Description of functions for linear singly linked list with references to head and tail

Полежанов Богдан

Февраль 2020

Список всех функций:

1 add_to_back(value,head,tail)

Метод добавляет значение **value** в конец связанного списка и возвращает значения **head** и **tail**

Функция 1: add_to_back

```
def add_to_back(value,head,tail):
    item=[value,None]
    if head is None:
        head=item
    else:
        tail[NEXT]=item
    tail=item
    return head,tail
```

2 add_to_front(value,head,tail)

Метод добавляет значение **value** в начало связанного списка и возвращает значения **head** и **tail**

Функция 2: add_to_front(value,head,tail)

```
def add_to_front(value,head,tail):
    item=[value,None]
    if head is None:
        head=item
        tail=item
    else:
        head=[value,head]
    return head,tail
```

3 remove_from_back(head,tail)

Метод удаляет последний элемент связанного списка и возвращает значения **head** и **tail**

Функция 3: remove_from_back

```
def remove_from_back(head,tail):
    if tail is None:
        print('List is Empty')
    elif head==tail:
        head=tail=None
    else:
        old=current=head
        while current is not tail:
            old=current
            current=current[NEXT]
        old[NEXT]=None
        tail=old
    return head,tail
```

4 remove_from_front(head,tail)

Метод удаляет первый элемент связанного списка и возвращает значения **head** и **tail**

Функция 4: remove_from_front

```
def remove_from_front(head,tail):
    if head is None:
        print('List is Empty')
    elif head==tail:
        head=tail=None
    else:
        head=head[NEXT]
    return head,tail
```

5 get_element_at(value, head, tail)

Метод возвращает значение по индексу **value**. Если элемента с индексом **value** не существует в списке, на консоль выволится соответсвующее сообщение, а возвращаемое значение будет **None**

Функция 5: get_element_at

```
def get_element_at(value, head, tail):
    int(value)
    if (value>length(head,tail)-1) or (value<0):
        print('Element at '+str(value)+' not found! Value is
            None')
        return None
    else:
        item=head
        for i in range(value):
            item=item[NEXT]
        return item[VALUE]
```

6 find(value,head,tail)

Метод выводит на консоль индекс первого вхождения элемента **value** в список. Если элемент не был найден, на консоль выводится соответствующее сообщение.

Функция 6: find

```
def find(value,head,tail):
    current=head
    for i in range(length(head,tail)):
        if value==current[VALUE]:
            print('Value '+str(value)+' was found at index = '
                +str(i))
            return
        current=current[NEXT]
    print('Value '+str(value)+' was not found!')
```

7 find_all(value,head,tail)

Метод выводит на консоль индексы всех вхождений элемента **value** в список. Если элемент не был найден, на консоль выводится соответствующее сообщение.

Функция 7: find_all

```
def find_all(value,head,tail):
    current=head
    success=False
    for i in range(length(head,tail)):
        if value==current[VALUE]:
            print('Value '+str(value)+' was found at index = '
                +str(i))
        current=current[NEXT]
        success=True
    if not success:
        print('Value '+str(value)+' was not found!')
```

8 print_elements(head,tail)

Метод выводит значения всех элементов в списке друг за другом, каждый с новой строки

Функция 8: print_elements

```
def print_elements(head,tail):
    if head is None:
        print('List is empty!')
        return
    item=head
    while item is not tail:
        print(item[VALUE])
        item=item[NEXT]
    print(item[VALUE])
```

9 length(head,tail)

Возвращает количество элементов в списке

Функция 9: length

```
def length(head,tail):
    if head is None:
        return 0
    item=head
    count=1
    while item is not tail:
        count+=1
        item=item[NEXT]
    return int(count)
```