## 2. Двунаправленный связанный список

Обычный список мы можем проходить только в одну сторону, от головы к хвосту, но имеется немало задач, когда желательно оставаться в рамках модели связанного списка, но бегать по нему в обе стороны. Некоторые операции в результате становятся ещё более эффективными (например, удаление узла), однако приходится выделять дополнительную "ячейку" памяти под второй указатель.

Общая схема работы двунаправленного связанного списка (класс LinkedList2) сохраняется как в прошлом уроке.

Концепция узла расширяется полем prev, который теперь указывает на предыдущий элемент в списке.

**class** **Node**:

**def** **\_\_init\_\_**(self, v):

self.value = v

self.prev = **None**

self.next = **None**

n1 = Node(**12**)

n2 = Node(**55**)

n1.next = n2 # 12 -> 55

n2.prev = n1

Немного изменить нам потребуется лишь метод add\_in\_tail():

**class** **LinkedList2**:

**def** **\_\_init\_\_**(self):

self.head = **None**

self.tail = **None**

**def** **add\_in\_tail**(self, item):

**if** self.head **is** **None**:

self.head = item

item.prev = **None**

item.next = **None**

**else**:

self.tail.next = item

item.prev = self.tail

self.tail = item

[[ предыдущее занятие ]](http://skillsmart.ru/algo/py-kf32y/fb50h6802e1s.html)

**Задания.**  
2.1. Добавьте в класс LinkedList2 метод поиска первого узла по его значению.

find(val)

2.2. Добавьте в класс LinkedList2 метод поиска всех узлов по конкретному значению (возвращается список найденных узлов).

find\_all(val)

2.3. Добавьте в класс LinkedList2 метод удаления одного узла по его значению.

delete(val, all=False)

где флажок all=False по умолчанию -- удаляем только первый нашедшийся элемент.

2.4. Дополните этот метод удалением всех узлов по конкретному значению (флажок all=True).

2.5. Добавьте в класс LinkedList2 метод вставки узла после заданного узла.

insert(afterNode, newNode)

Если afterNode = None и список пустой, добавьте новый элемент первым в списке.  
Если afterNode = None и список непустой, добавьте новый элемент последним в списке.

2.6. Добавьте в класс LinkedList2 метод вставки узла самым первым элементом.

add\_in\_head(newNode)

2.7. Добавьте в класс LinkedList2 метод очистки всего содержимого (создание пустого списка) -- clean()

2.8. Добавьте в класс LinkedList2 метод вычисления текущей длины списка -- len()

2.9. Напишите проверочные тесты для каждого из предыдущих заданий.

## заготовка класса для автоматической проверки

**class** **Node**:

**def** **\_\_init\_\_**(self, v):

self.value = v

self.prev = **None**

self.next = **None**

**class** **LinkedList2**:

**def** **\_\_init\_\_**(self):

self.head = **None**

self.tail = **None**

**def** **add\_in\_tail**(self, item):

**if** self.head **is** **None**:

self.head = item

item.prev = **None**

item.next = **None**

**else**:

self.tail.next = item

item.prev = self.tail

self.tail = item

**def** **find**(self, val):

**return** **None** # здесь будет ваш код

**def** **find\_all**(self, val):

**return** [] # здесь будет ваш код

**def** **delete**(self, val, all=**False**):

**pass** # здесь будет ваш код

**def** **clean**(self):

**pass** # здесь будет ваш код

**def** **len**(self):

**return** **0** # здесь будет ваш код

**def** **insert**(self, afterNode, newNode):

**pass** # здесь будет ваш код

**def** **add\_in\_head**(self, newNode)

**pass** # здесь будет ваш код