

PL Assignment #02 : Make Linked List

과제물 부과일 : 2016-03-09(목)
Program Upload 마감일 : 2016-03-15(수) 23:59:59

문제

임의의 character 문자열을 저장하고 있는 배열을 읽어서 주어진 자료구조를 이용하여 List를 Python으로 생성하시요.

예를 들어, 입력 배열이 다음과 같으면,

```
['a', 'p', 'w']
```

생성되는 linked list는 다음과 같다.

```
[a] -> [p] -> [w] -> None
```

(자바의 null이 파이썬에서는 None이다.)

(즉, 하나의 linked list는 다수의 노드로 구성되며, 각 노드는 데이터를 의미하는 element와 다음 노드를 가리키는 next를 갖는다. 주어진 linked list의 마지막 노드의 next는 None이 된다. 자세한 자료구조는 뒤의 python 코드를 참고하시오.)

과정

주어진 아래와 같은 함수들의 구현을 완성하시요.

<필수로 완성할 함수>

```
(1) def _link_last(self, element, node):  
(2) def _get_node(self, index, x):  
(3) def __str__(self):  
(4) def __len__(self):  
(5) def _reverse(self, x, pred):
```

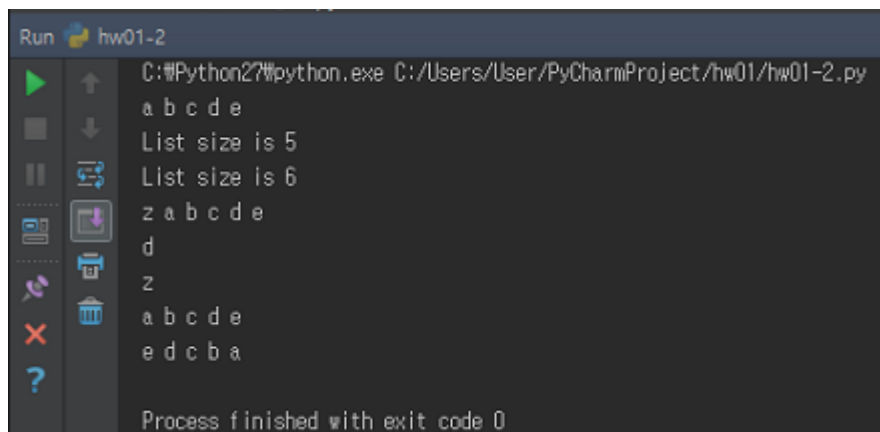
주의

- “Iteration을 절대 사용하지 마시오. 즉, for, while, goto 등이 나타나면 0점 처리함”
- 주어진 Class에서 새로운 함수나 변수를 절대 추가하지 마시오.
- 주어진 코드에서 새로운 함수나 전역 변수를 절대 추가하지 마시오.
- 주어진 양식을 따르지 않아 실행이 되지 않을 경우, 감점한다.
- 기타 과제 제출에 관한 구체적인 제반 사항은 각 TA의 지침에 따른다.

테스트 예

```
def test_recursion_linked_list():  
    INPUT = ['a', 'b', 'c', 'd', 'e']  
    test_RLL = RecursionLinkedList()  
    for i in INPUT:  
        test_RLL.add(i)  
    print str(test_RLL)  
    print "List size is " + str(len(test_RLL))  
    test_RLL.add('z', 0)  
    print "List size is " + str(len(test_RLL))  
    print str(test_RLL)  
    print test_RLL.get_node(4).element  
    print test_RLL.remove(0)  
    print str(test_RLL)  
    test_RLL.reverse()  
    print str(test_RLL)
```

<결과>



```
Run hw01-2  
C:\Python27\python.exe C:/Users/User/PyCharmProject/hw01/hw01-2.py  
a b c d e  
List size is 5  
List size is 6  
z a b c d e  
d  
z  
a b c d e  
e d c b a  
  
Process finished with exit code 0
```

1. RecursionLinkedList.py : List를 나타내는 클래스

```
class RecursionLinkedList(object):

    def __init__(self):
        self.head = None

    def _link_first(self, element):
        # connect newly created node the beginning of the list
        if self is not None:
            self.head = Node(element, self.head)
        else:
            self.head = Node(element, None)

    def _link_last(self, element, node):
        """
        :type node: Node
        :type element: char
        """
        # assignment(1) connect given node the next of the last linked node
        if node.next is None: # create next node

            else: # visit next node

    def _link_next(self, index, element):
        """
        :type index: Int
        :type element: Char
        :param index
        :param element
        """
        next = self.get_node(index).next
        self.get_node(index).next = Node(element, next)

    def _unlink_first(self):
        # unlink first node of list
        x = self.head
        """ :type : Node """
        element = x.element
        self.head = x.next
        return element

    def _unlink_next(self, pred):
        """
        :type pred: Node
        :param pred:
        """
        x = pred.next
        element = x.element
        pred.next = x.next
        return element

    def _get_node(self, index, x):
        """
        :type index:int
        :type x:Node
        :param index:
        :param x:
        """
        # assignment(2) Get nth(index) node
        if index == 0: # return current node

            elif index > 0: # return result of call _get_node
```

```

def get_node(self, index):
    return self._get_node(index, self.head)

def __len__(self):
    if self.head is None:
        return 0
    return len(self.head)

def add(self, element, index=None):
    if index is None:
        if self.head is None:
            self._link_first(element)
        else:
            self._link_last(element, self.head)
        return

    if index < 0 or index > len(self):
        print "ERROR"
    elif index == 0:
        self._link_first(element)
    else:
        self._link_next(index-1, element)

def remove(self, index):
    if index < 0 or index > len(self):
        print "ERROR"
    elif index == 0:
        return self._unlink_first()
    else:
        return self._unlink_next(self._get_node(index - 1, self.head))

def __str__(self):
    if self.head is None:
        return "List is null"
    return str(self.head)

def _reverse(self, x, pred):
    """
    :type x: Node
    :type pred: Node
    :param x:
    """
    # assignment(5)
    # Fill out, Use recursion

def reverse(self):
    self._reverse(self.head, None)

class Node(object):
    """
    :type element:char
    :type next: Node
    """

    def __init__(self, element, next):
        """
        :type element : char
        :type next : Node
        """
        self.element = element
        self.next = next

```

```

def __str__(self):
    # assignment(3)
    if self.next is None: # Return string of self.element

    else:
        # Return self.element and string of next

def __len__(self):
    # assignment(4) Return size of entire node


def test_recursion_linked_list():
    INPUT = ['a', 'b', 'c', 'd', 'e']
    test_RLL = RecursionLinkedList()
    for i in INPUT:
        test_RLL.add(i)
    print str(test_RLL)
    print "List size is " + str(len(test_RLL))
    test_RLL.add('z', 0)
    print "List size is " + str(len(test_RLL))
    print str(test_RLL)
    print test_RLL.get_node(4).element
    print test_RLL.remove(0)
    print str(test_RLL)
    test_RLL.reverse()
    print str(test_RLL)

test_recursion_linked_list()

```

PL Assignment #1-3 : Selection Sort Recursion (Bonus Assignment)

문제

1-2에서 했던 링크드리스트를 Selection Sort 하시오.

예를 들어, 입력 배열이 다음과 같으면,

[4, 6, 3]

생성되는 linked list는 다음과 같다.

[4] -> [6] -> [3] -> None

이를 정렬하면

[3] -> [4] -> [6] -> None

과정

주어진 아래와 같은 함수들의 구현을 완성하시오.

<완성할 함수>

```
(1) def _selection(self, current_node):  
(2) def compare(self, current_node, compare_node, min_node):
```

주의

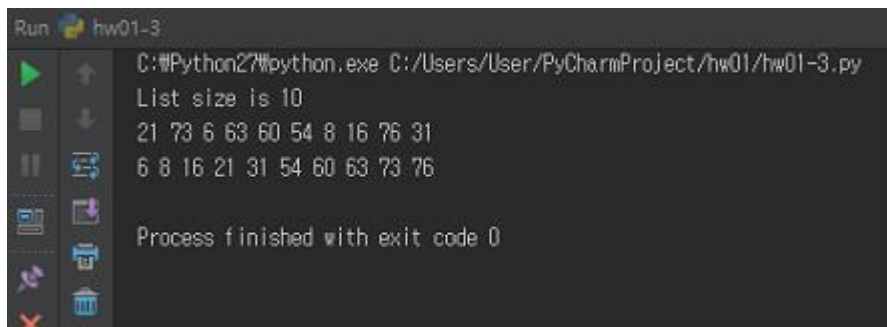
- “Iteration을 절대 사용하지 마시오. 즉, for, while, goto 등이 나타나면 0점 처리함”
- 주어진 Class에서 새로운 함수나 변수를 절대 추가하지 마시오.
- 주어진 코드에서 새로운 함수나 전역 변수를 절대 추가하지 마시오.
- 주어진 양식을 따르지 않아 실행이 되지 않을 경우, 감점한다.
- 기타 과제 제출에 관한 구체적인 제반 사항은 각 TA의 지침에 따른다.

테스트 예

```
def test_selection_sort():
    random_numbers=[]
    for i in range(10):
        random_numbers.append(random.randrange(0, 100))

    test_RSS = RecursionLinkedList()
    for i in random_numbers:
        test_RSS.add(i)
    print "List size is " + str(len(test_RSS))
    print str(test_RSS)
    #test_RLL.iter_selection_sort()
    test_RSS.iter_selection_sort()
    print str(test_RSS)
```

<결과>



```
Run hw01-3
C:\Python27\python.exe C:/Users/User/PyCharmProject/hw01/hw01-3.py
List size is 10
21 73 6 63 60 54 8 16 76 31
6 8 16 21 31 54 60 63 73 76

Process finished with exit code 0
```

1. 주어진 코드를 RLinkedList.py 에 추가한다.

RecursionLinkedList에 추가해야 할 함수

```
def iter_selection_sort(self):
    current_node = self.head
    compare_node = self.head
    min_node = self.head
    while current_node.next is not None:
        while compare_node is not None:
            if min_node.element > compare_node.element:
                min_node = compare_node
            compare_node = compare_node.next
        current_node.element, min_node.element = min_node.element, current_node.element
        current_node = current_node.next
        compare_node = current_node
        min_node = current_node

def selection_sort(self):
    self._selection(self.head)

# Bonus Assignment
def _selection(self, current_node):
    # Fill out, Use recursion

def compare(self, current_node, compare_node, min_node):
    # Fill out, Use recursion
```

➤ Import random 추가 할 것!