실습 5장 리스트-20231765 박종승

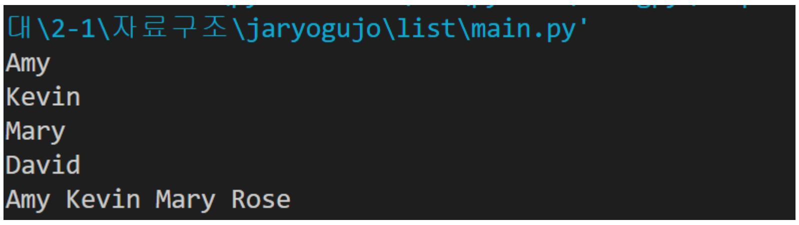
linkedListBasic.py

```python
from listNode import *

class LinkedListBasic:
    def __init__(self):
        self.__head = ListNode('dummy', None)
        self.__numItems = 0

    def insert(self, i:int, newItem):
        if i >= 0 and i <= self.__numItems:
            prev = self.__getNode( i - 1 )
            newNode = ListNode(newItem, prev.next)
            prev.next = newNode
            self.__numItems += 1
        else:
            print('index', i, ': out of bound in insert()')

    def append(self, newItem):
        prev = self.__getNode(self.__numItems - 1)
        newNode = ListNode(newItem, prev.next)
        prev.next = newNode
        self.__numItems += 1

    def pop(self, *args):

        if len(args) != 0:
            i = args[0]

        if len(args) == 0 or i == -1:
            i = self.__numItems - 1

        if (i >= 0 and i <= self.__numItems-1):
            prev = self.__getNode(i - 1)
            retItem = prev.next.item
            prev.next = prev.next.next
            self.__numItems -= 1
            return retItem
        else:
            return None

    def remove(self, x):
        (prev, curr) = self.__findNode(x)
        if curr != None:
            prev.next = curr.next
            self.__numItems -= 1
            return x
        else:
            return None

    def get(self, i:int):
        if self.isEmpty():
            return None
        if (i >= 0 and i <= self.__numItems - 1):
            return self.__getNode(i).item
        else:
            return None

    def index(self, x) -> int:
        curr = self.__head.next
        for index in range(self.__numItems):
            if curr.item == x:
                return index
            else:
                curr = curr.next
        return -2

    def isEmpty(self) -> bool:
        return self.__numItems == 0

    def size(self) -> int:
        return self.__numItems

    def clear(self):
        self.__head = ListNode('dummy', None)
        self.__numItems = 0

    def count(self, x) -> int:
        cnt = 0
        curr = self.__head.next
        while curr != None:
            if curr.item == x:
                cnt += 1
            curr = curr.next
        return cnt

    def extend(self, a):
        for index in range(a.size()):
            self.append(a.get(index))

    def copy(self):
        a = LinkedListBasic()
        for index in range(self.__numItems):
            a.append(self.get(index))
        return a
```

```python
def reverse(self):
    a = LinkedListBasic()
    for index in range(self.__numItems):
        a.insert(0, self.get(index))
    self.clear()
    for index in range(a.size()):
        self.append(a.get(index))

def sort(self) -> None:
    a = []
    for index in range(self.__numItems):
        a.append(self.get(index))
    a.sort()
    self.clear()
    for index in range(len(a)):
        self.append(a[index])

def __findNode(self, x):
    prev = self.__head
    curr = prev.next
    while curr != None:
        if curr.item == x:
            return (prev, curr)
        else:
            prev = curr; curr = curr.next
    return (None, None)

def __getNode(self, i:int) -> ListNode:
    curr = self.__head
    for index in range(i+1):
        curr = curr.next
    return curr

def printList(self):
    curr = self.__head.next
    while curr != None:
        print(curr.item, end = ' ')
        curr = curr.next
    print()

def __iter__(self):
    self.__iter_node = self.__head.next
    return self

def __next__(self):
    if self.__iter_node is not None:
        item = self.__iter_node.item
```

```python
            self.__iter_node = self.__iter_node.next
            return item
        else:
            raise StopIteration

    def __next__(self):
        if self.__iter_node is not None:
            item = self.__iter_node.item
            self.__iter_node = self.__iter_node.next
            return item
        else:
            raise StopIteration
```

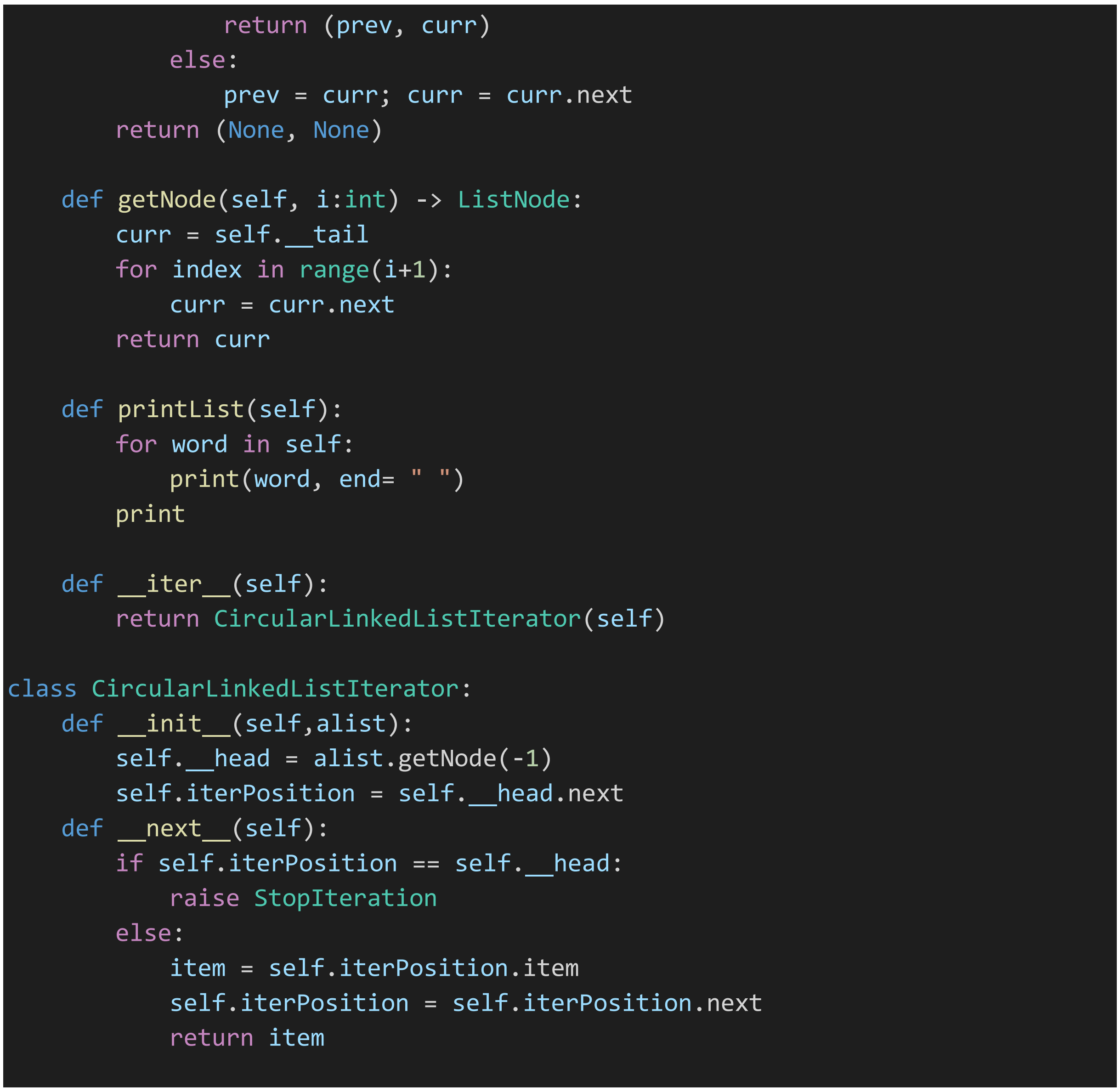
실행결과 :



```
대\2-1\자료구조\jaryogujo\list\main.py'
Amy
Kevin
Mary
David
Amy Kevin Mary Rose
```

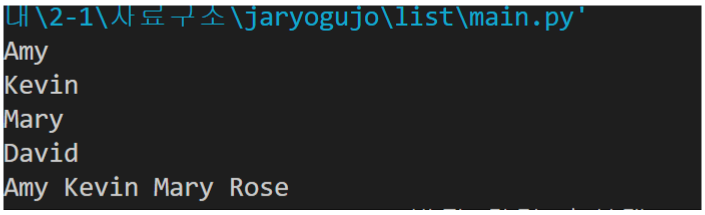
circularLinkedList.py

```python
from listNode import *

class CircularLinkedList:
    def __init__(self):
        self.__tail = ListNode('dummy', None)
        self.__tail.next = self.__tail
        self.__numItems = 0

    def insert(self, i:int, newItem):
        if i >= 0 and i <= self.__numItems:
            prev = self.getNode( i - 1 )
            newNode = ListNode(newItem, prev.next)
            prev.next = newNode
            self.__numItems += 1
        else:
            print('index', i, ': out of bound in insert()')

    def append(self, newItem):
        prev = self.getNode(self.__numItems - 1)
        newNode = ListNode(newItem, prev.next)
        prev.next = newNode
        self.__numItems += 1
    def pop(self, *args):

        if len(args) != 0:
            i = args[0]

        if len(args) == 0 or i == -1:
            i = self.__numItems - 1

        if (i >= 0 and i <= self.__numItems-1):
            prev = self.getNode(i - 1)
            retItem = prev.next.item
            prev.next = prev.next.next
            self.__numItems -= 1
            return retItem
        else:
            return None

    def remove(self, x):
        (prev, curr) = self.__findNode(x)
        if curr != None:
            prev.next = curr.next
            self.__numItems -= 1
            return x
        else:
            return None

    def get(self, i:int):
        if self.isEmpty():
            return None
        if (i >= 0 and i <= self.__numItems - 1):
            return self.getNode(i).item
        else:
            return None

    def index(self, x) -> int:
        curr = self.__head.next
        for index in range(self.__numItems):
            if curr.item == x:
                return index
            else:
                curr = curr.next
        return -2

    def isEmpty(self) -> bool:
        return self.__numItems == 0

    def size(self) -> int:
        return self.__numItems

    def clear(self):
        self.__tail = ListNode('dummy', None)
        self.__tail.next = self.__tail
        self.__numItems = 0

    def count(self, x) -> int:
        cnt = 0
        curr = self.__head.next
        while curr != None:
            if curr.item == x:
                cnt += 1
            curr = curr.next
        return cnt

    def extend(self, a):
        for index in range(a.size()):
            self.append(a.get(index))

    def copy(self):
        a = CircularLinkedList()
        for index in range(self.__numItems):
            a.append(self.get(index))
        return a

    def reverse(self):
        a = CircularLinkedList()
        for index in range(self.__numItems):
            a.insert(0, self.get(index))
        self.clear()
        for index in range(a.size()):
            self.append(a.get(index))

    def sort(self) -> None:
        a = []
        for index in range(self.__numItems):
            a.append(self.get(index))
        a.sort()
        self.clear()
        for index in range(len(a)):
            self.append(a[index])

    def __findNode(self, x):
        prev = self.__head
        curr = prev.next
        while curr != None:
            if curr.item == x:
```



```python
                return (prev, curr)
            else:
                prev = curr; curr = curr.next
        return (None, None)

    def getNode(self, i:int) -> ListNode:
        curr = self.__tail
        for index in range(i+1):
            curr = curr.next
        return curr

    def printList(self):
        for word in self:
            print(word, end= " ")
        print

    def __iter__(self):
        return CircularLinkedListIterator(self)

class CircularLinkedListIterator:
    def __init__(self,alist):
        self.__head = alist.getNode(-1)
        self.iterPosition = self.__head.next
    def __next__(self):
        if self.iterPosition == self.__head:
            raise StopIteration
        else:
            item = self.iterPosition.item
            self.iterPosition = self.iterPosition.next
            return item
```

실행결과:



```
내\2-1\자료구조\jaryogujo\list\main.py'
Amy
Kevin
Mary
David
Amy Kevin Mary Rose
```