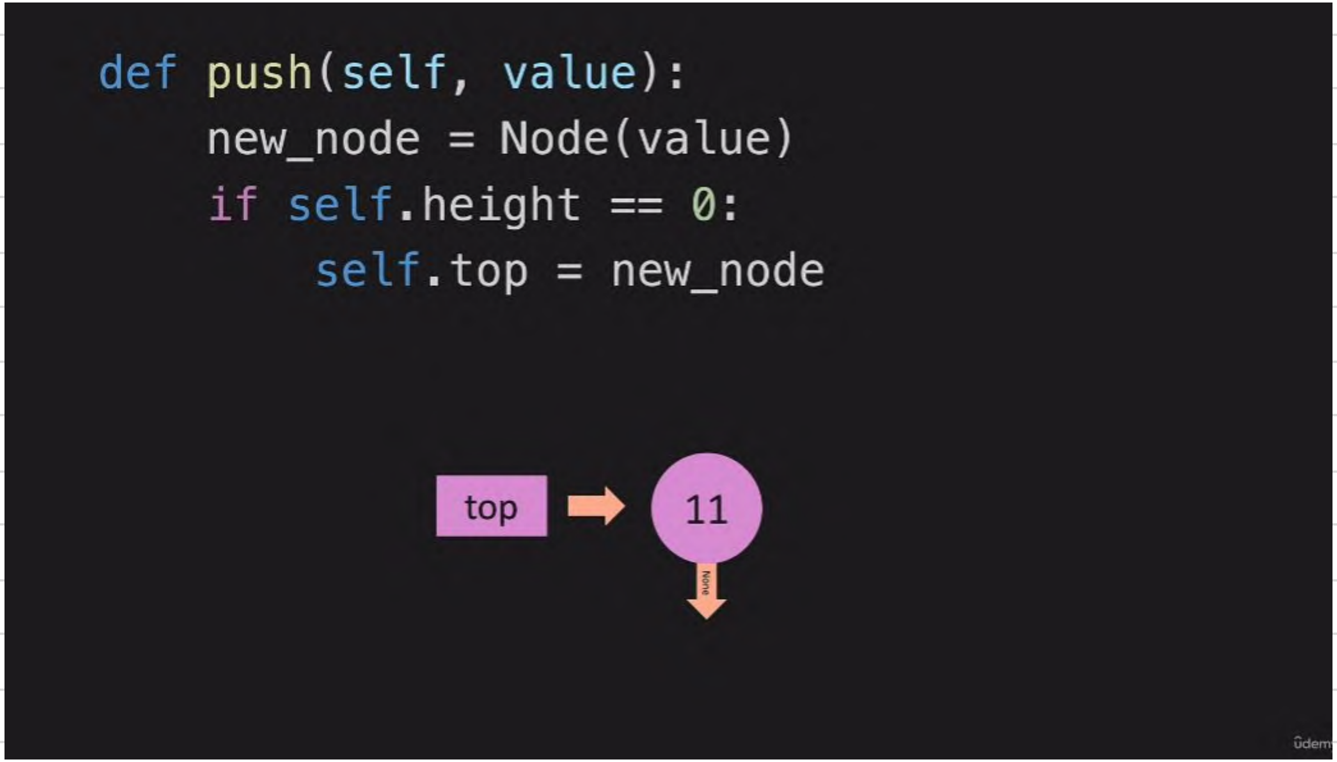
```
def push(self, value):
    new_node = Node(value)
    if self.height == 0:
```

Stack : Push Code



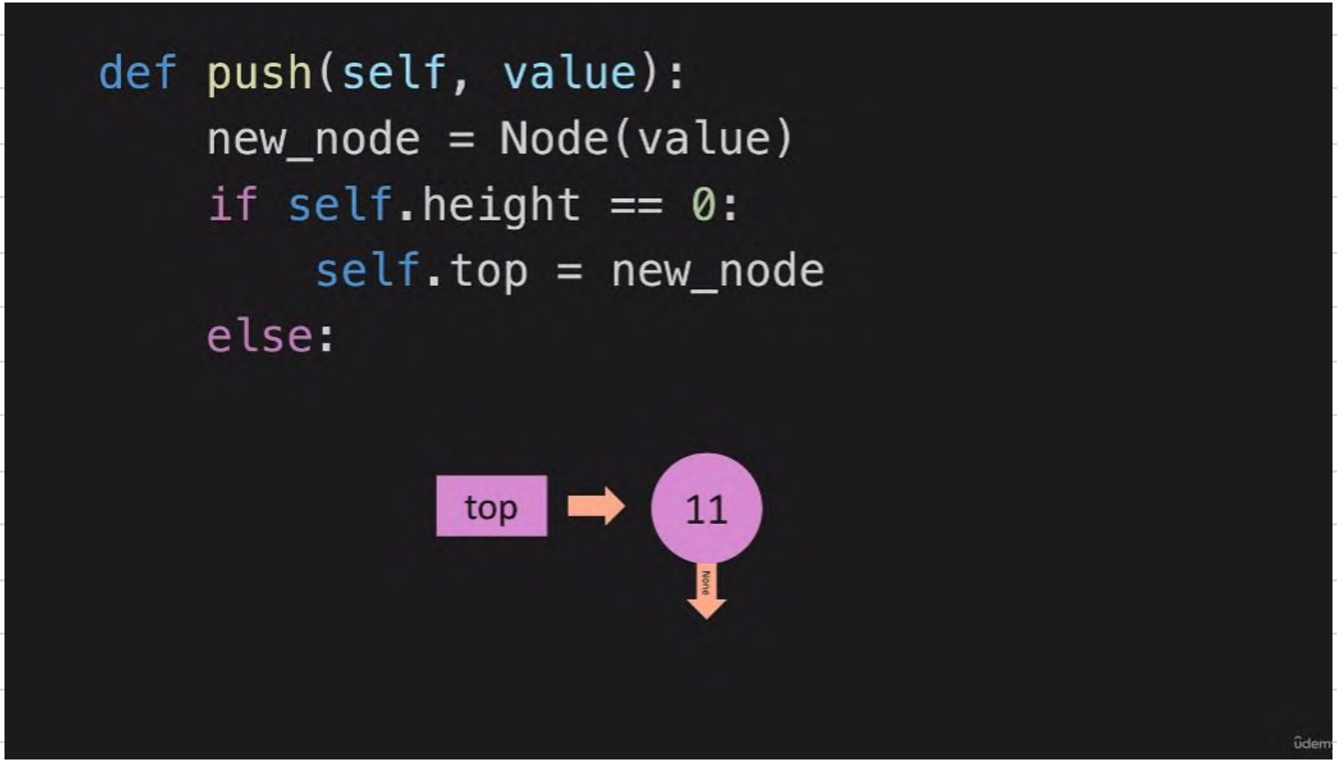
```
def push(self, value):
    new_node = Node(value)
    if self.height == 0:
```

Stack : Push Code



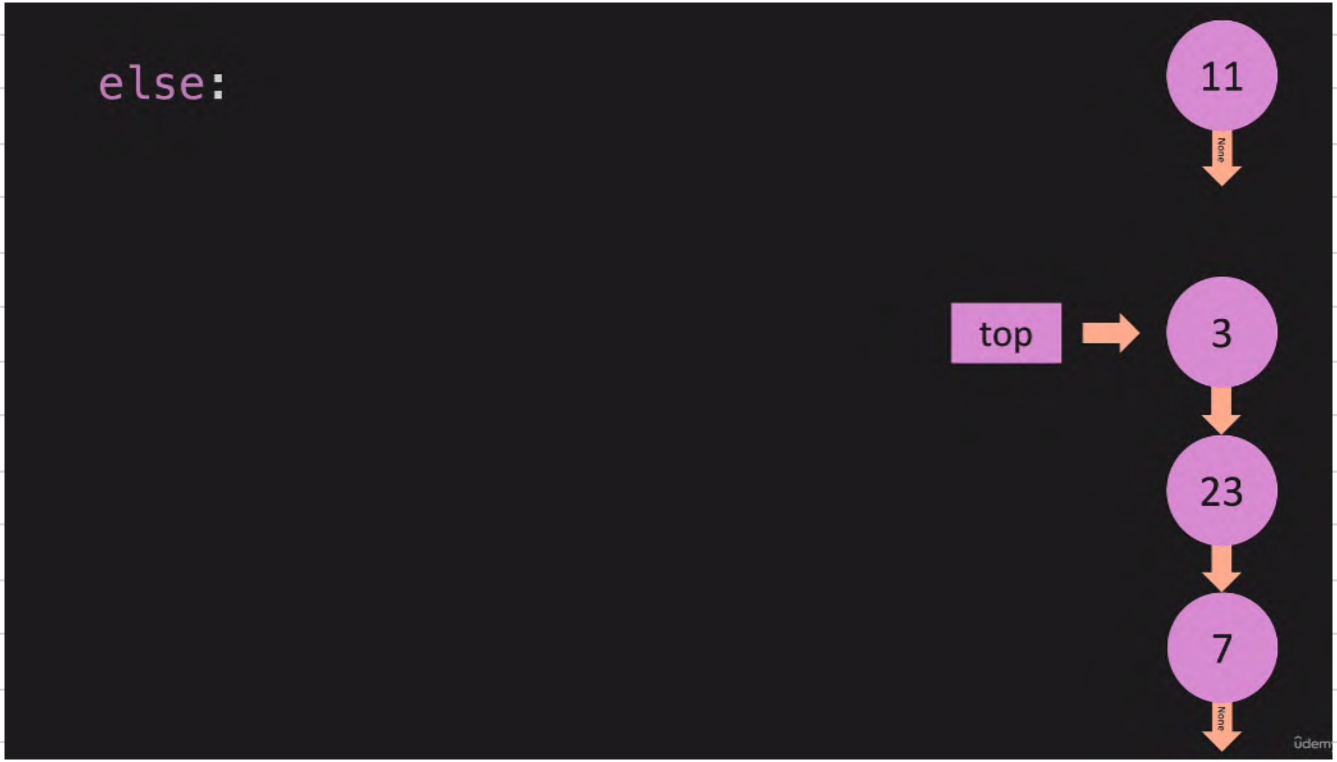
```
def push(self, value):
    new_node = Node(value)
    if self.height == 0:
        self.top = new_node
```

Stack : Push Code



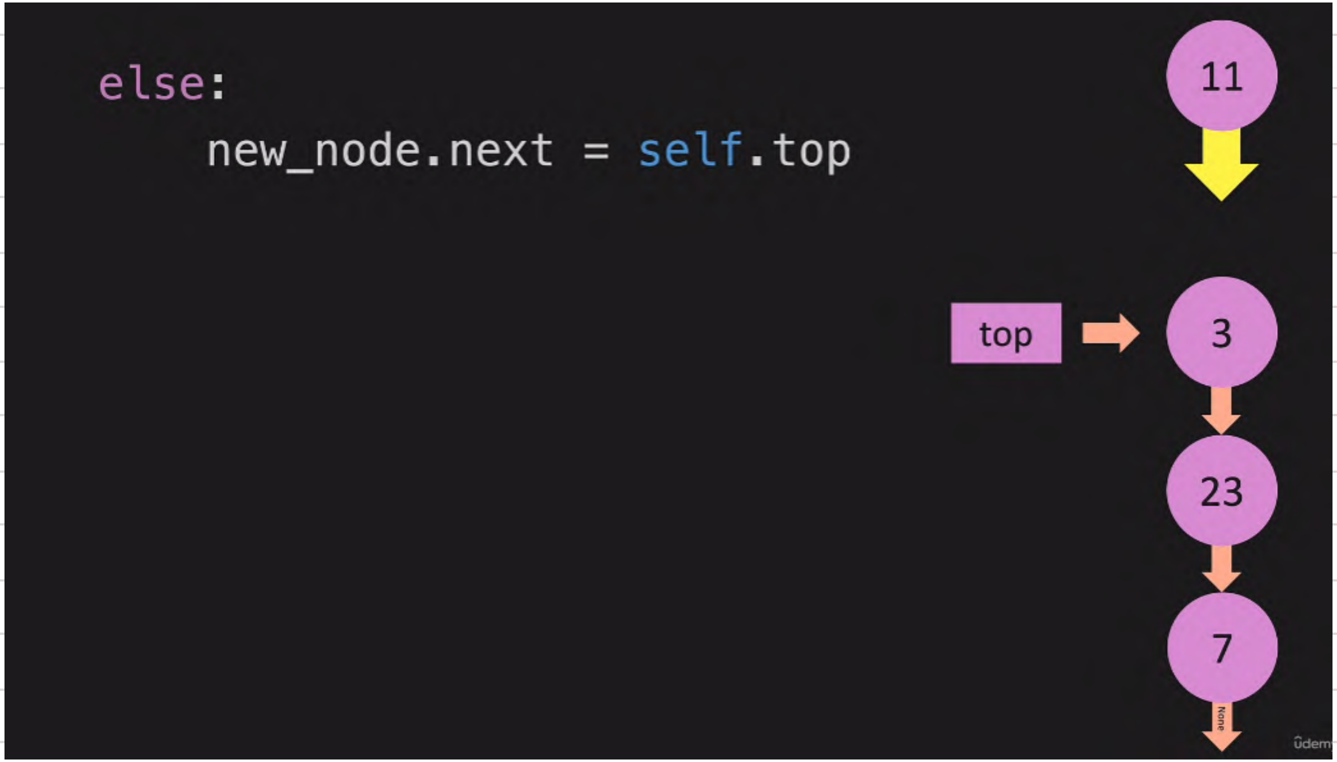
```
def push(self, value):
    new_node = Node(value)
    if self.height == 0:
        self.top = new_node
    else:
```

Stack : Push Code



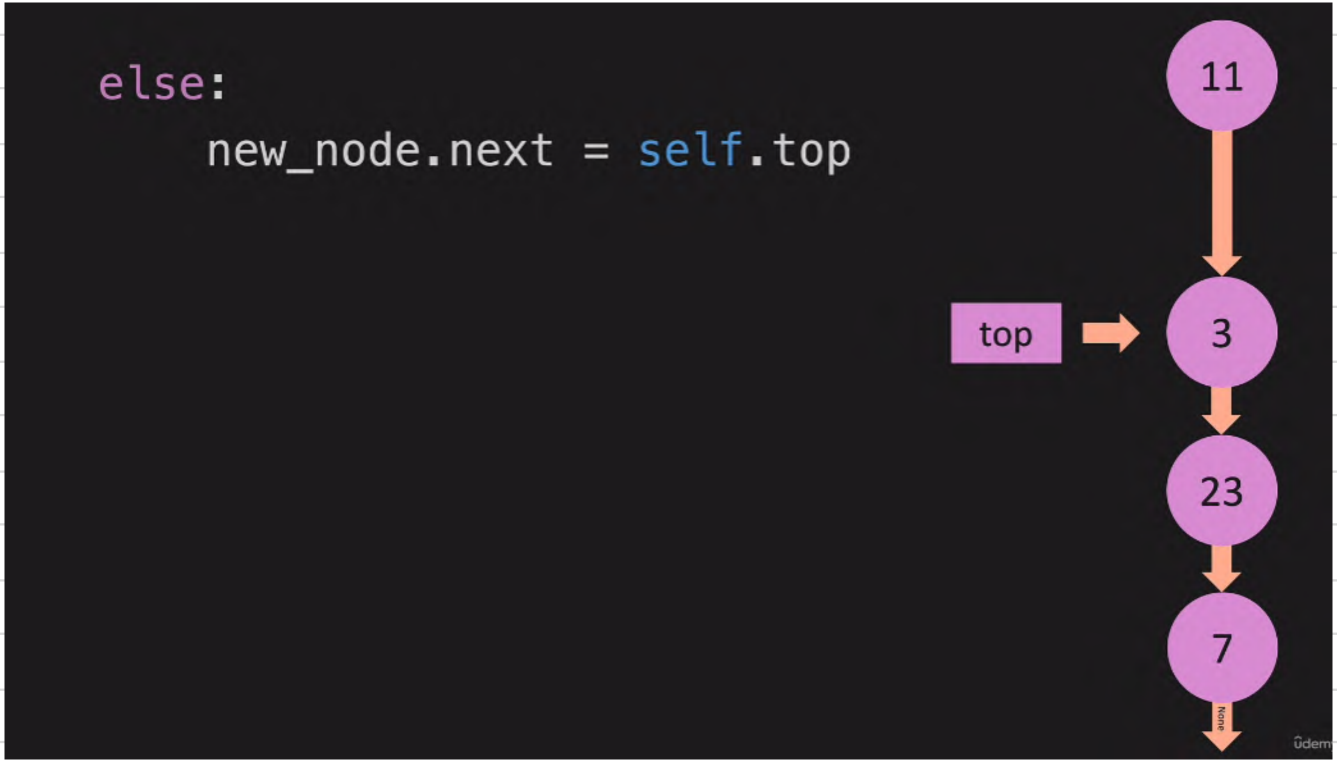
```
else:
```

Stack : Push Code



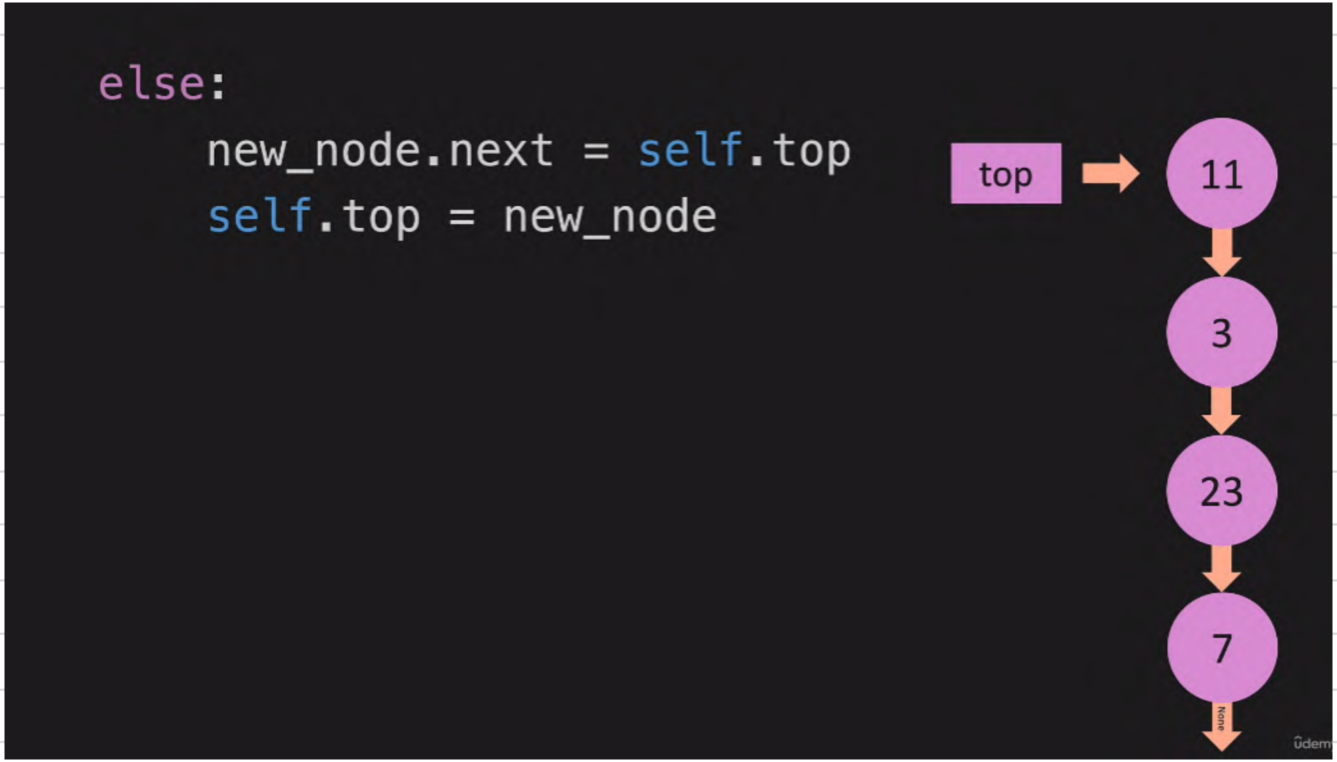
```
else:
    new_node.next = self.top
```

Stack : Push Code



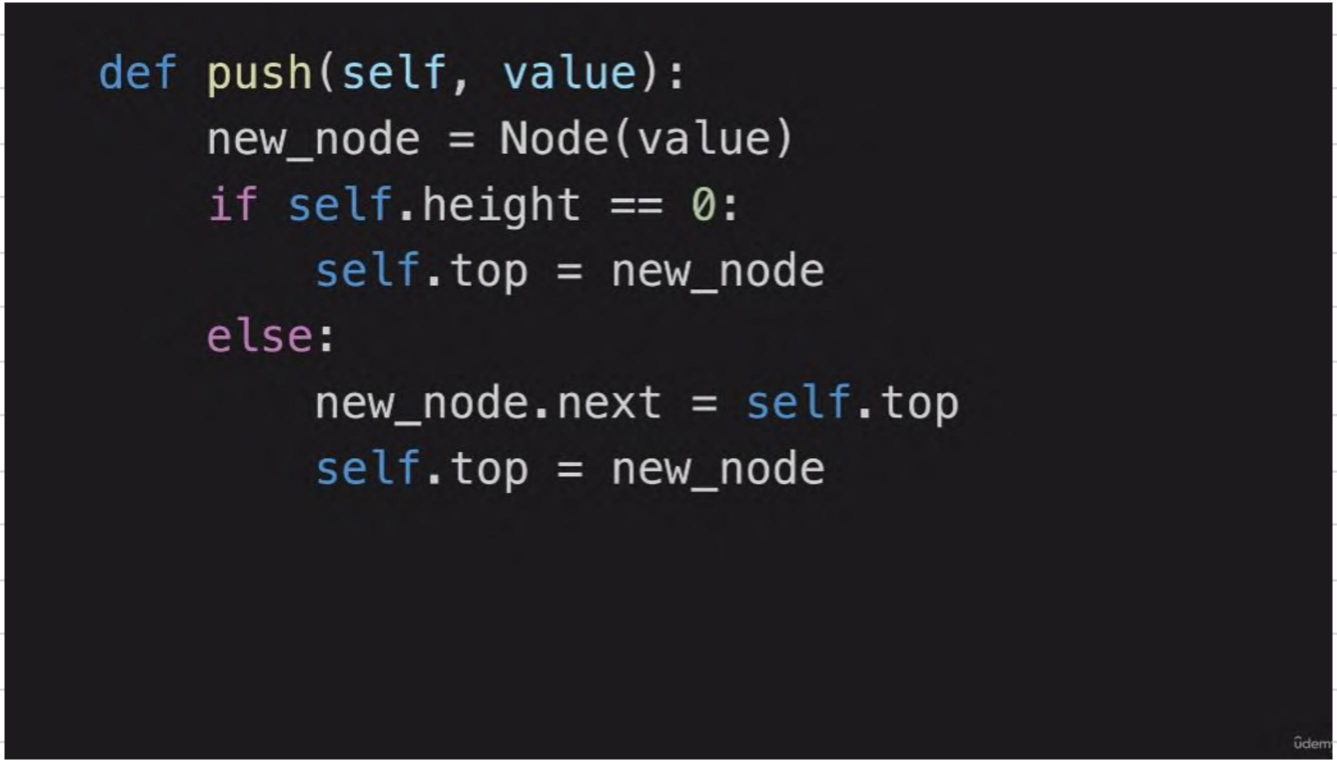
```python
else:
    new_node.next = self.top
```

Stack : Push Code



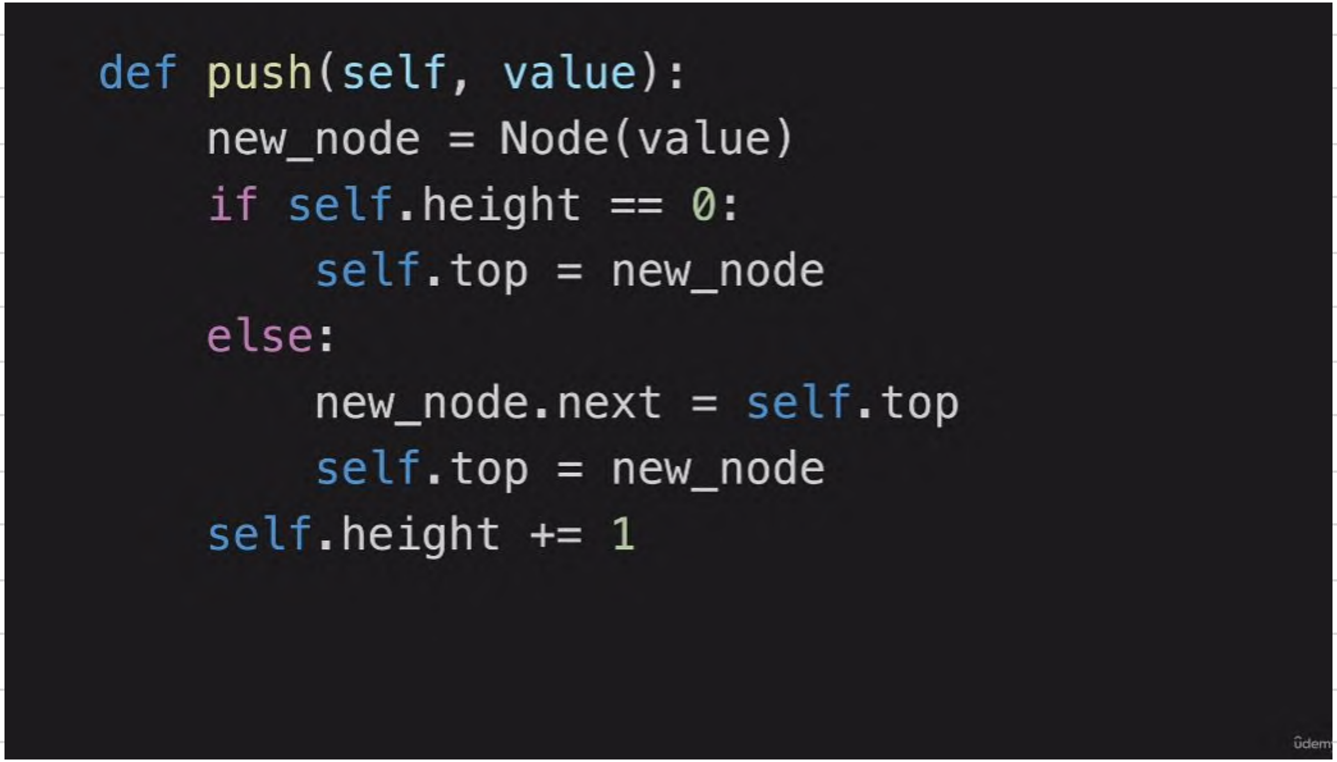
```
else:
    new_node.next = self.top
    self.top = new_node
```

Stack : Push Code



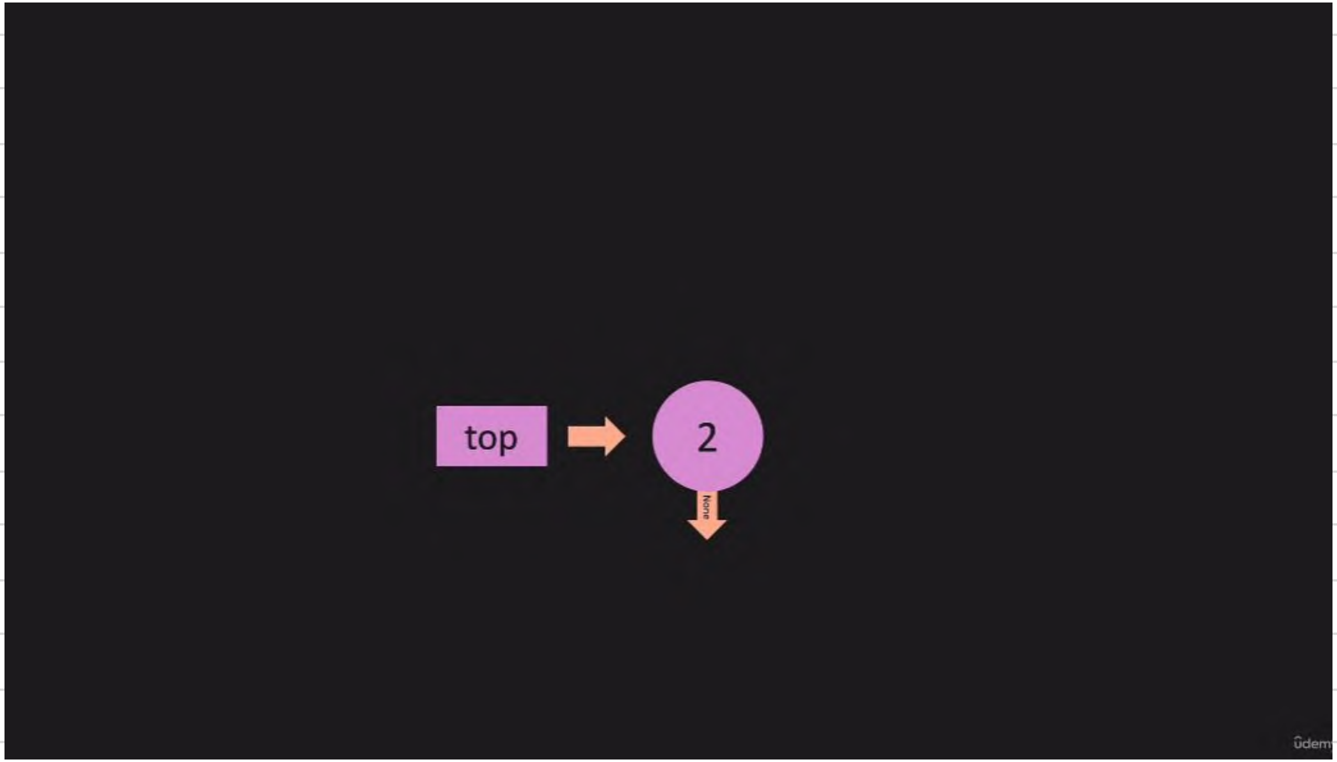
```
def push(self, value):
    new_node = Node(value)
    if self.height == 0:
        self.top = new_node
    else:
        new_node.next = self.top
        self.top = new_node
```

Stack : Push Code

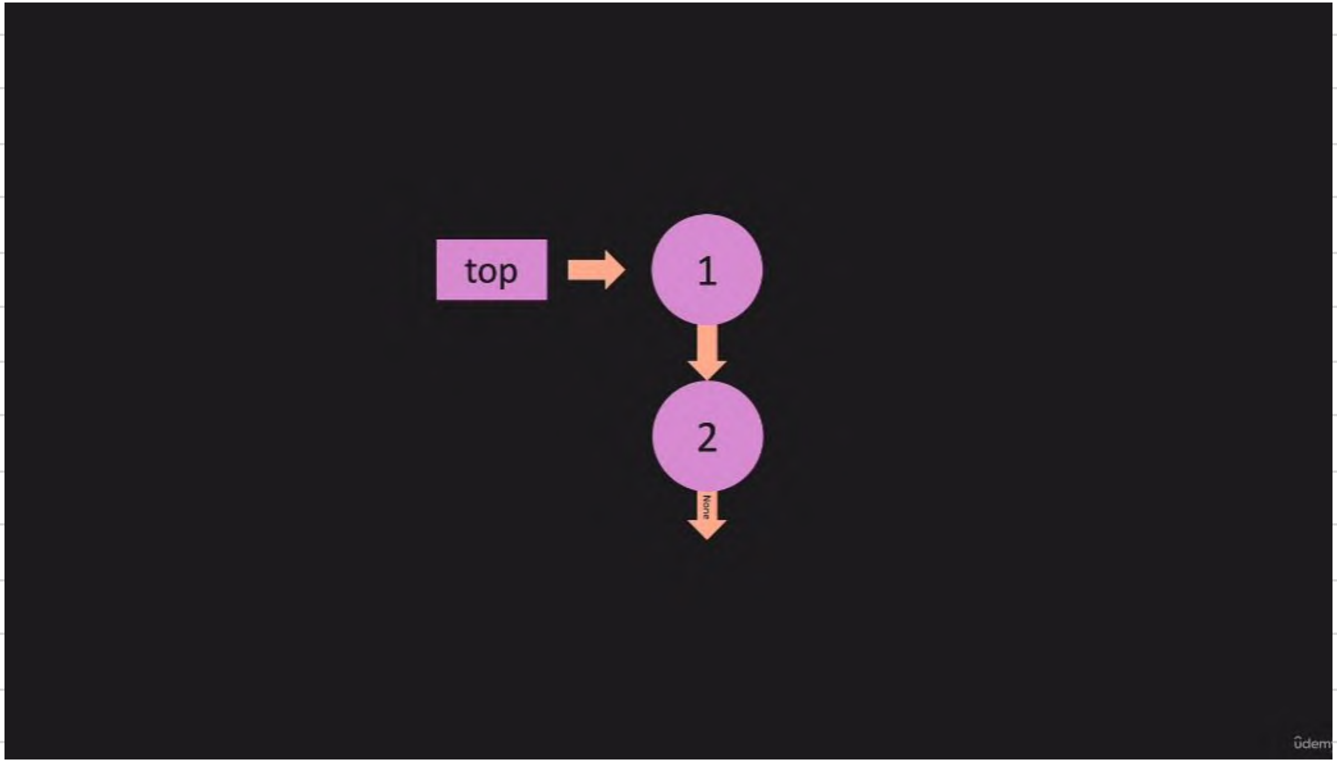


```
def push(self, value):
    new_node = Node(value)
    if self.height == 0:
        self.top = new_node
    else:
        new_node.next = self.top
        self.top = new_node
    self.height += 1
```

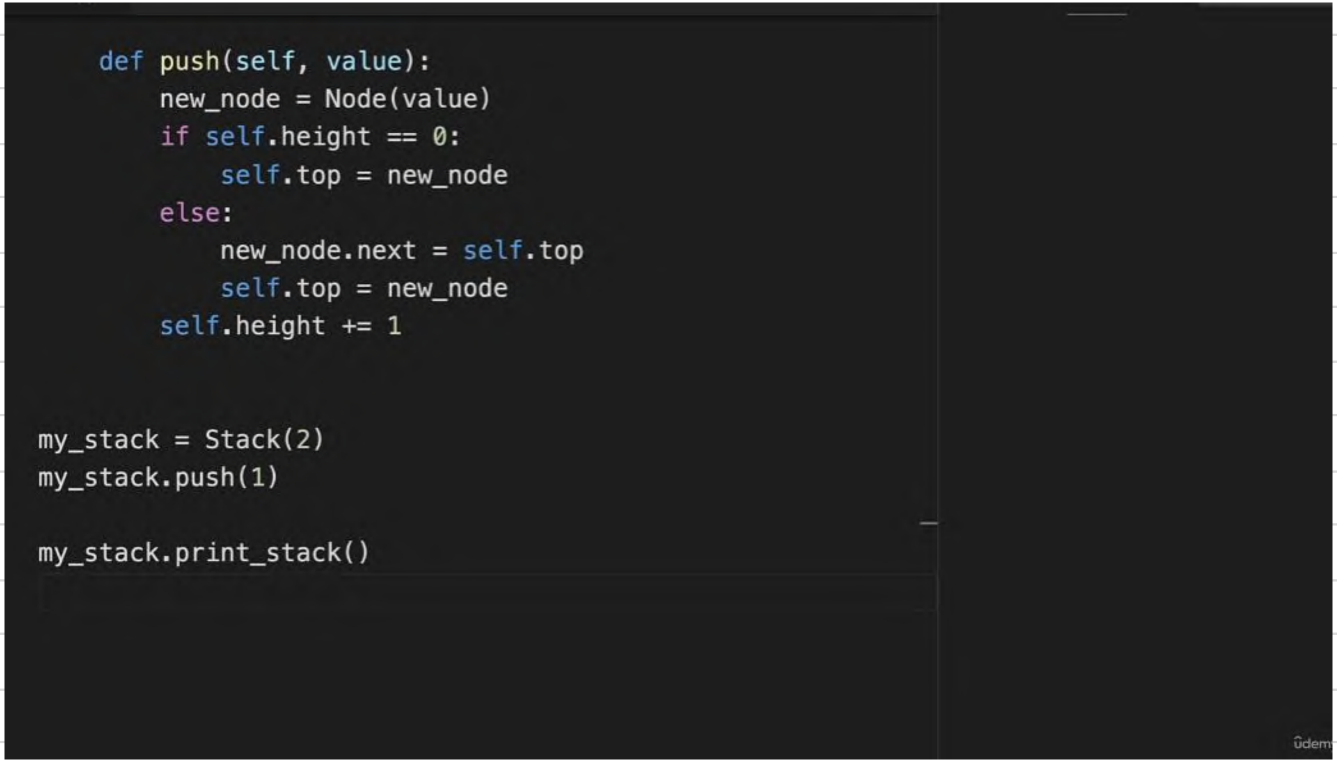
Stack : Push Skenario Percobaan



Stack : Push Skenario Percobaan

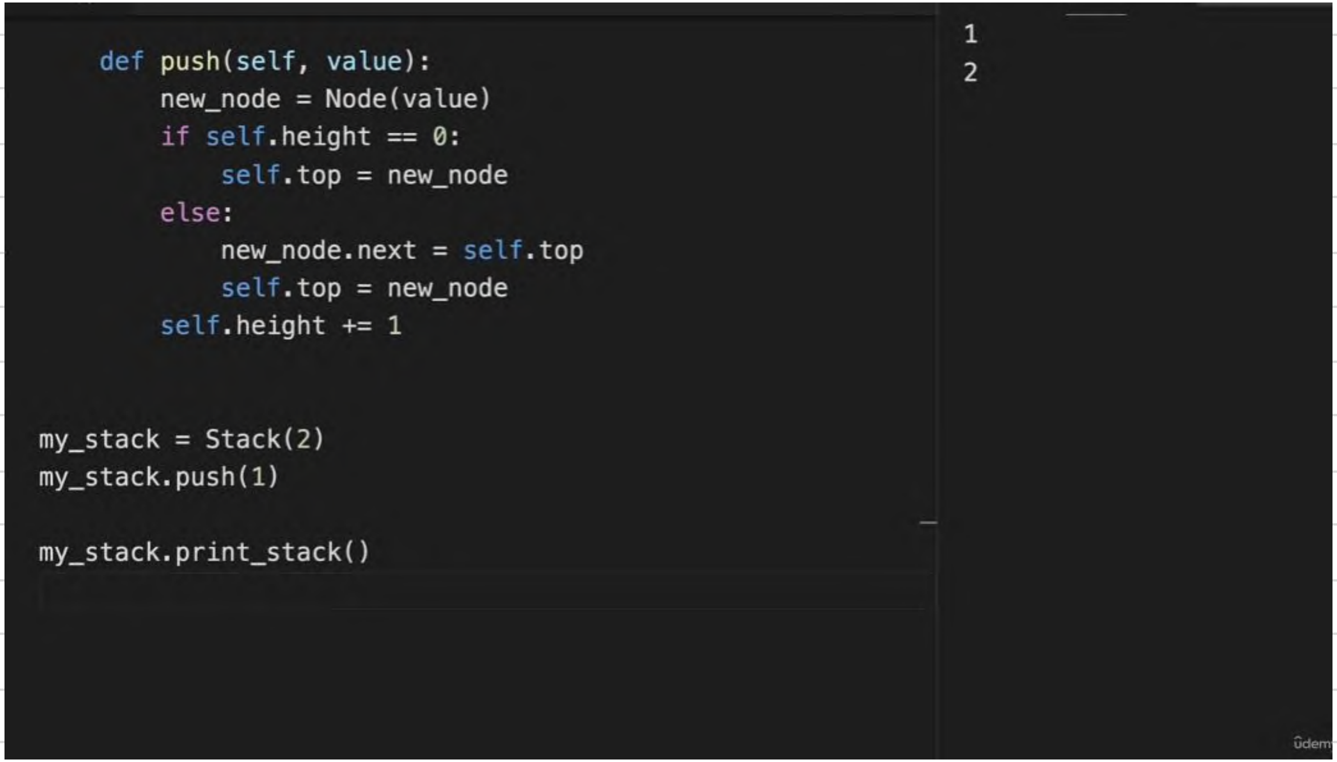


Stack : Push Skenario Percobaan



```
    def push(self, value):
        new_node = Node(value)
        if self.height == 0:
            self.top = new_node
        else:
            new_node.next = self.top
            self.top = new_node
        self.height += 1


my_stack = Stack(2)
my_stack.push(1)

my_stack.print_stack()
```

Stack : Push Skenario Percobaan



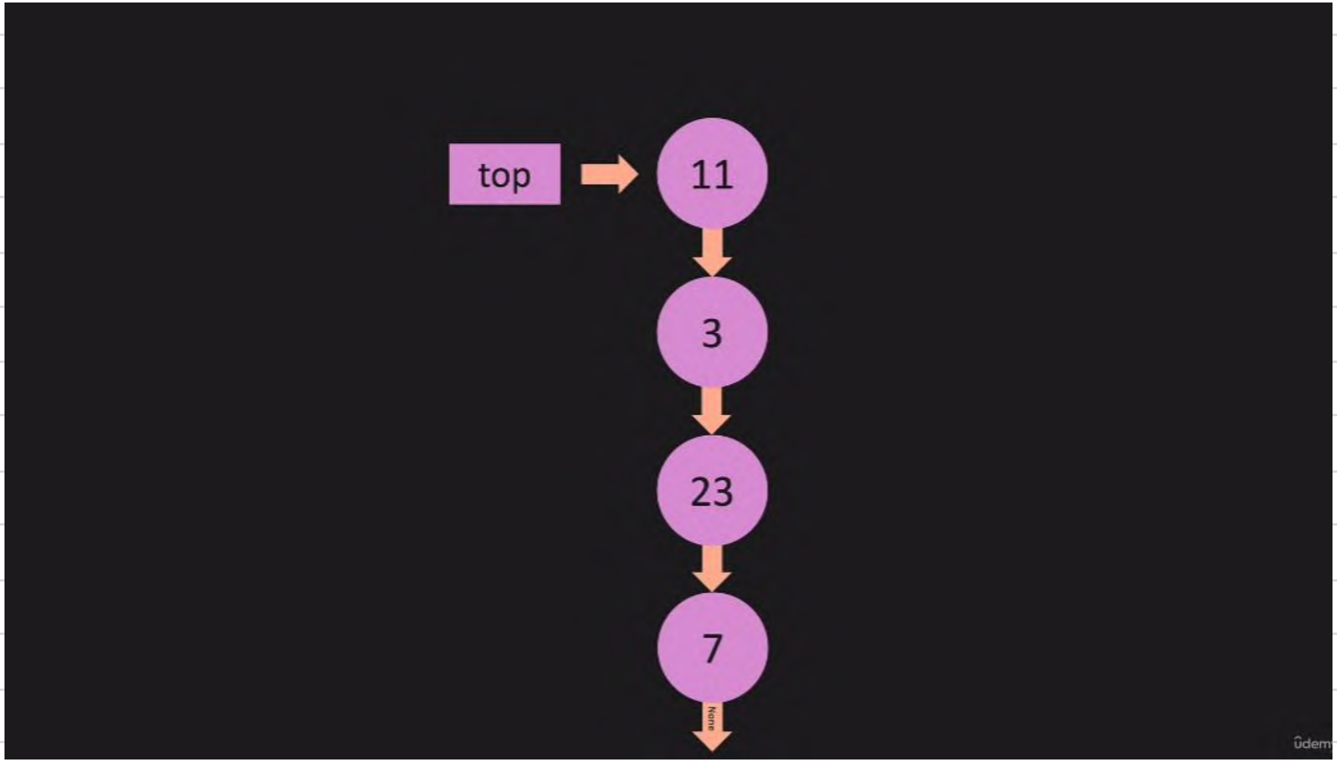
```
    def push(self, value):
        new_node = Node(value)
        if self.height == 0:
            self.top = new_node
        else:
            new_node.next = self.top
            self.top = new_node
        self.height += 1


my_stack = Stack(2)
my_stack.push(1)

my_stack.print_stack()
```

```
1
2
```

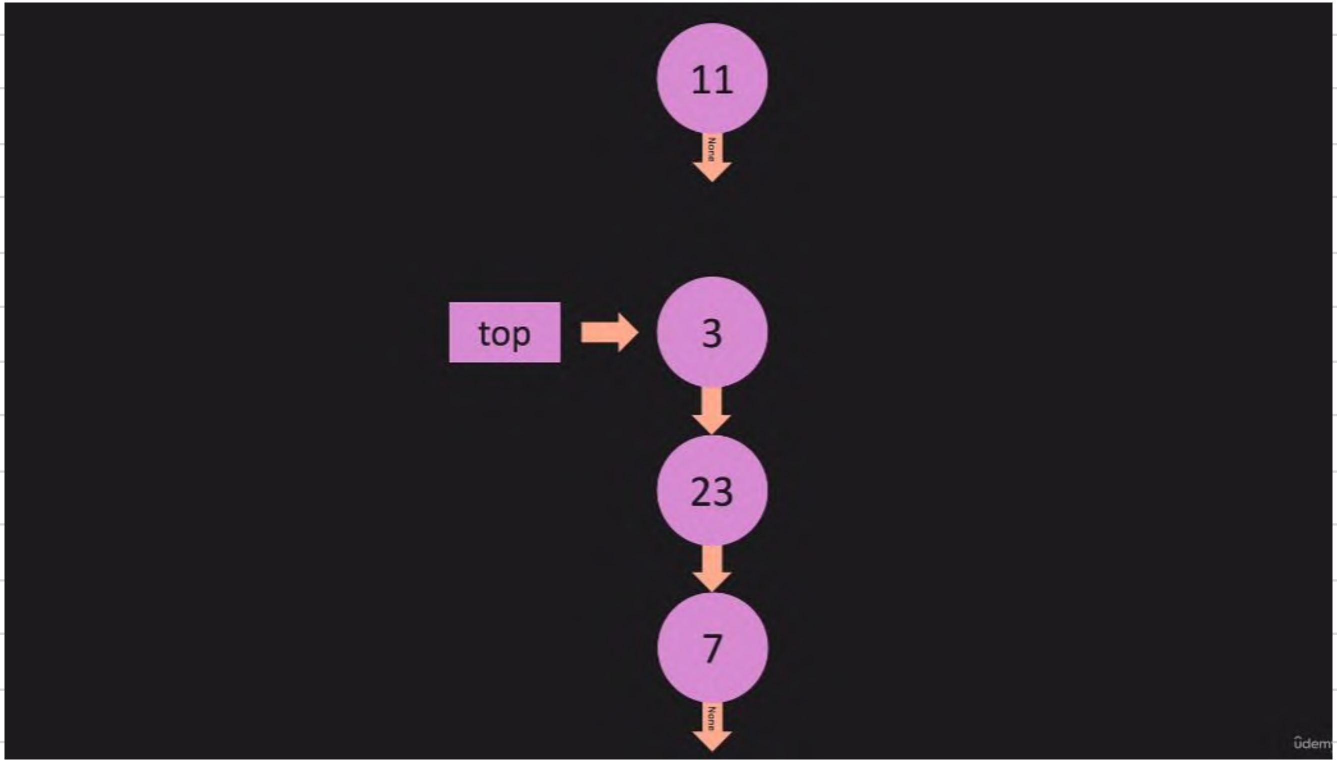
04

Stack Pop

Stack : Pop

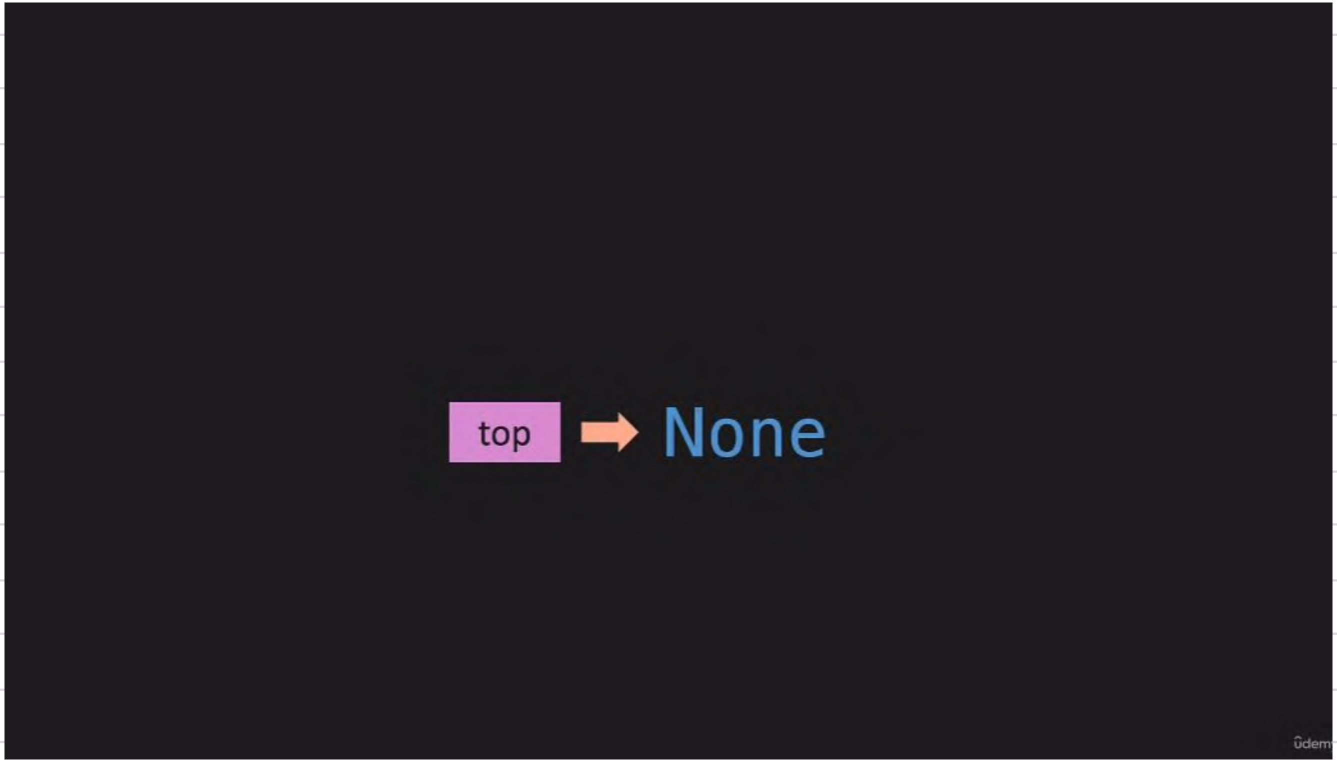


Stack : Pop

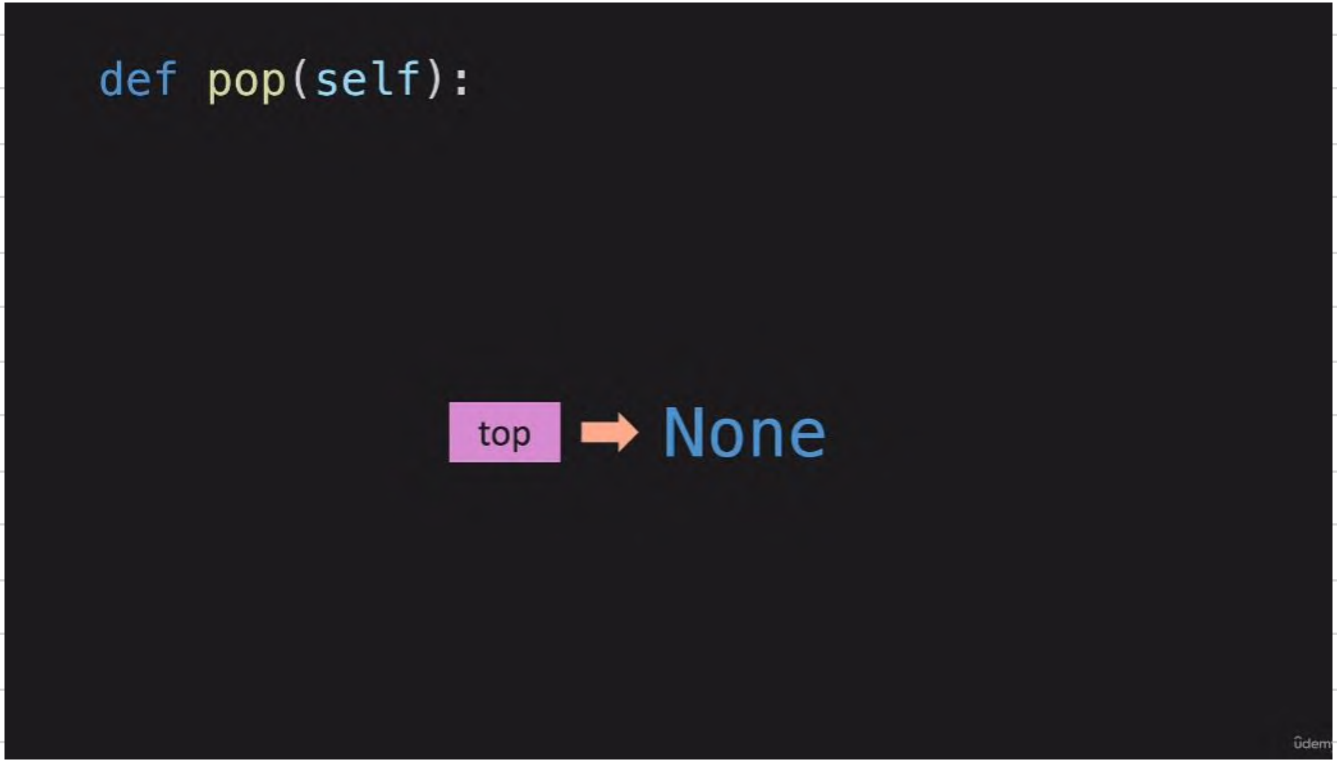


Edge Condition #1

Stack : Pop

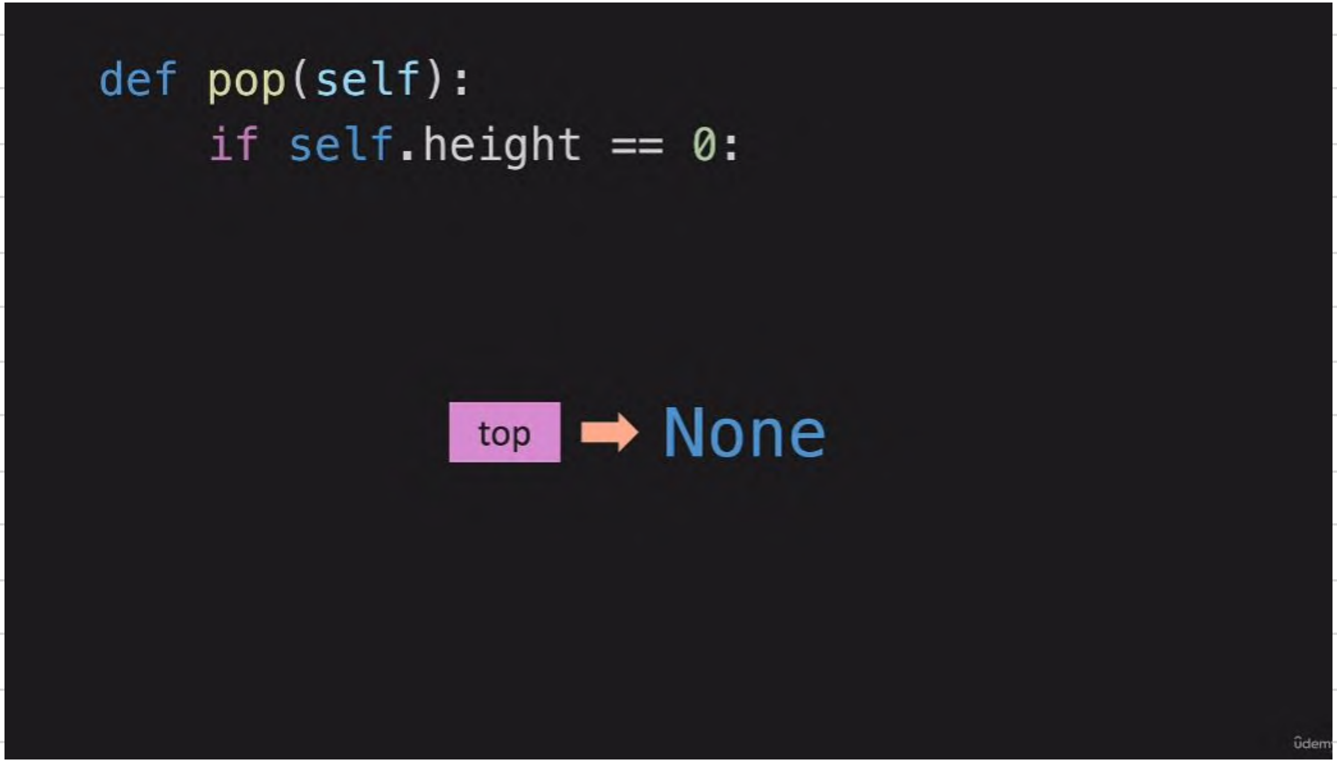


Stack : Pop Code



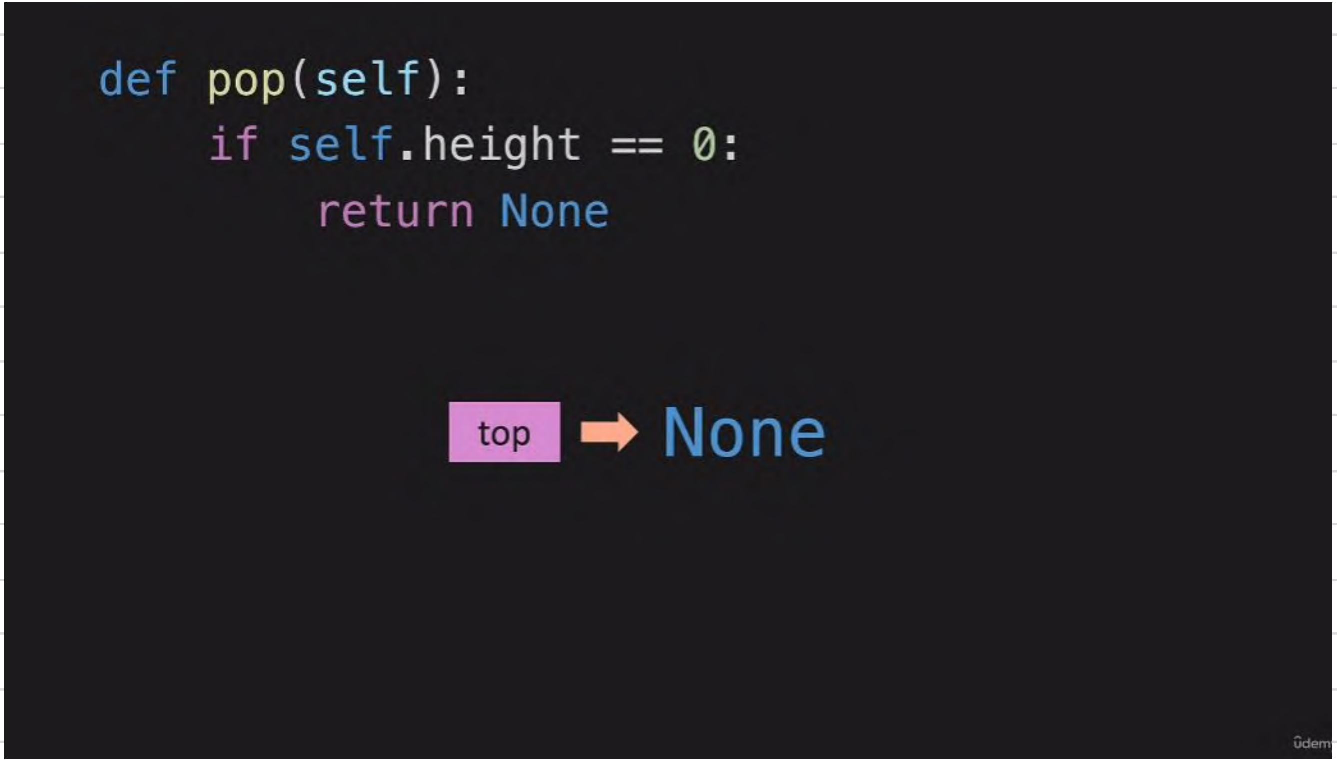
```
def pop(self):
```

Stack : Pop Code



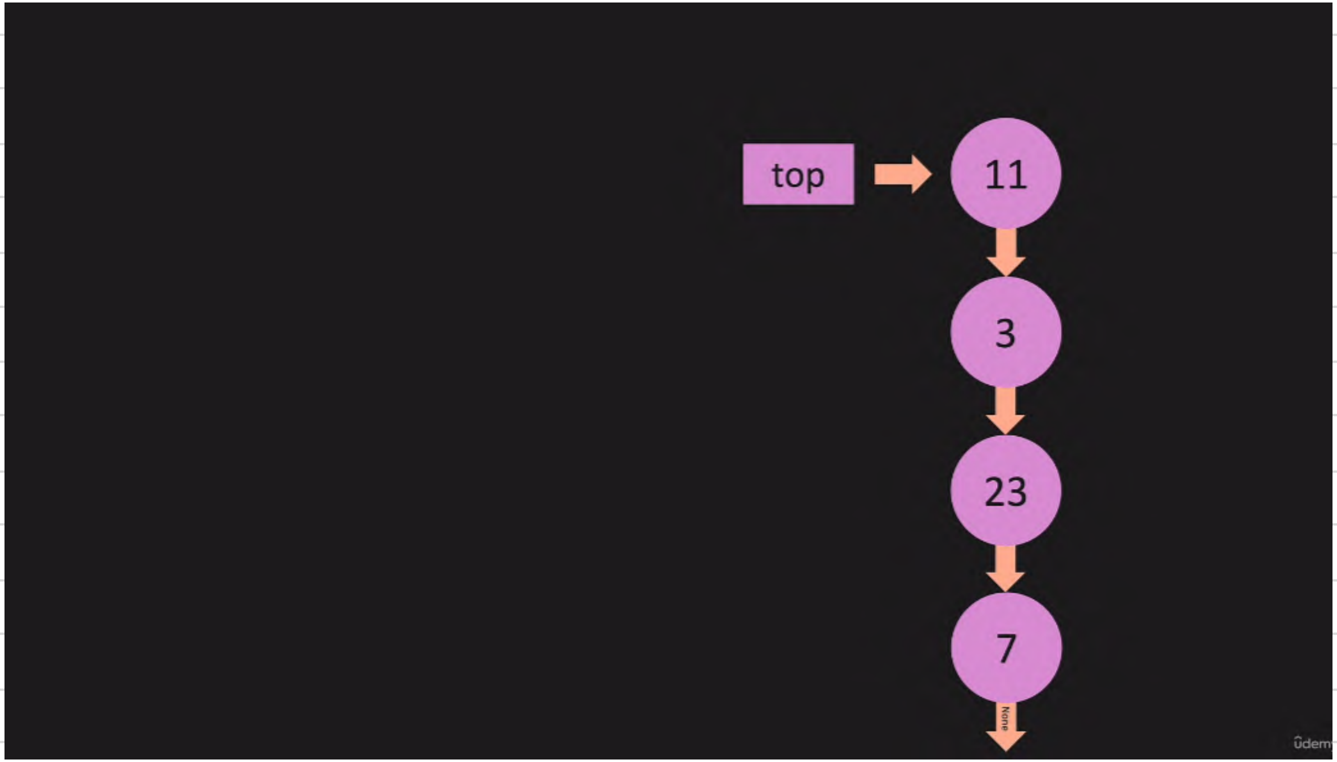
```
def pop(self):
    if self.height == 0:
```

Stack : Pop Code

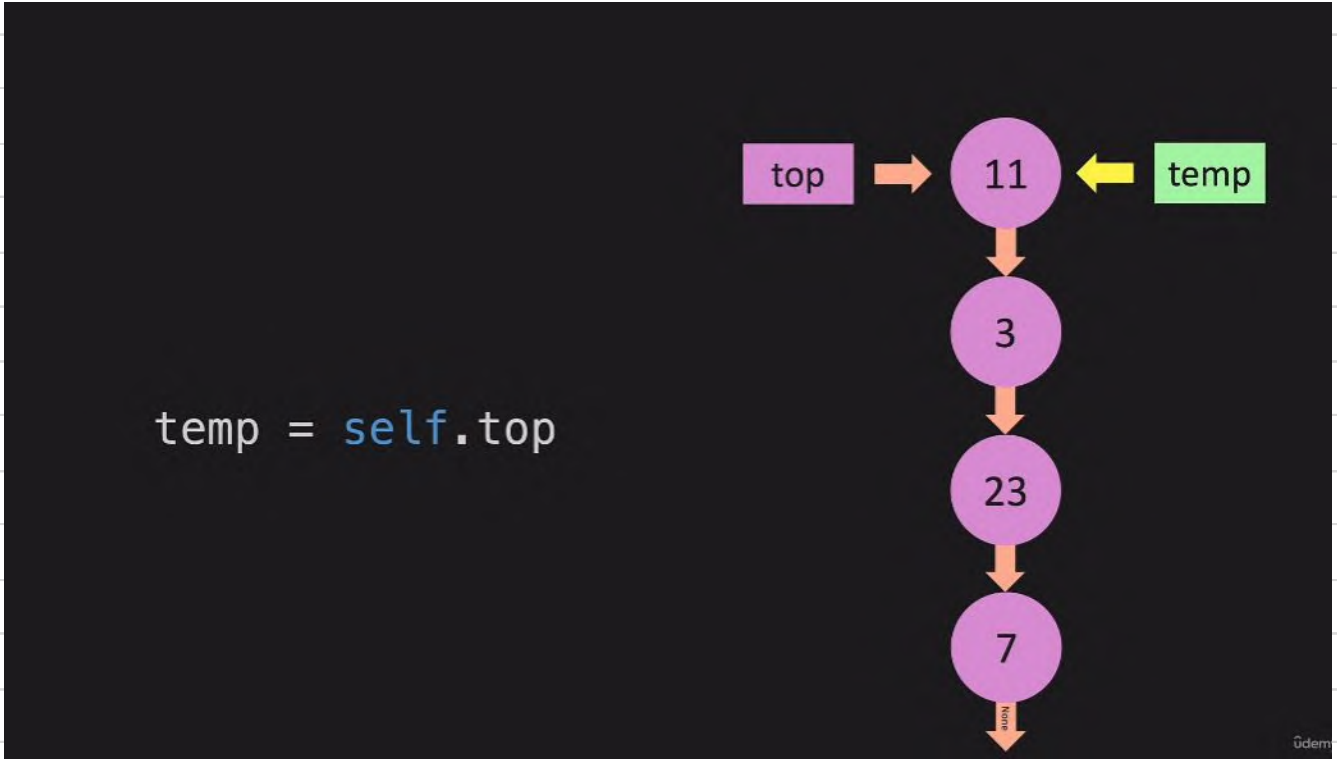


```
def pop(self):
    if self.height == 0:
        return None
```

Stack : Pop Code

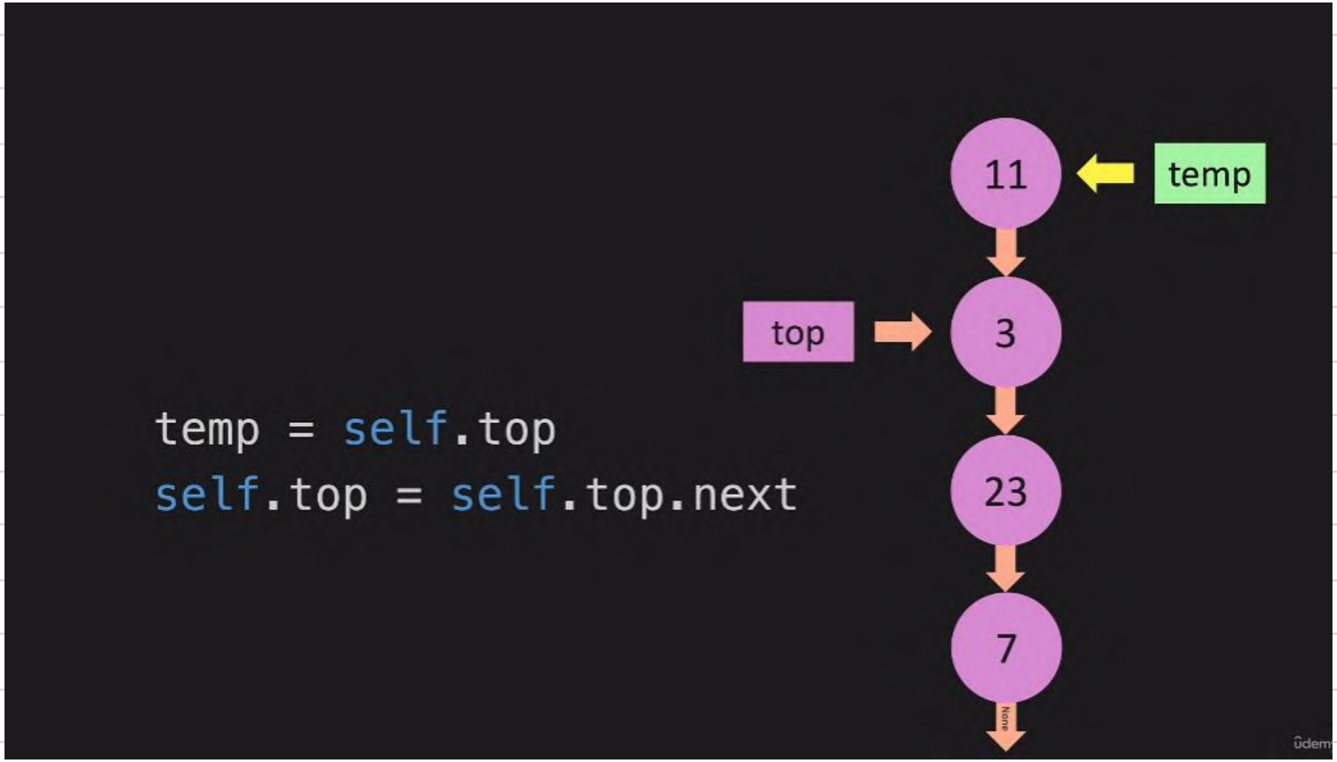


Stack : Pop Code



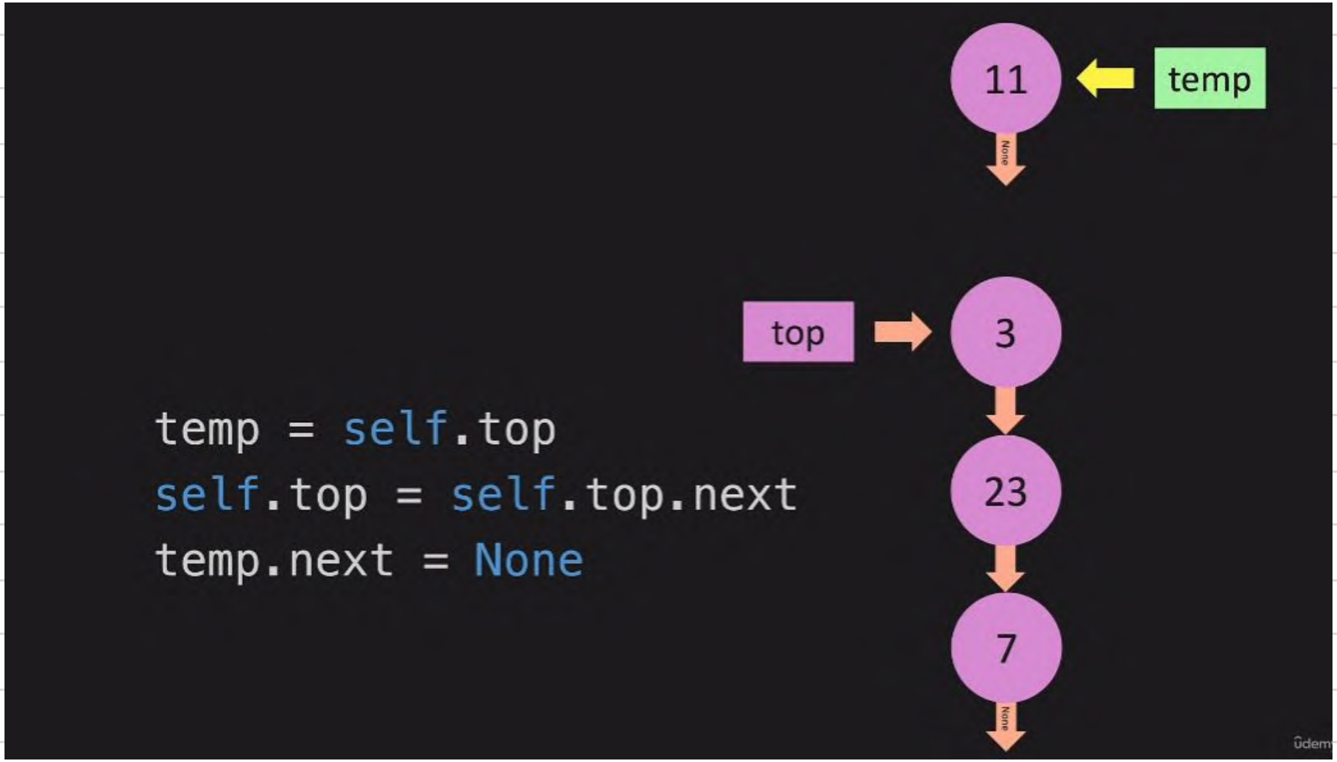
```
temp = self.top
```

Stack : Pop Code



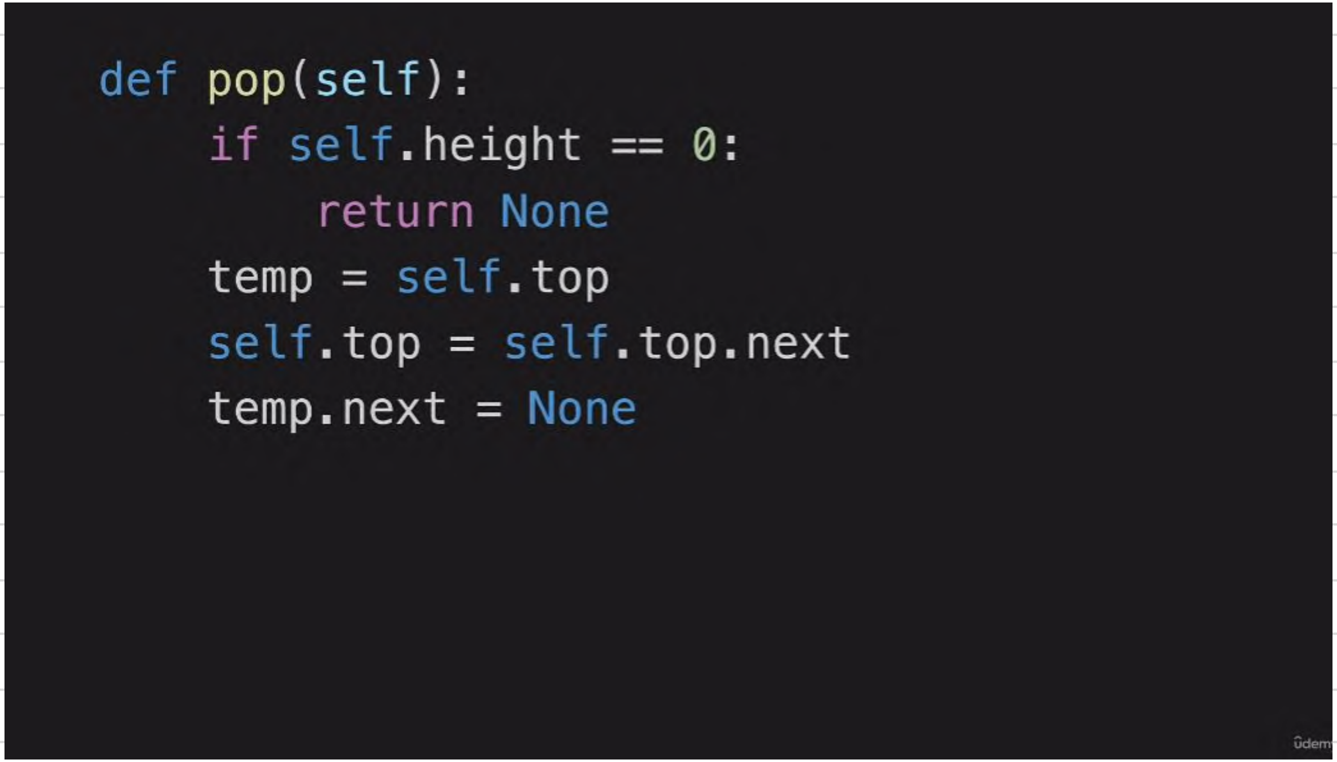
```
temp = self.top
self.top = self.top.next
```

Stack : Pop Code



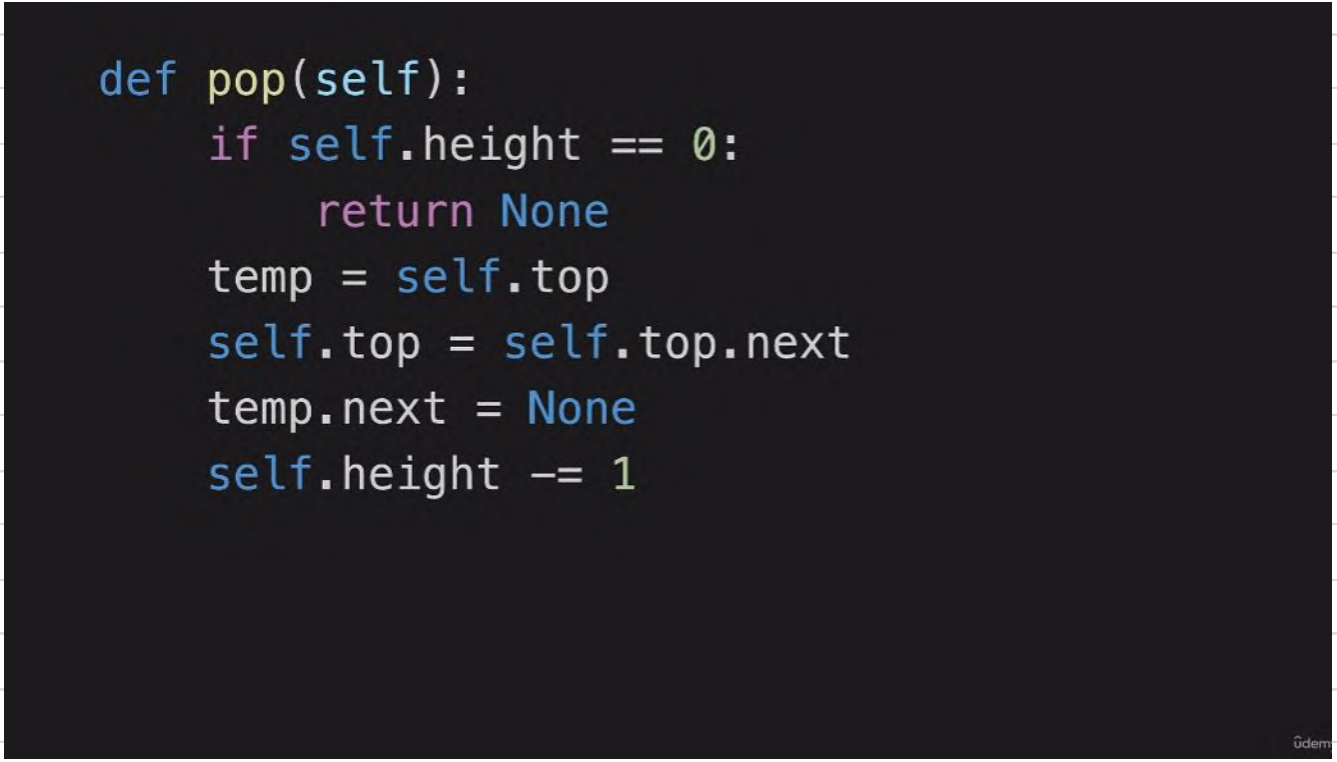
```
temp = self.top
self.top = self.top.next
temp.next = None
```

Stack : Pop Code



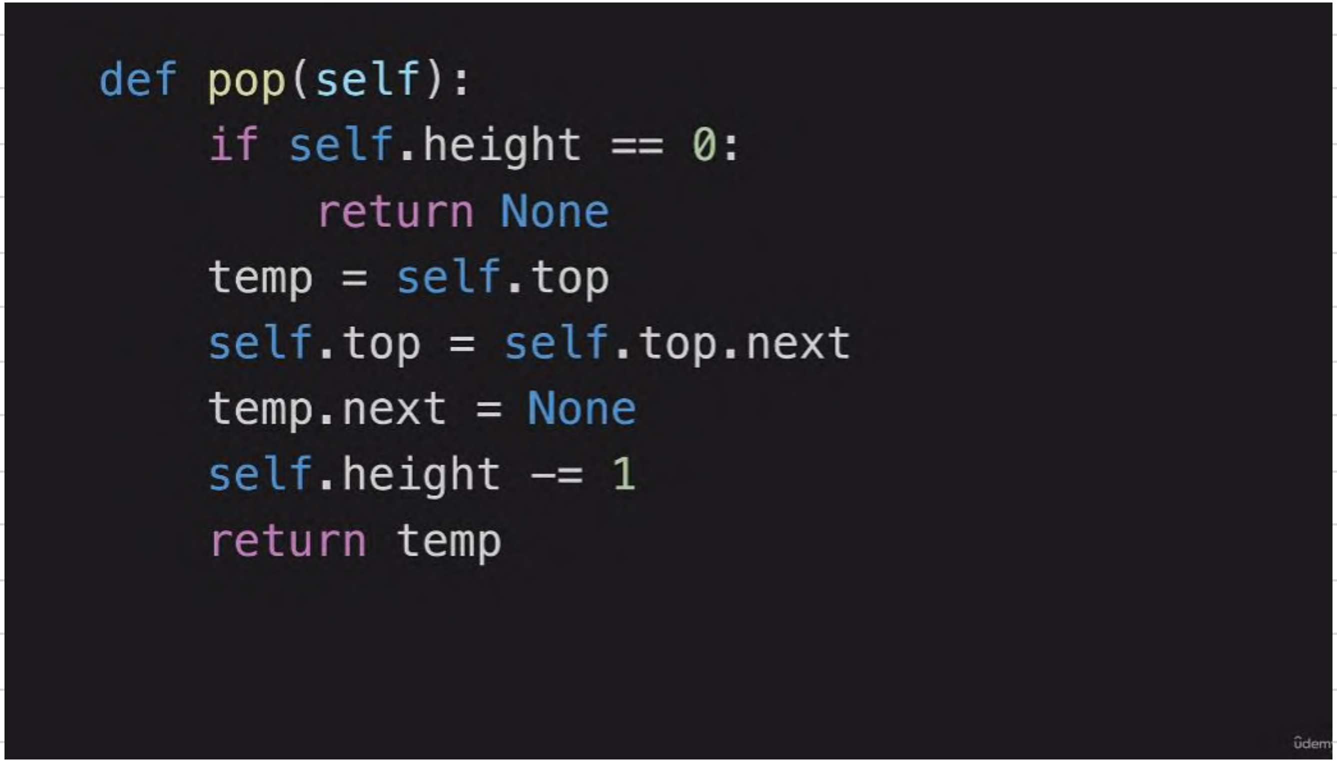
```
def pop(self):
    if self.height == 0:
        return None
    temp = self.top
    self.top = self.top.next
    temp.next = None
```

Stack : Pop Code



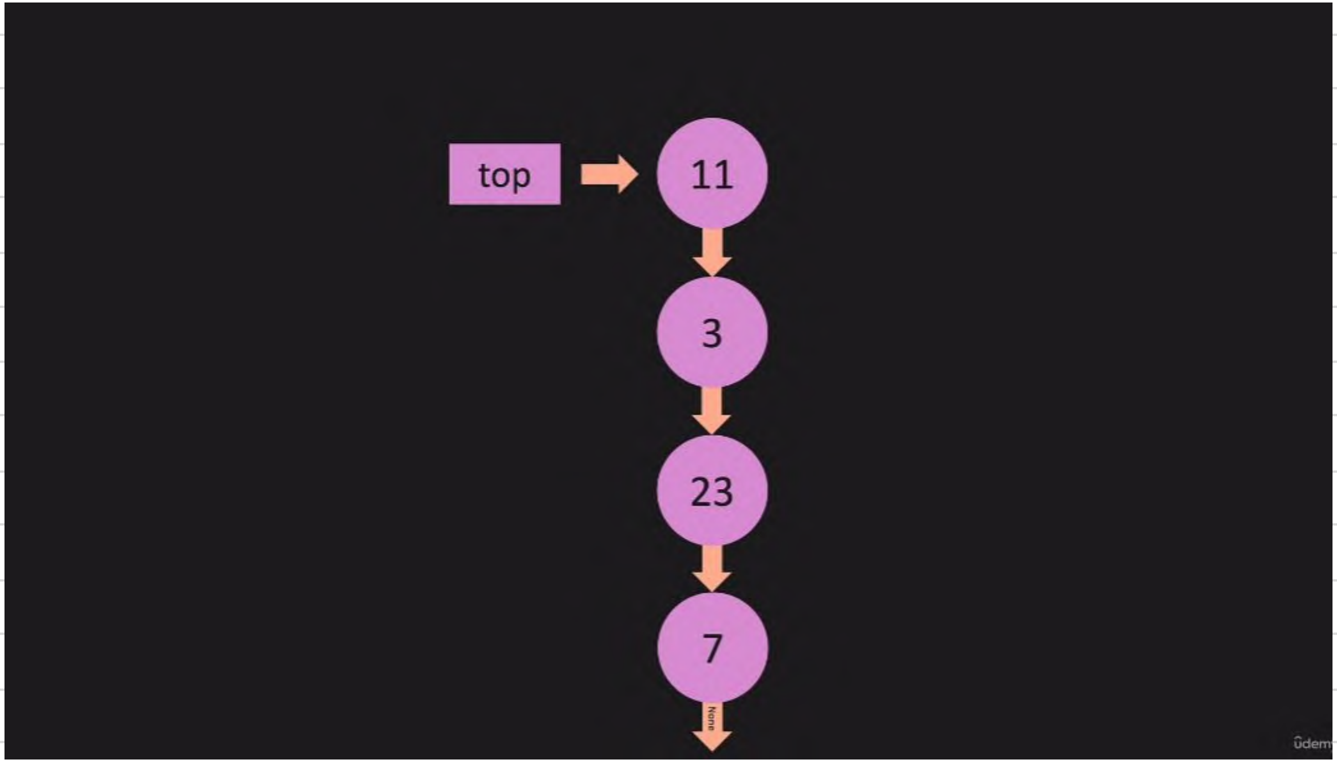
```
def pop(self):
    if self.height == 0:
        return None
    temp = self.top
    self.top = self.top.next
    temp.next = None
    self.height -= 1
```

Stack : Pop Code

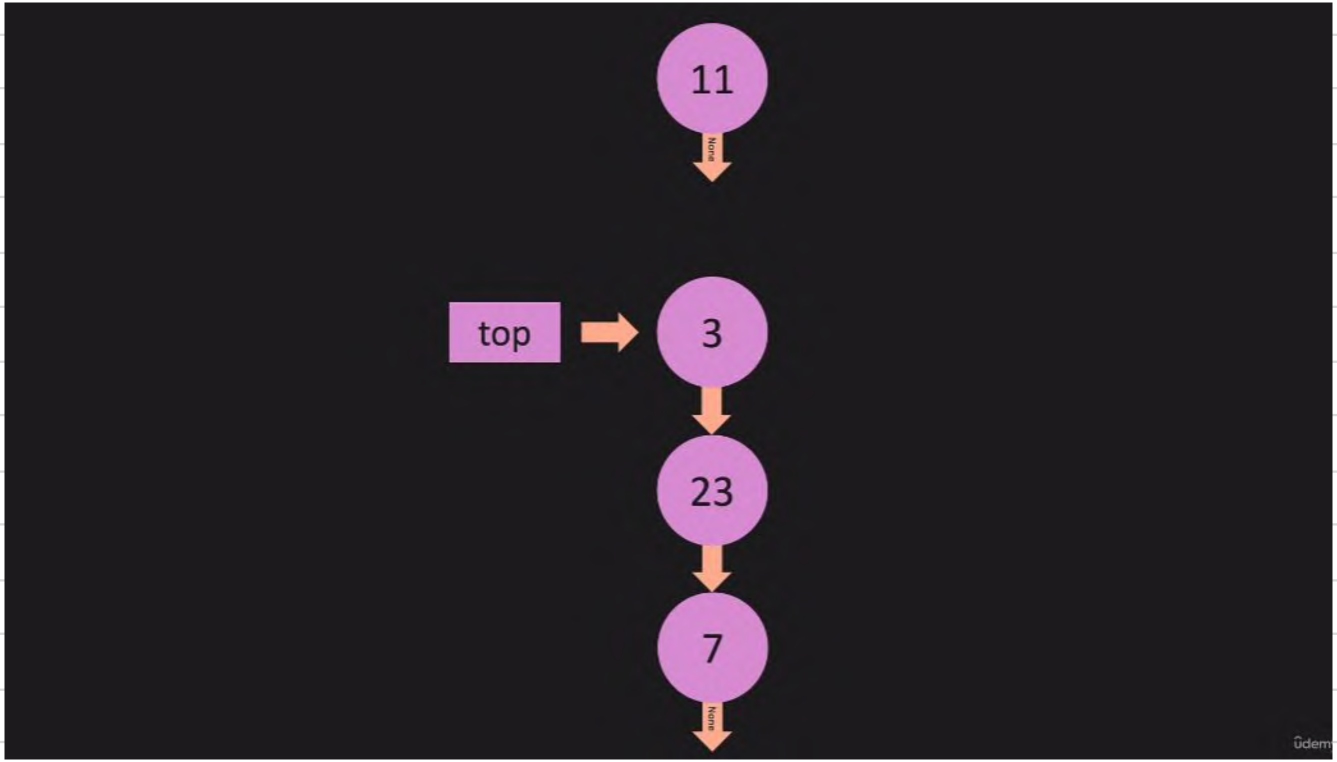


```
def pop(self):
    if self.height == 0:
        return None
    temp = self.top
    self.top = self.top.next
    temp.next = None
    self.height -= 1
    return temp
```

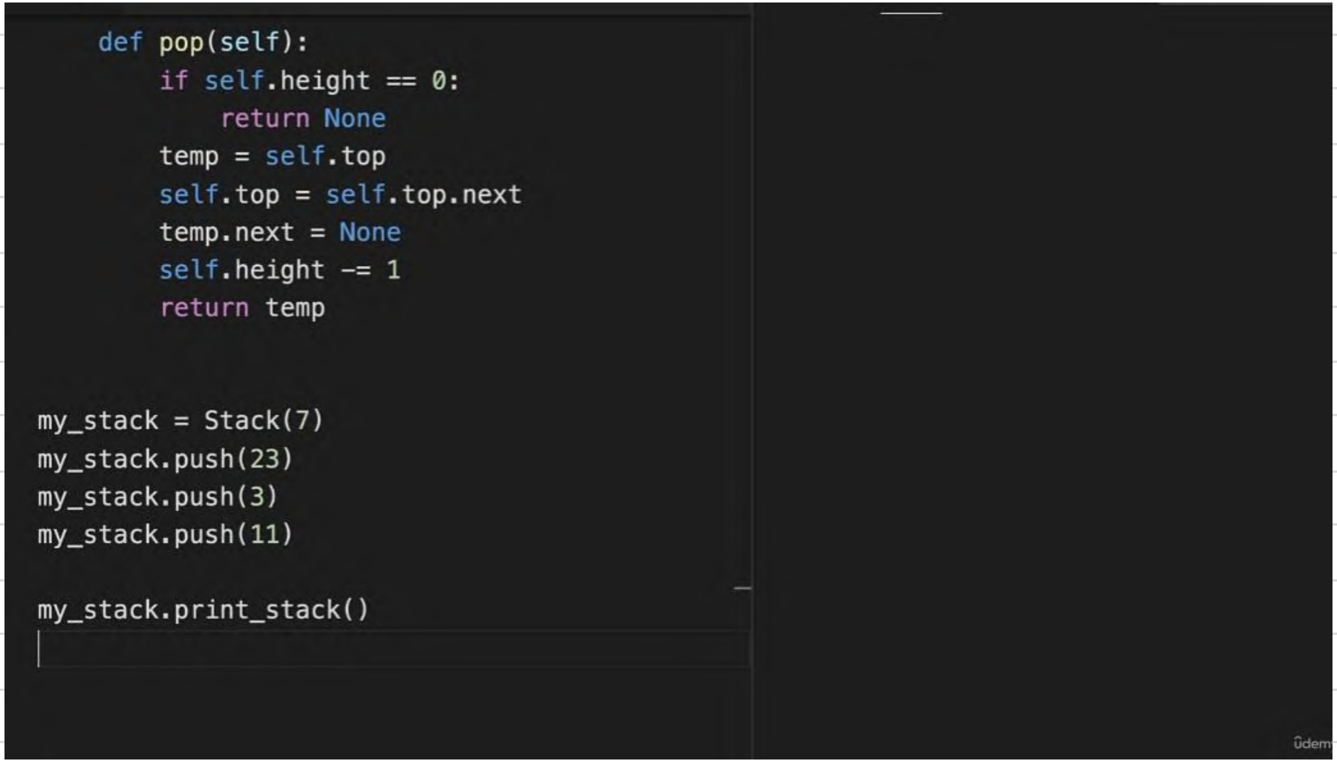
Stack : Pop Skenario Percobaan



Stack : Pop Skenario Percobaan

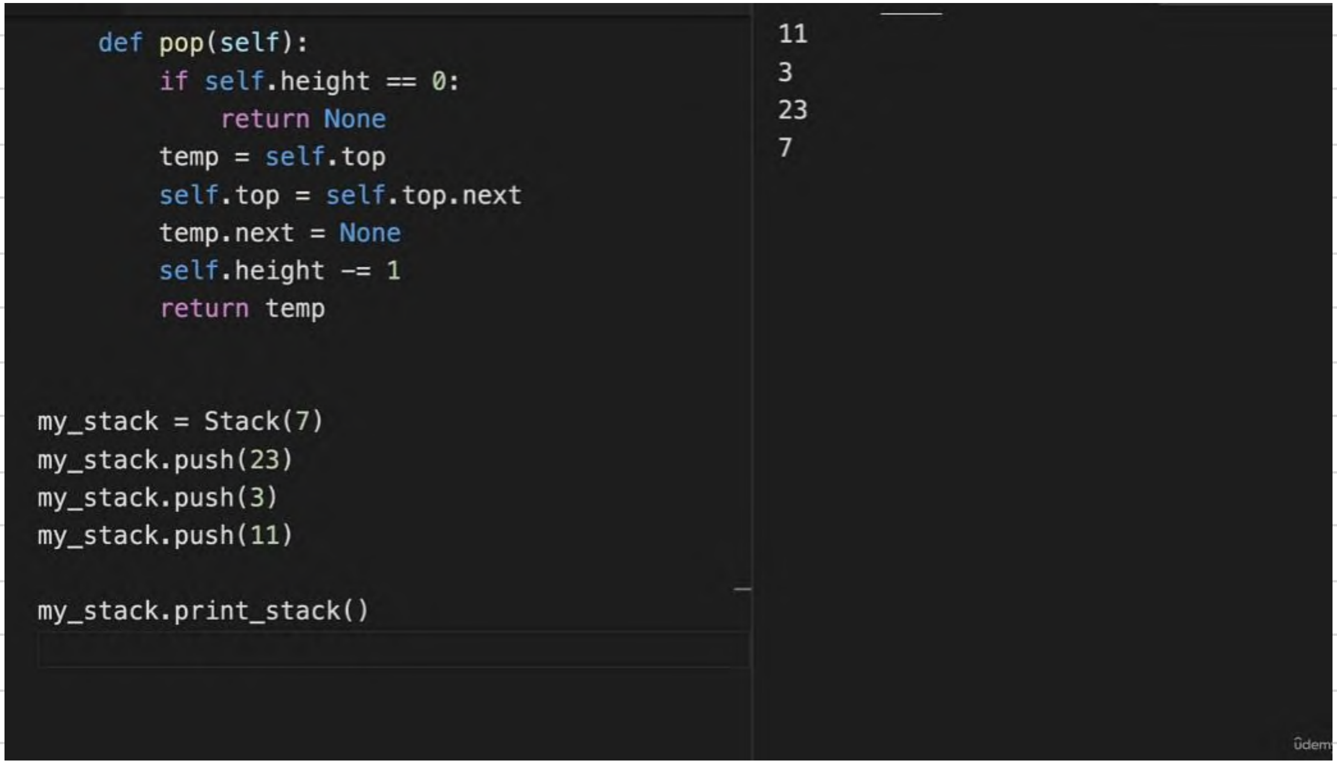


Stack : Pop Skenario Percobaan



```
    def pop(self):
        if self.height == 0:
            return None
        temp = self.top
        self.top = self.top.next
        temp.next = None
        self.height -= 1
        return temp


my_stack = Stack(7)
my_stack.push(23)
my_stack.push(3)
my_stack.push(11)

my_stack.print_stack()
```

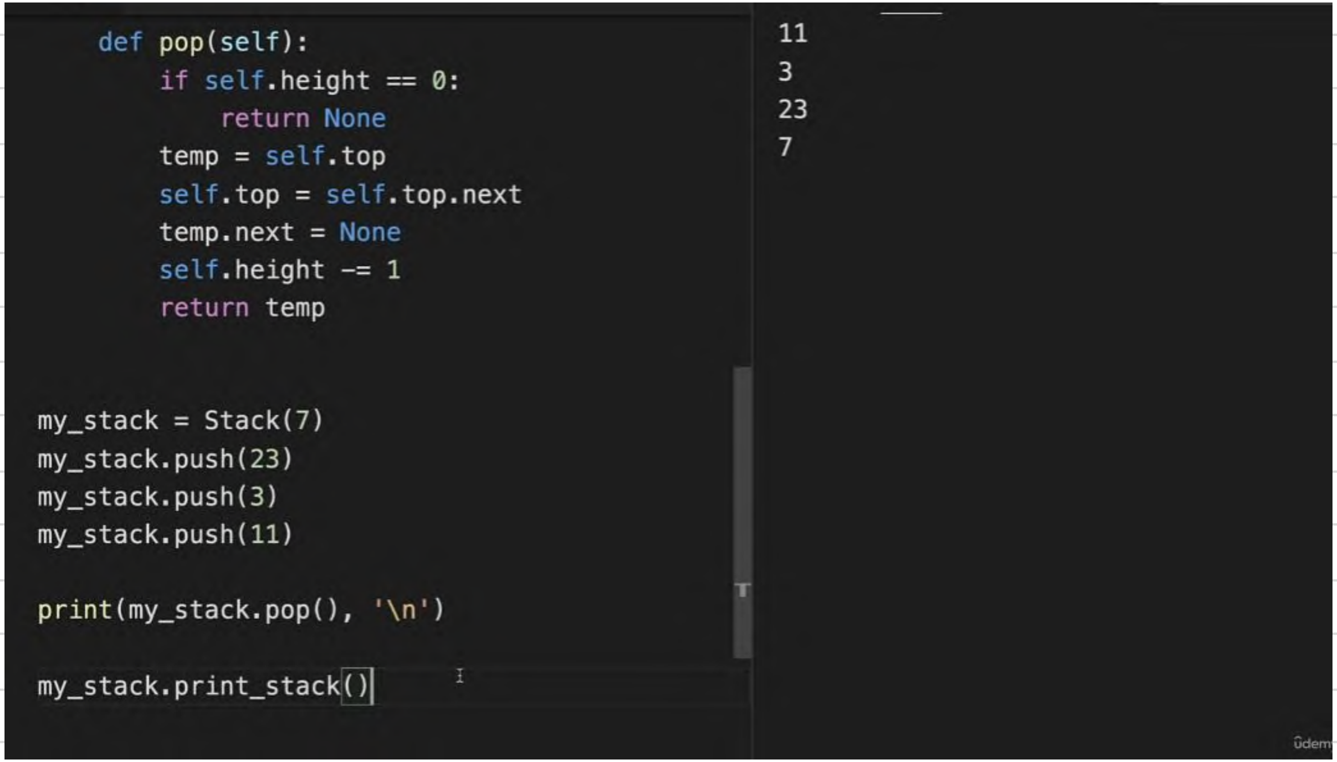
Stack : Pop Skenario Percobaan



```
    def pop(self):
        if self.height == 0:
            return None
        temp = self.top
        self.top = self.top.next
        temp.next = None
        self.height -= 1
        return temp


my_stack = Stack(7)
my_stack.push(23)
my_stack.push(3)
my_stack.push(11)

my_stack.print_stack()
```

```
11
3
23
7
```

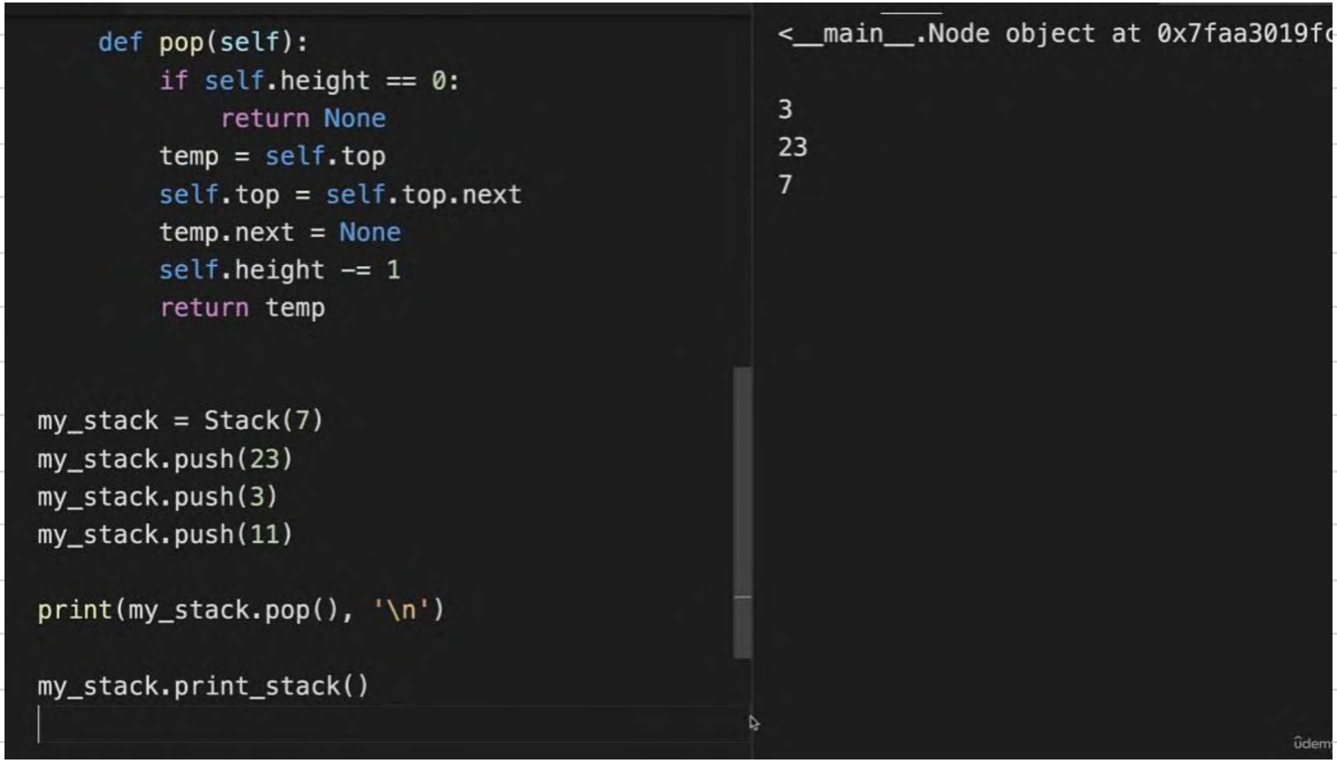
Stack : Pop Skenario Percobaan



```
    def pop(self):
        if self.height == 0:
            return None
        temp = self.top
        self.top = self.top.next
        temp.next = None
        self.height -= 1
        return temp


my_stack = Stack(7)
my_stack.push(23)
my_stack.push(3)
my_stack.push(11)

print(my_stack.pop(), '\n')

my_stack.print_stack()
```

```
11
3
23
7
```

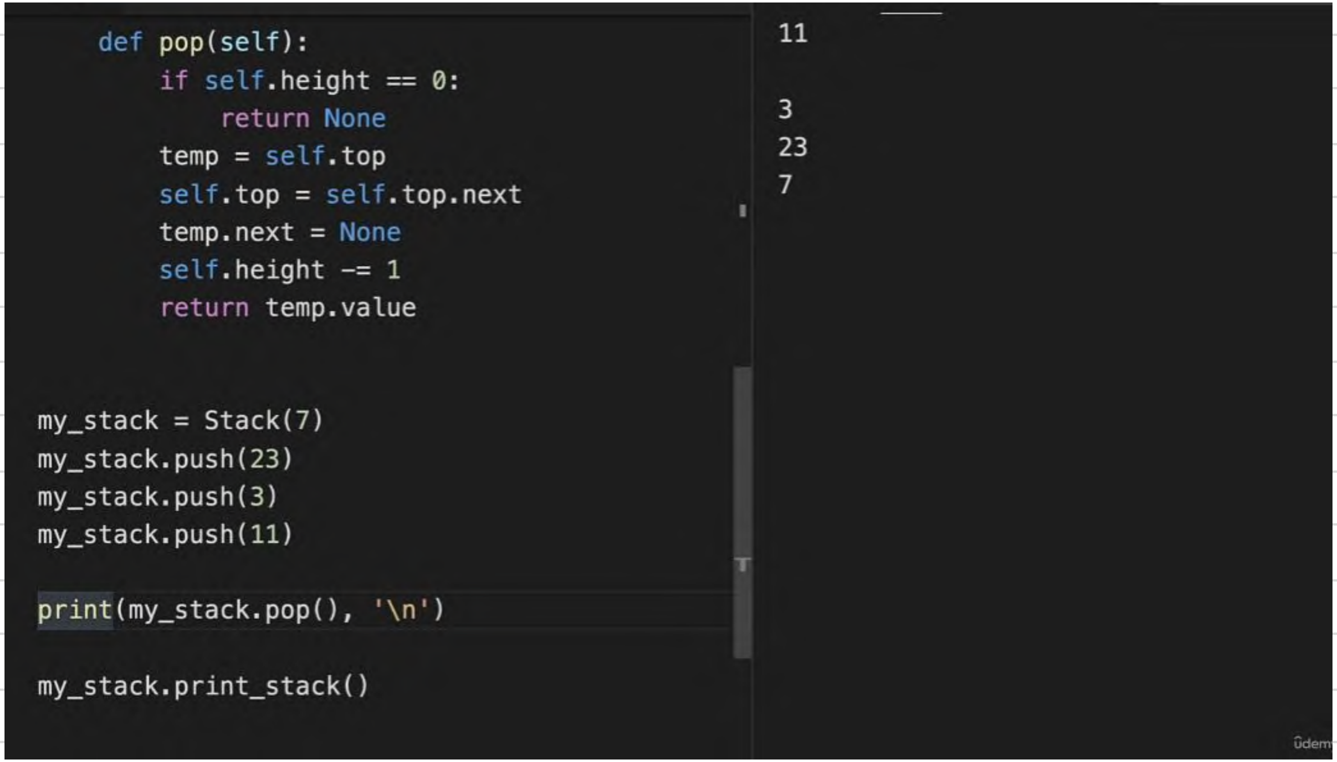
Stack : Pop Skenario Percobaan



```
    def pop(self):
        if self.height == 0:
            return None
        temp = self.top
        self.top = self.top.next
        temp.next = None
        self.height -= 1
        return temp


my_stack = Stack(7)
my_stack.push(23)
my_stack.push(3)
my_stack.push(11)

print(my_stack.pop(), '\n')

my_stack.print_stack()
```

```
<__main__.Node object at 0x7faa3019f

3
23
7
```

Stack : Pop Skenario Percobaan



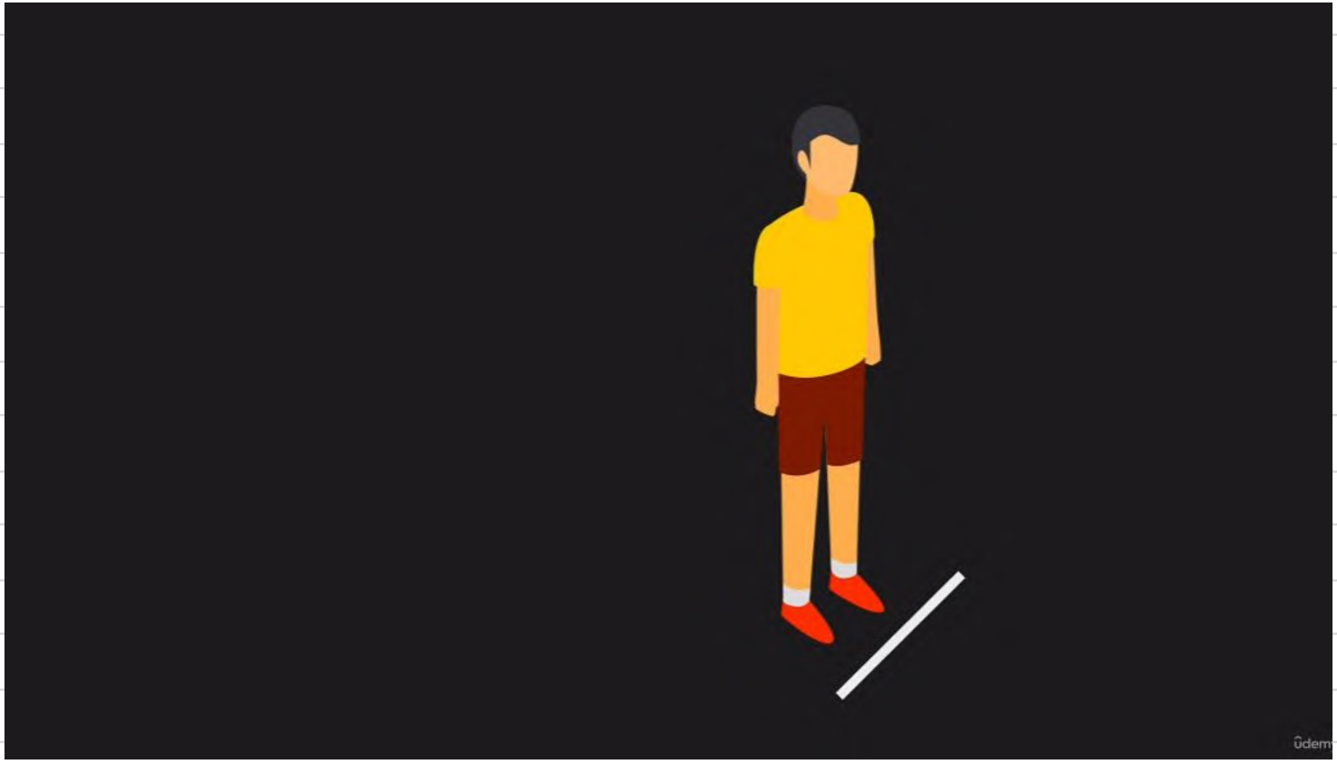
```
    def pop(self):
        if self.height == 0:
            return None
        temp = self.top
        self.top = self.top.next
        temp.next = None
        self.height -= 1
        return temp.value


my_stack = Stack(7)
my_stack.push(23)
my_stack.push(3)
my_stack.push(11)

print(my_stack.pop(), '\n')

my_stack.print_stack()
```

```
11

3
23
7
```

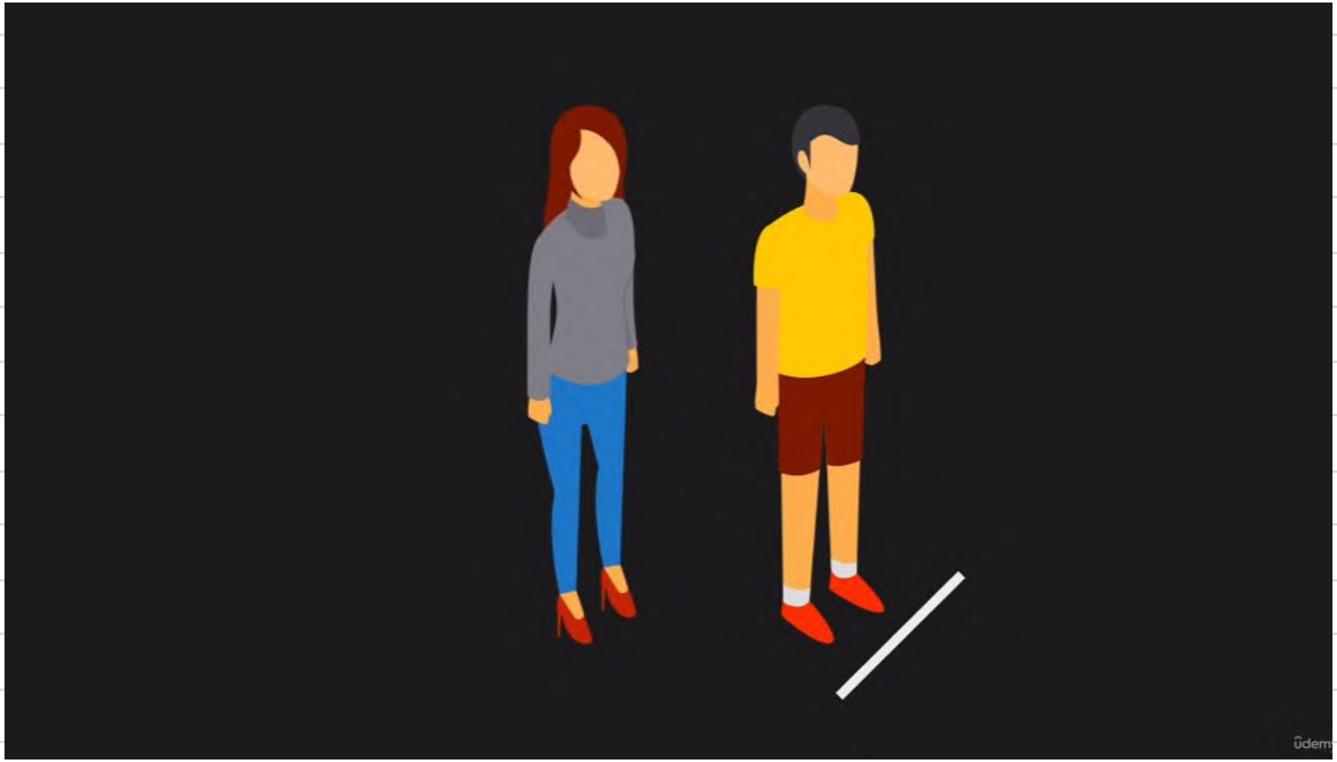
05

Queue Intro

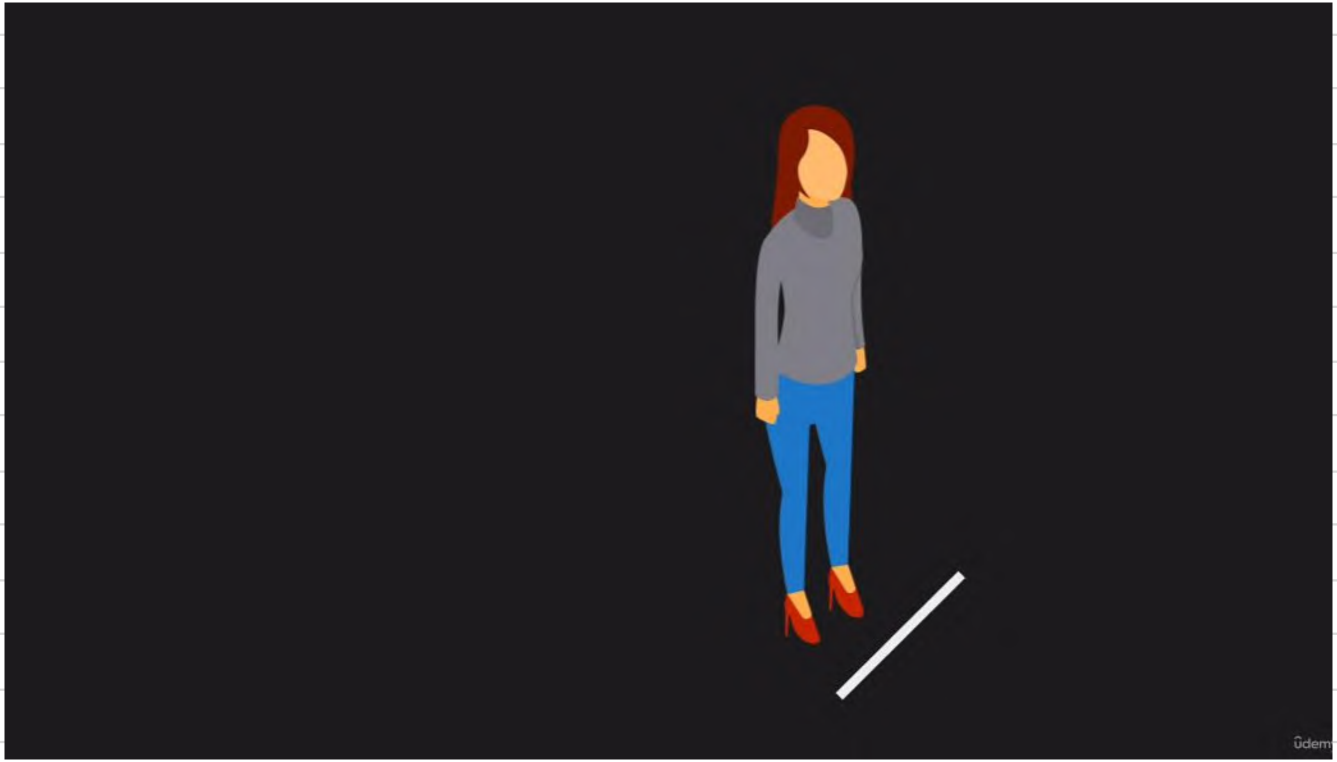
Queue



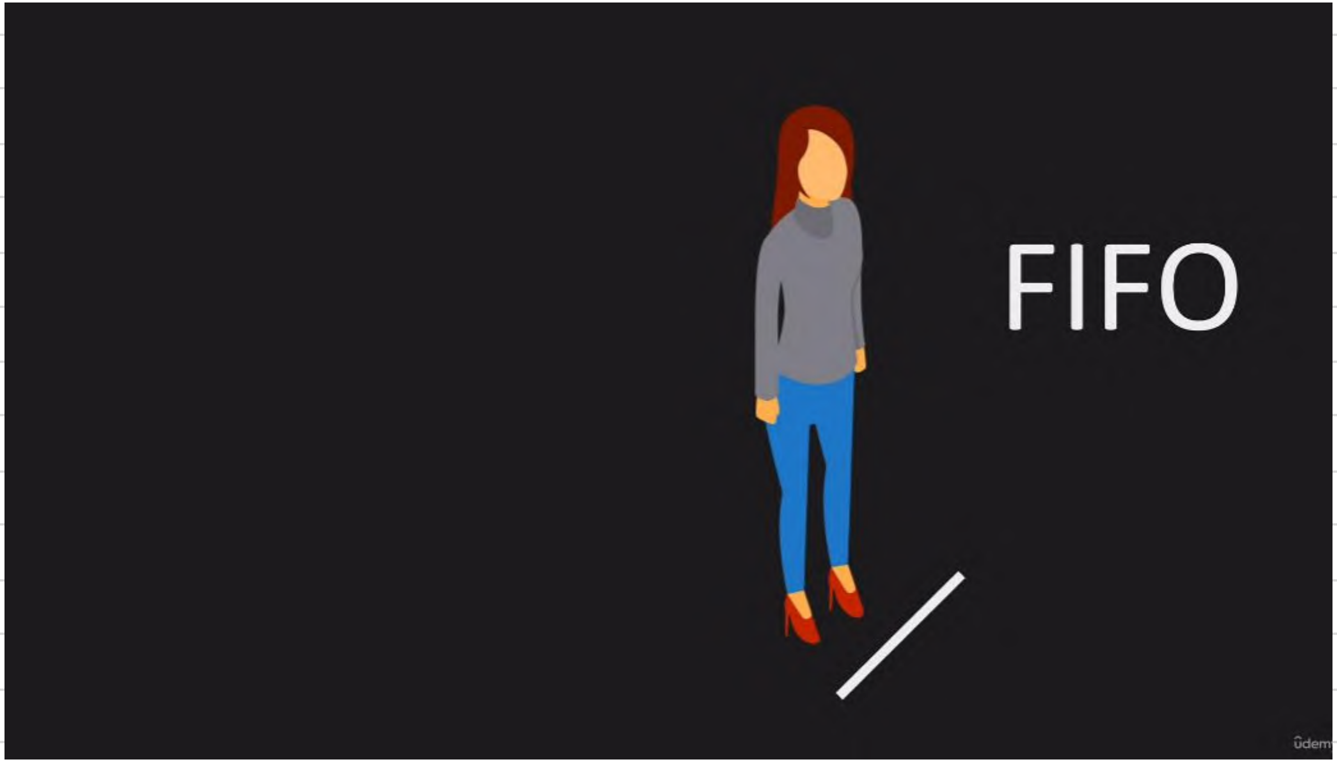
Queue



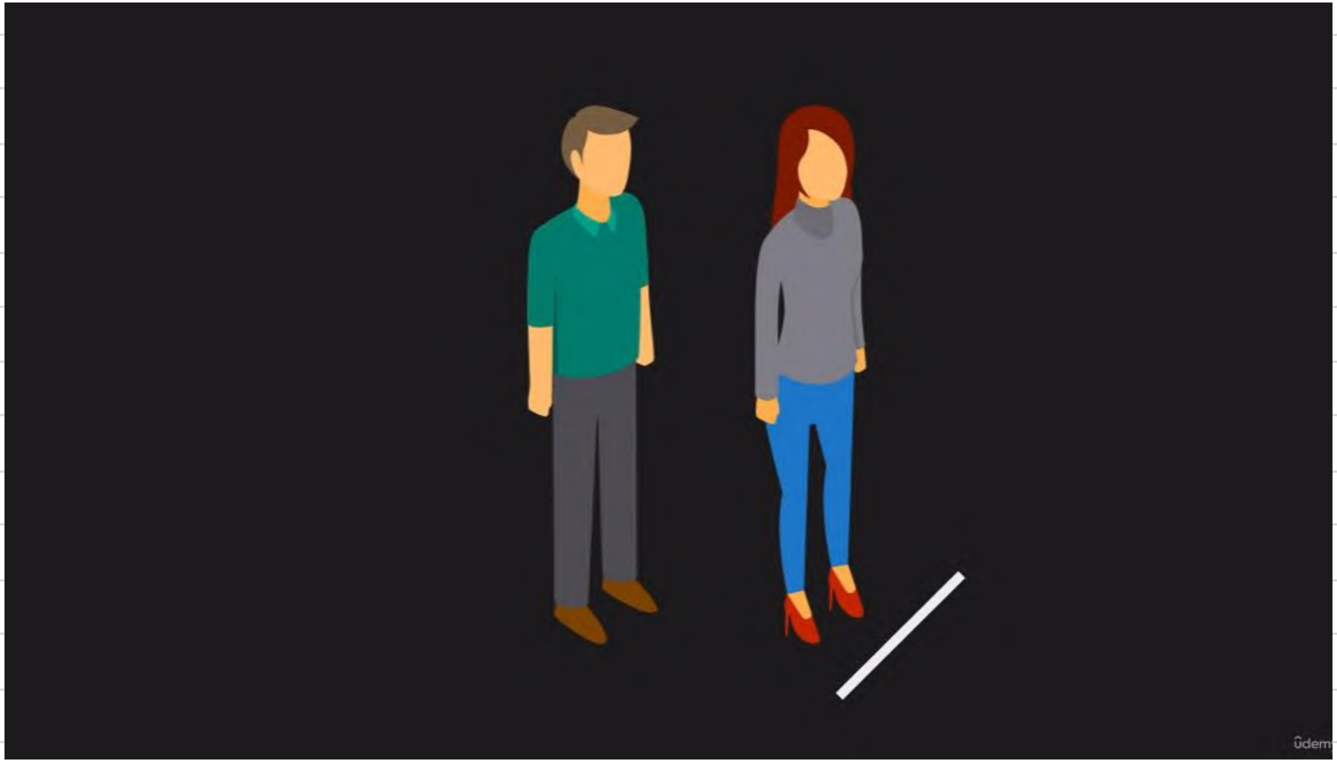
Queue



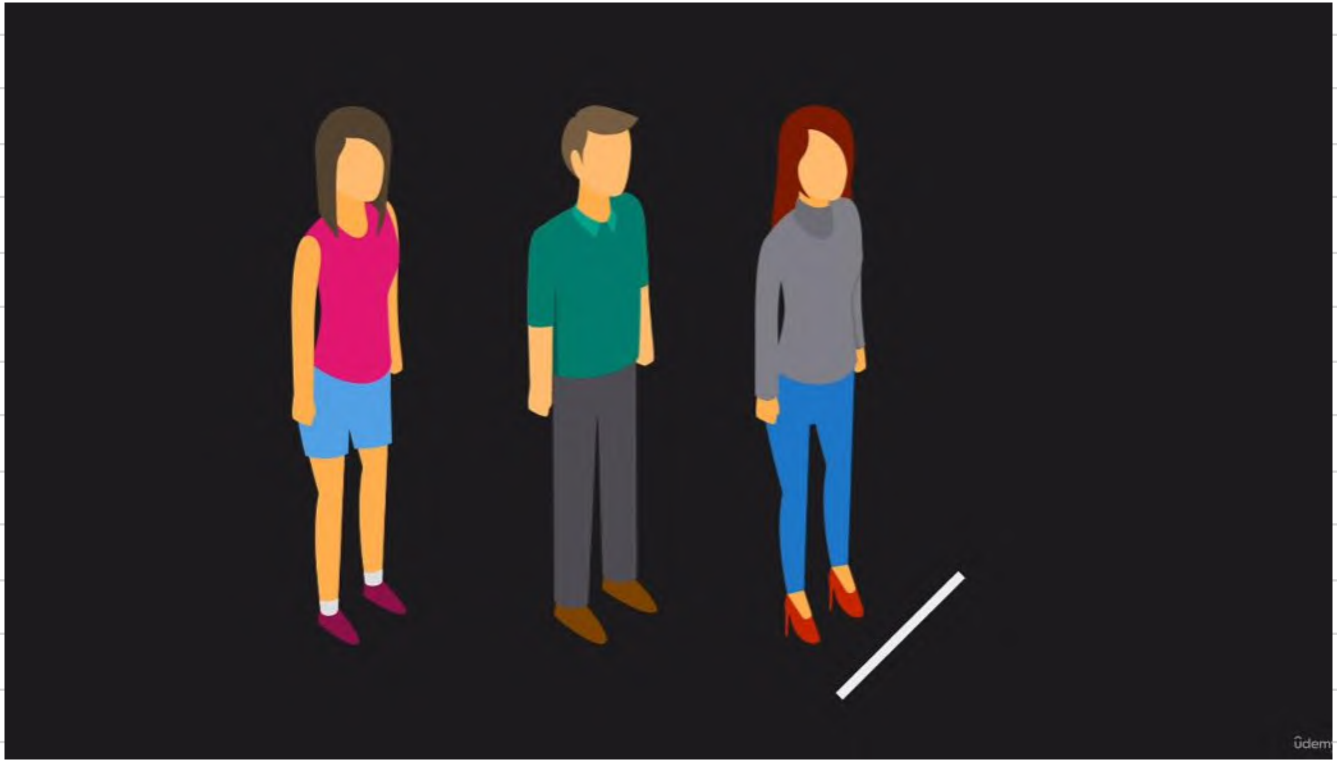
Queue



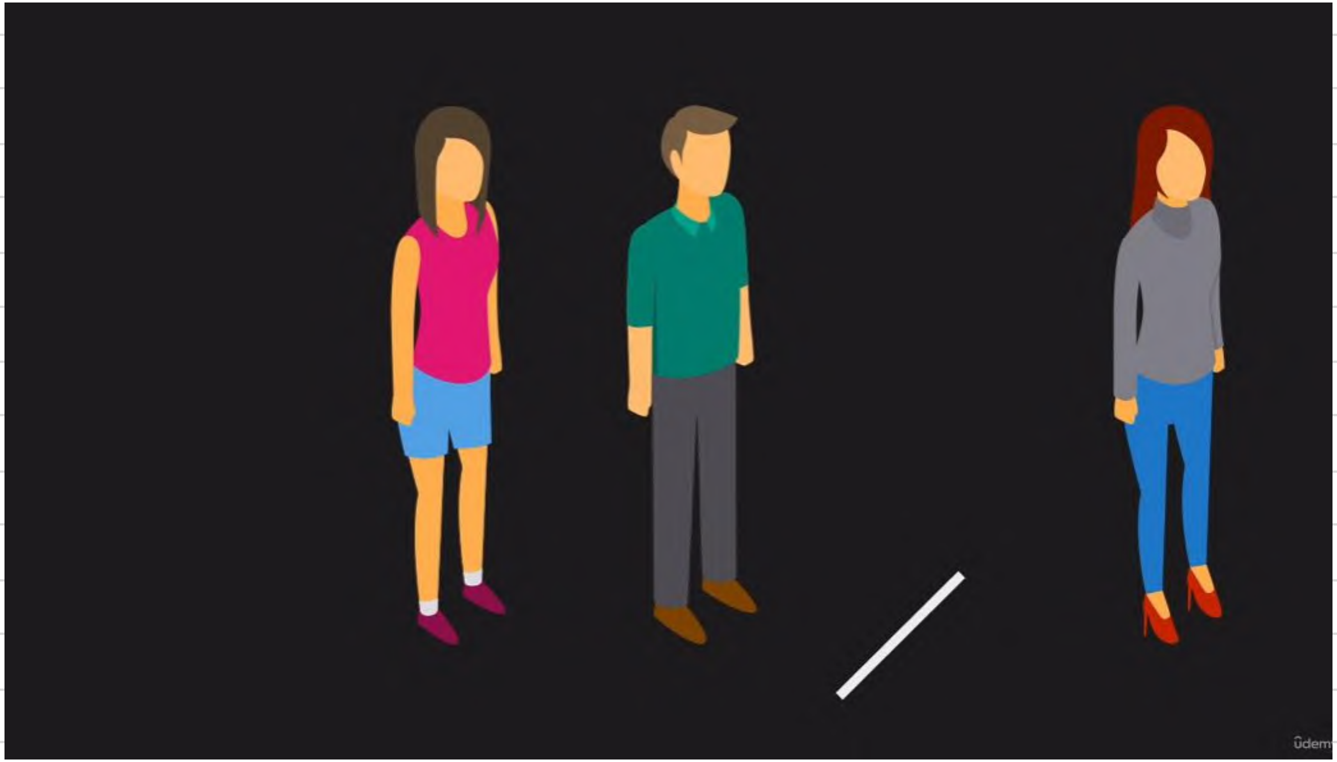
Queue



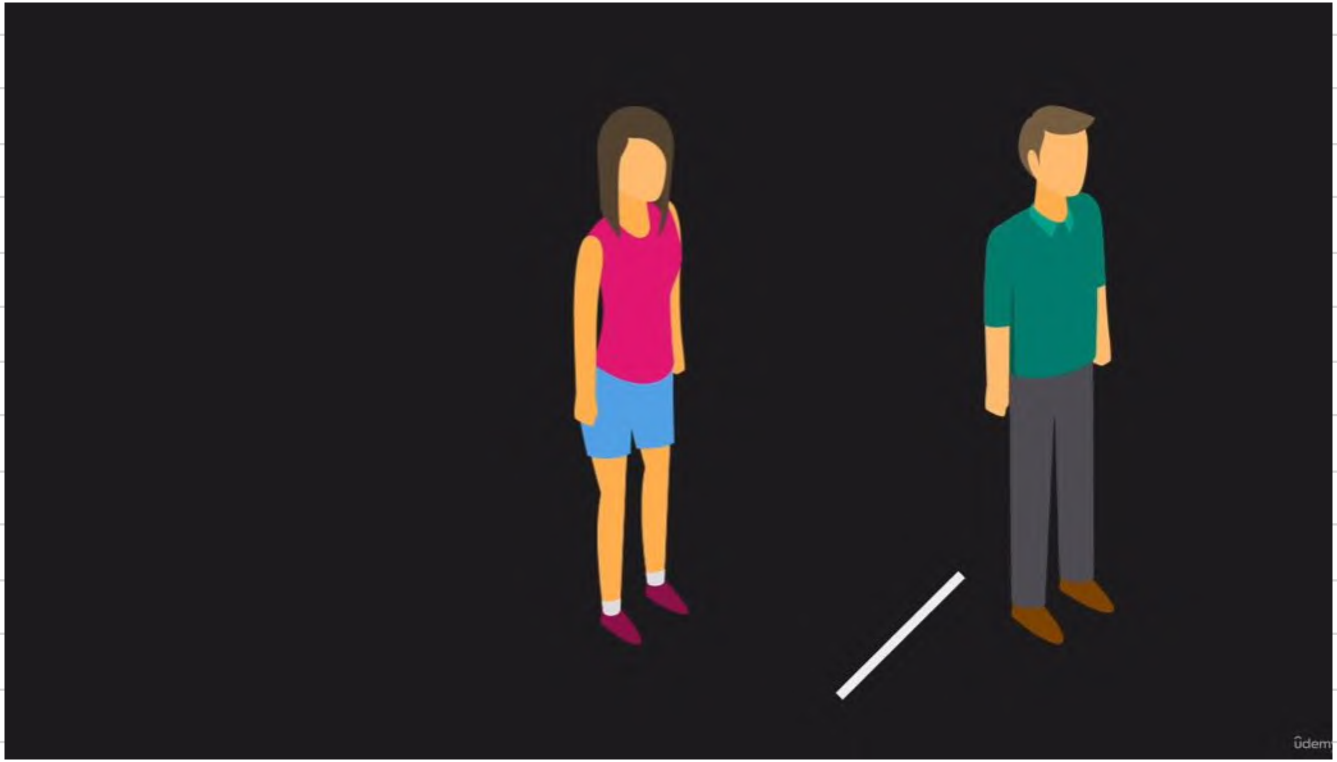
Queue



Queue

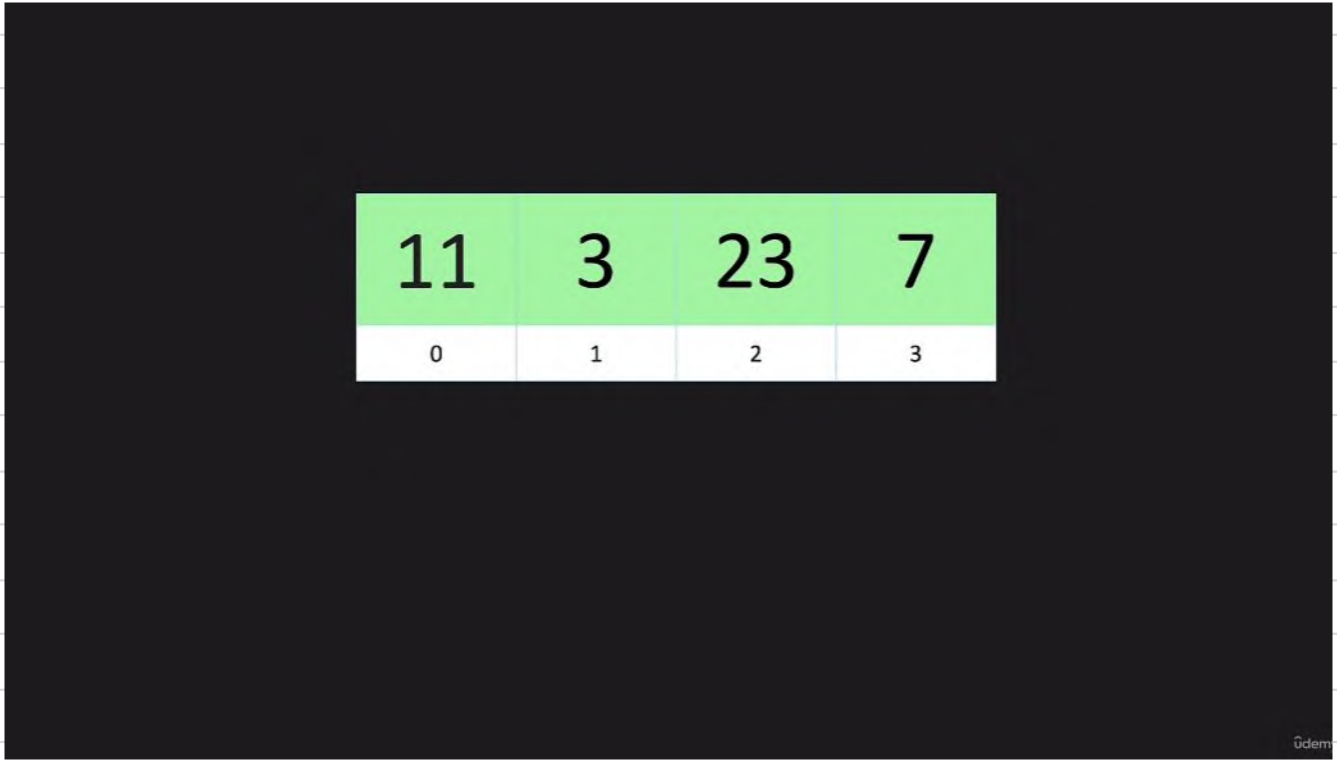


Queue

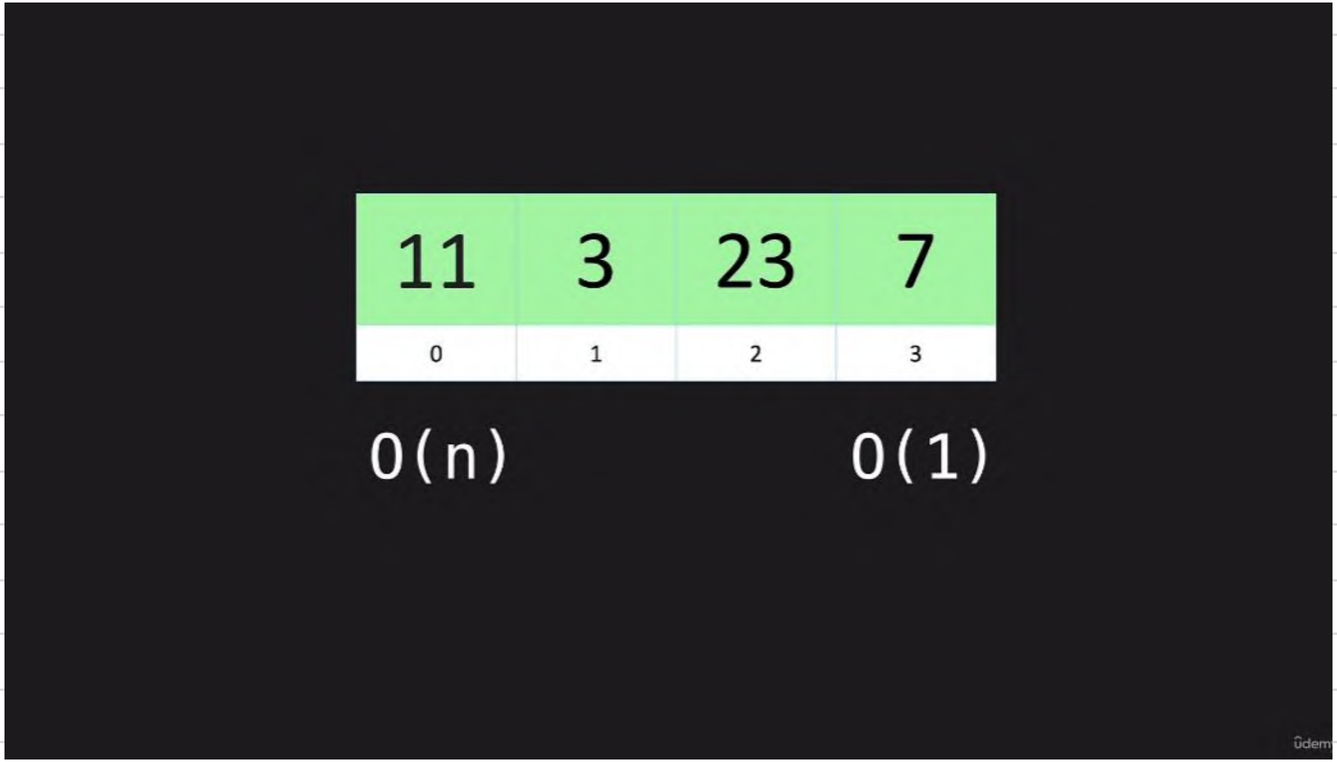


Queue

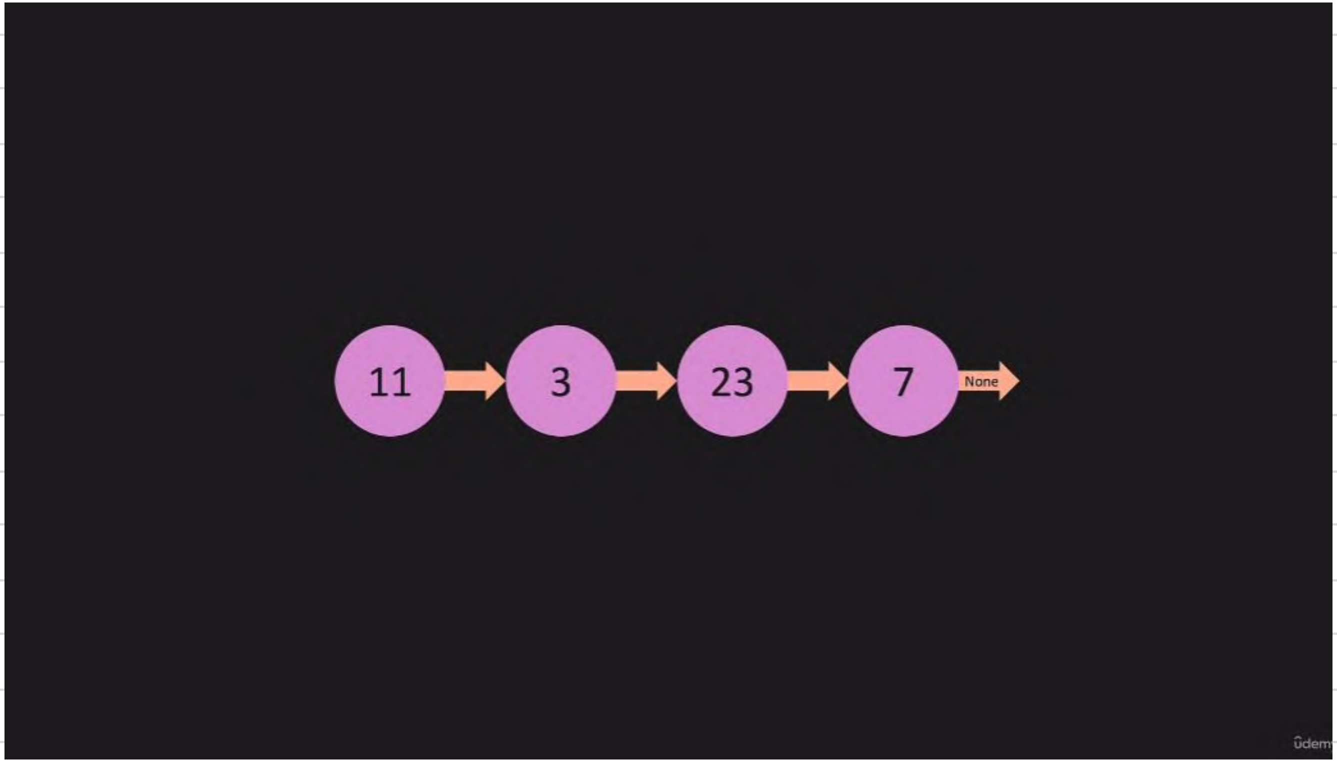
Queue



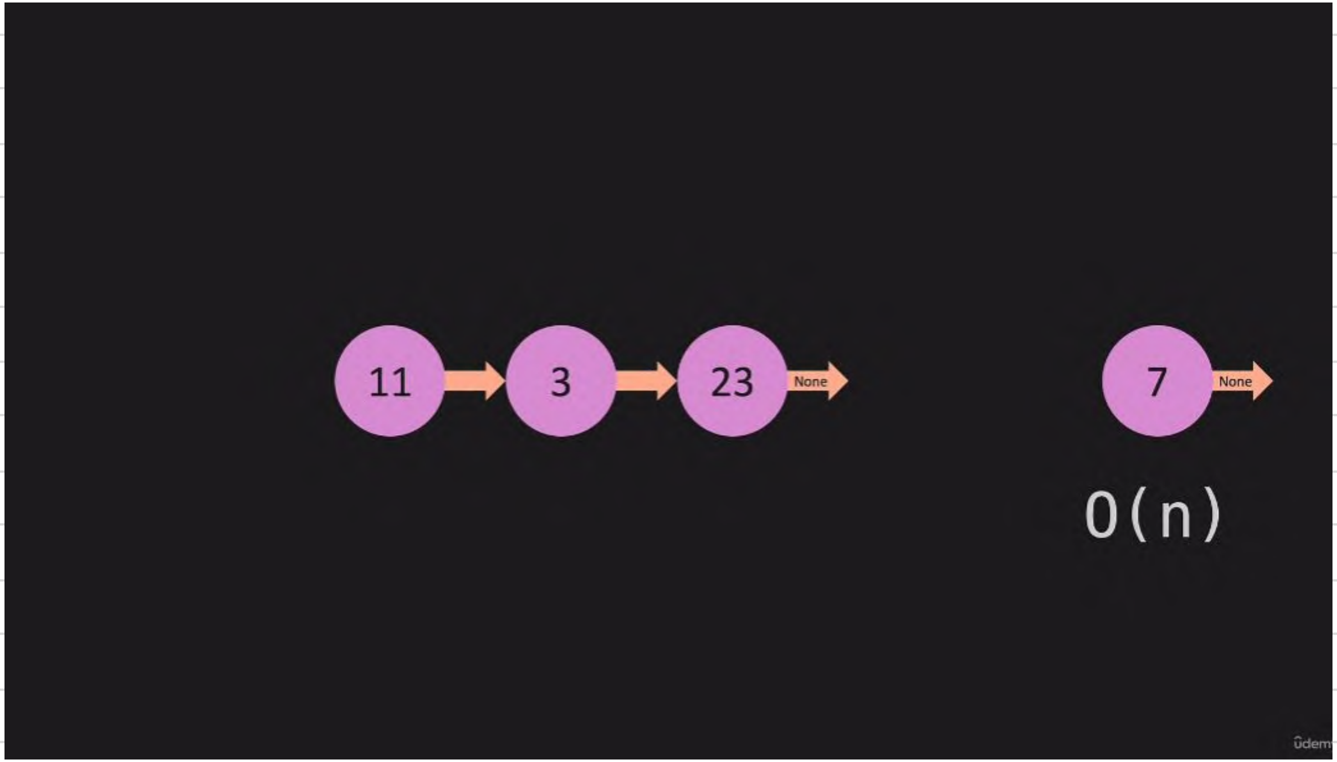
Queue



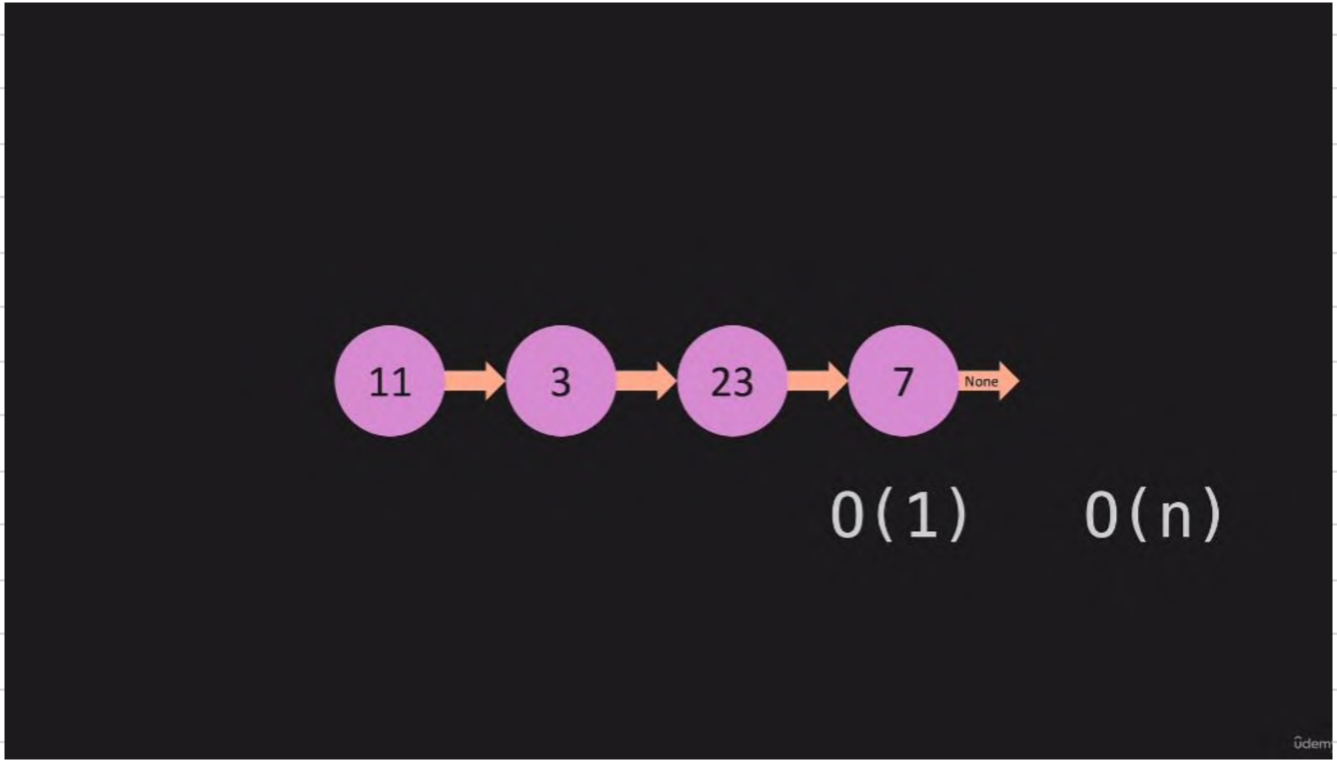
Queue



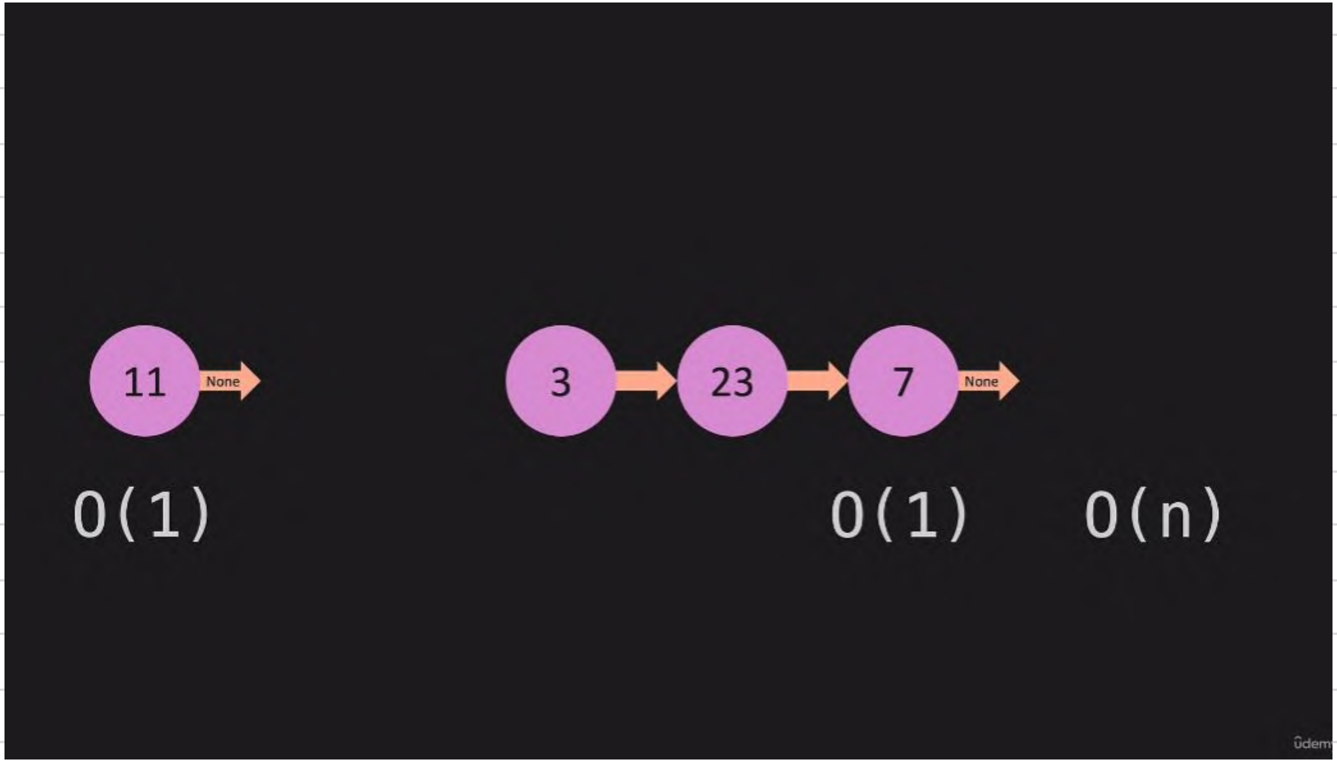
Queue



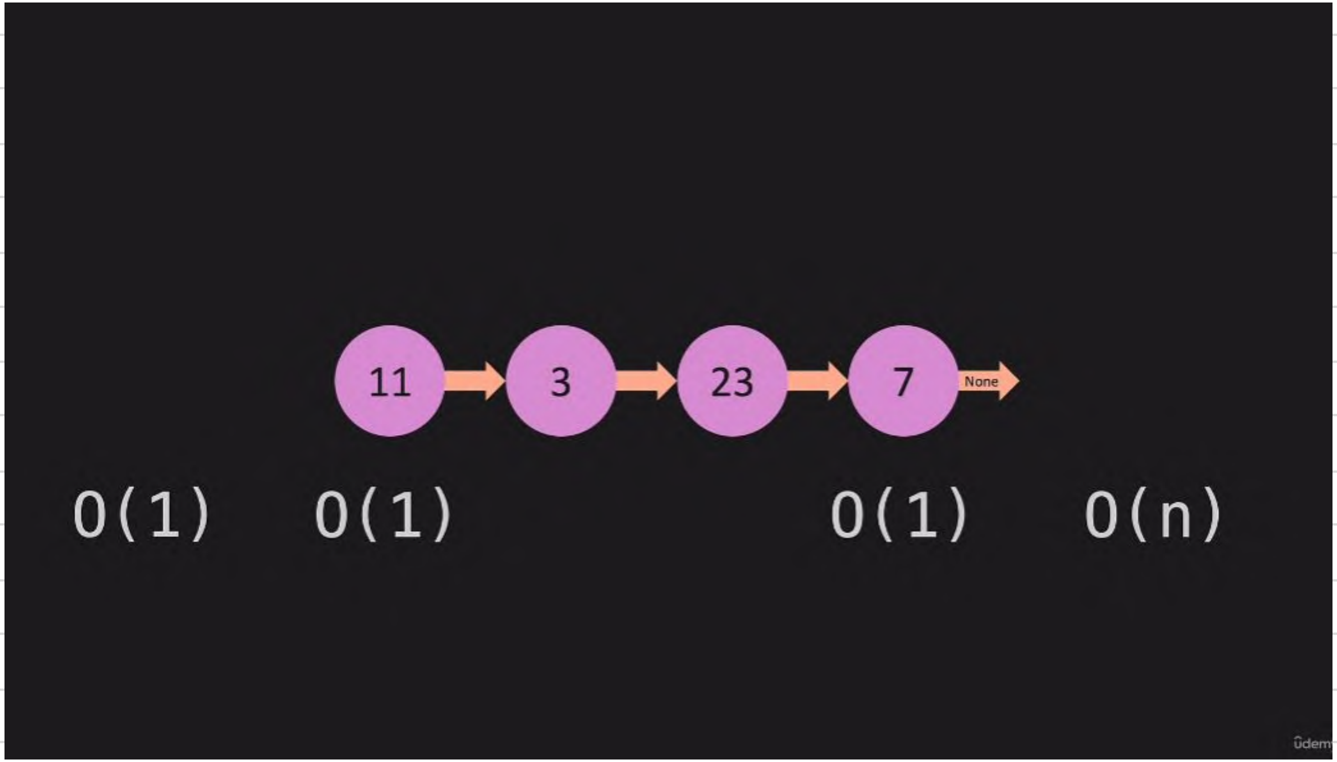
Queue



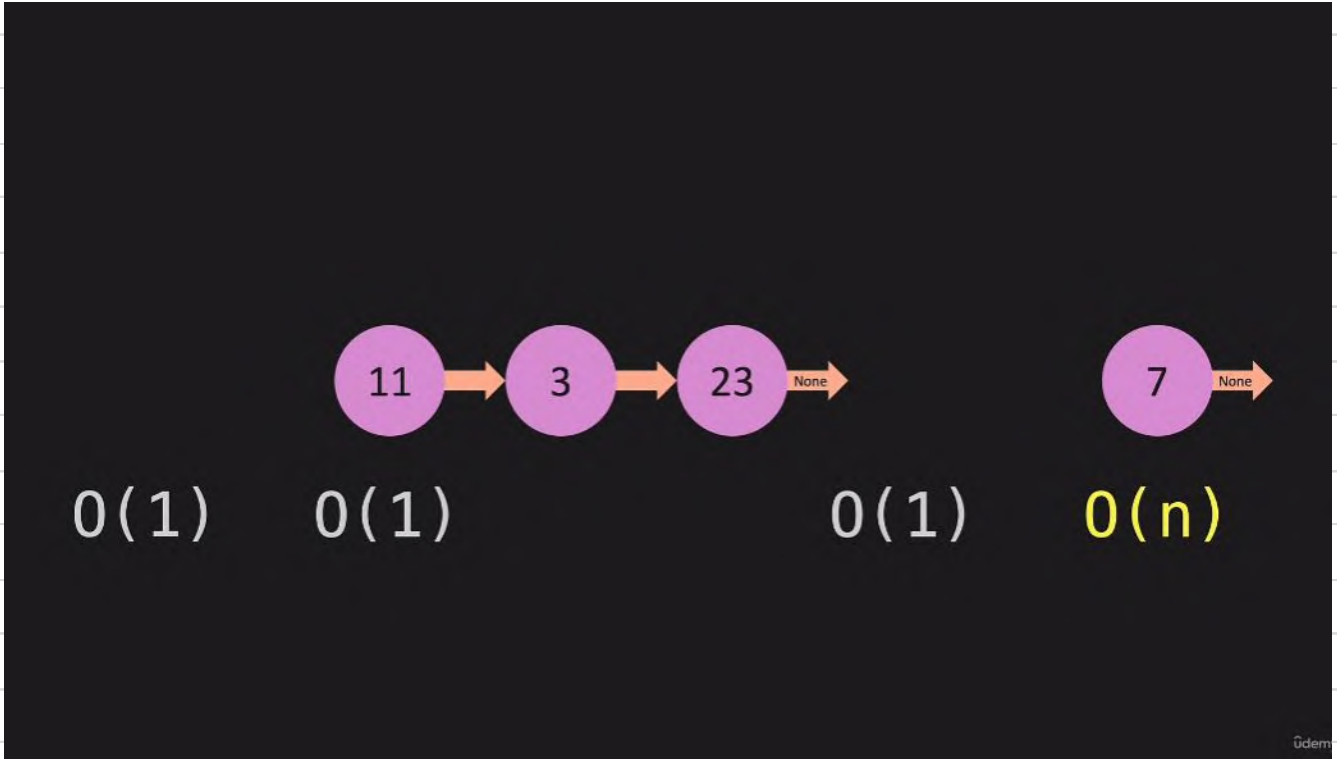
Queue



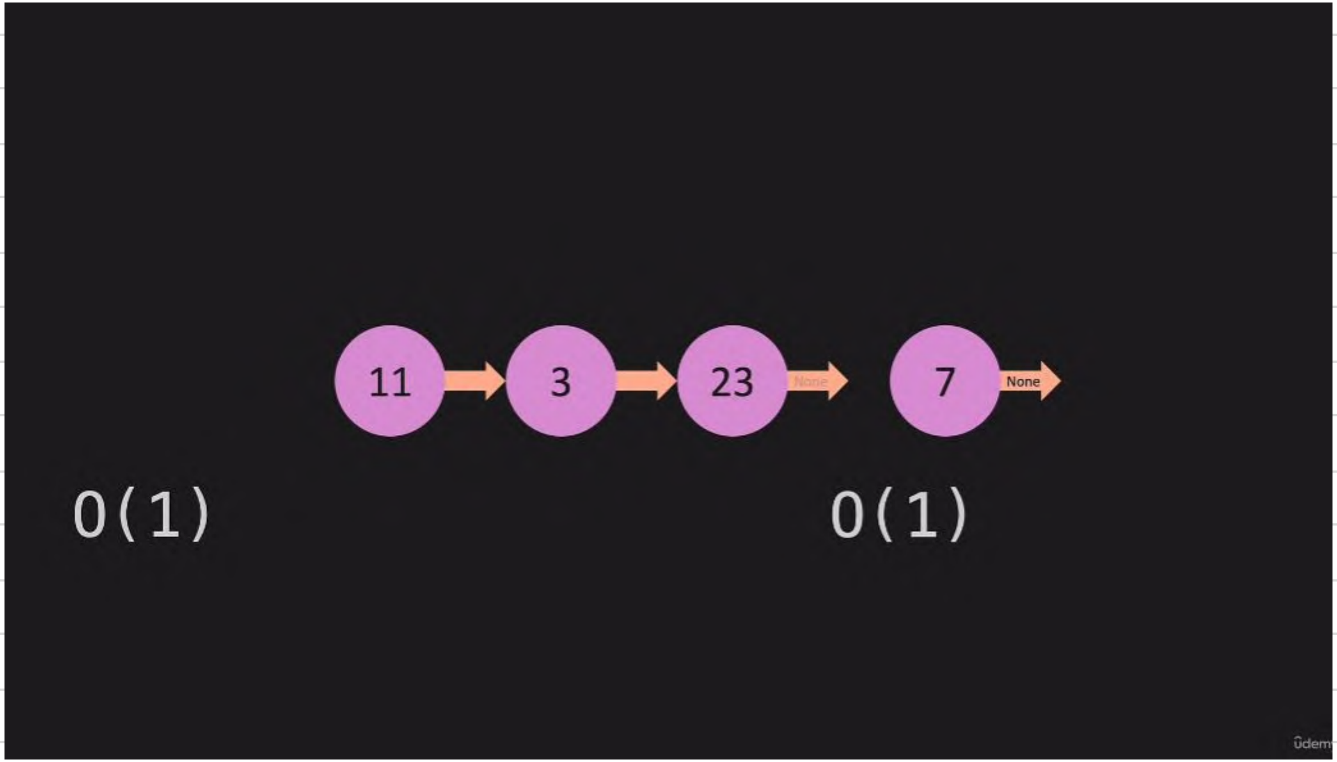
Queue



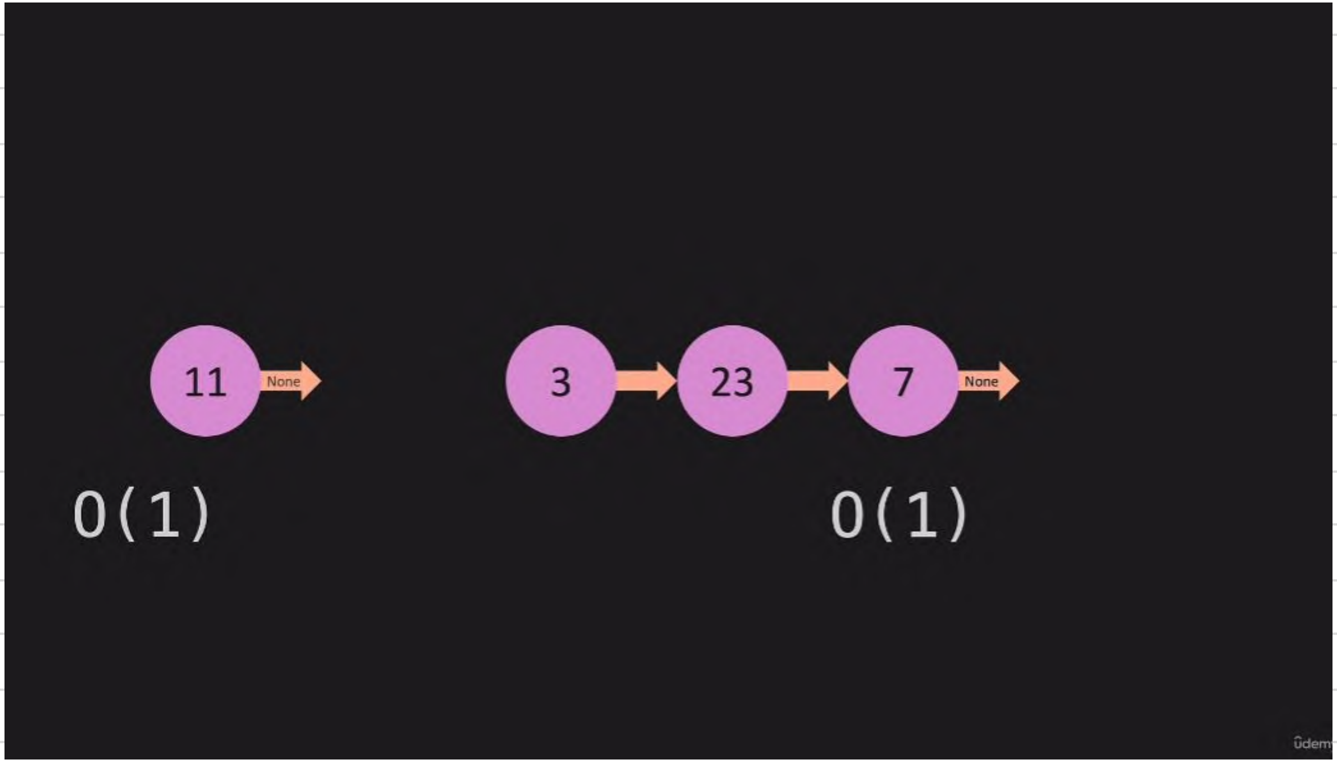
Queue



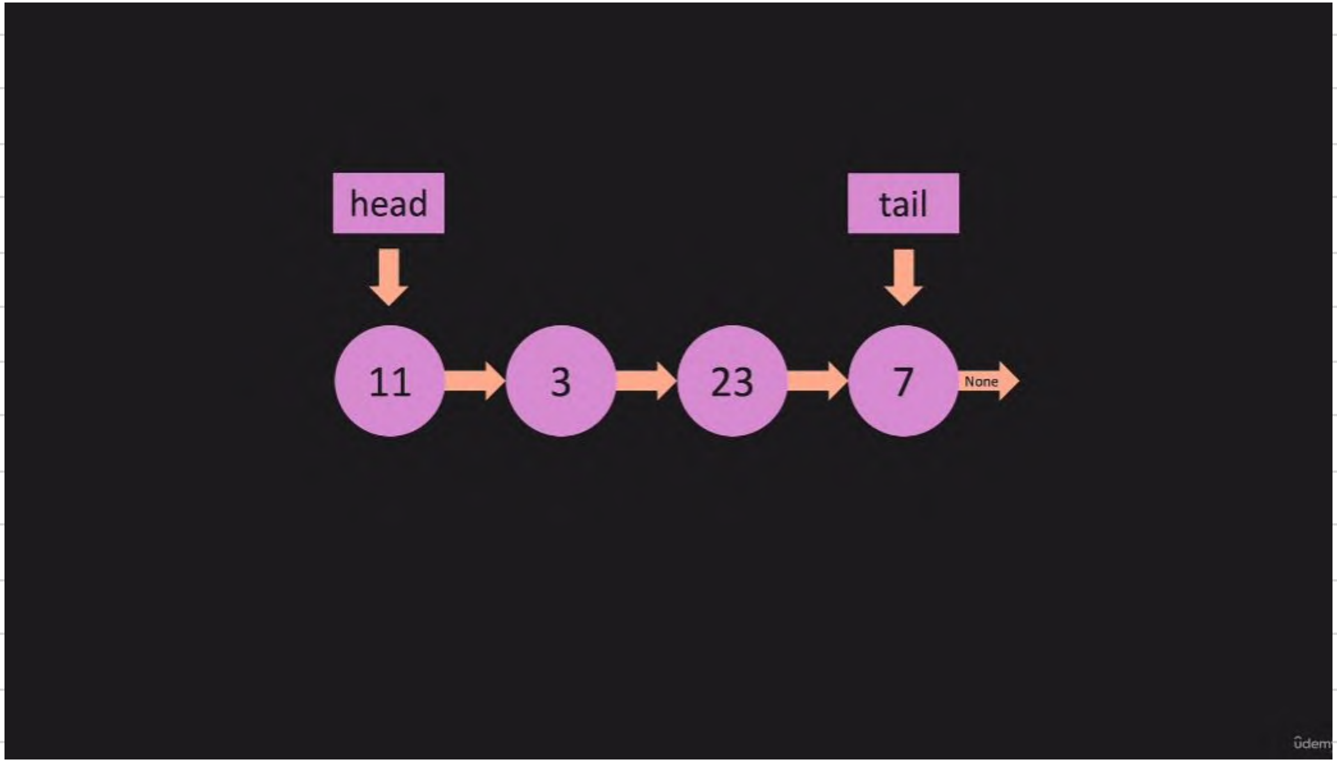
Queue



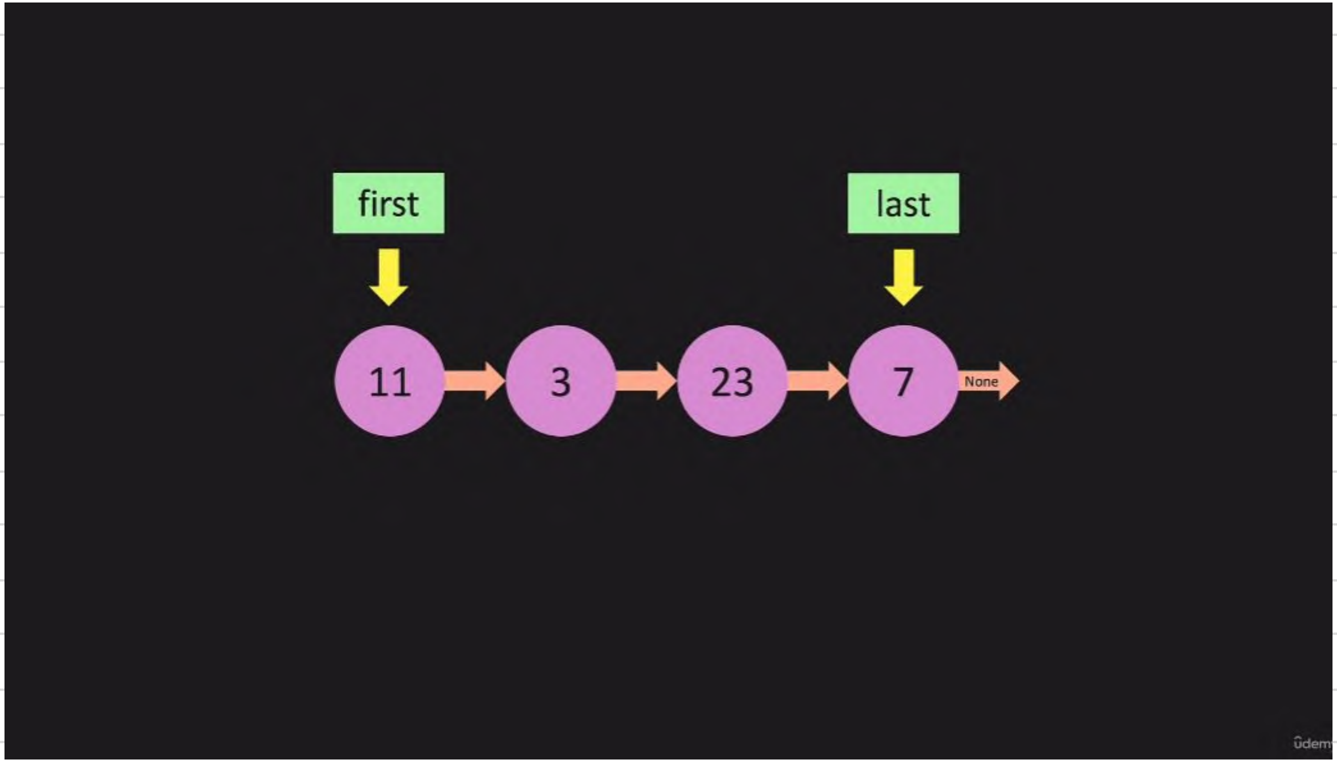
Queue



Queue



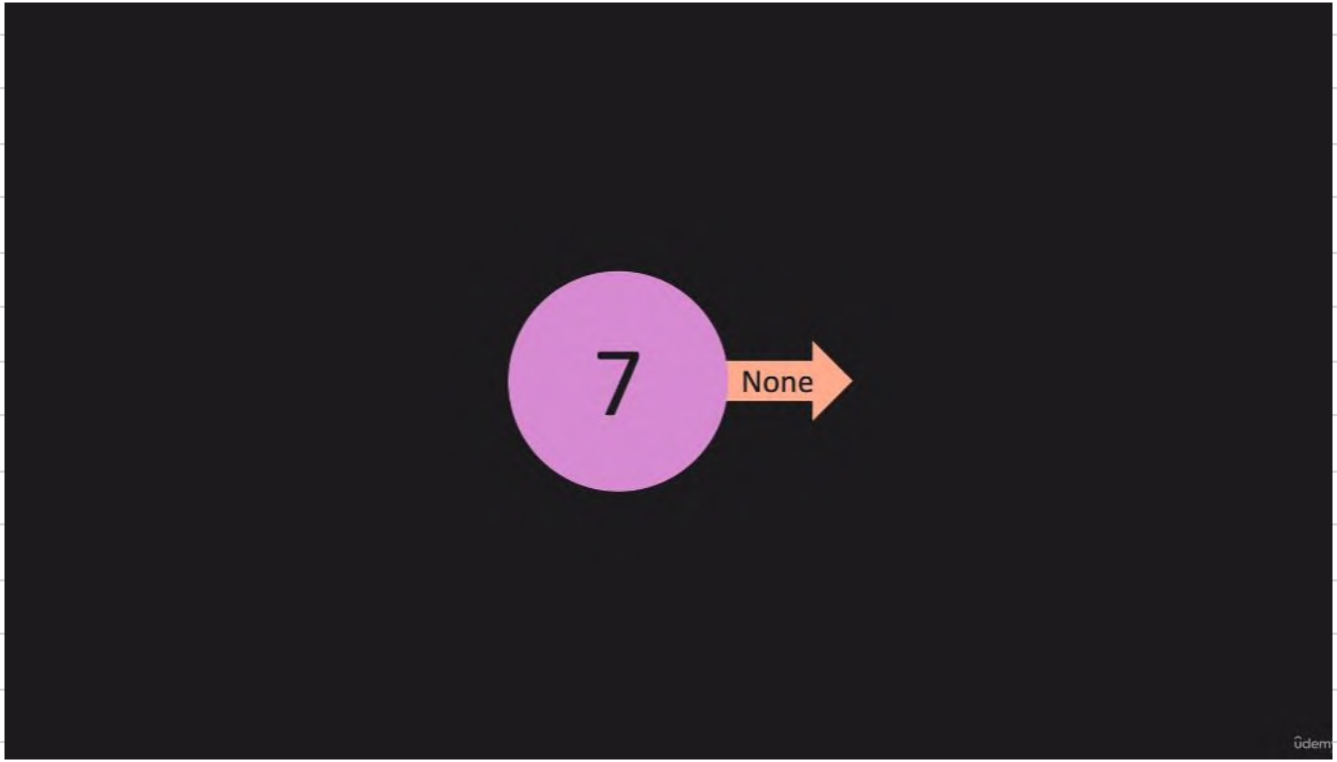
Queue



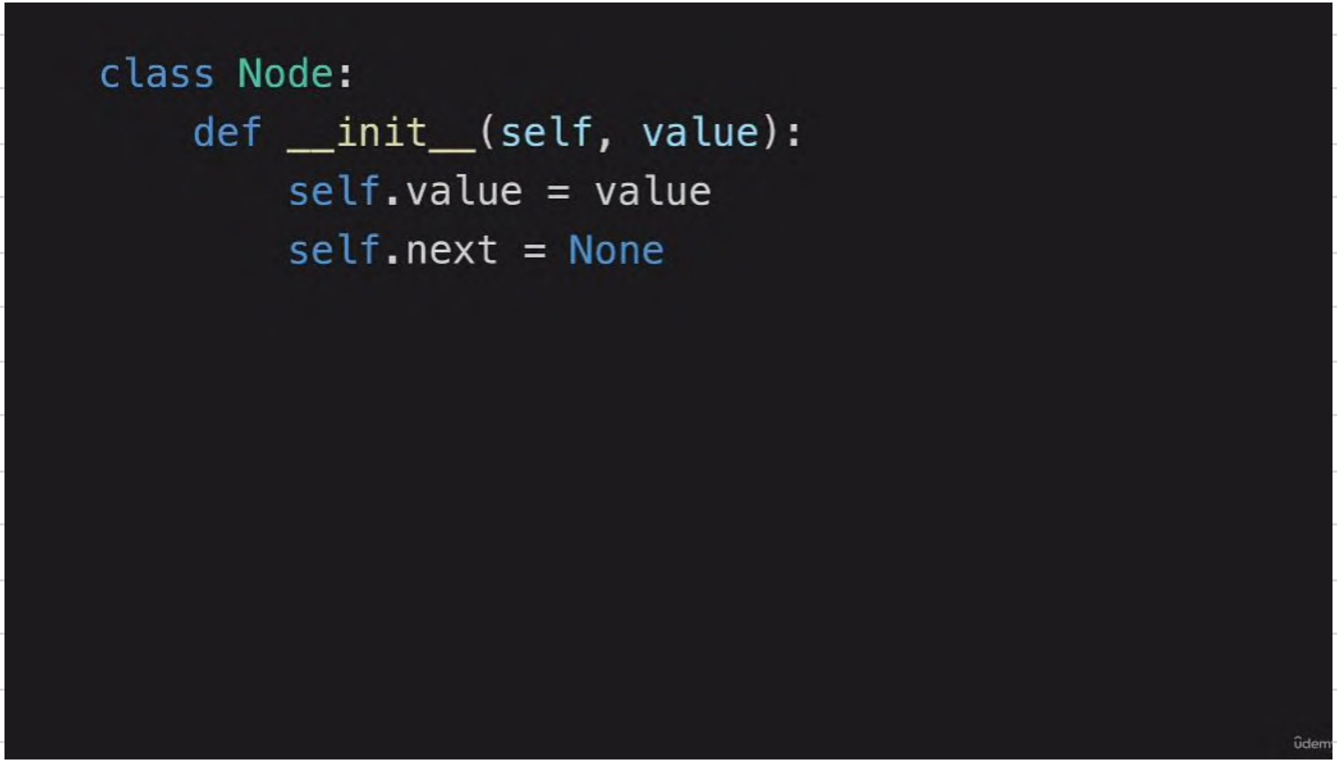
06

Queue Constructor

Queue : Constructor

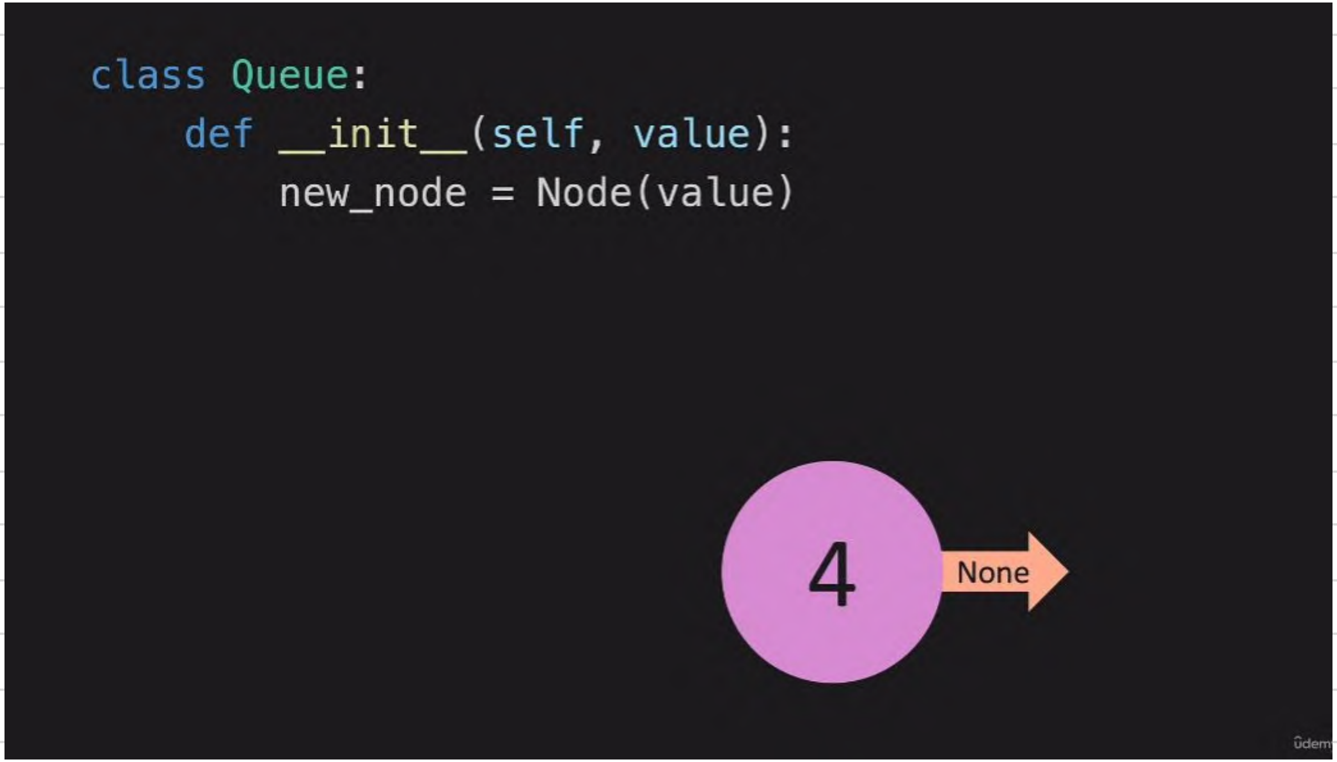


Queue : Constructor



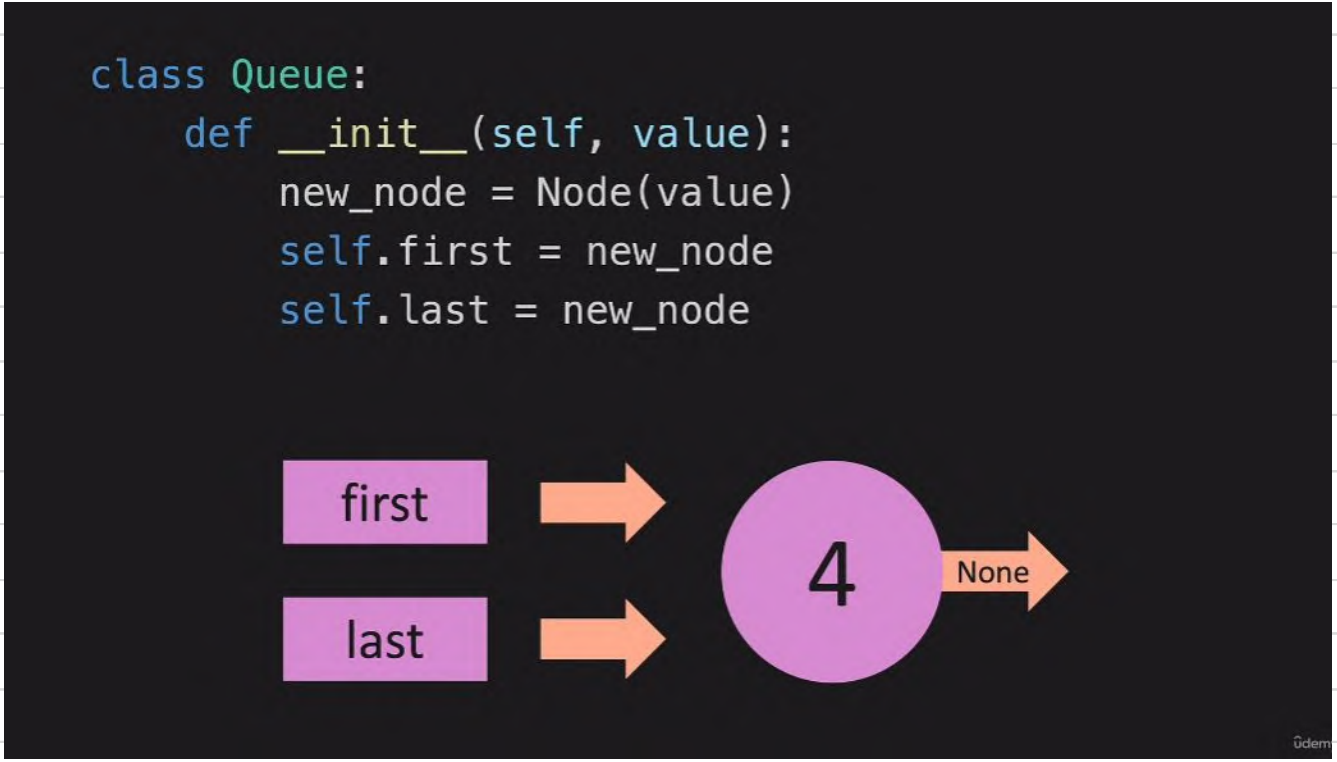
```
class Node:
    def __init__(self, value):
        self.value = value
        self.next = None
```

Queue : Constructor



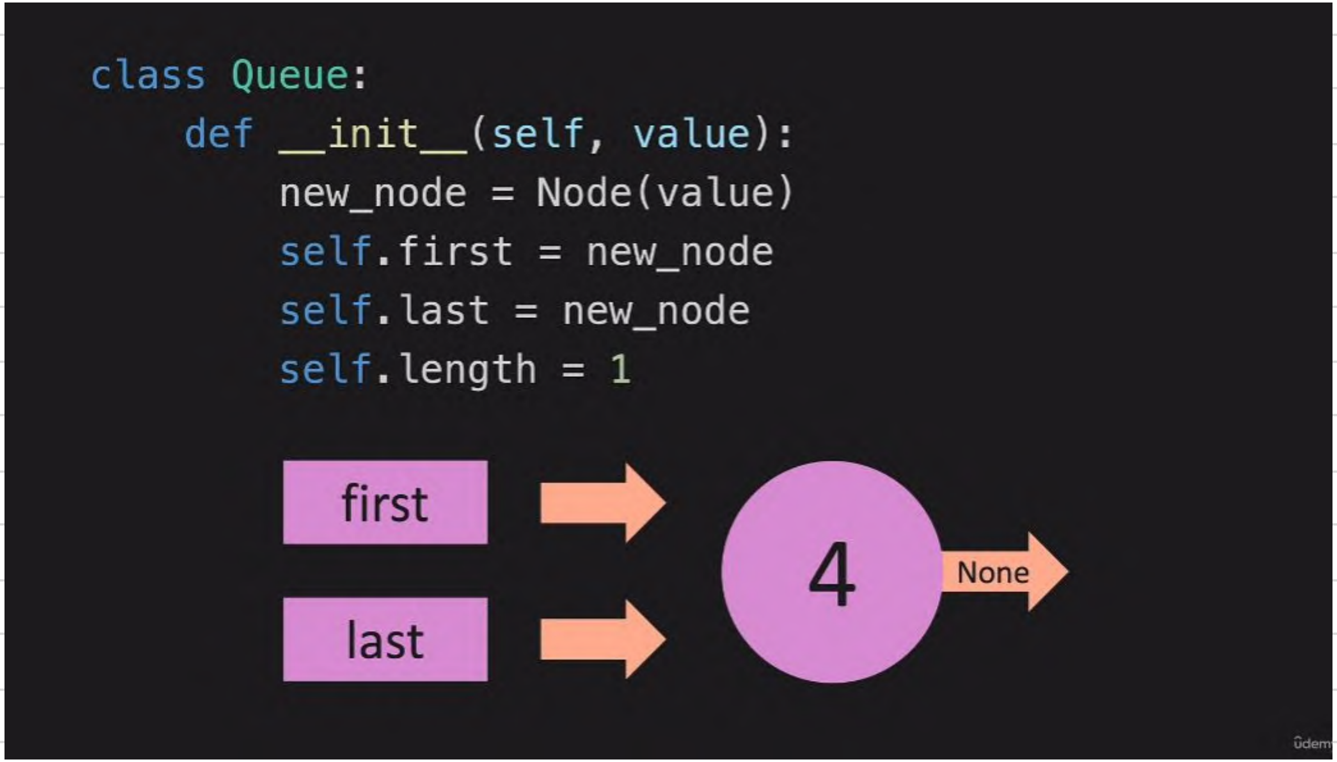
```
class Queue:
    def __init__(self, value):
        new_node = Node(value)
```

Queue : Constructor



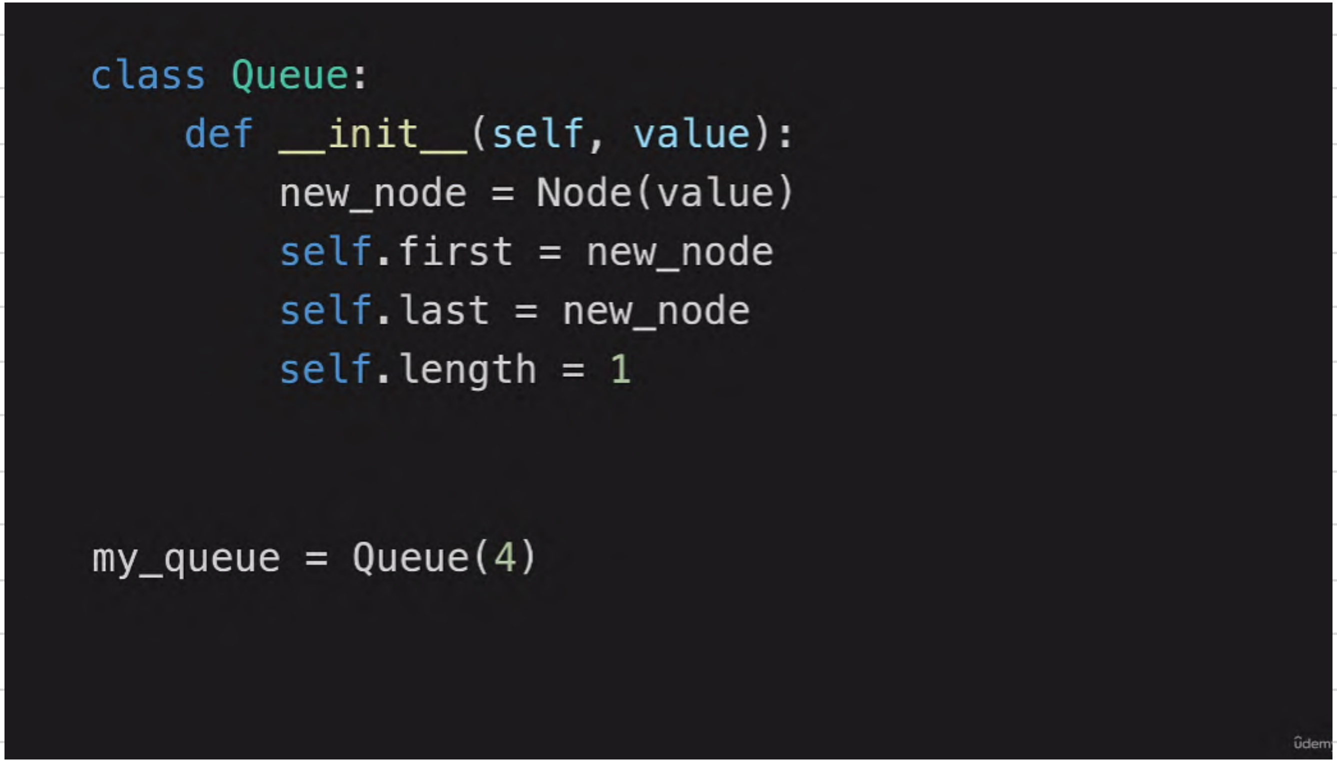
```
class Queue:
    def __init__(self, value):
        new_node = Node(value)
        self.first = new_node
        self.last = new_node
```

Queue : Constructor



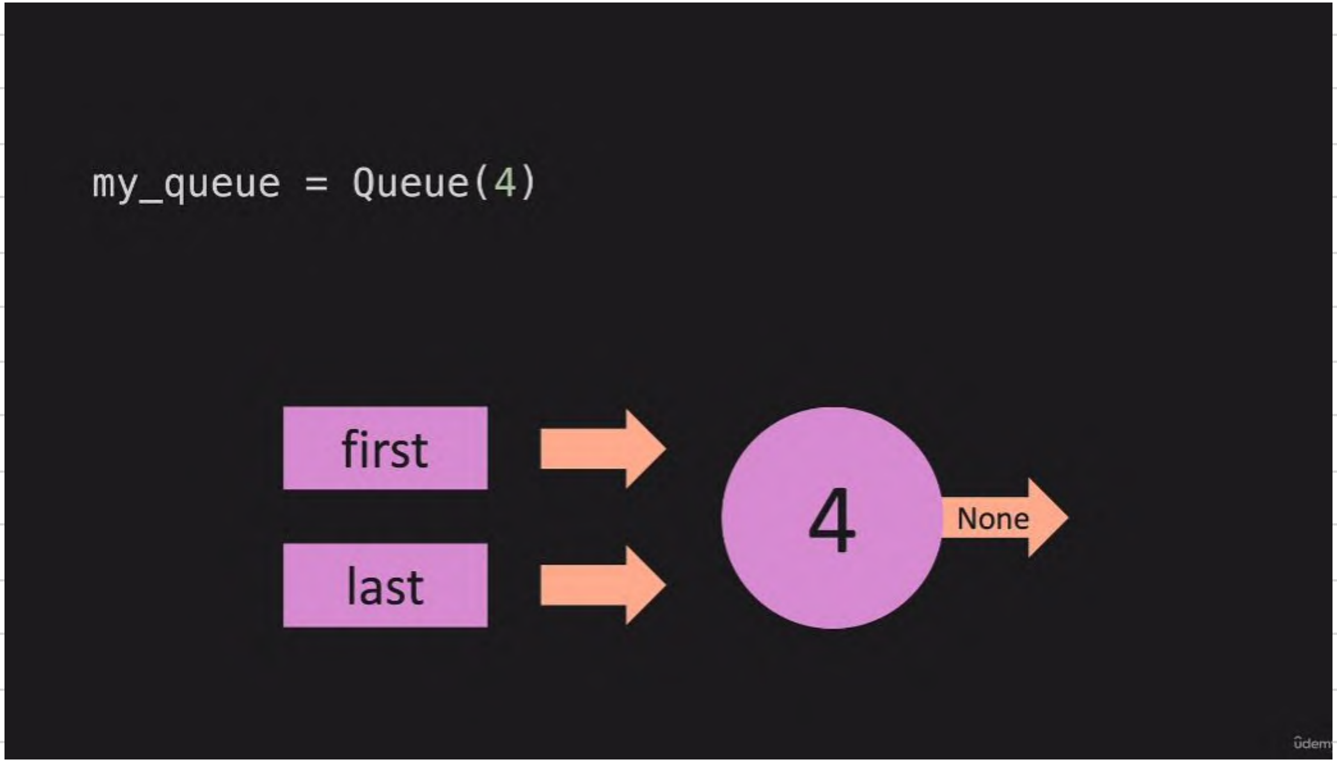
```
class Queue:
    def __init__(self, value):
        new_node = Node(value)
        self.first = new_node
        self.last = new_node
        self.length = 1
```

Queue : Constructor



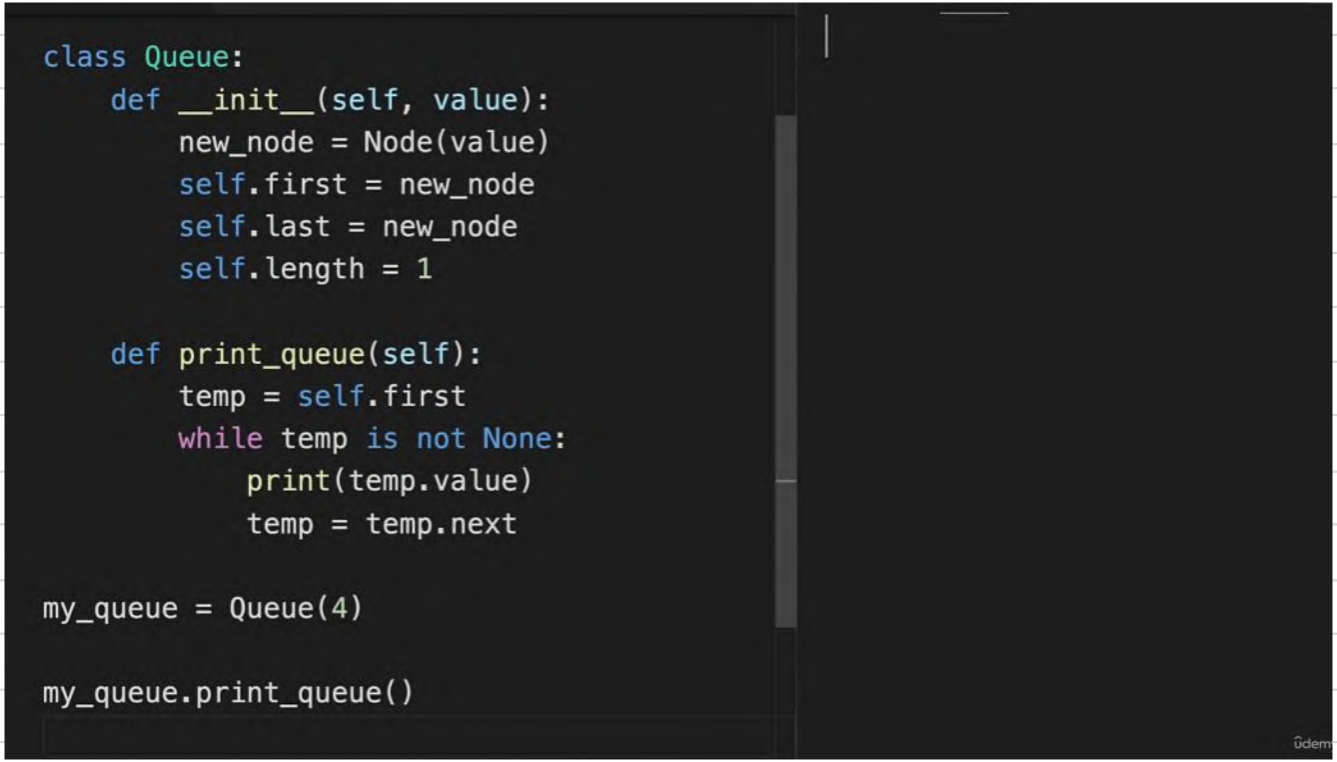
```python
class Queue:
    def __init__(self, value):
        new_node = Node(value)
        self.first = new_node
        self.last = new_node
        self.length = 1


my_queue = Queue(4)
```

Queue : Constructor



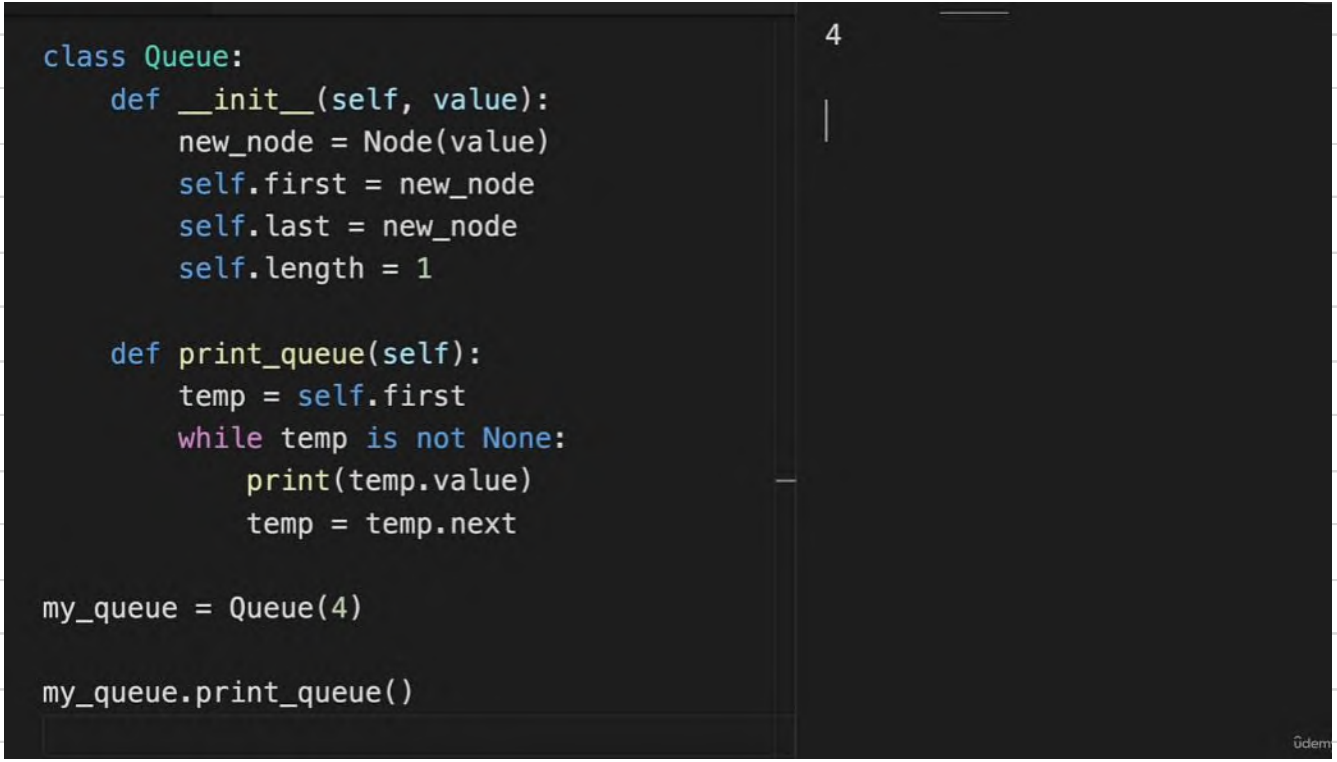
```
my_queue = Queue(4)
```

Queue : Constructor



```python
class Queue:
    def __init__(self, value):
        new_node = Node(value)
        self.first = new_node
        self.last = new_node
        self.length = 1

    def print_queue(self):
        temp = self.first
        while temp is not None:
            print(temp.value)
            temp = temp.next

my_queue = Queue(4)

my_queue.print_queue()
```

Queue : Constructor



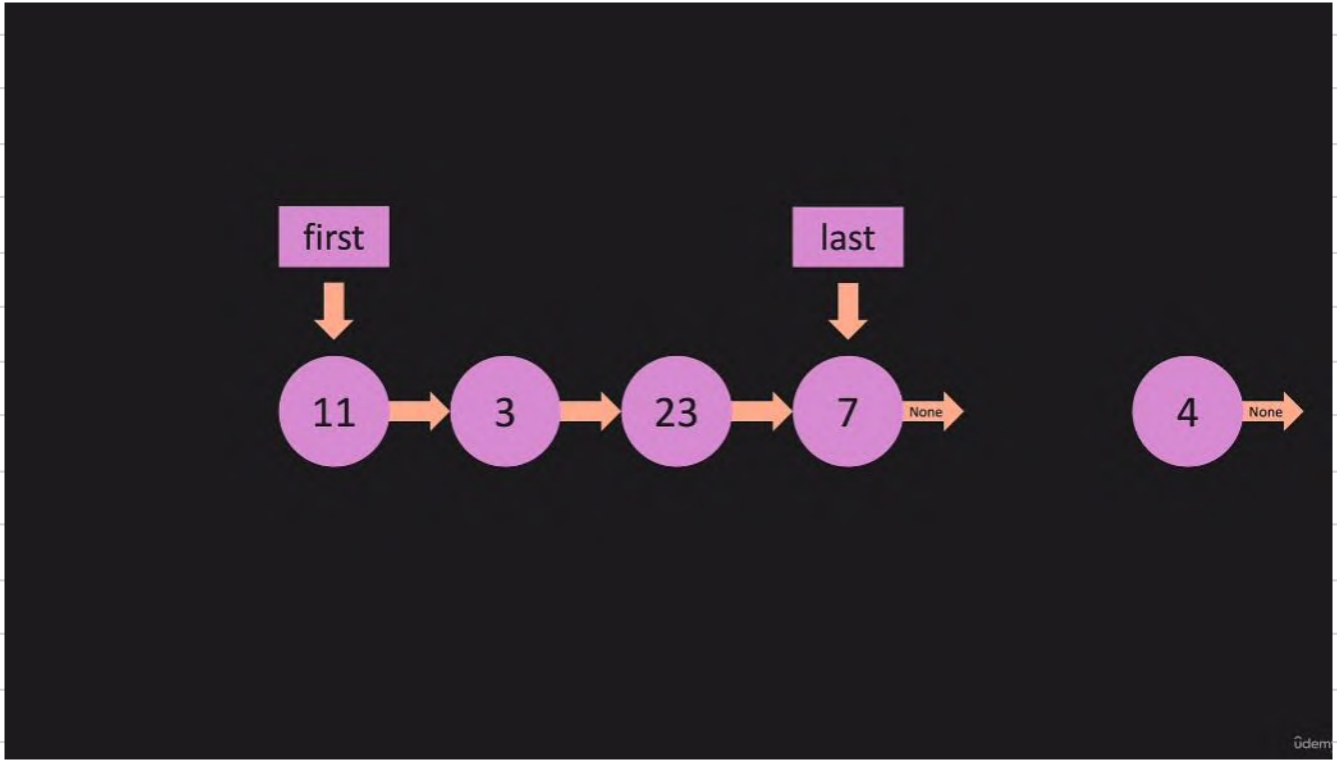
```
class Queue:
    def __init__(self, value):
        new_node = Node(value)
        self.first = new_node
        self.last = new_node
        self.length = 1

    def print_queue(self):
        temp = self.first
        while temp is not None:
            print(temp.value)
            temp = temp.next

my_queue = Queue(4)

my_queue.print_queue()
```

```
4
```

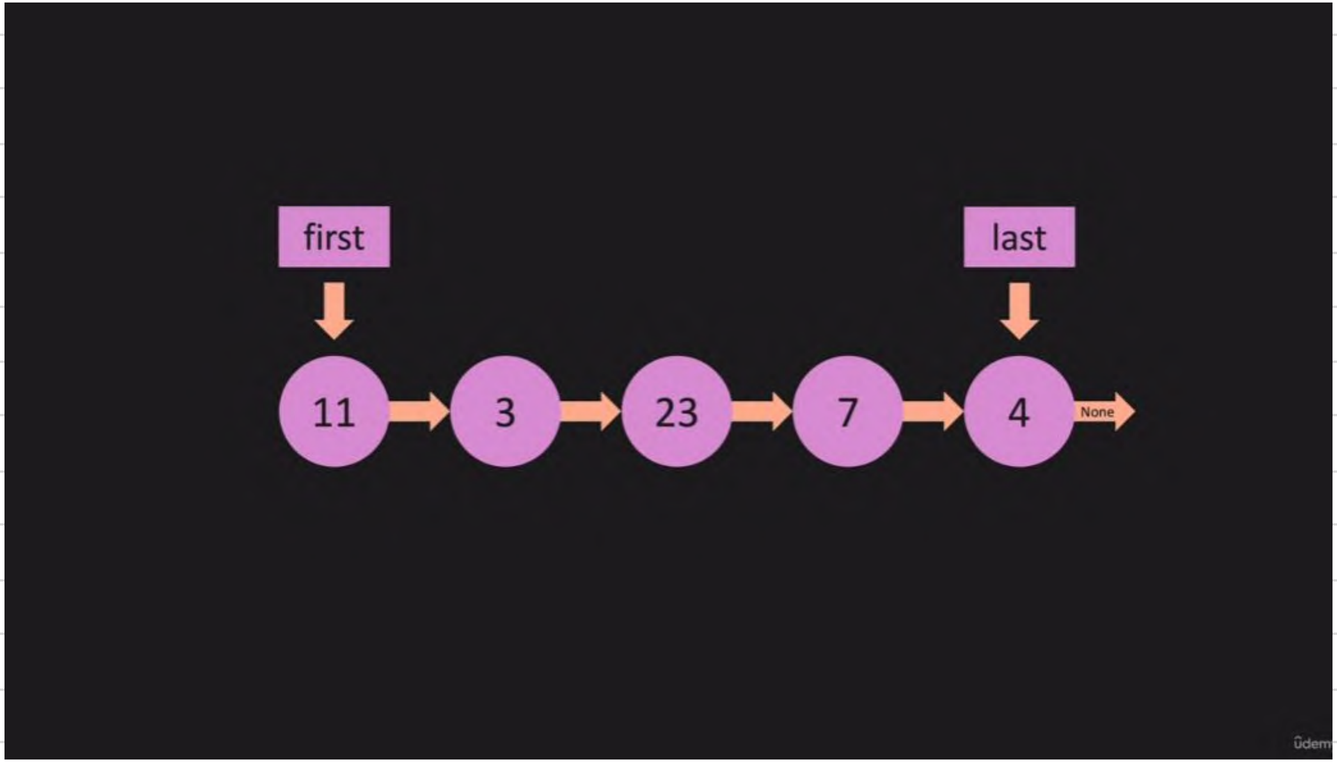
07

Queue Enqueue

Queue : Enqueue

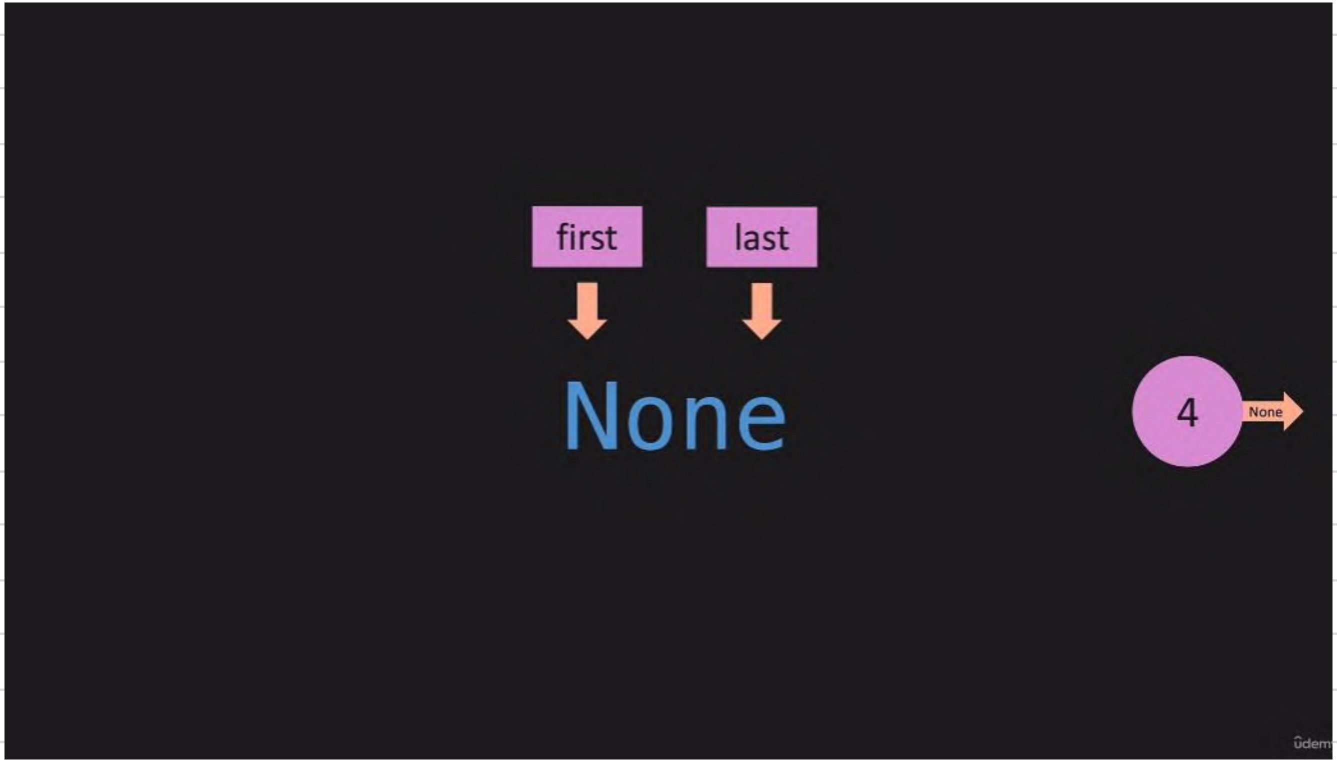


Queue : Enqueue



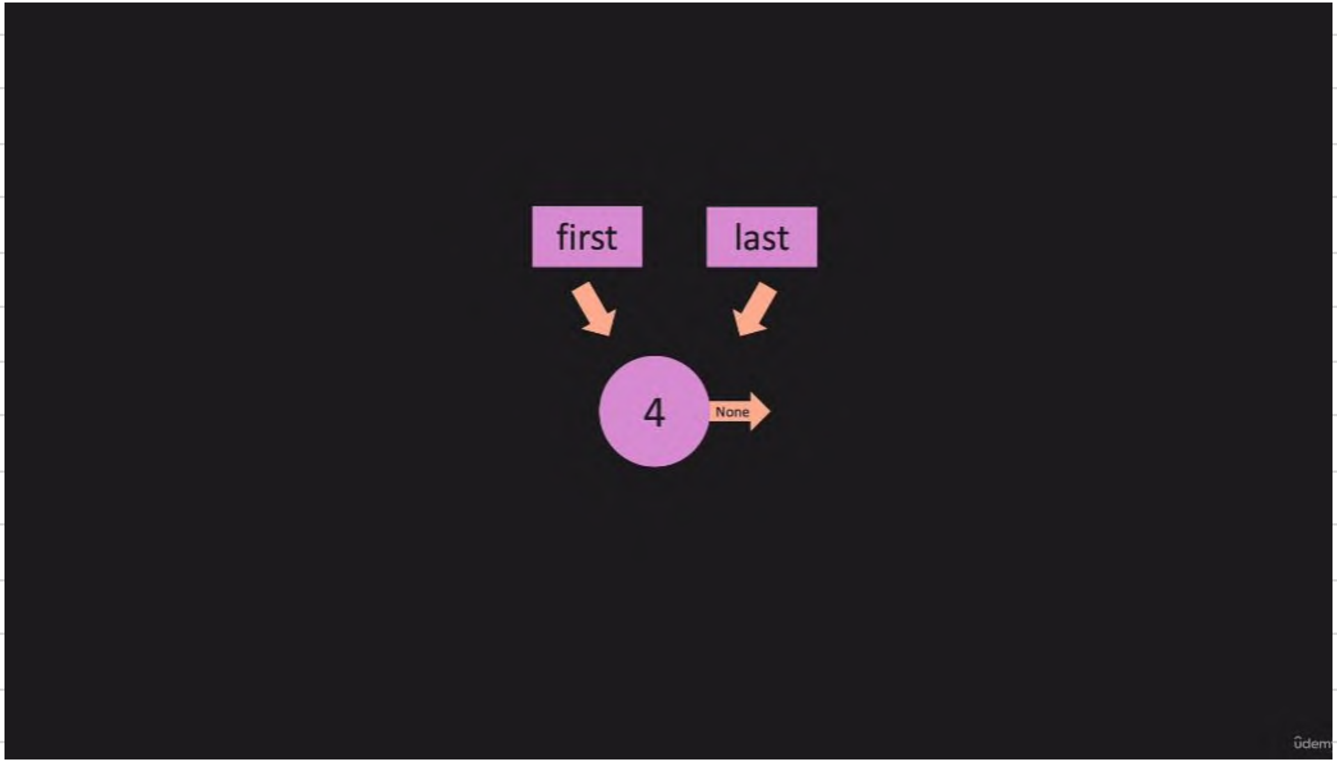
Edge Condition #1

Queue : Enqueue

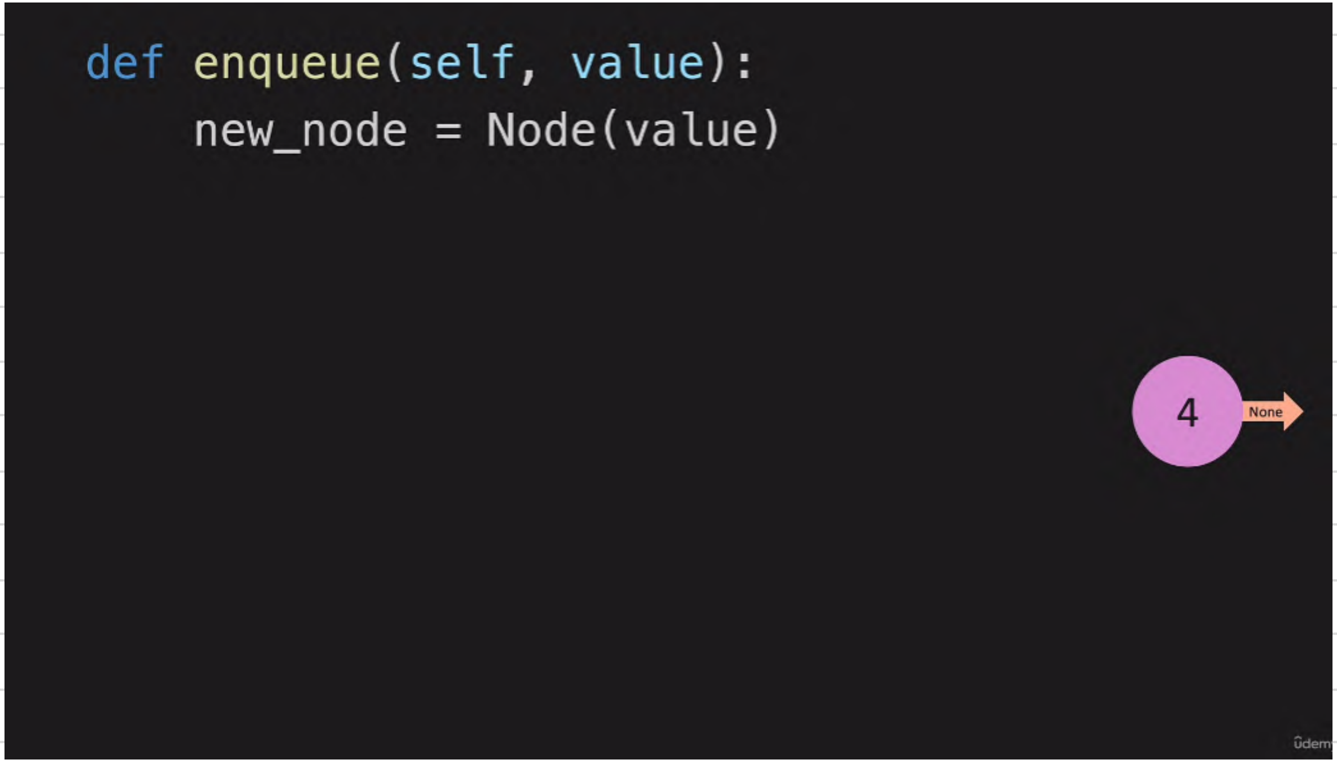


Seharusnya

Queue : Enqueue

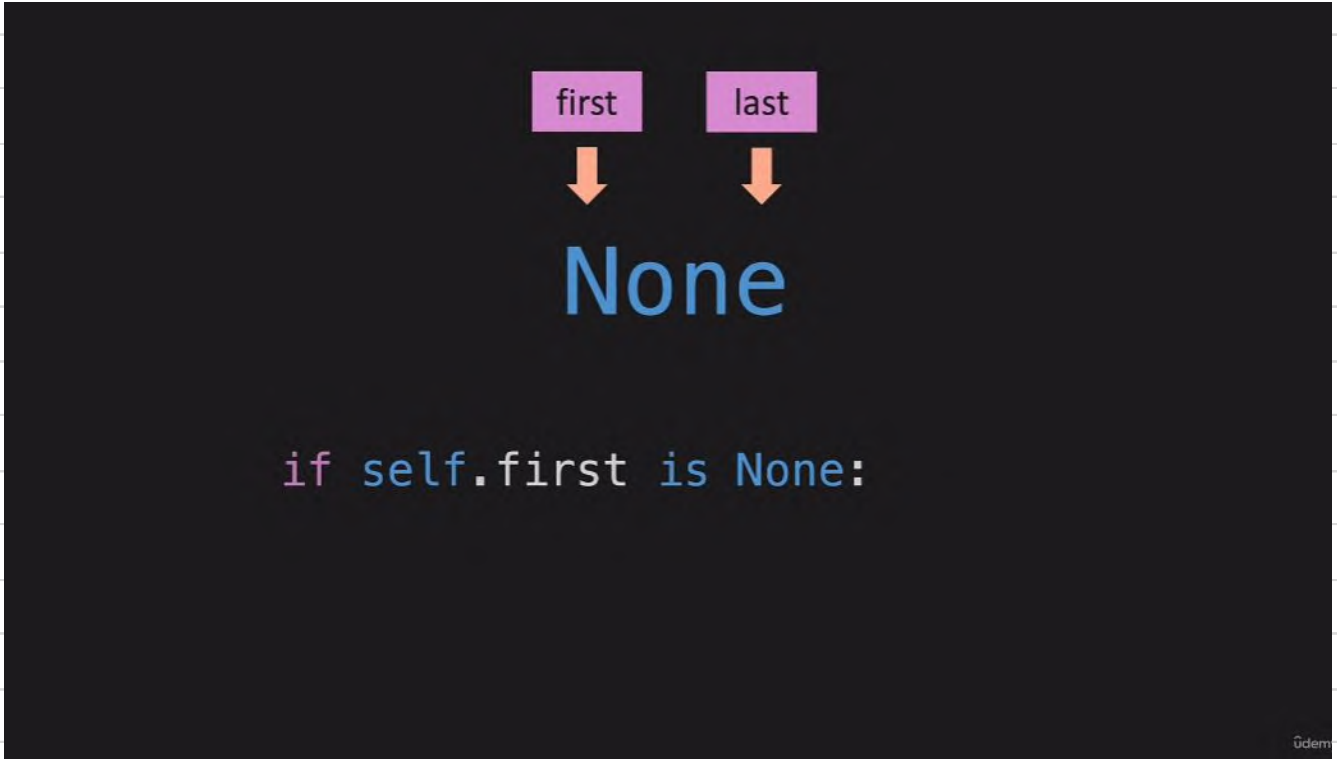


Queue : Enqueue Code



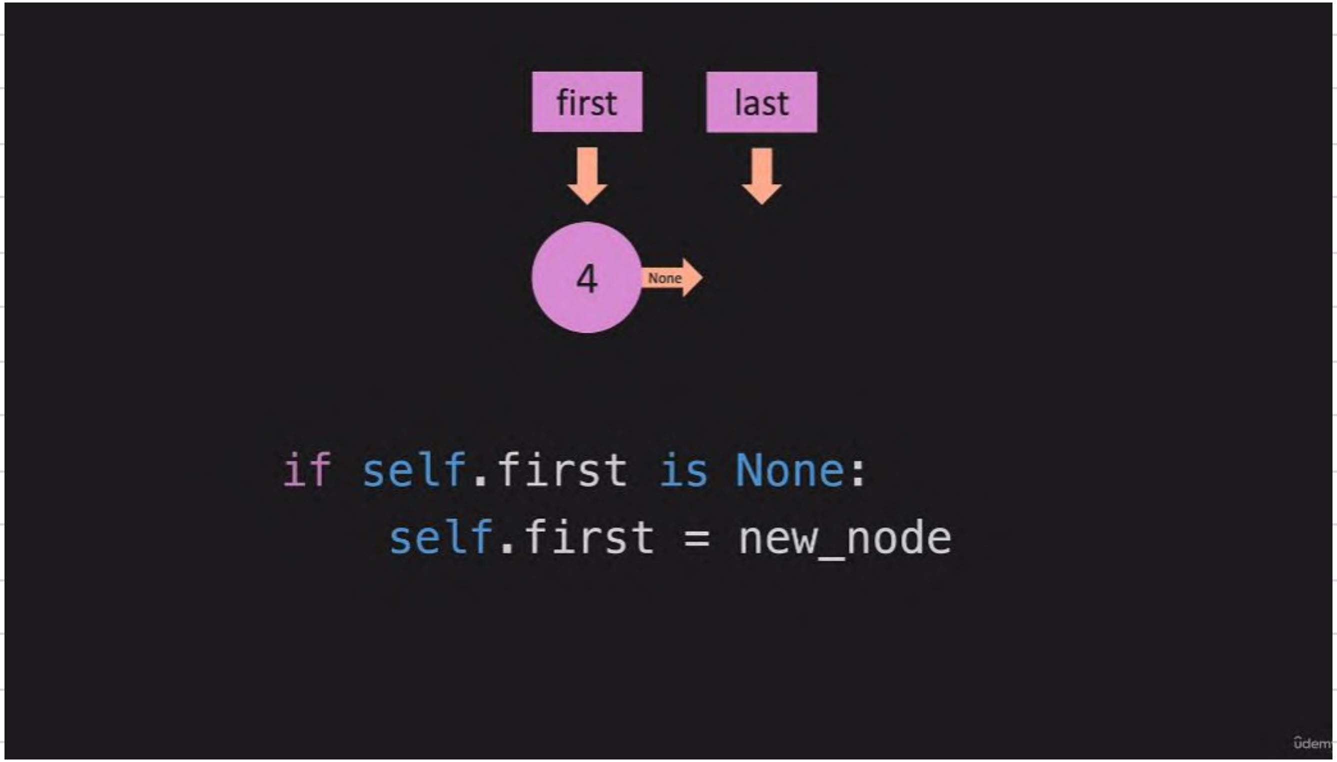
```
def enqueue(self, value):
    new_node = Node(value)
```

Queue : Enqueue Code



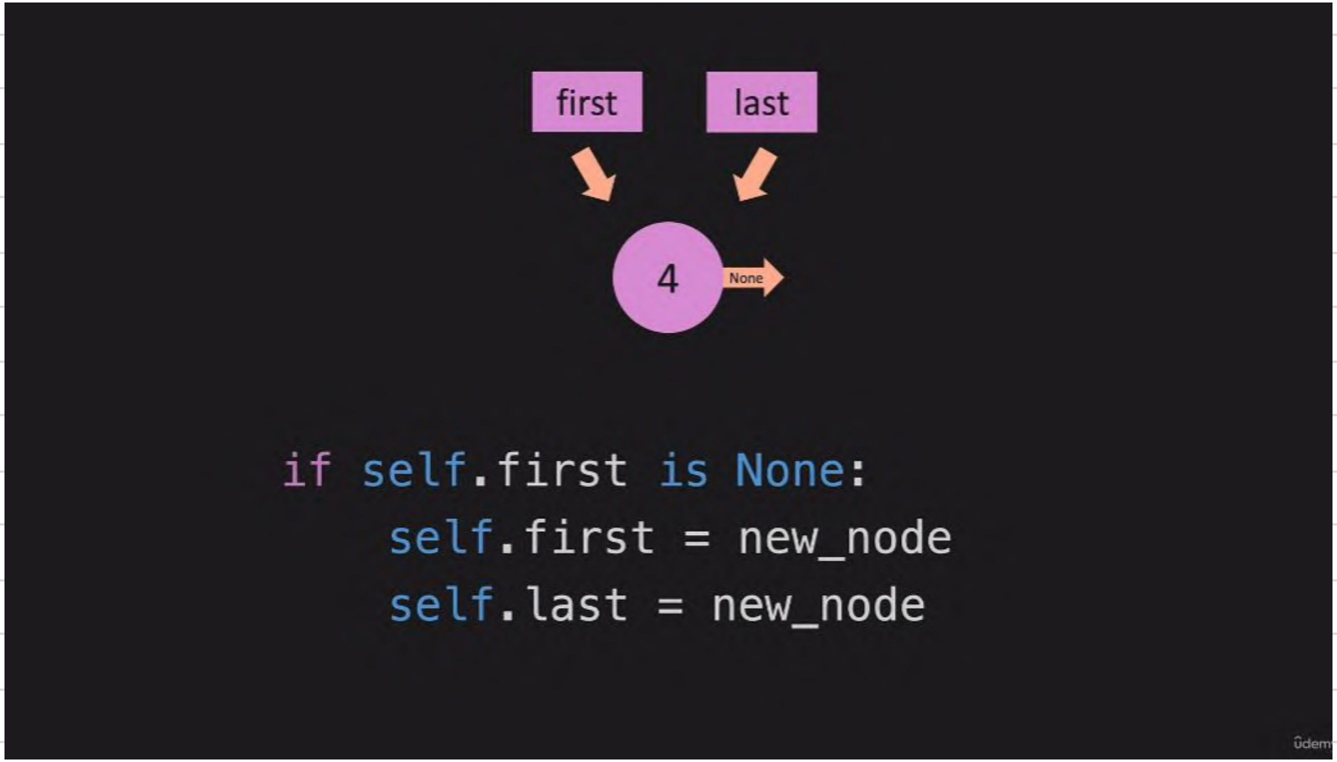
```
if self.first is None:
```

Queue : Enqueue Code



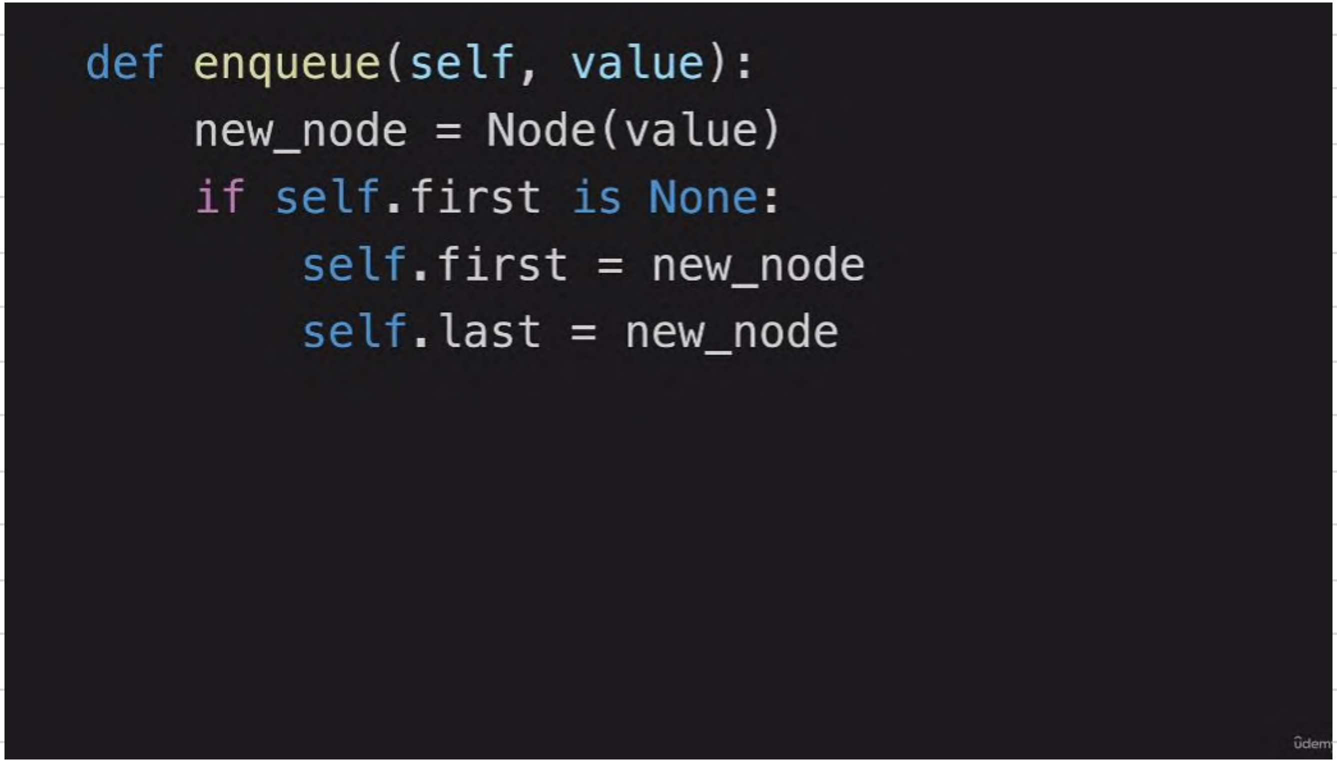
```
if self.first is None:
    self.first = new_node
```

Queue : Enqueue Code



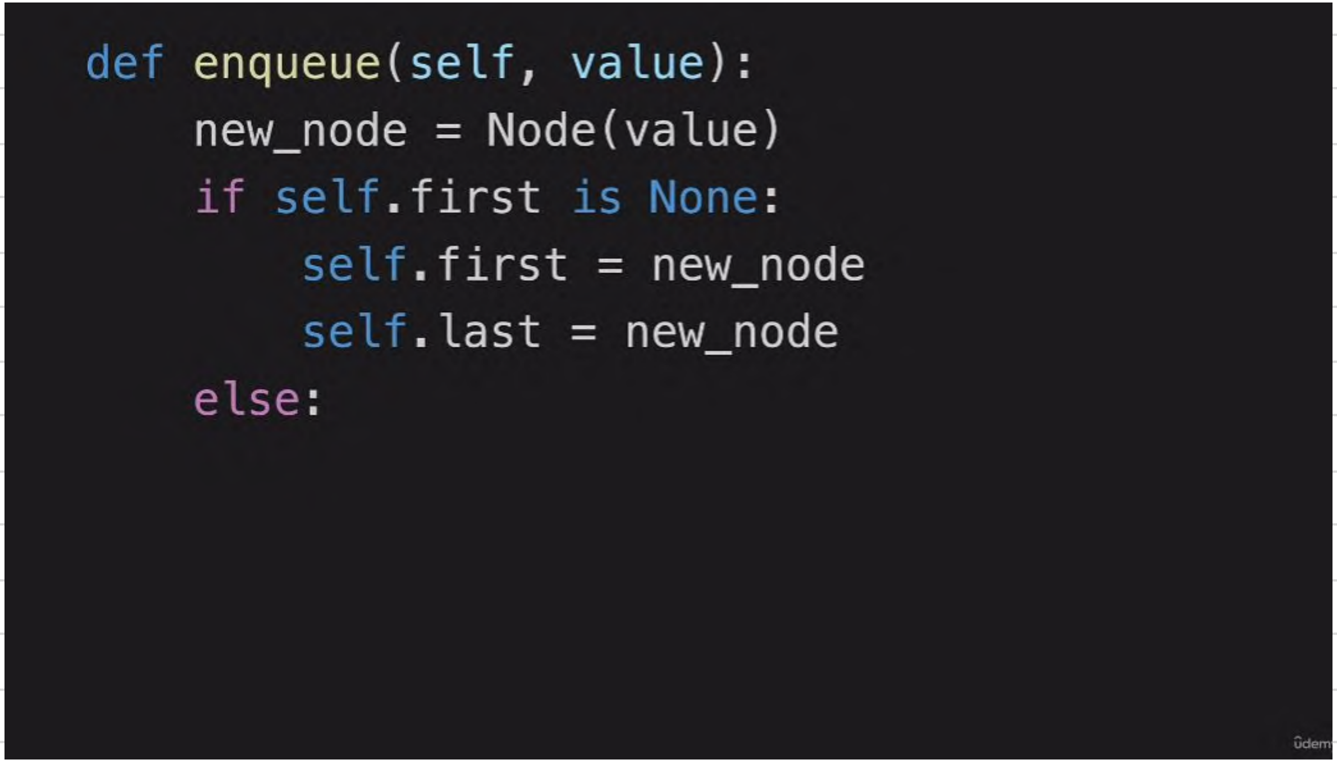
```
if self.first is None:
    self.first = new_node
    self.last = new_node
```

Queue : Enqueue Code



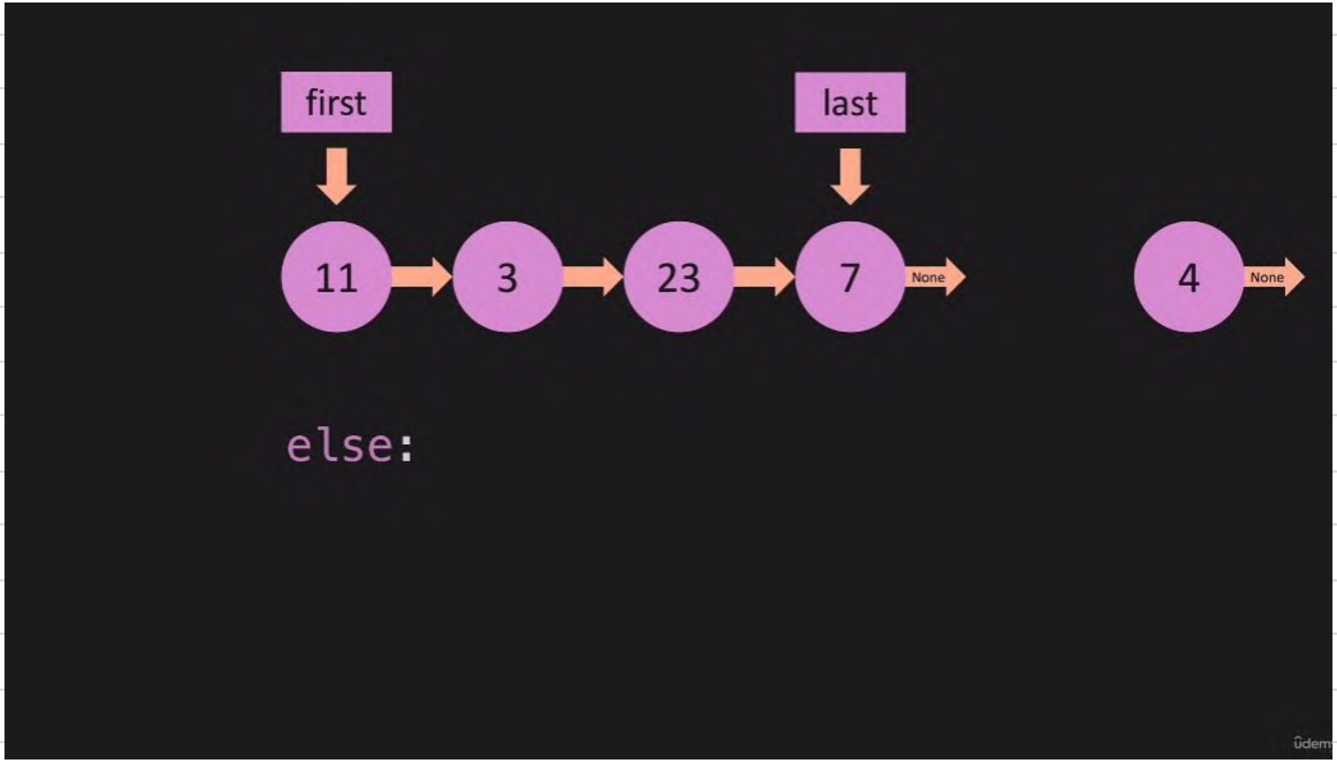
```
def enqueue(self, value):
    new_node = Node(value)
    if self.first is None:
        self.first = new_node
        self.last = new_node
```

Queue : Enqueue Code



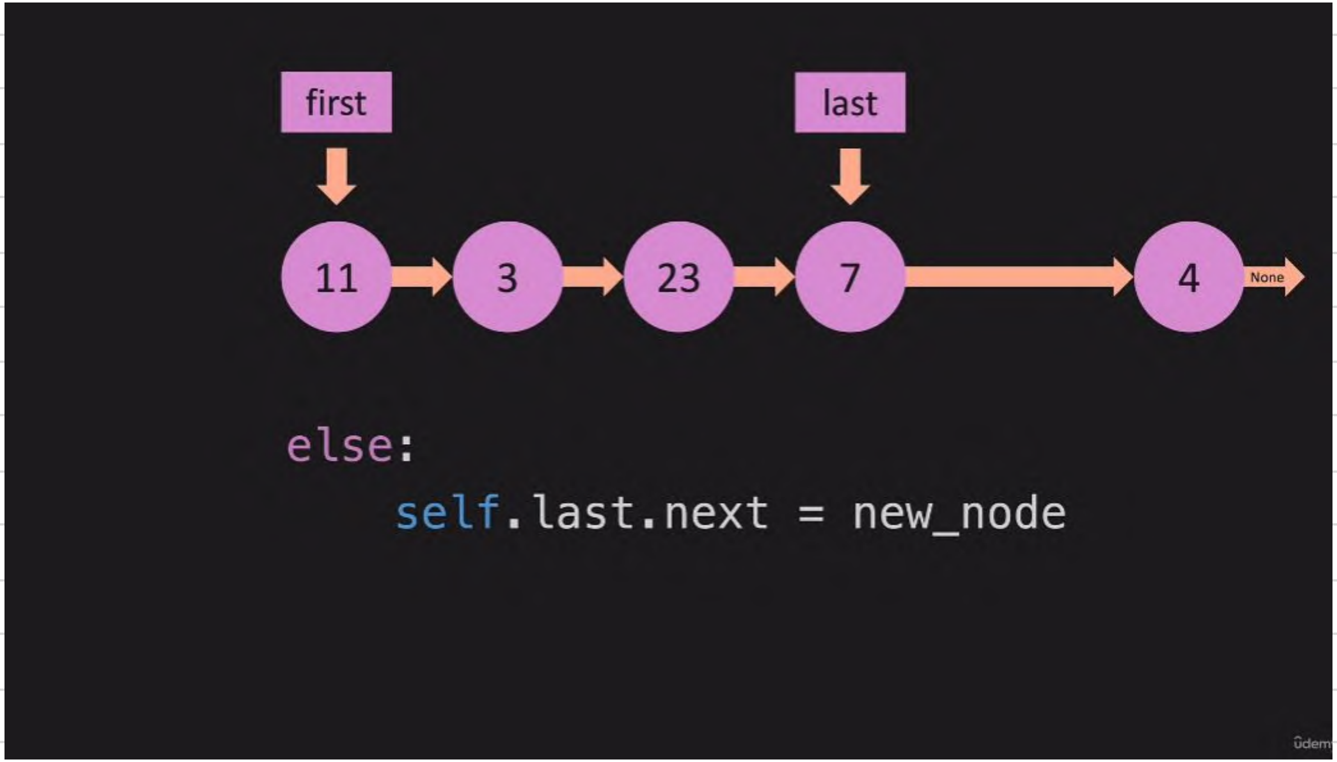
```
def enqueue(self, value):
    new_node = Node(value)
    if self.first is None:
        self.first = new_node
        self.last = new_node
    else:
```

Queue : Enqueue Code



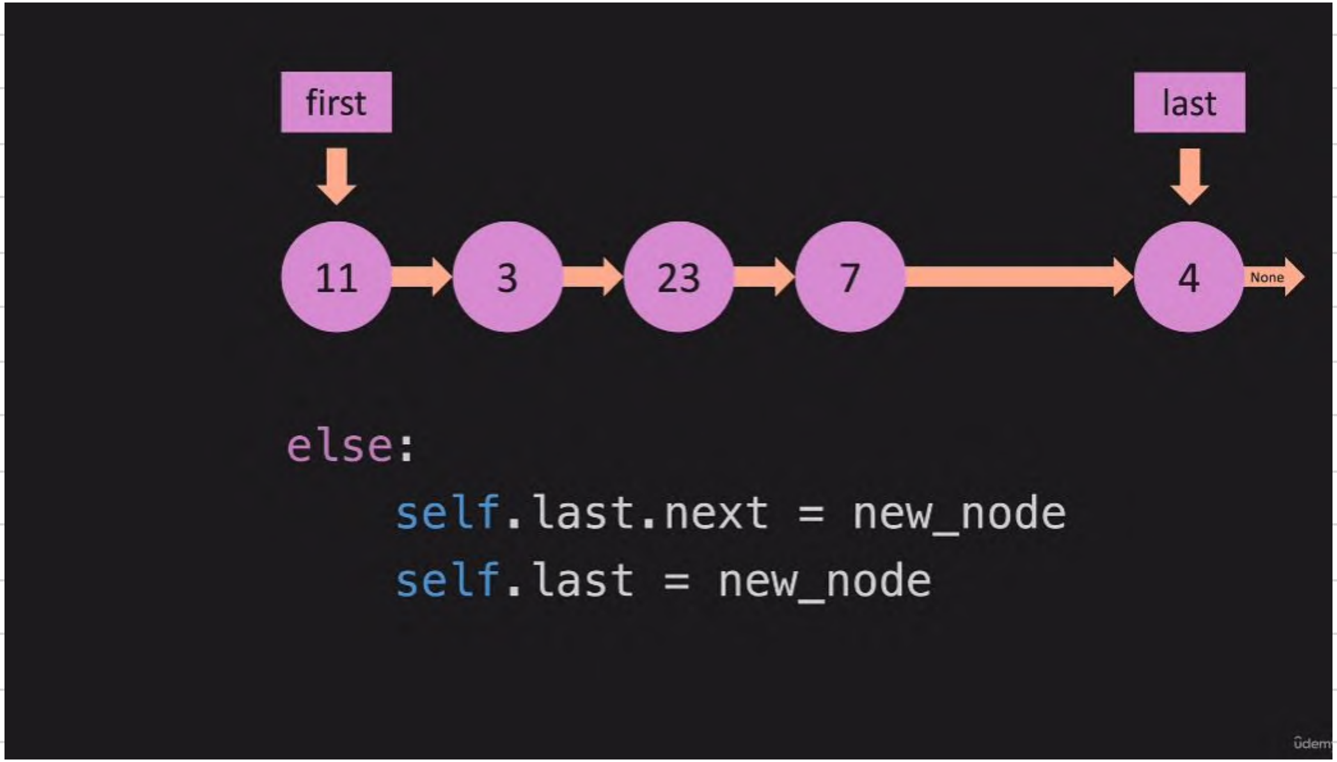
```
else:
```

Queue : Enqueue Code



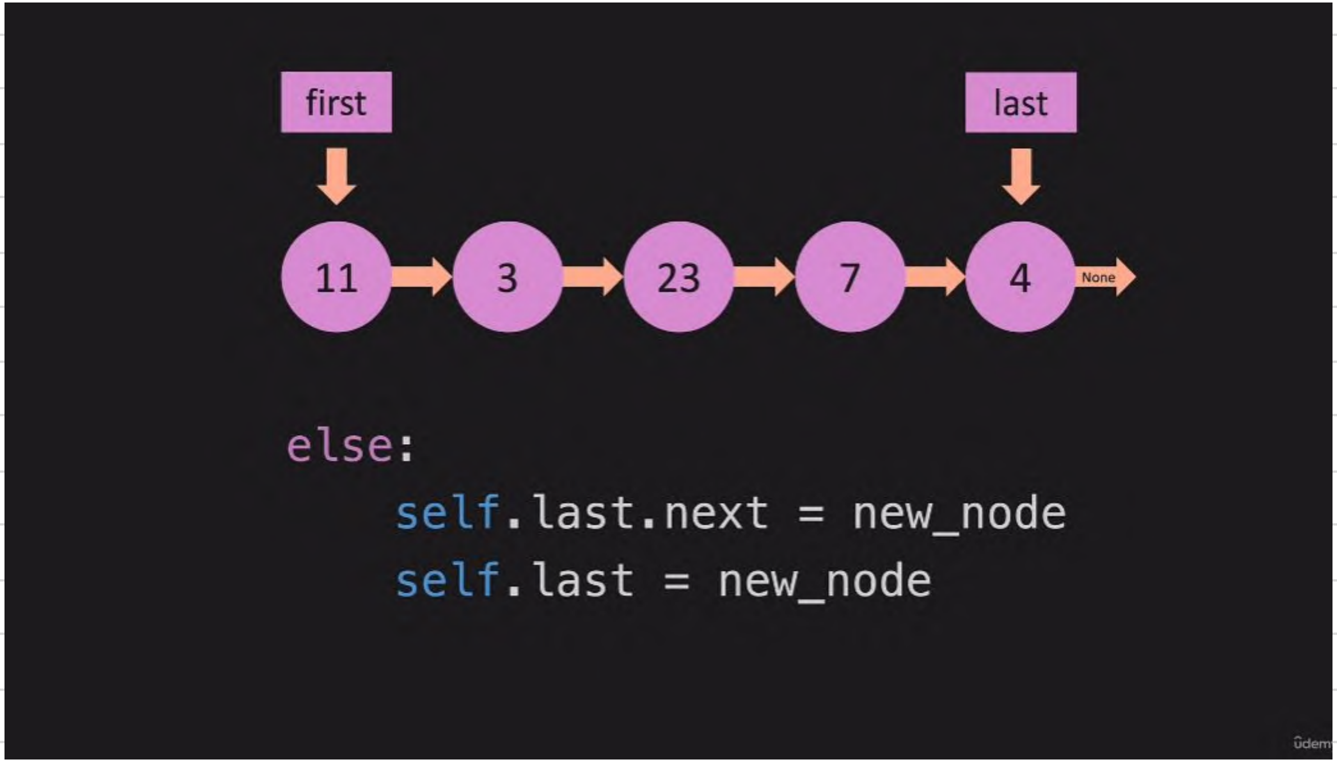
```
else:
    self.last.next = new_node
```

Queue : Enqueue Code



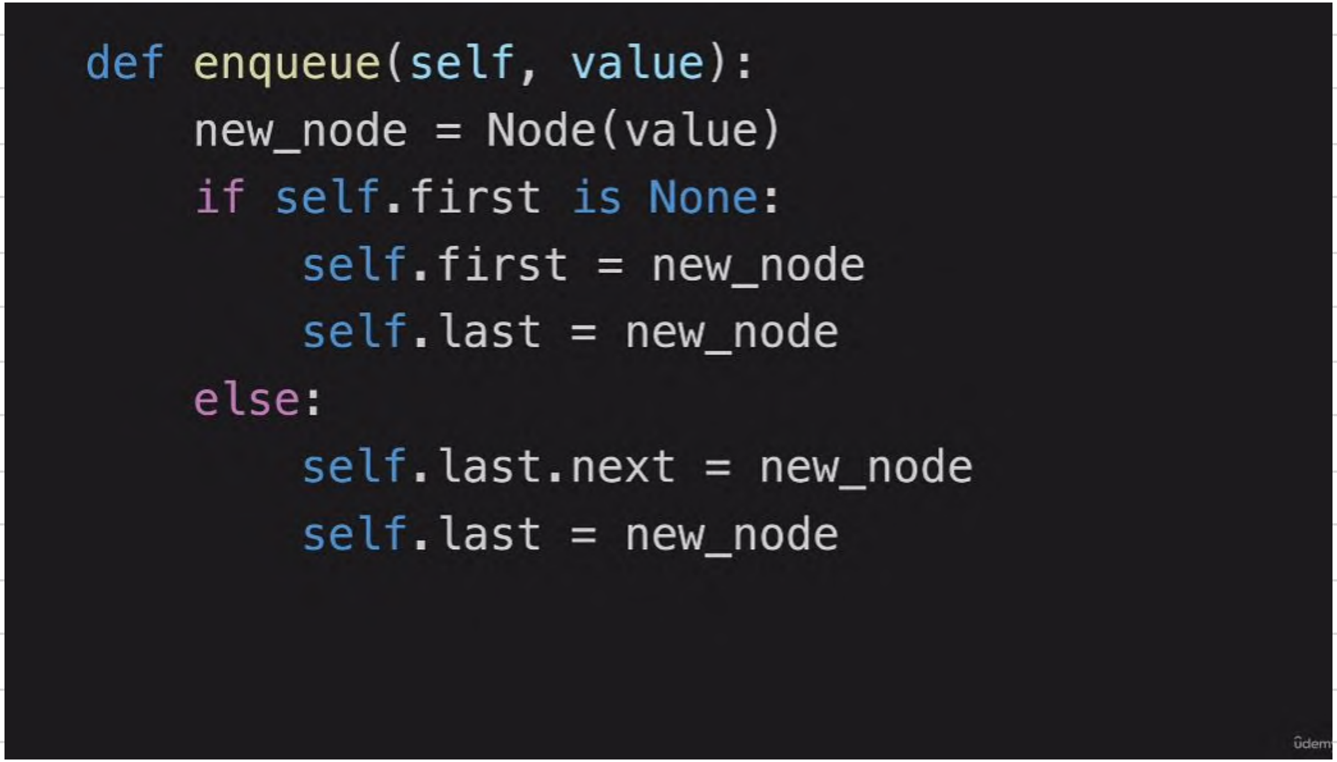
```
else:
    self.last.next = new_node
    self.last = new_node
```

Queue : Enqueue Code



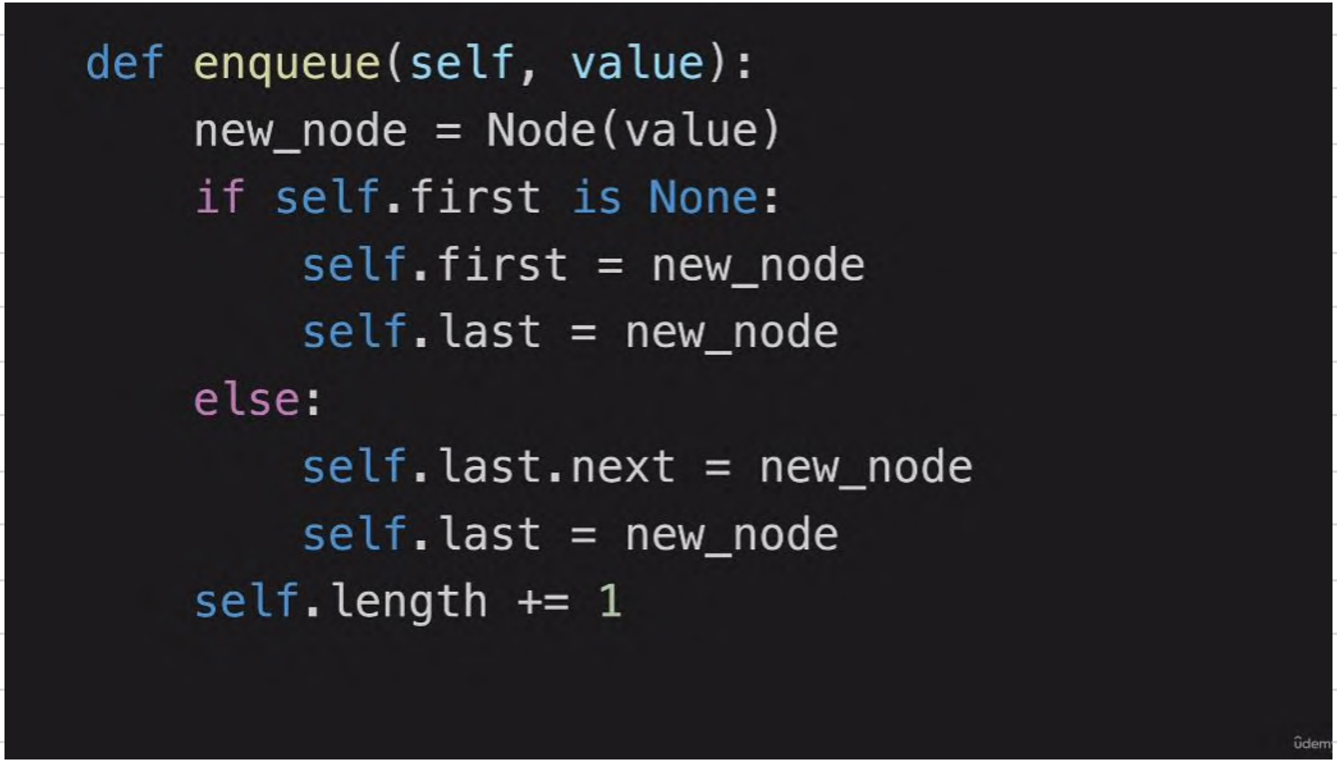
```
else:
    self.last.next = new_node
    self.last = new_node
```

Queue : Enqueue Code



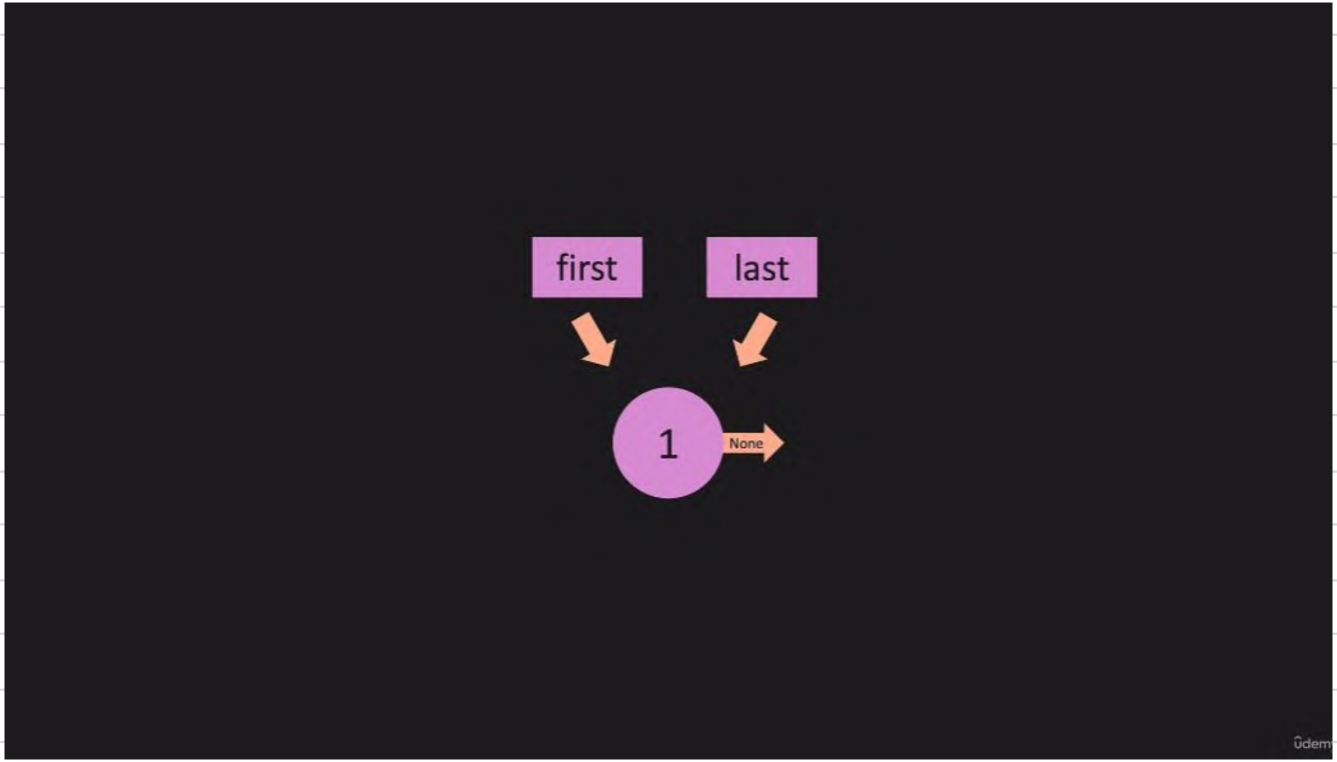
```
def enqueue(self, value):
    new_node = Node(value)
    if self.first is None:
        self.first = new_node
        self.last = new_node
    else:
        self.last.next = new_node
        self.last = new_node
```

Queue : Enqueue Code

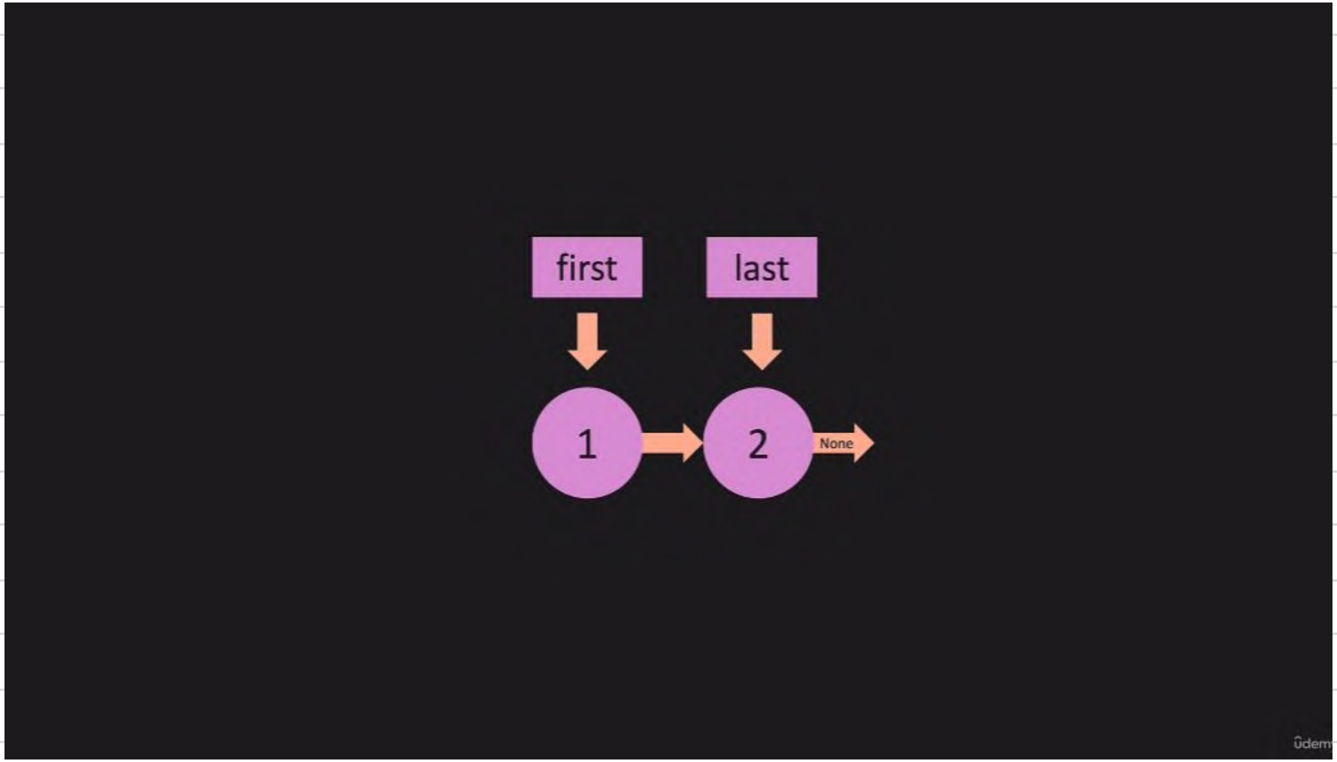


```python
def enqueue(self, value):
    new_node = Node(value)
    if self.first is None:
        self.first = new_node
        self.last = new_node
    else:
        self.last.next = new_node
        self.last = new_node
    self.length += 1
```

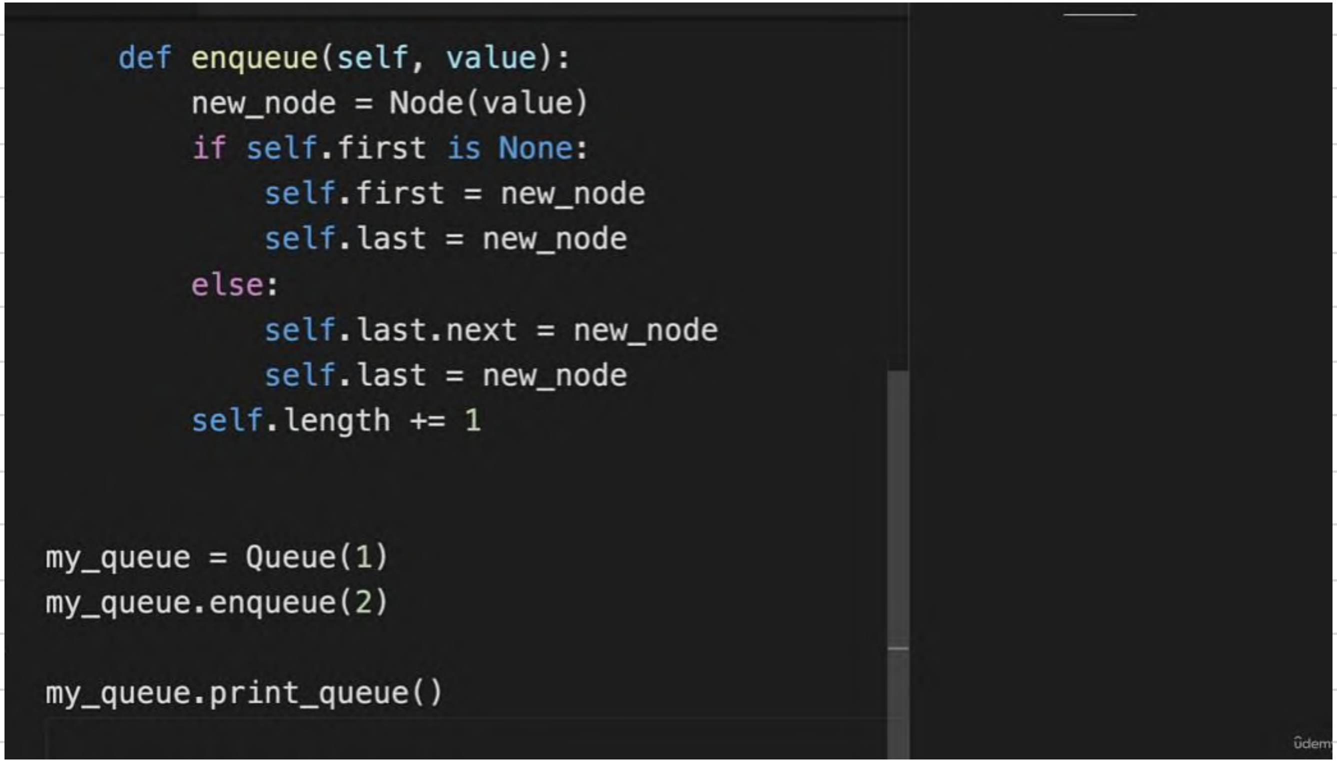
Queue : Enqueue Skenario Percobaan



Queue : Enqueue Skenario Percobaan

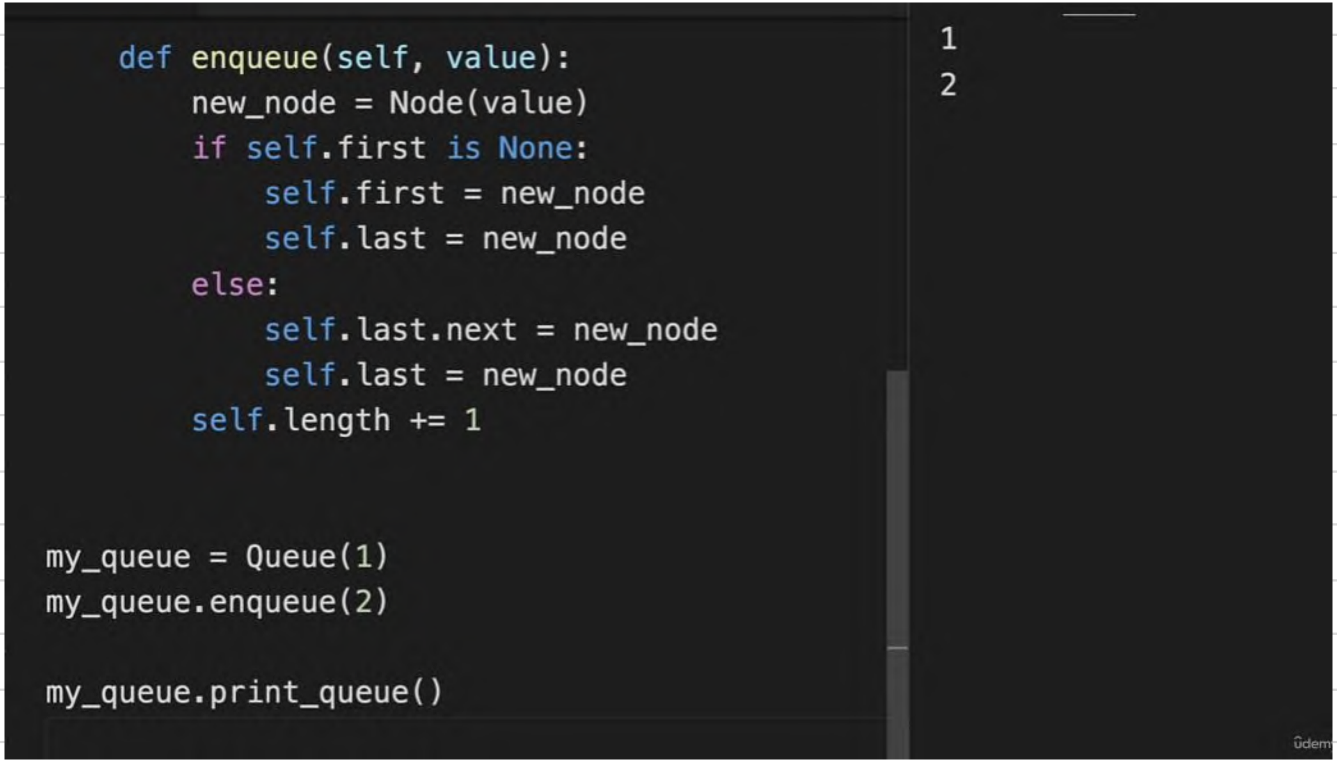


Queue : Enqueue Skenario Percobaan



```
    def enqueue(self, value):
        new_node = Node(value)
        if self.first is None:
            self.first = new_node
            self.last = new_node
        else:
            self.last.next = new_node
            self.last = new_node
        self.length += 1


my_queue = Queue(1)
my_queue.enqueue(2)

my_queue.print_queue()
```

Queue : Enqueue Skenario Percobaan



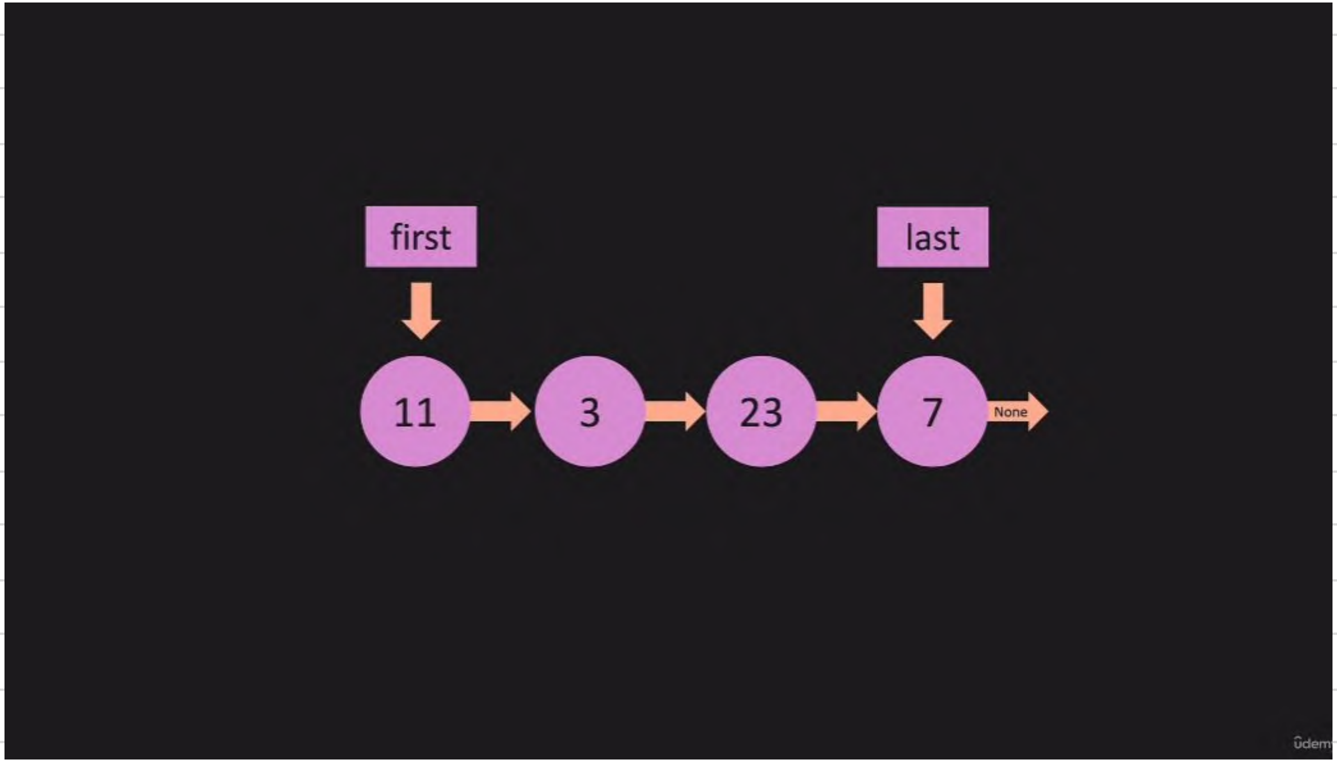
```
    def enqueue(self, value):
        new_node = Node(value)
        if self.first is None:
            self.first = new_node
            self.last = new_node
        else:
            self.last.next = new_node
            self.last = new_node
        self.length += 1


my_queue = Queue(1)
my_queue.enqueue(2)

my_queue.print_queue()
```

```
1
2
```

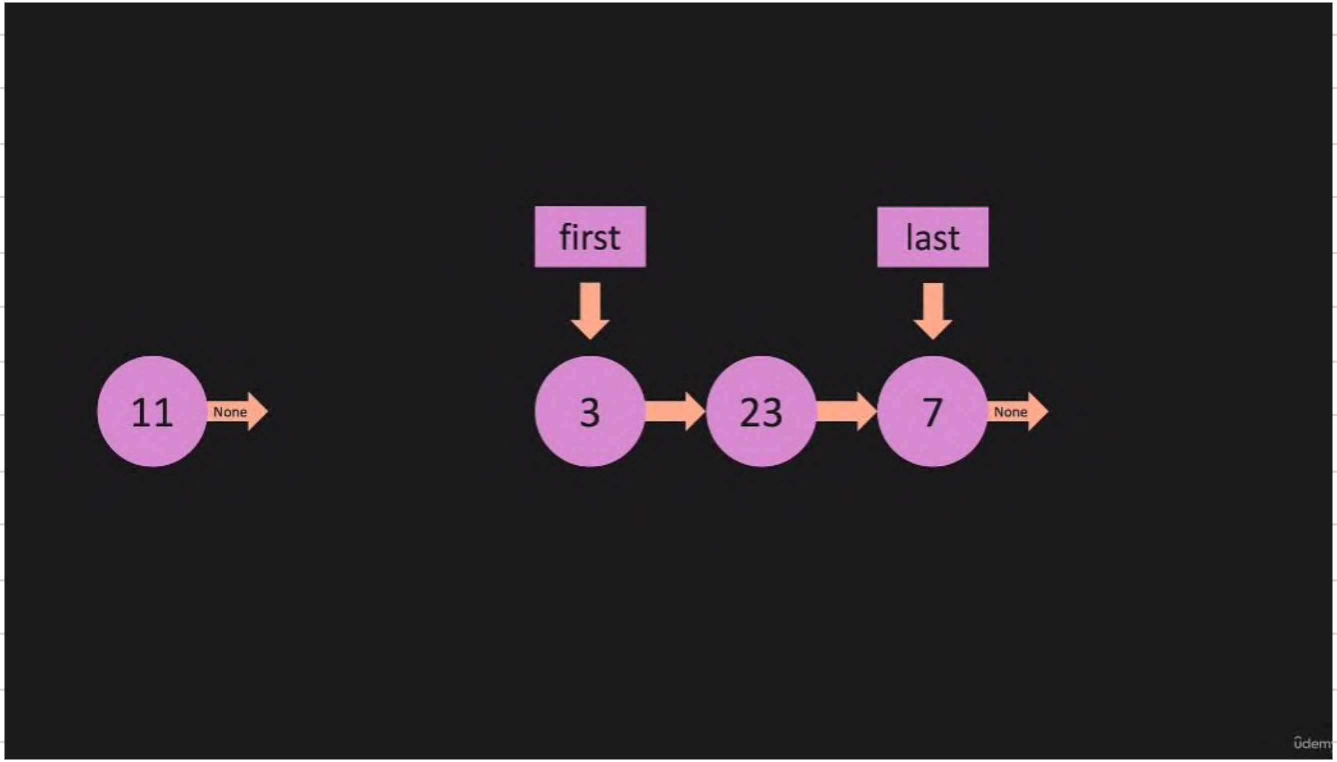
08

Queue
Dequeue

Queue : Dequeue

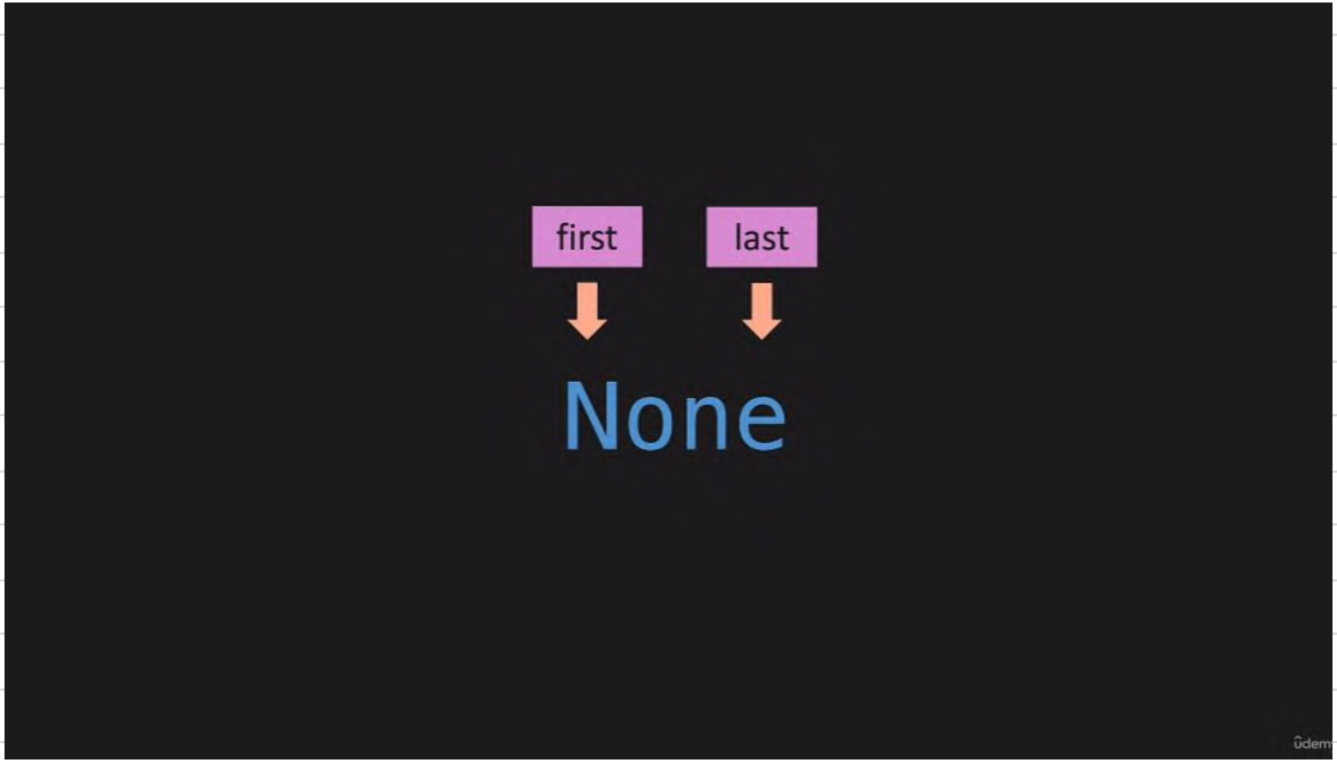


Queue : Dequeue



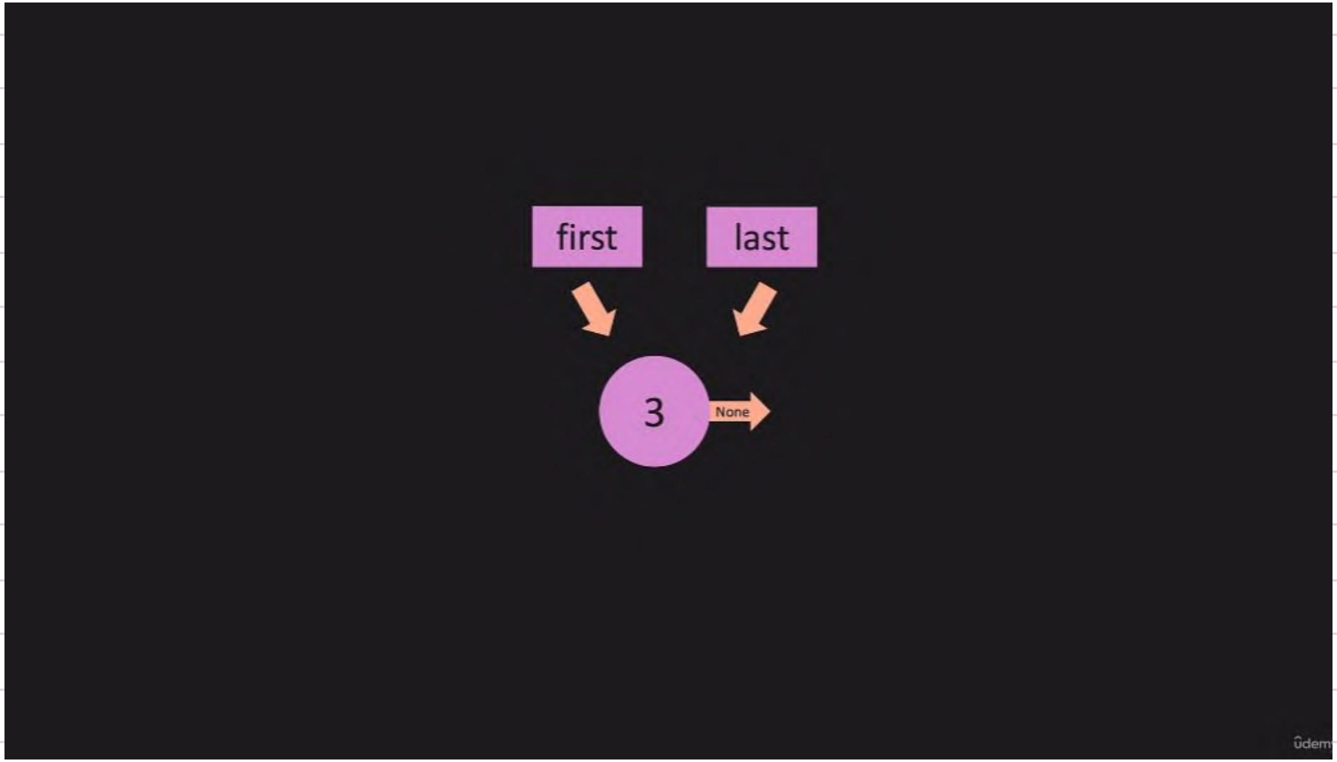
Edge Condition #1

Queue : Dequeue

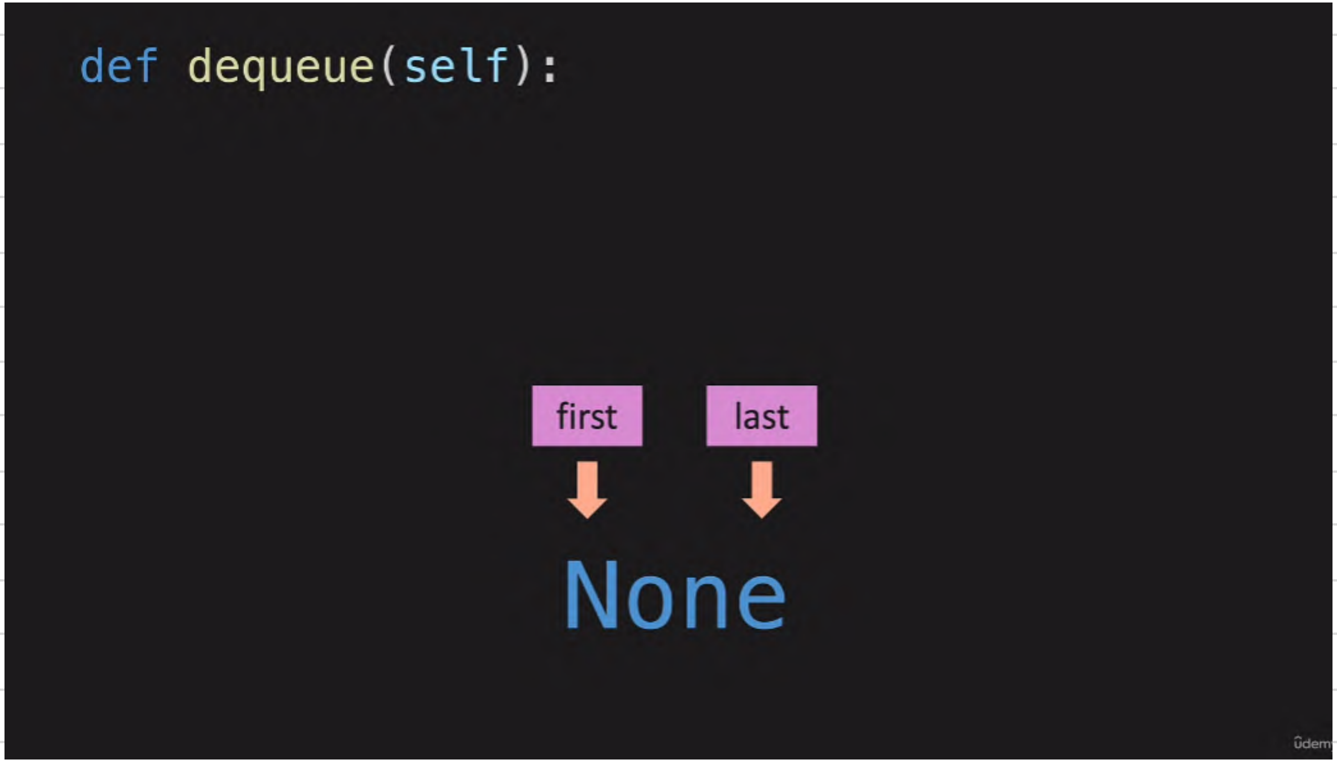


Edge Condition #2

Queue : Dequeue

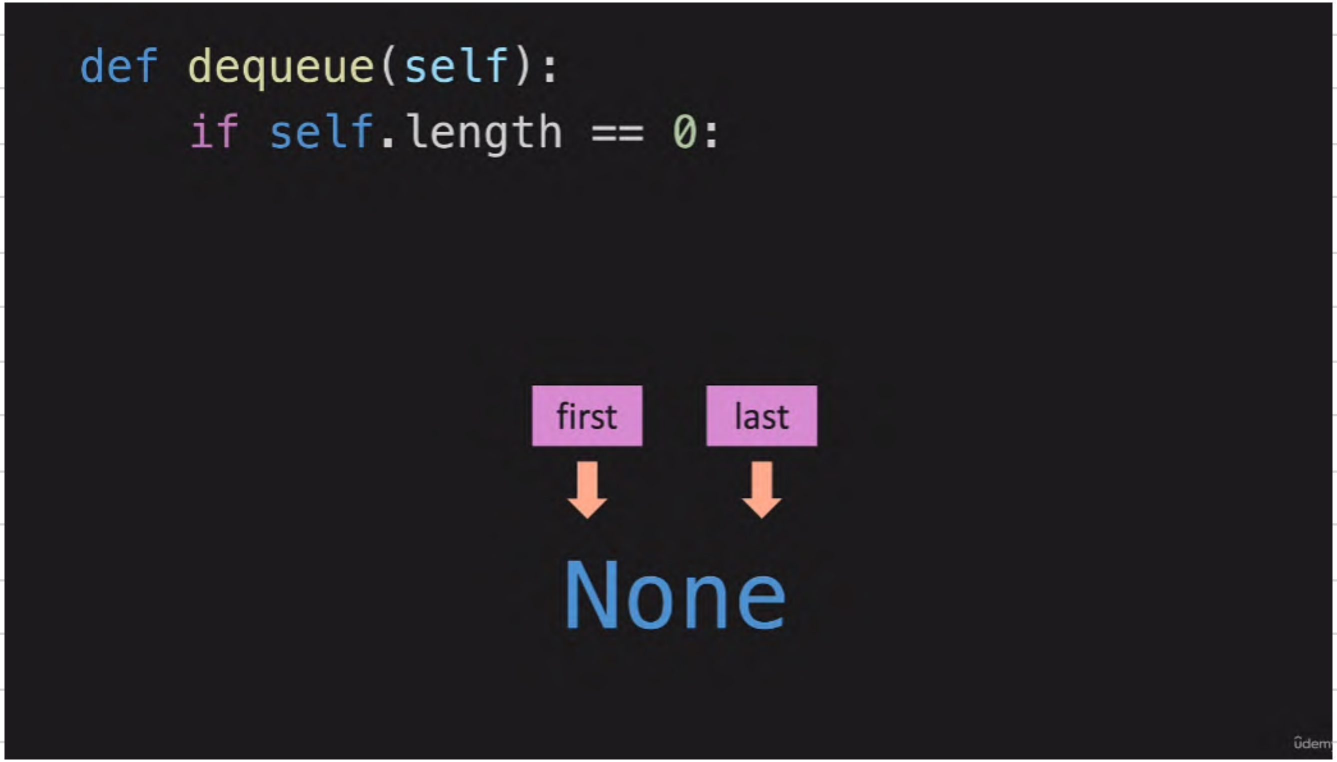


Queue : Dequeue Code



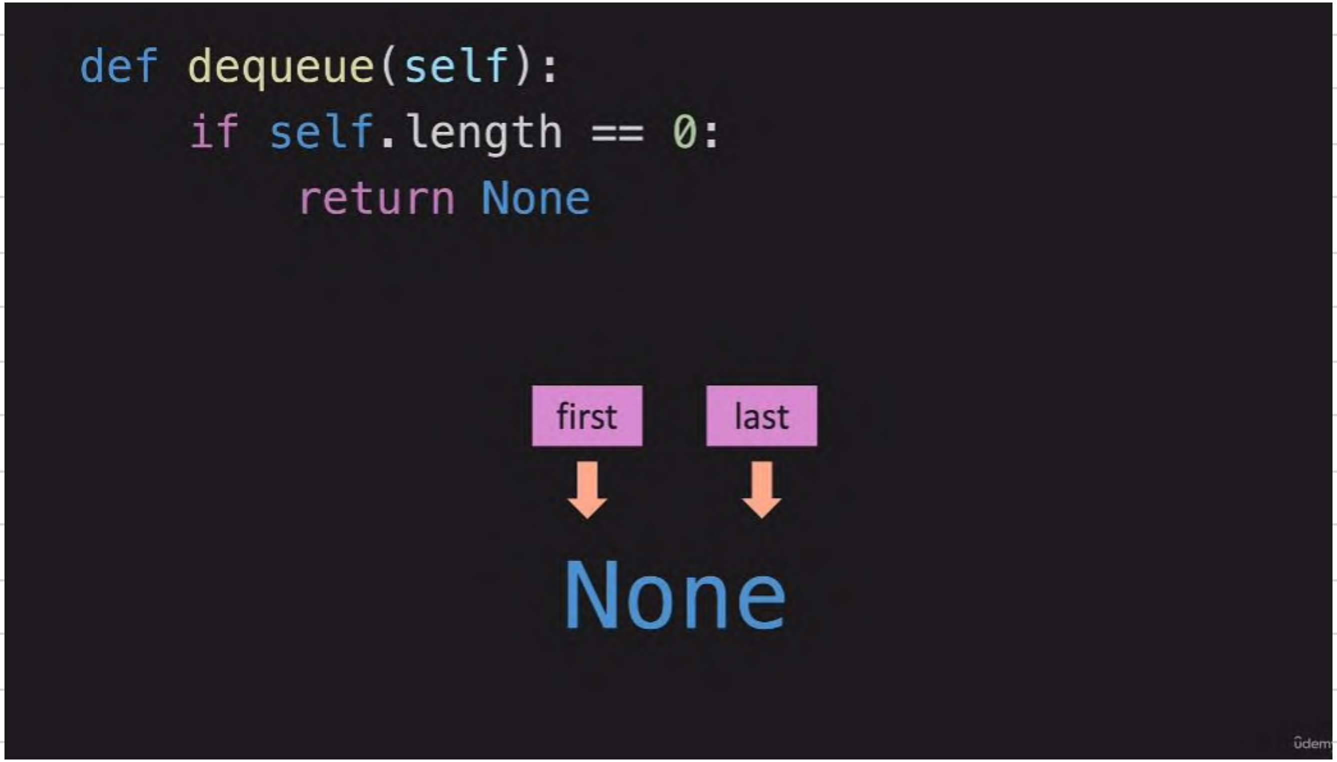
```
def dequeue(self):
```

Queue : Dequeue Code



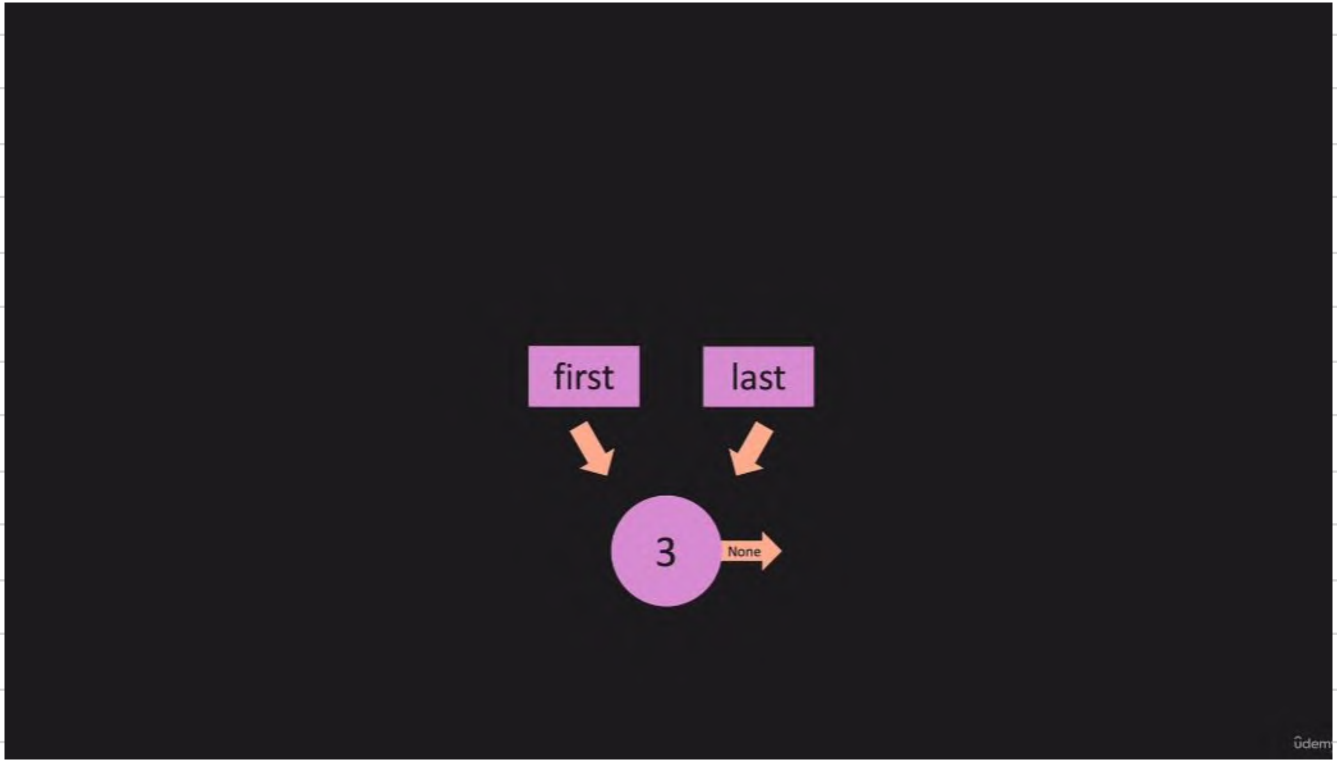
```
def dequeue(self):
    if self.length == 0:
```

Queue : Dequeue Code

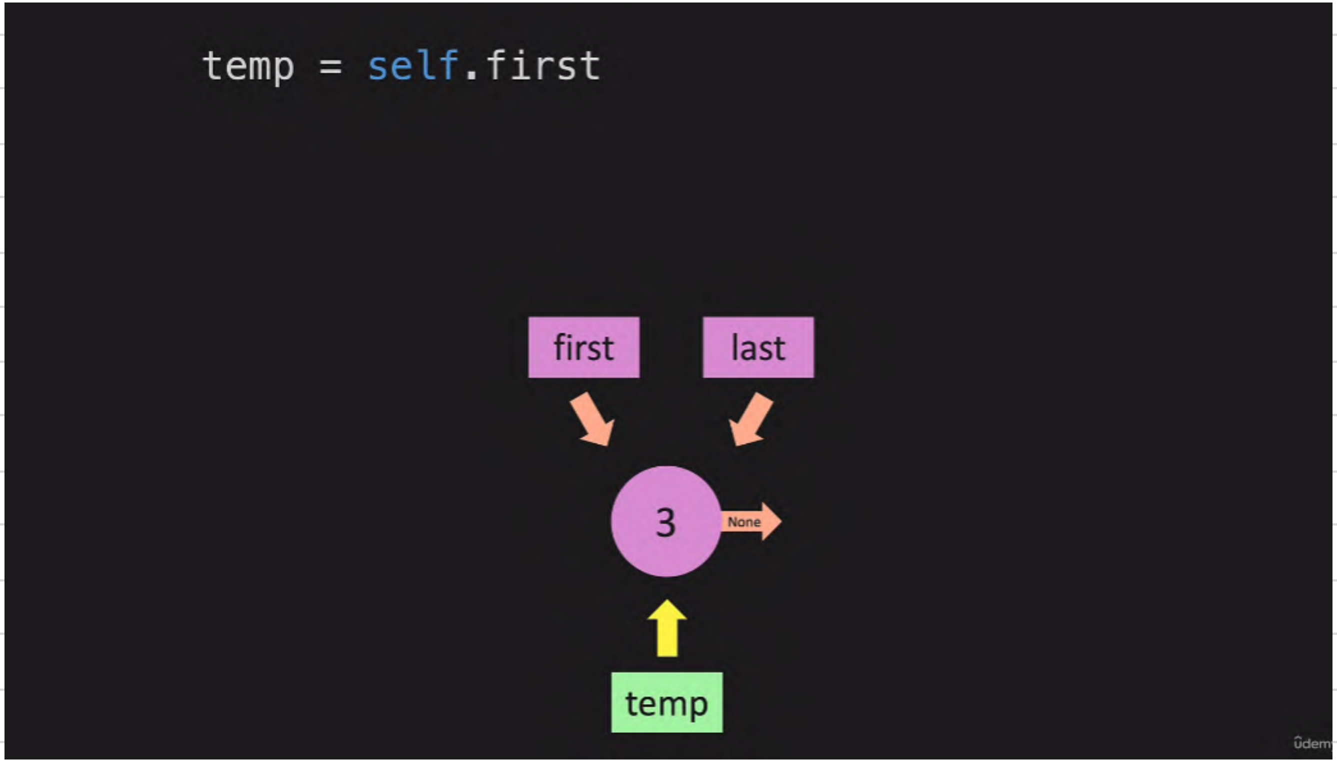


```
def dequeue(self):
    if self.length == 0:
        return None
```

Queue : Dequeue Code

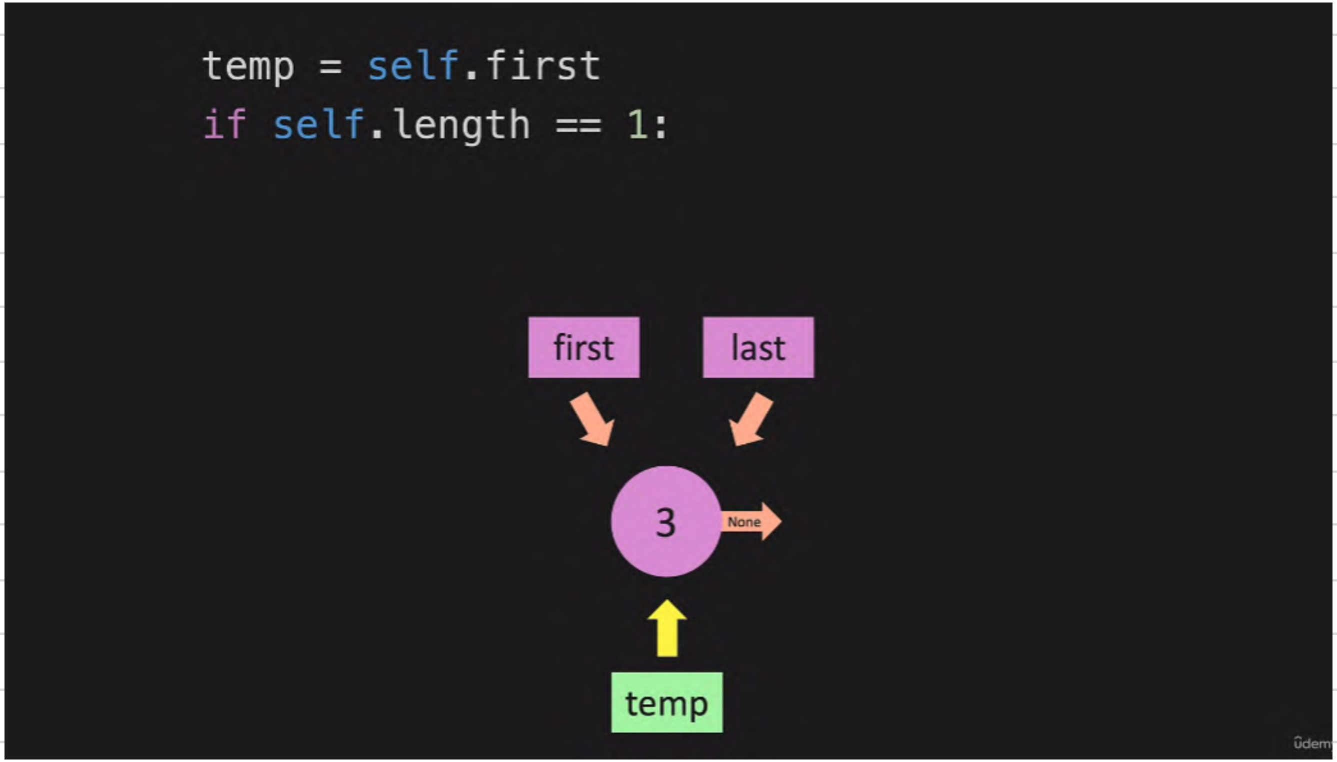


Queue : Dequeue Code



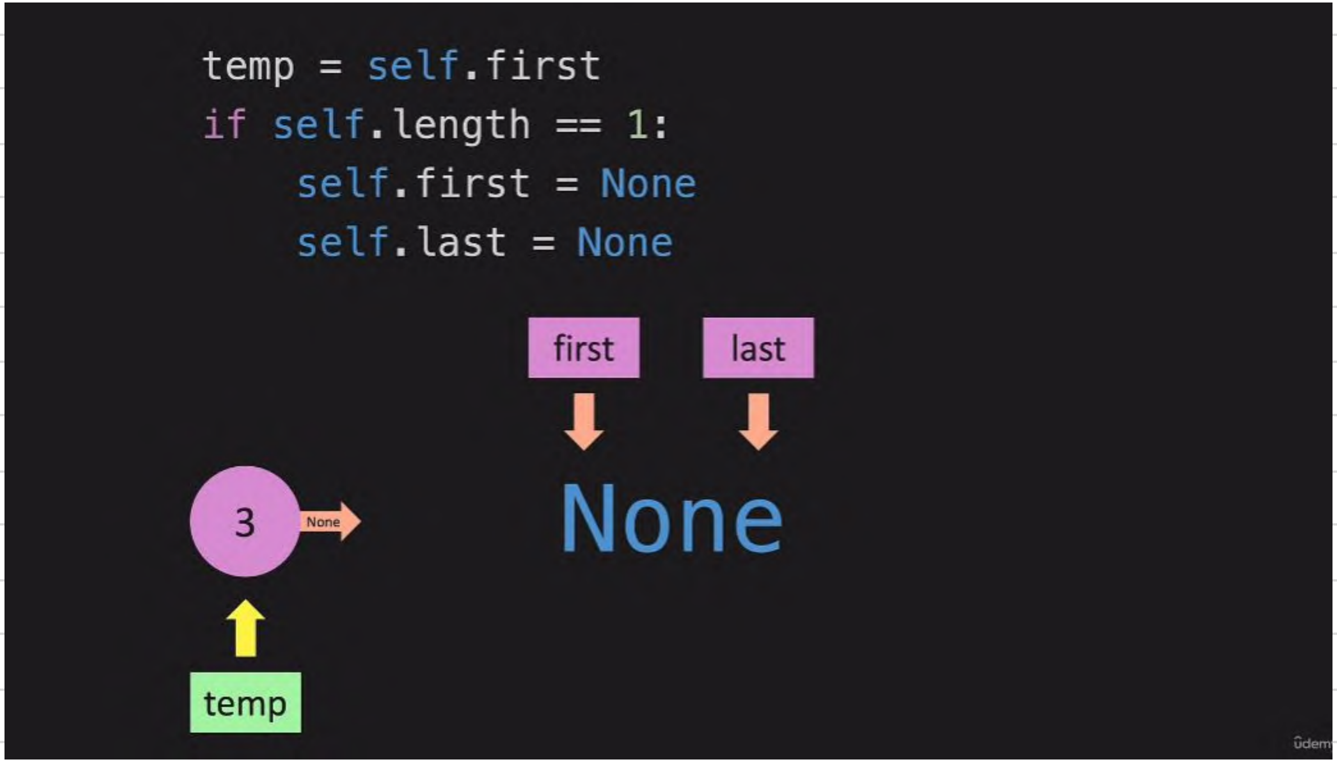
```
temp = self.first
```

Queue : Dequeue Code



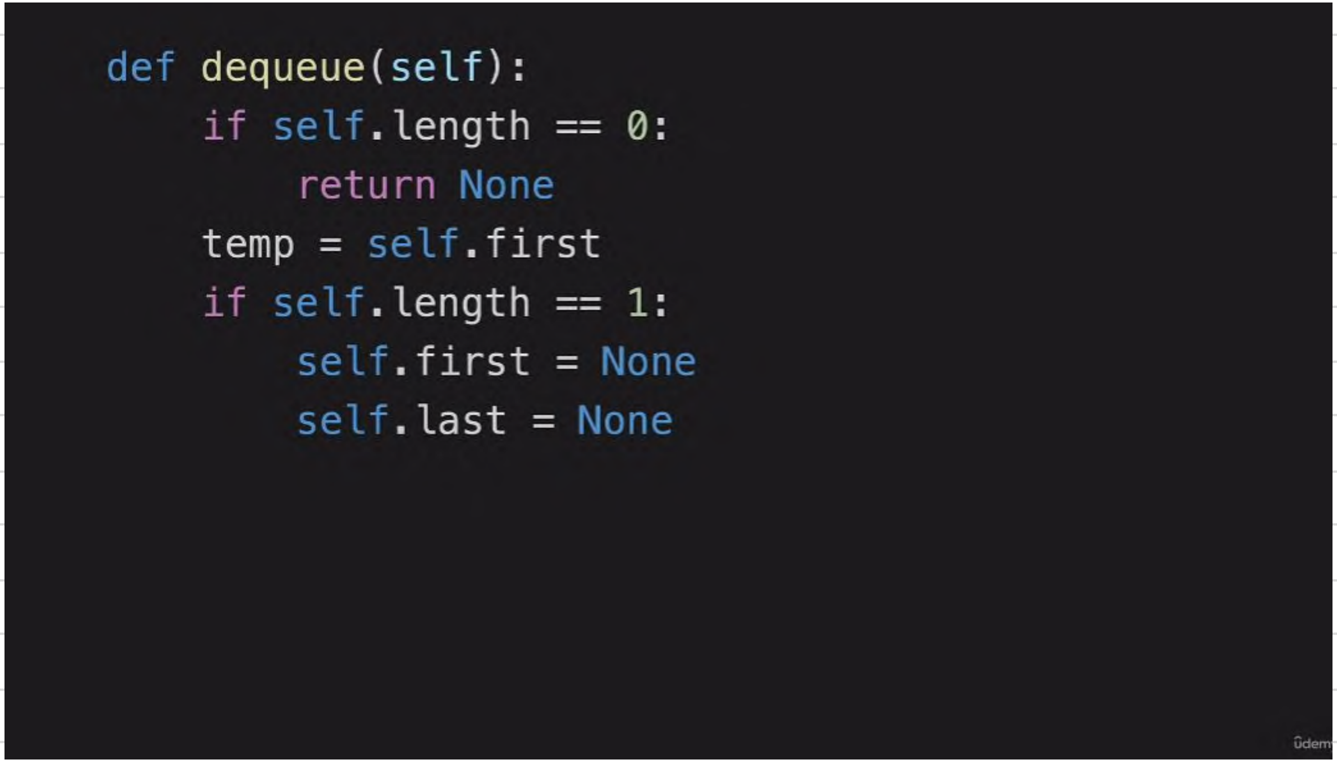
```
temp = self.first
if self.length == 1:
```

Queue : Dequeue Code



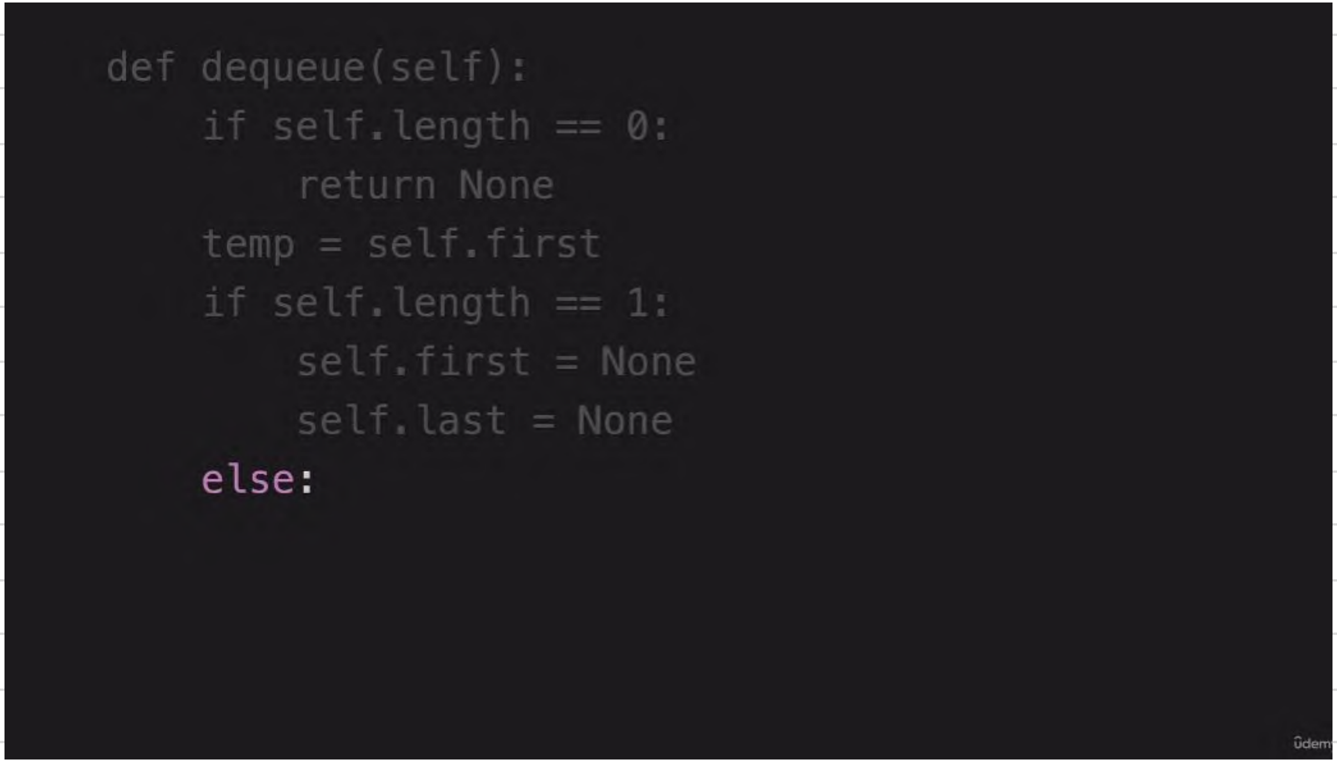
```
temp = self.first
if self.length == 1:
    self.first = None
    self.last = None
```

Queue : Dequeue Code



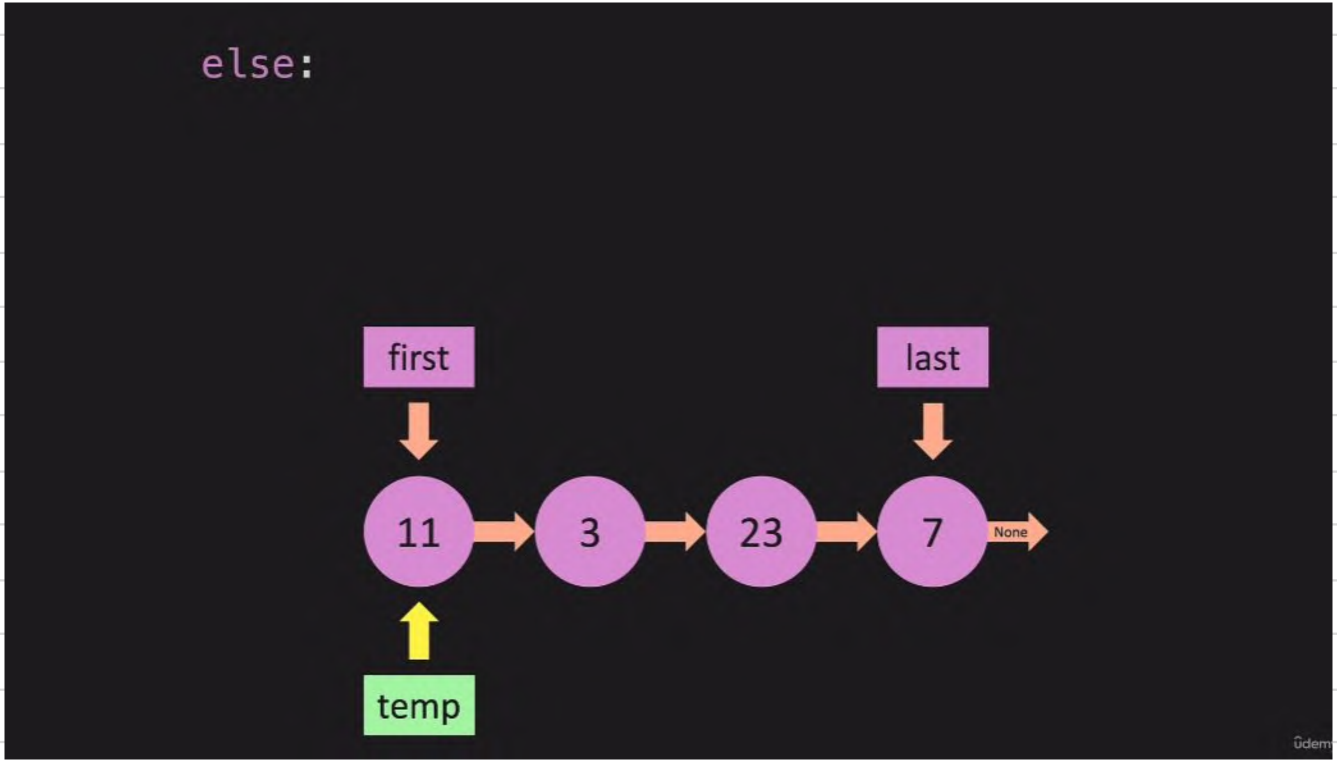
```
def dequeue(self):
    if self.length == 0:
        return None
    temp = self.first
    if self.length == 1:
        self.first = None
        self.last = None
```

Queue : Dequeue Code



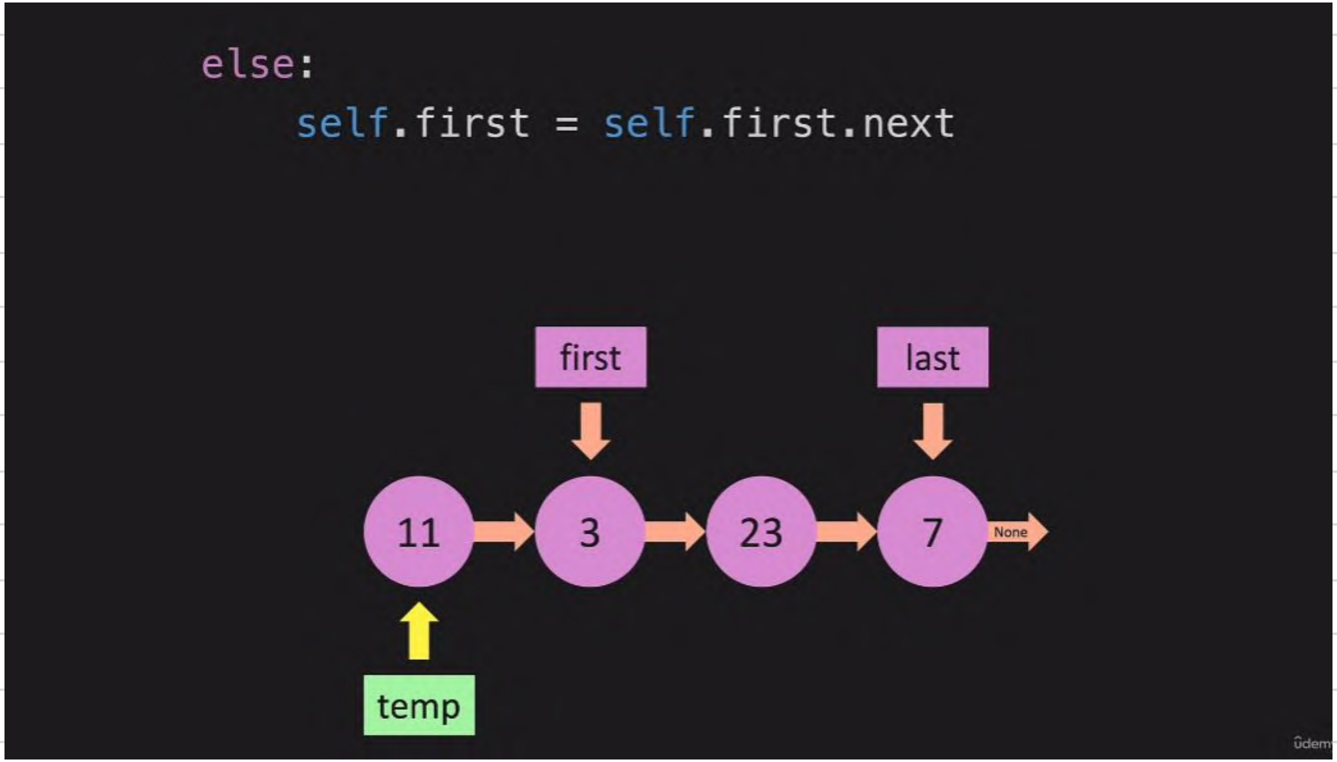
```
def dequeue(self):
    if self.length == 0:
        return None
    temp = self.first
    if self.length == 1:
        self.first = None
        self.last = None
    else:
```

Queue : Dequeue Code



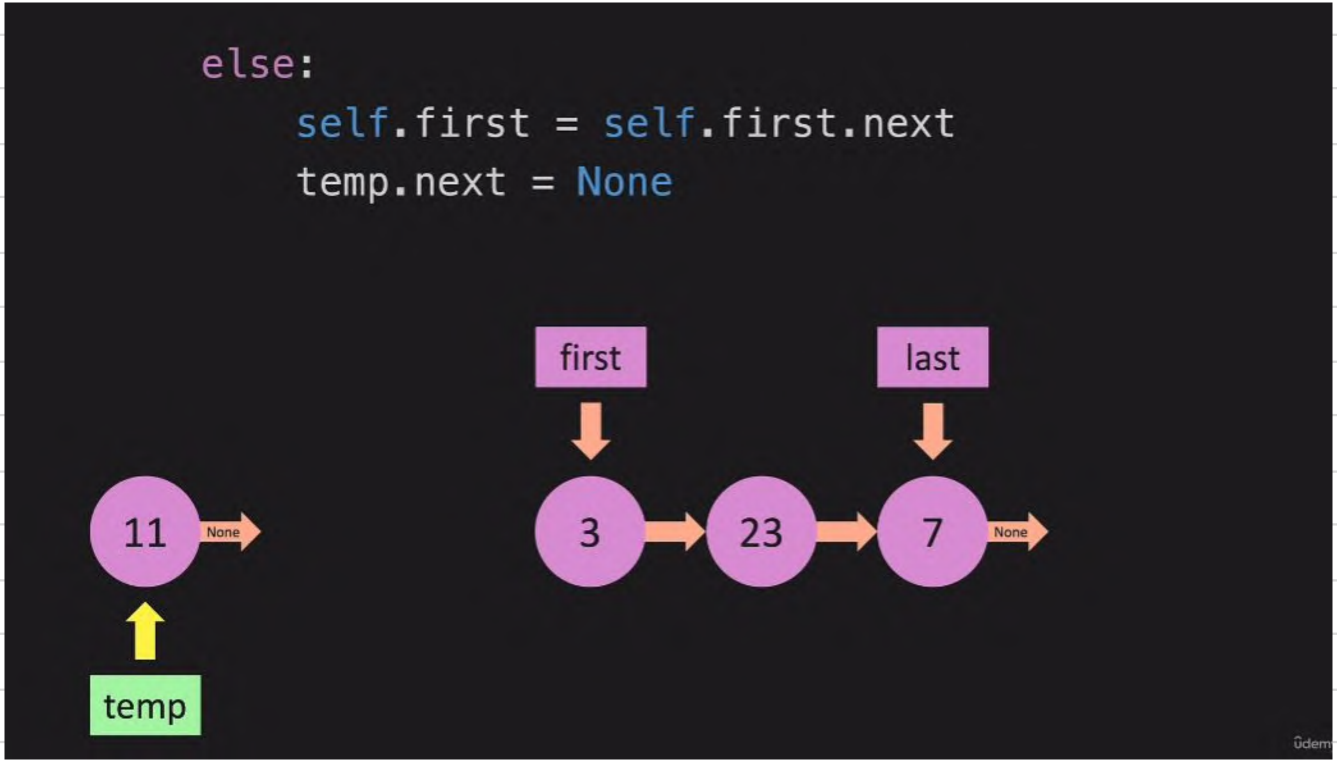
```
else:
```

Queue : Dequeue Code



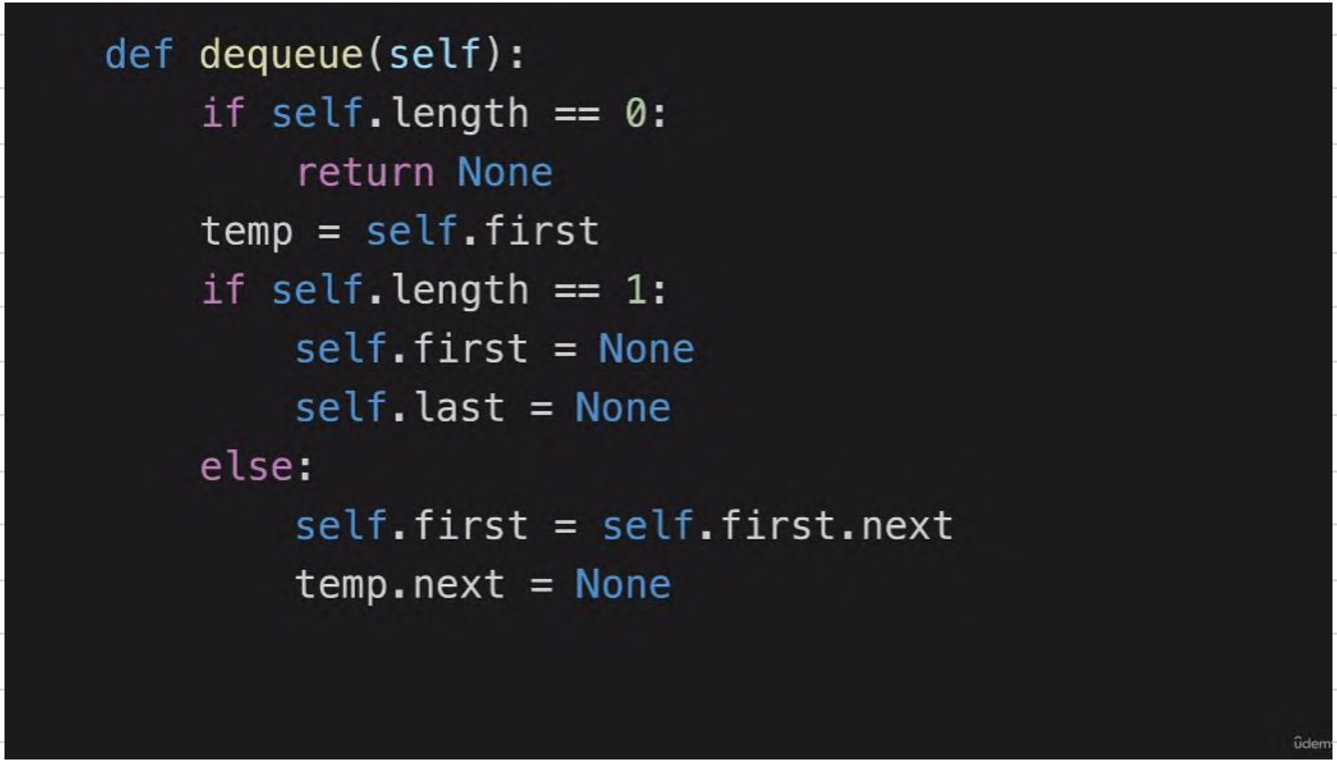
```
else:
    self.first = self.first.next
```

Queue : Dequeue Code



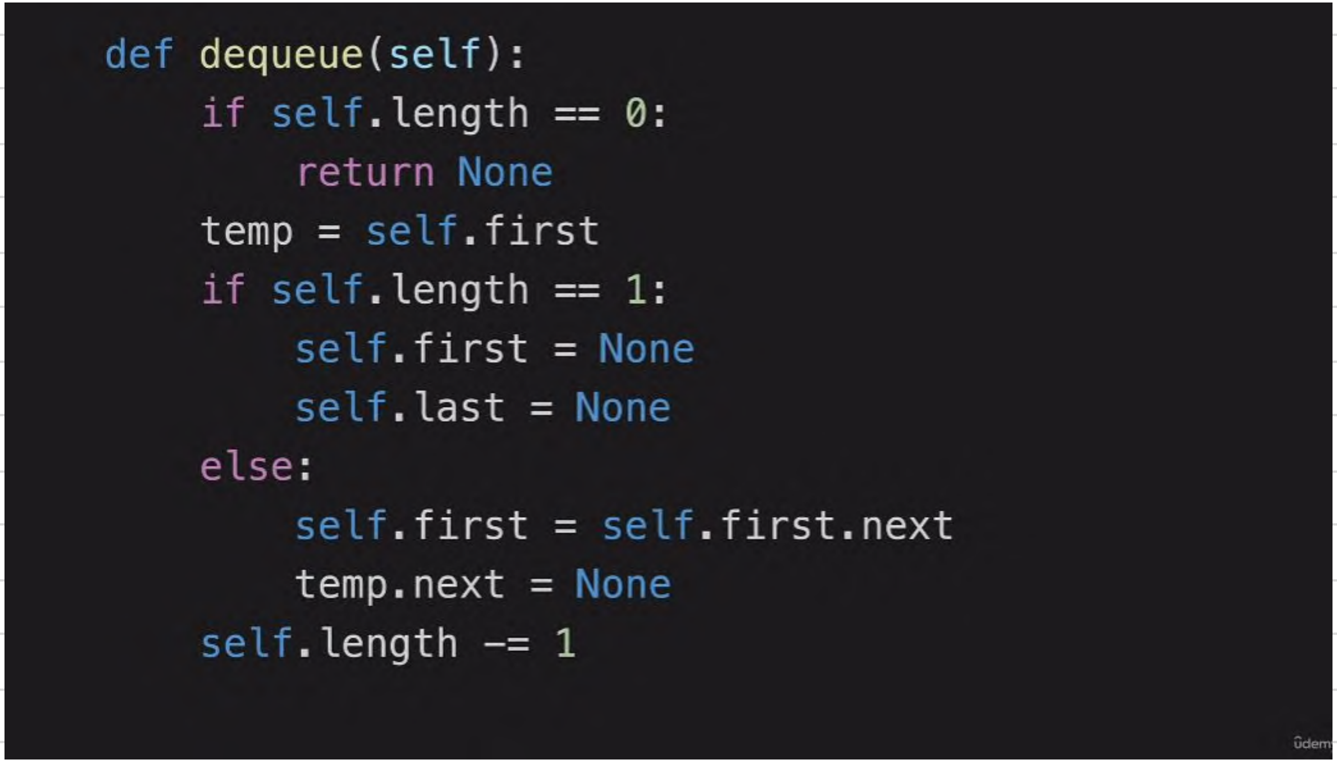
```
else:
    self.first = self.first.next
    temp.next = None
```

Queue : Dequeue Code



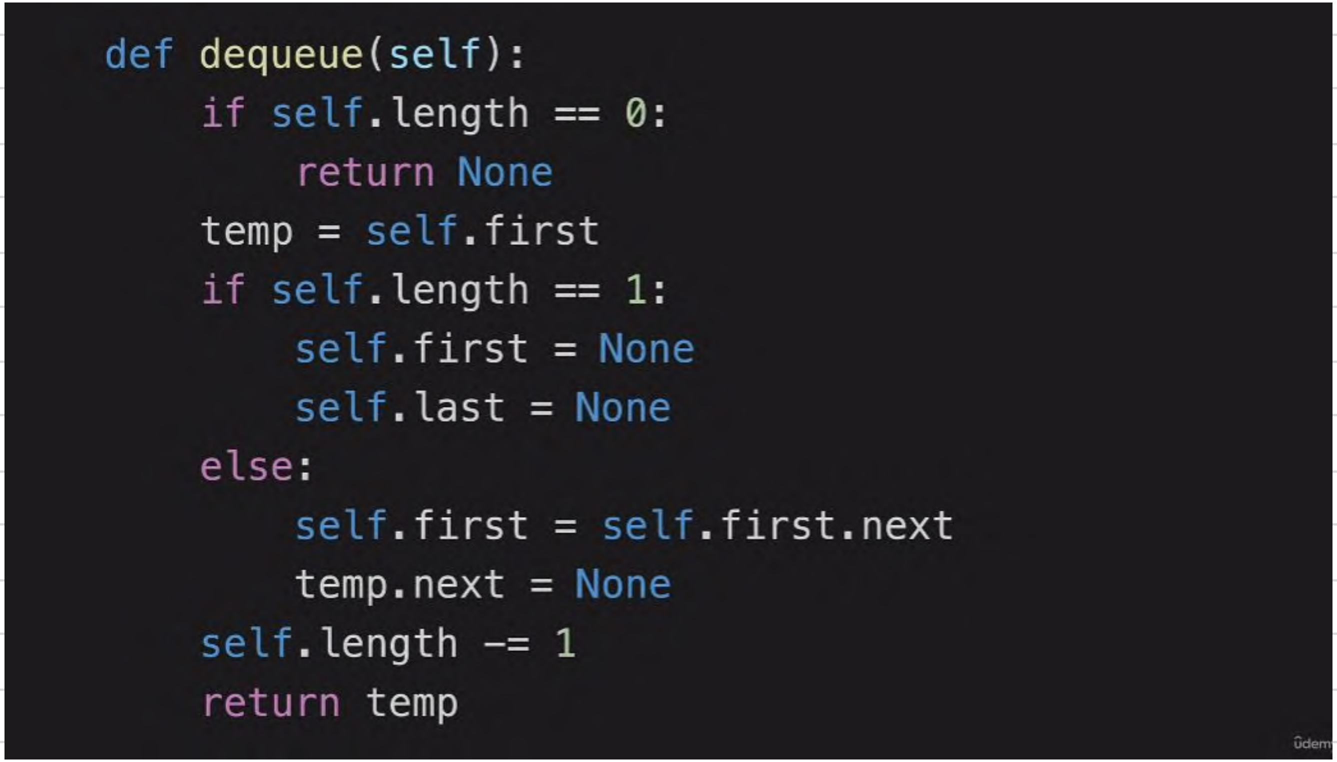
```
def dequeue(self):
    if self.length == 0:
        return None
    temp = self.first
    if self.length == 1:
        self.first = None
        self.last = None
    else:
        self.first = self.first.next
        temp.next = None
```

Queue : Dequeue Code



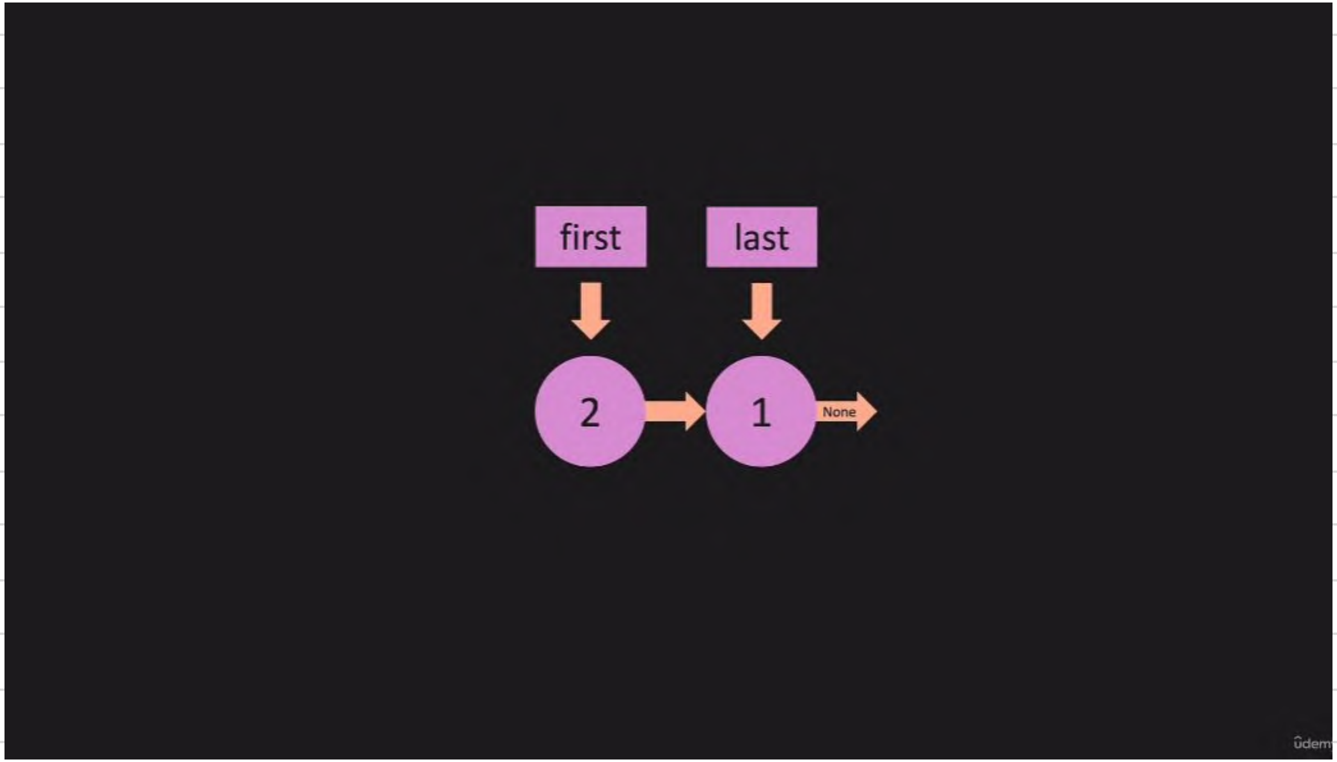
```
def dequeue(self):
    if self.length == 0:
        return None
    temp = self.first
    if self.length == 1:
        self.first = None
        self.last = None
    else:
        self.first = self.first.next
        temp.next = None
    self.length -= 1
```

Queue : Dequeue Code

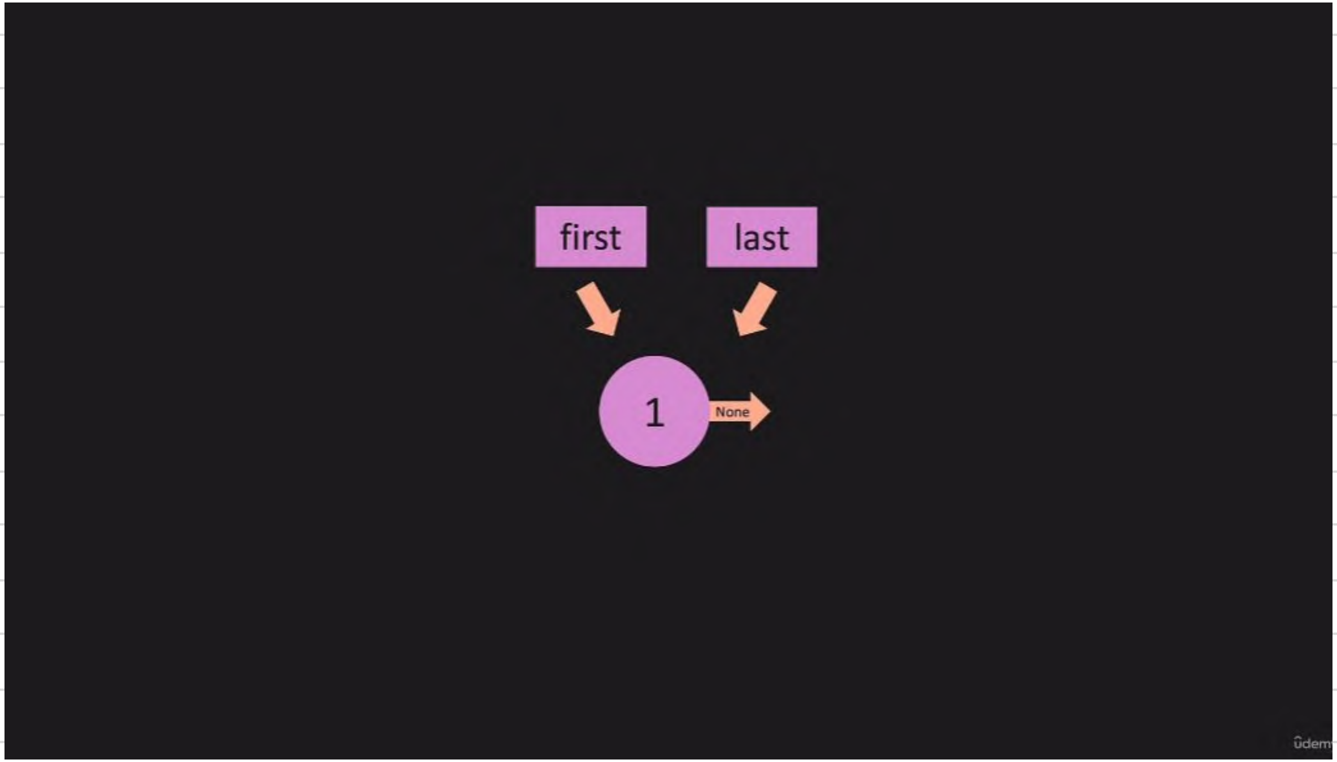


```python
def dequeue(self):
    if self.length == 0:
        return None
    temp = self.first
    if self.length == 1:
        self.first = None
        self.last = None
    else:
        self.first = self.first.next
        temp.next = None
    self.length -= 1
    return temp
```

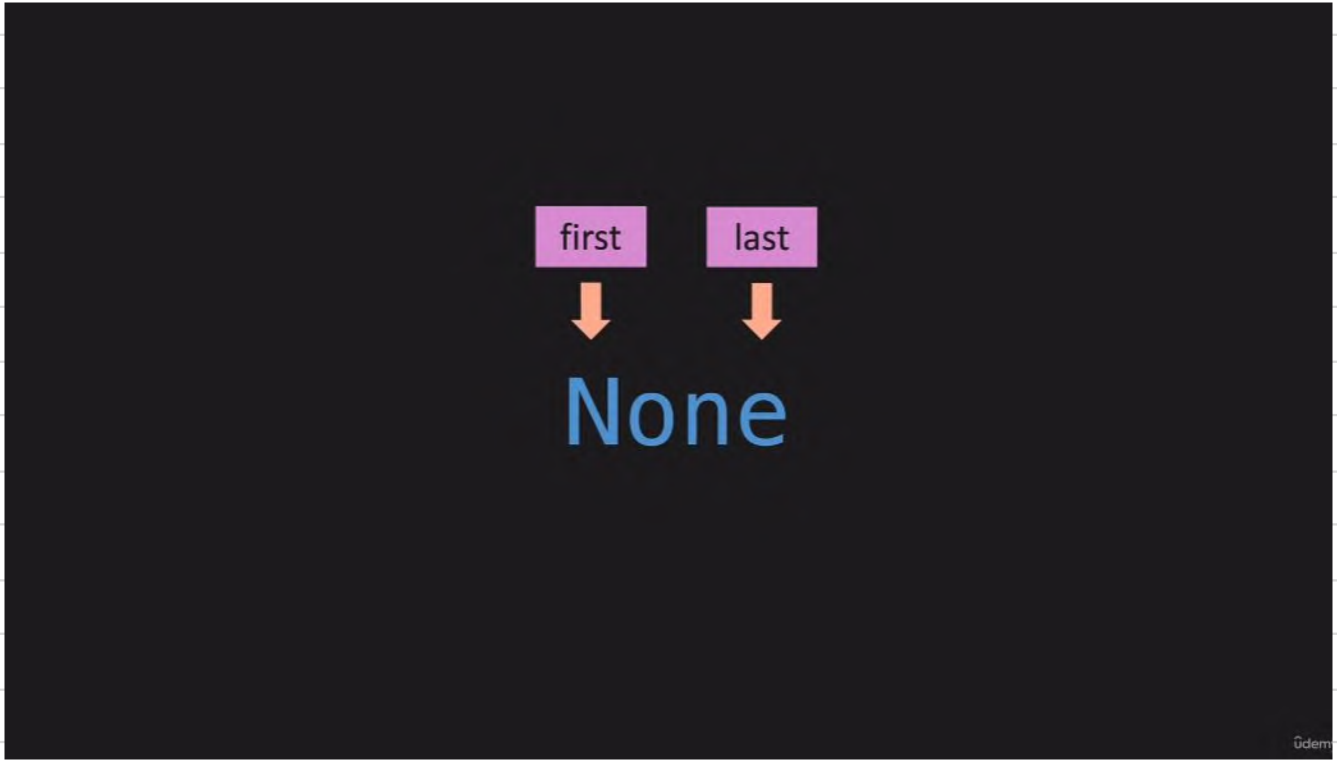
Queue : Dequeue Skenario Percobaan



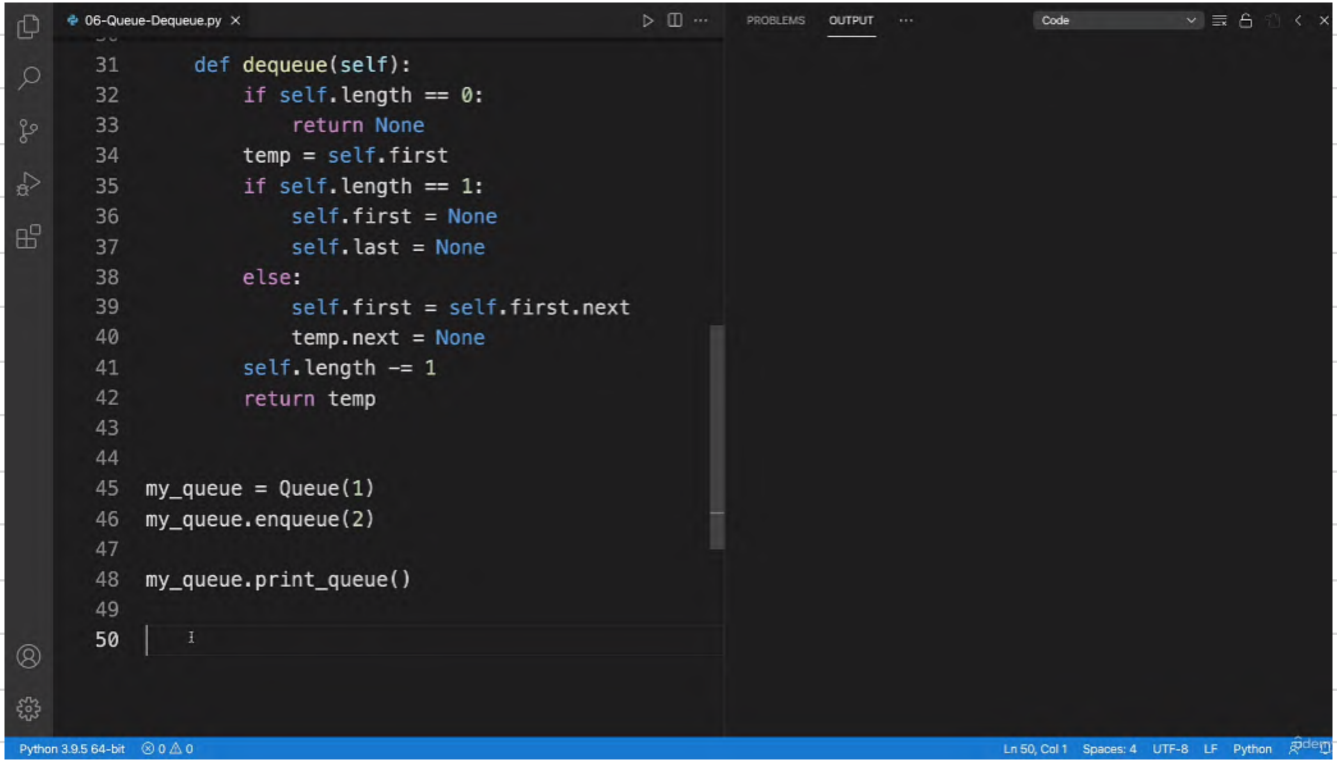
Queue : Dequeue Skenario Percobaan



Queue : Dequeue Skenario Percobaan

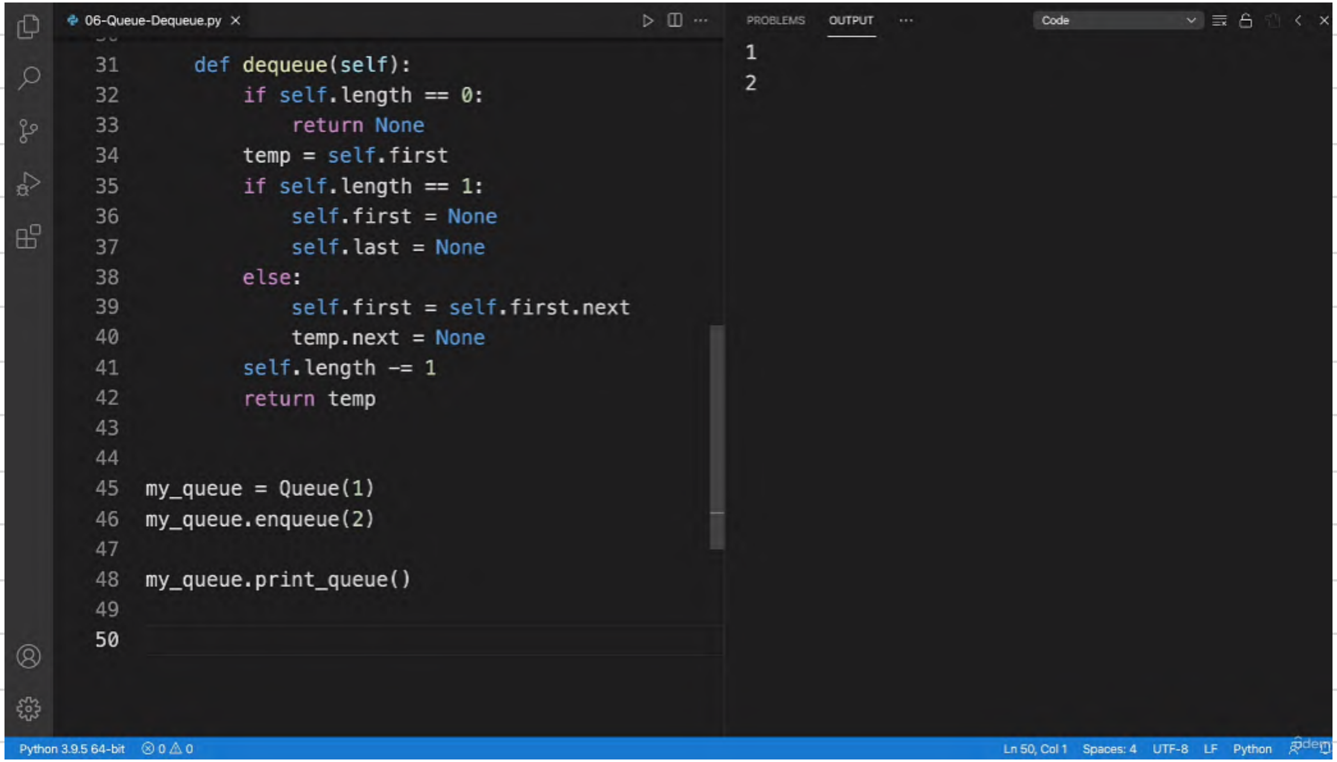


Queue : Dequeue Skenario Percobaan



```
    def dequeue(self):
        if self.length == 0:
            return None
        temp = self.first
        if self.length == 1:
            self.first = None
            self.last = None
        else:
            self.first = self.first.next
            temp.next = None
        self.length -= 1
        return temp


my_queue = Queue(1)
my_queue.enqueue(2)

my_queue.print_queue()

```

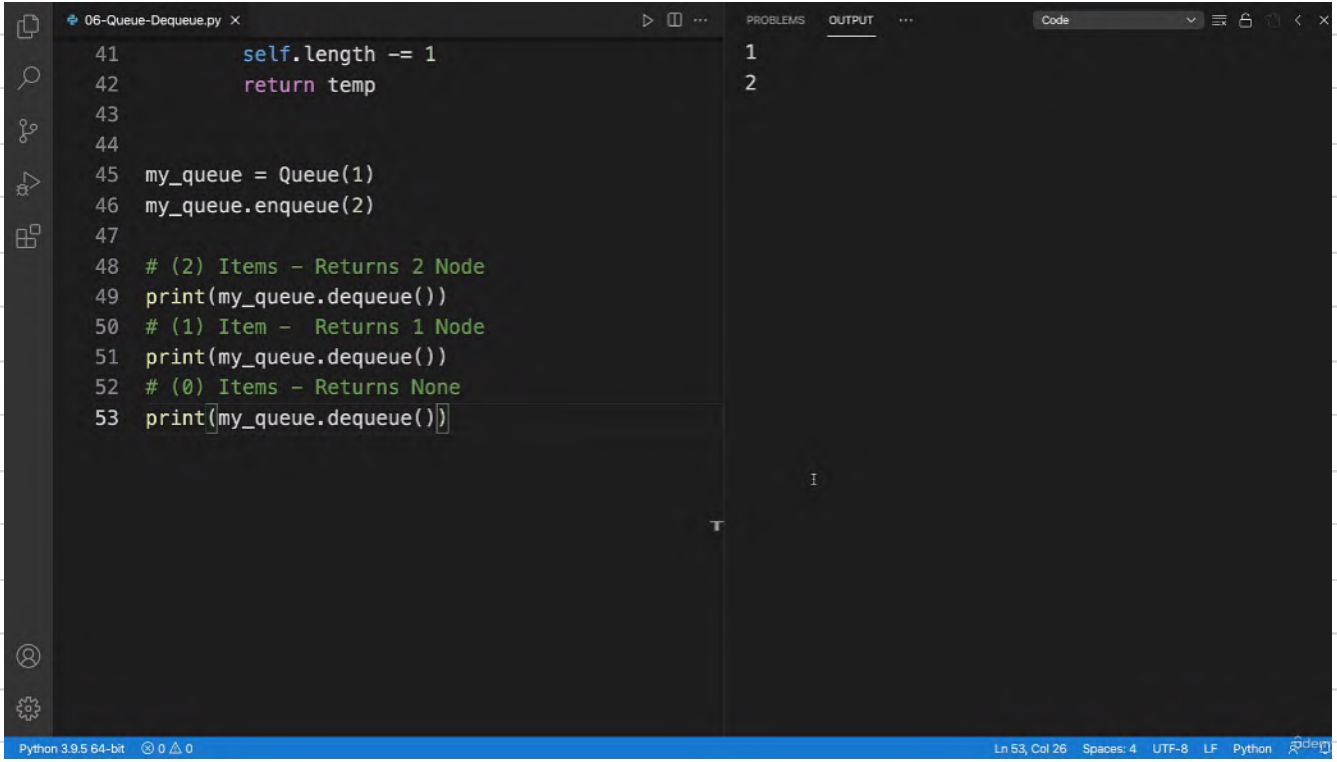
Queue : Dequeue Skenario Percobaan



```python
    def dequeue(self):
        if self.length == 0:
            return None
        temp = self.first
        if self.length == 1:
            self.first = None
            self.last = None
        else:
            self.first = self.first.next
            temp.next = None
        self.length -= 1
        return temp


my_queue = Queue(1)
my_queue.enqueue(2)

my_queue.print_queue()

```

```
1
2
```

Queue : Dequeue Skenario Percobaan

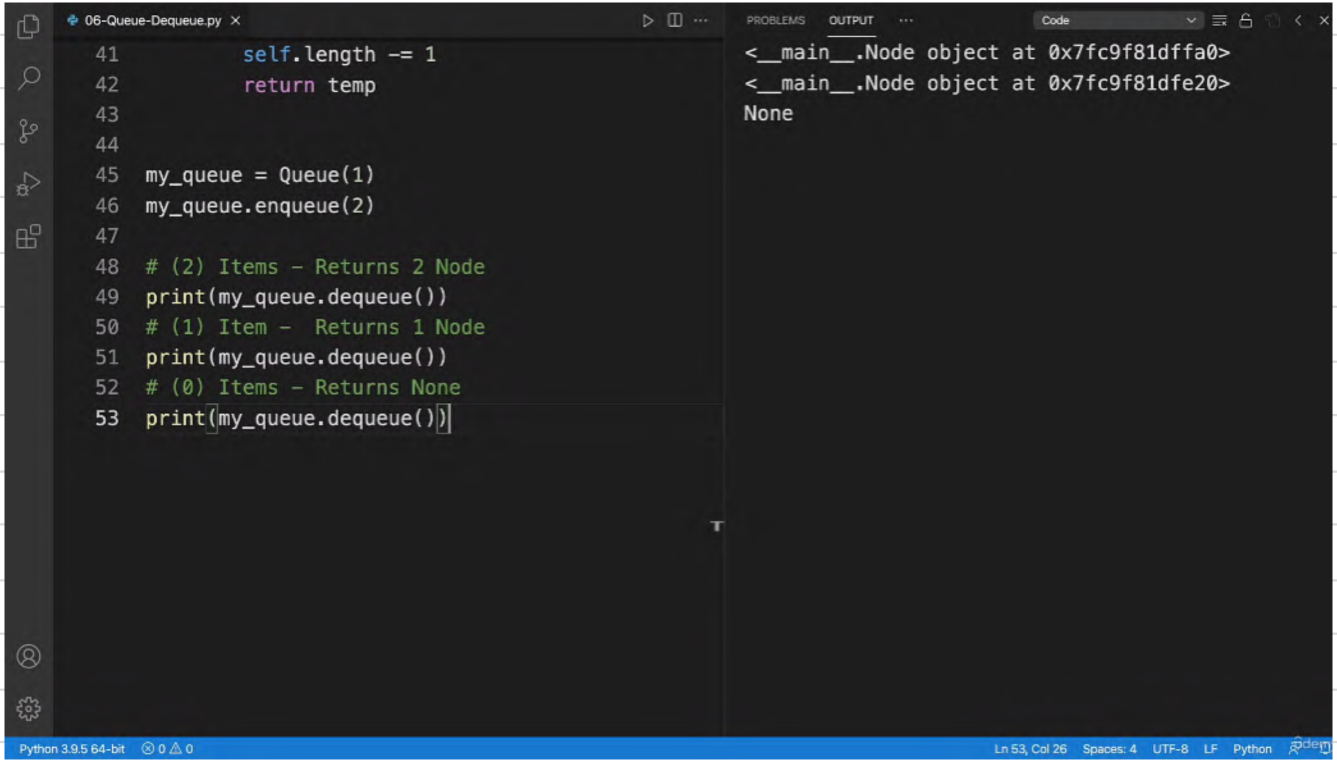


06-Queue-Dequeue.py

```python
        self.length -= 1
        return temp


my_queue = Queue(1)
my_queue.enqueue(2)

# (2) Items - Returns 2 Node
print(my_queue.dequeue())
# (1) Item -  Returns 1 Node
print(my_queue.dequeue())
# (0) Items - Returns None
print(my_queue.dequeue())
```

```
1
2
```

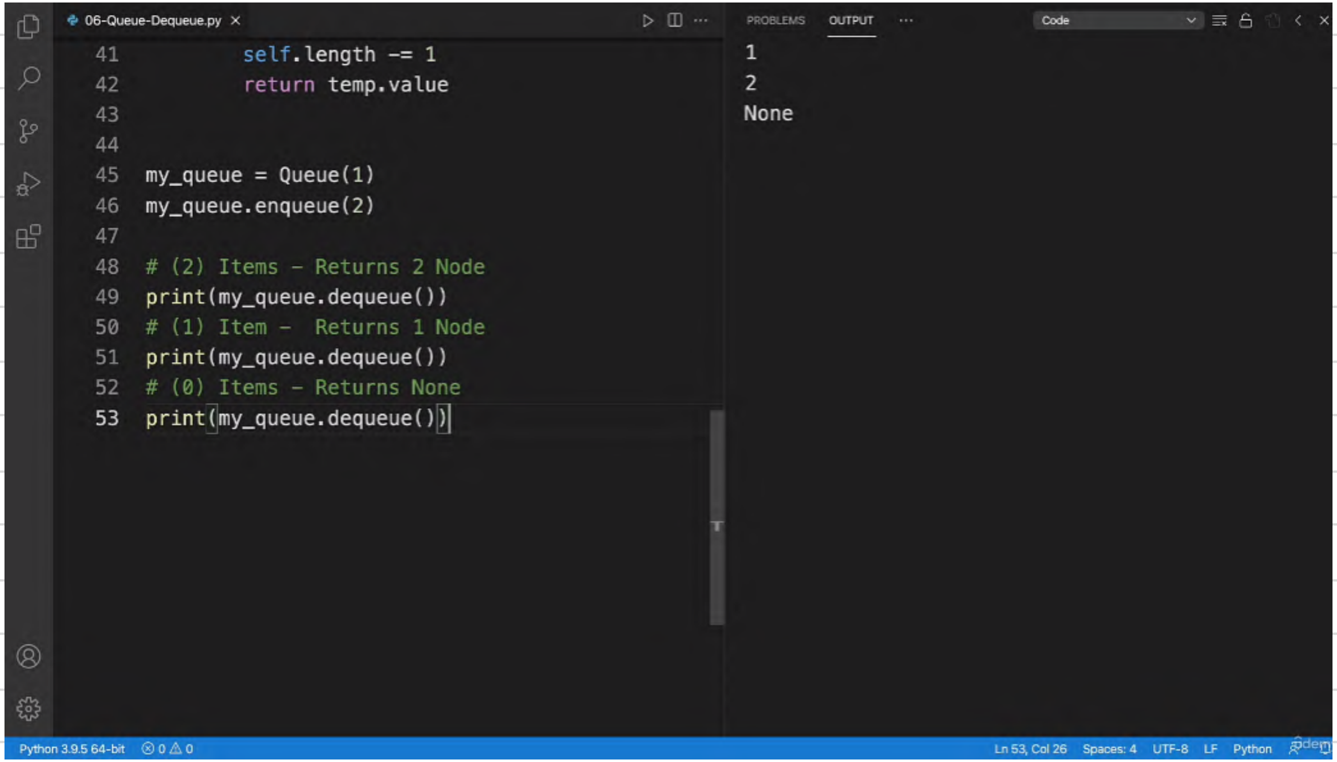
Queue : Dequeue Skenario Percobaan



```python
        self.length -= 1
        return temp


my_queue = Queue(1)
my_queue.enqueue(2)

# (2) Items - Returns 2 Node
print(my_queue.dequeue())
# (1) Item -  Returns 1 Node
print(my_queue.dequeue())
# (0) Items - Returns None
print(my_queue.dequeue())
```

```
<__main__.Node object at 0x7fc9f81dffa0>
<__main__.Node object at 0x7fc9f81dfe20>
None
```

Queue : Dequeue Skenario Percobaan



```python
            self.length -= 1
            return temp.value


my_queue = Queue(1)
my_queue.enqueue(2)

# (2) Items - Returns 2 Node
print(my_queue.dequeue())
# (1) Item -  Returns 1 Node
print(my_queue.dequeue())
# (0) Items - Returns None
print(my_queue.dequeue())
```

```
1
2
None
```

Terima Kasih

Ada **Pertanyaan?**