

```

class Node:
    def __init__(self, item):
        self.item = item
        self.link = None

class CircularLinkedList:
    def __init__(self):
        self.root = None
        self.current = self.root

    def append(self, item):
        # 새로운 노드를 만든다.
        newnode = Node(item)
        # 부트가 비어 있으면 부트와 테일을 새 노드로 지정한다.
        if self.root is None:
            self.root = newnode
            self.tail = newnode
            newnode.link = self.root
        # 부트가 비어 있지 않으면 테일 뒤에 새 노드를 추가하고 테일을 업데이트한다.
        else:
            self.tail.link = newnode
            self.tail = newnode
            newnode.link = self.root

    # 리스트 안에 모든 노드를 프린트 한다.
    def print(self):
        curNode = self.root
        while curNode.link != self.root:
            print(curNode.item)
            curNode = curNode.link
        print(curNode.item)

    # 리스트에 존재하는 노드 수를 구한다.
    def count(self):
        curNode = self.root
        cnt = 1
        while curNode.link != self.root:
            curNode = curNode.link
            cnt += 1
        return cnt

    # 리스트에 item을 가지는 노드를 current 노드로 지정한다.
    def setCurrent(self, item):
        curNode = self.root
        if item in self.listSize():
            self.current = curNode
            curNode.item = item
            curNode.link = curNode.link
        else:
            curNode = curNode.link

    # current 노드를 다음 노드로 바꾼다.
    def nextNode(self):
        self.current = self.current.link
        print("현재 위치는 ", self.current.item, "입니다.")

    # current 다음에 item을 삽입한다.
    def insert(self, item):
        # current가 None이면
        if self.current is None:
            # current 노드의 링크에 새 노드를 지정하고 새 노드의 링크에 current 노드가 연결되어 있던 링크를 지정하여 연결한다.
            newnode = Node(item)
            self.current.link = newnode
            newnode.link = self.root
        # 만약 item이 tail 노드 였다면 새 노드를 tail 노드로 재지정한다.
        elif self.current == self.tail:
            self.tail = newnode
            self.link = newnode

    # item을 가지는 노드를 삭제한다.
    def delete(self, item):
        delNode = None
        # 부트가 삭제 대상 노드인지 확인한다. 부트를 삭제할 경우, 부트 다음 노드를 부트로 결산하고 tail 노드의 링크를 결산한 부트로 재지정한다.
        if self.root == item:
            curNode = self.root
            curNode.item = item
            self.root = self.nextNode()
            self.tail.link = self.root
            delNode = self.root
        # 부트가 삭제 대상이 아닐 경우, 끝까지 탐색하면서 item이 존재하는 노드를 찾는다.
        # 찾았다면 preNode의 링크에 대상노드 링크를 넘겨준다. 만약, 대상 노드가 tail 노드라면 tail 노드를 preNode로 변경한다.
        else:
            while curNode.link != self.root:
                preNode = curNode
                curNode = curNode.link
                if curNode.item == item:
                    preNode.link = curNode.link
                    if curNode == self.tail:
                        self.tail = preNode
                    delNode = curNode
            # 삭제된 것이 맞으면 error message를 출력한다.
            if delNode == False:
                print("delete failed")

# fruits = CircularLinkedList()
# fruits.append('사과')
# fruits.append('딸기')
# fruits.append('오렌지')
# fruits.append('포도')

# fruits.print()
# print(fruits.listSize())
# fruits.setCurrent('포도')
# fruits.insert('오렌지')
# print(fruits.tail.item)

# fruits.setCurrent('오렌지')
# for i in range(10):
#     fruits.nextNode()

# fruits.delete('포도')
# fruits.setCurrent('오렌지')
# for i in range(10):
#     fruits.nextNode()

```

① self.tail
 ② self.current
 마지막에 있는 변수명.
 self.root
 Circular linked list 내 마지막 노드의 link에는 root 노드가 저장된다. **핵심!**
핵심!!!
 끝 부분이 새로운 노드를 추가하고, 이것을 'self.tail' 변수에 저장해야함!!!
 Circular linked list란 가지는 어노드.
 일반적인 linked list의 insert 메소드는 다르게, current 노드 뒤에 새로운 노드를 삽입한다.

끝 부분에 새로운 노드가 추가된 것이라면,
 이 새로운 노드를 'self.tail'에 저장해야함!!!

```

# item을 가지는 노드를 삭제한다.
def delete(self, item):
    delNode = None
    # 부트가 삭제 대상 노드인지 확인한다. 부트를 삭제할 경우, 부트 다음 노드를 부트로 결산하고 tail 노드의 링크를 결산한 부트로 재지정한다.
    if self.root == item:
        curNode = self.root
        curNode.item = item
        self.root = self.nextNode()
        self.tail.link = self.root
        delNode = self.root
    # 부트가 삭제 대상이 아닐 경우, 끝까지 탐색하면서 item이 존재하는 노드를 찾는다.
    # 찾았다면 preNode의 링크에 대상노드 링크를 넘겨준다. 만약, 대상 노드가 tail 노드라면 tail 노드를 preNode로 변경한다.
    else:
        while curNode.link != self.root:
            preNode = curNode
            curNode = curNode.link
            if curNode.item == item:
                preNode.link = curNode.link
                if curNode == self.tail:
                    self.tail = preNode
                delNode = curNode
        # 삭제된 것이 맞으면 error message를 출력한다.
        if delNode == False:
            print("delete failed")

# fruits = CircularLinkedList()
# fruits.append('사과')
# fruits.append('딸기')
# fruits.append('오렌지')
# fruits.append('포도')

# fruits.print()
# print(fruits.listSize())
# fruits.setCurrent('포도')
# fruits.insert('오렌지')
# print(fruits.tail.item)

# fruits.setCurrent('오렌지')
# for i in range(10):
#     fruits.nextNode()

# fruits.delete('포도')
# fruits.setCurrent('오렌지')
# for i in range(10):
#     fruits.nextNode()

```

del 메소드에 사용되는 변수
 만약, 맨 마지막 노드를 지운 거라면,
 맨 마지막 전의 노드를 'self.tail' 변수에 저장해야함.