11.6 Методы списков. Часть 2

Шаг 1

Тема урока: методы списков

1. Метод `insert()`
2. Метод `index()`
3. Метод `remove()`
4. Метод `pop()`
5. Метод `reverse()`
6. Метод `count()`
7. Метод `clear()`
8. Метод `copy()`
9. Метод `sort()`

Аннотация. Другие методы списков.

Мы уже познакомились с двумя списочными методами `append()` и `extend()`. Первый добавляет в конец списка один новый элемент, а второй расширяет список другим списком. К спискам в Python применимы и другие удобные методы, с которыми мы познакомимся в этом уроке.

Метод insert()

Метод `insert()` позволяет вставлять значение в список в заданной позиции. В него передается два аргумента:

1. `index` : индекс, задающий место вставки значения;
2. `value` : значение, которое требуется вставить.

Когда значение вставляется в список, список расширяется в размере, чтобы разместить новое значение. Значение, которое ранее находилось в заданной индексной позиции, и все элементы после него сдвигаются на одну позицию к концу списка.

Приведённый ниже код:

```
names = ['Gvido', 'Roman', 'Timur']
print(names)
names.insert(0, 'Anders')
print(names)
names.insert(3, 'Josef')
print(names)
```

выводит:

```
['Gvido', 'Roman', 'Timur']
['Anders', 'Gvido', 'Roman', 'Timur']
['Anders', 'Gvido', 'Roman', 'Josef', 'Timur']
```

Если указан недопустимый индекс, то во время выполнения программы не происходит ошибки. Если задан индекс за пределами конца списка, то значение будет добавлено в конец списка. Если применен отрицательный индекс, который указывает на недопустимую позицию, то значение будет вставлено в начало списка.

Метод index()

Метод `index()` возвращает индекс первого элемента, значение которого равняется переданному в метод значению. Таким образом, в метод передается один параметр:

1. `value` : значение, индекс которого требуется найти.

Если элемент в списке не найден, то во время выполнения происходит ошибка.

Приведённый ниже код:

```
names = ['Gvido', 'Roman', 'Timur']
position = names.index('Timur')
print(position)
```

выводит:

```
2
```

Приведённый ниже код:

```
names = ['Gvido', 'Roman', 'Timur']
position = names.index('Anders')
print(position)
```

приводит к возникновению ошибки:

```
ValueError: 'Anders' is not in list
```

Чтобы избежать таких ошибок, можно использовать метод `index()` вместе с оператором принадлежности `in` :

```
names = ['Gvido', 'Roman', 'Timur']
if 'Anders' in names:
    position = names.index('Anders')
    print(position)
else:
    print('Такого значения нет в списке')
```

Метод remove()

Метод `remove()` удаляет первый элемент, значение которого равняется переданному в метод значению. В метод передается один параметр:

1. `value` : значение, которое требуется удалить.

Метод уменьшает размер списка на один элемент. Все элементы после удаленного элемента смещаются на одну позицию к началу списка. Если элемент в списке не найден, то во время выполнения происходит ошибка.

Приведённый ниже код:

```
food = ['Рис', 'Курица', 'Рыба', 'Брокколи', 'Рис']
print(food)
food.remove('Рис')
print(food)
```

выводит:

```
['Рис', 'Курица', 'Рыба', 'Брокколи', 'Рис']
['Курица', 'Рыба', 'Брокколи', 'Рис']
```

Важно: метод `remove()` удаляет только первый элемент с указанным значением. Все последующие его вхождения остаются в списке. Чтобы удалить все вхождения, нужно использовать цикл `while` в связке с оператором принадлежности `in` и методом `remove` .

Метод pop()

Метод `pop()` удаляет элемент по указанному индексу и возвращает его. В метод `pop()` передается один **необязательный** аргумент:

1. `index` : индекс элемента, который требуется удалить.

Если индекс не указан, то метод удаляет и возвращает последний элемент списка. Если список пуст или указан индекс за пределами диапазона, то во время выполнения происходит ошибка.

Приведённый ниже код:

```
names = ['Gvido', 'Roman', 'Timur']
item = names.pop(1)
print(item)
print(names)
```

выводит:

```
Roman
['Gvido', 'Timur']
```

Метод count()

Метод `count()` возвращает количество элементов в списке, значения которых равны переданному в метод значению.

Таким образом, в метод передается один параметр:

1. `value` : значение, количество элементов, равных которому, нужно посчитать.

Если значение в списке не найдено, то метод возвращает 0.

Приведённый ниже код:

```
names = ['Timur', 'Gvido', 'Roman', 'Timur', 'Anders', 'Timur']
cnt1 = names.count('Timur')
cnt2 = names.count('Gvido')
cnt3 = names.count('Josef')

print(cnt1)
print(cnt2)
print(cnt3)
```

выводит:

```
3
1
0
```

Метод reverse()

Метод `reverse()` инвертирует порядок следования значений в списке, то есть меняет его на противоположный.

Приведённый ниже код:

```
names = ['Gvido', 'Roman', 'Timur']
names.reverse()
print(names)
```

выводит:

```
['Timur', 'Roman', 'Gvido']
```

Существует большая разница между вызовом метода `names.reverse()` и использованием среза `names[::-1]`. Метод `reverse()` меняет порядок элементов на обратный **в текущем списке**, а срез создаёт копию списка, в котором элементы следуют в обратном порядке.

Метод clear()

Метод `clear()` удаляет все элементы из списка.

Приведённый ниже код:

```
names = ['Gvido', 'Roman', 'Timur']
names.clear()
print(names)
```

выводит:

```
[]
```

Метод copy()

Метод `copy()` создаёт поверхностную копию списка.

Приведённый ниже код:

```
names = ['Gvido', 'Roman', 'Timur']
names_copy = names.copy()              # создаем поверхностную копию списка names

print(names)
print(names_copy)
```

выводит:

```
['Gvido', 'Roman', 'Timur']
['Gvido', 'Roman', 'Timur']
```

Аналогичного результата можно достичь с помощью срезов или функции `list()`:

```
names = ['Gvido', 'Roman', 'Timur']
names_copy1 = list(names)              # создаем поверхностную копию с помощью функции list()
names_copy2 = names[:]                 # создаем поверхностную копию с помощью среза от начала до конца
```

Примечания

Примечание. Существует большая разница в работе строковых и списочных методов. Строковые методы не изменяют содержимого объекта, к которому они применяются, а возвращают новое значение. Списочные методы, напротив, меняют содержимое объекта, к которому применяются.

❤️ Happy Pythoning!

Шаг 2

Установите соответствие между списочным методом и тем, что он выполняет.

Чтобы решить это задание откройте https://stepik.org/lesson/324754/step/2

Шаг 3

Что покажет приведённый ниже код?

```
colors = ['Orange']
colors.append('Red')
colors.append('Blue')
colors.append('Green')
colors.insert(0, 'Violet')
colors.insert(2, 'Purple')

print(colors)
```

Чтобы решить это задание откройте https://stepik.org/lesson/324754/step/3

Шаг 4

Что покажет приведённый ниже код?

```
colors = ['Red', 'Blue', 'Green', 'Black', 'White']
del colors[-1]
colors.remove('Green')

print(colors)
```

Чтобы решить это задание откройте https://stepik.org/lesson/324754/step/4

Шаг 5

Всё сразу 2 🌶

Дополните приведённый ниже код, чтобы он:

1. Заменил второй (по порядку) элемент списка на `17`;
2. Добавил числа `4`, `5` и `6` в конец списка;
3. Удалил первый (по порядку) элемент списка;
4. Удвоил список;
5. Вставил число `25` по индексу `3`;
6. Вывел список с помощью функции `print()`

Примечание. Под удвоением списка мы понимаем добавление к исходному списку всех его элементов, сохраняя порядок. Например, при удвоении списка `['py', 'gen']` мы получаем список `['py', 'gen', 'py', 'gen']`.

Чтобы решить это задание откройте https://stepik.org/lesson/324754/step/5

Шаг 6

Переставить min и max

На вход программе подаётся строка текста, содержащая **различные** натуральные числа. Вам необходимо переставить максимальный и минимальный элементы местами и вывести изменённую строку.

Формат входных данных
На вход программе подаётся строка текста, содержащая различные натуральные числа, разделённые символом пробела.

Формат выходных данных
Программа должна вывести текст в соответствии с условием задачи.

Примечание. Используйте подходящие встроенные функции и списочные методы.

▶ Тестовые данные

Sample Input 1:

```
3 4 5 2 1
```

Sample Output 1:

```
3 4 1 2 5
```

Sample Input 2:

```
10 9 8 7 6 5 4 3 2 1
```

Sample Output 2:

```
1 9 8 7 6 5 4 3 2 10
```

Sample Input 3:

```
1 2
```

Sample Output 3:

```
2 1
```

Sample Input 4:

```
1
```

Sample Output 4:

```
1
```

Чтобы решить это задание откройте https://stepik.org/lesson/324754/step/6

Шаг 7

Количество артиклей

На вход программе подаётся строка, содержащая английский текст. Напишите программу, которая подсчитывает общее количество артиклей: `a` , `an` , `the` .

Формат входных данных

На вход программе подаётся строка, содержащая английский текст. Слова текста разделены символом пробела.

Формат выходных данных

Программа должна вывести общее количество артиклей `a`, `an`, `the` вместе с поясняющим текстом.

Примечание. Артикли могут начинаться с заглавной буквы `A`, `An`, `The`.

▶ Тестовые данные

Sample Input:

```
William Shakespeare was born in the town of Stratford, England, in the year 1564. When he was a young man, Shakespeare moved to the city of London, where he began writing plays. His plays were soon very successful, and were enjoyed both by the common people of London and also by the rich and famous. In addition to his plays, Shakespeare wrote many short poems and a few longer poems. Like his plays, these poems are still famous today.
```

Sample Output:

```
Общее количество артиклей: 7
```

Чтобы решить это задание откройте https://stepik.org/lesson/324754/step/7

Шаг 8

Взлом Братства Стали

Немалоизвестный в пустошах Мохаве Курьер забрел в Хидден-Вэли – секретный бункер Братства Стали и любезно соглашается помочь им в решении их проблем. Одной из такой проблем являлся странный компьютерный вирус, который проявлялся в виде появления комментариев к программам на терминалах Братства Стали. Известно, что программисты Братства никогда не оставляют комментарии к коду и пишут программы на Python, поэтому удаление всех этих комментариев никак не навредит им. Помогите писцу Ибсену удалить все комментарии из программы.

Формат входных данных

На первой строке подаётся символ решётки и сразу же натуральное число n – количество строк в программе, не считая первой. Далее следуют n строк кода.

Формат выходных данных

Нужно вывести те же строки, но удалить комментарии и символы пустого пространства в конце строк. Пустую строку вместо первой строки ввода выводить не надо.

Примечание 1. Обращайте внимание на лишние пробелы в конце строк. Проверяющая система не пропустит ваше решение с ними.

Примечание 2. Гарантируется, что в самом коде программы отсутствуют символы `#`.

▶ Тестовые данные

Sample Input:

```
#12
print("Введите своё имя")
name = input()
print("Введите пароль, если имеется")    # ахахахах вам не поймать меня
password = input()
if password == "hoover":
    print("Здравствуйте, рыцарь", name)         #долой Макнамару
elif password == "noncr":
    print("Здравствуйте, паладин", name)
elif password == "gelios":
    print("Здравствуйте, старейшина", name)          #Элайджа вперёд
else:
    print("Здравствуйте, послушник", name)
```

Sample Output:

```
print("Введите своё имя")
name = input()
print("Введите пароль, если имеется")
password = input()
if password == "hoover":
    print("Здравствуйте, рыцарь", name)
elif password == "noncr":
    print("Здравствуйте, паладин", name)
elif password == "gelios":
    print("Здравствуйте, старейшина", name)
else:
    print("Здравствуйте, послушник", name)
```

Чтобы решить это задание откройте https://stepik.org/lesson/324754/step/8

Шаг 9

Метод sort()

В Python списки имеют встроенный метод `sort()`, который сортирует элементы списка по возрастанию или убыванию.

Приведённый ниже код:

```
a = [1, 7, -3, 9, 0, -67, 34, 12, 45, 1000, 6,  8, -2, 99]
a.sort()
print('Отсортированный список:', a)
```

выводит:

```
Отсортированный список: [-67, -3, -2, 0, 1, 6, 7, 8, 9, 12, 34, 45, 99, 1000]
```

По умолчанию метод `sort()` сортирует список по возрастанию. Если требуется отсортировать список по убыванию, необходимо явно указать параметр `reverse = True`.

Приведённый ниже код:

```
a = [1, 7, -3, 9, 0, -67, 34, 12, 45, 1000, 6,  8, -2, 99]
a.sort(reverse=True)  # сортируем по убыванию
print('Отсортированный список:', a)
```

выводит:

```
Отсортированный список: [1000, 99, 45, 34, 12, 9, 8, 7, 6, 1, 0, -2, -3, -67]
```

Примечания

Примечание 1. С помощью метода `sort()` можно сортировать списки содержащие не только числа, но и строки. В таком случае элементы списка сортируются в соответствии с лексикографическим порядком (https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BA%D1%81%D0%B8%D0%BA%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA).

Приведённый ниже код:

```
a = ['бета', 'альфа', 'дельта', 'гамма']
a.sort()
print('Отсортированный список:', a)
```

выводит:

```
Отсортированный список: ['альфа', 'бета', 'гамма', 'дельта']
```

Примечание 2. Метод `sort()` использует алгоритм Timsort (https://ru.wikipedia.org/wiki/Timsort).

❤️ Happy Pythoning!

Шаг 10

Что покажет приведённый ниже код?

```
numbers = [4, 2, 8, 6, 5, 3, 10, 4, 100, 1, -7]
numbers.sort()
del numbers[0]
del numbers[-1]
numbers.sort(reverse=True)

print(numbers)
```

Чтобы решить это задание откройте https://stepik.org/lesson/324754/step/10

Шаг 11

Сортировка чисел

На вход программе подаётся строка текста, содержащая целые числа. Из данной строки формируется список чисел. Напишите программу, которая сортирует и выводит данный список сначала по возрастанию, а затем по убыванию.

Формат входных данных

На вход программе подаётся строка текста, содержащая целые числа, разделённые символом пробела.

Формат выходных данных

Программа должна вывести две строки текста в соответствии с условием задачи.

▶ Тестовые данные

Sample Input:

```
4 5 1 2 3 8
```

Sample Output:

```
1 2 3 4 5 8
8 5 4 3 2 1
```

Чтобы решить это задание откройте https://stepik.org/lesson/324754/step/11