Строки, кортежи и списки представляют собой последовательности, а это значит, что мы можем обратиться к любому их элементу по индексу.

Для выполнения такой операции в питоне используются квадратные скобки [] после объекта. В квадратных скобках указывается желаемый индекс. Индексирование начинается с 0.

In [24]:
```
s = 'Welcome to "Brasil!"' # заодно обратите внимание на кавычки внутри кавычек (исп
print(s)

print(s[0]) # первый элемент
print(s[1]) # второй
print(s[2]) # третий
print(s[-1]) # последний
print(s[-2]) # второй с конца

#Предыдущие действия никак не изменили строку
print(s)
```

```
Welcome to "Brasil!"
W
e
l
"
!
Welcome to "Brasil!"
```

Кроме выбора одного элемента с помощью индексирования можно получить подстроку. Для этого надо указать индексы границ подстроки через двоеточие.

Первое число - от какого индекса начинаем (если здесь ничего не написать, то начнем сначала). Второе число (после первого двоеточия) - каким индексом заканчивается срез (если ничего не написать, то питон возьмет последний символ). Третье число (необязательное, после второго двоеточия) - шаг, по умолчанию там стоит 1 (каждая буква).

Таким образом, использовав одно число без двоеточий, мы получим один символ. Использовав два числа через двоеточие - срез строки, включая первый индекс и не включая второй (первое число обязательно меньше второго). Использовав три числа через два двоеточих - срез строки с определенным шагом, заданным третьим числом.

In [25]:
```
print(s[1:])
print(s[:4]) # четыре первых символа до порядкового номера 4
print(s[:]) # копия строки
print(s[:-1]) # вся строка кроме последнего символа
print(s[::2]) # также можно выбирать символы из строки с каким-то шагом
print(s[::-1]) # например, с помощью шага -1 можно получить строку наоборот
```

```
elcome to "Brasil!"
Welc
Welcome to "Brasil!"
Welcome to "Brasil!
Wloet Bai!
"!lisarB" ot emocleW
```

По аналогии со строками, у списков и кортежей тоже можно брать срезы.

In [1]:
```
myList = [0, 1, 2, 3, 4, 5]
myTuple = (0, 1, 2, 3, 4, 5)
```

In [2]:
```
print(myList[1:]) # берем все, начиная с элемента с первым индексом
print(myTuple[2:])
```

```
[1, 2, 3, 4, 5]
(2, 3, 4, 5)
```

In [3]:
```
print(myList[:3]) # берем все до элемента с третьим индексом (невключительно)
print(myTuple[:-1])
```

```
[0, 1, 2]
(0, 1, 2, 3, 4)
```

In [4]:
```
print(myList[::2]) # берем все четные элементы
print(myTuple[1::2]) # берем все нечетные элементы
```

```
[0, 2, 4]
(1, 3, 5)
```

Метод .find()

Когда мы работаем со строками, у нас часто стоит задача брать срез не с конкретного индекса, а привязывать его к поиску определенного символа. У нас есть специальный метод строковых переменных .find().

Методы - это методы классы. Грубо говоря, это функции, которые будут работать только с определенным типом данных. Синтаксис метода следующий: `{название переменной или данные}.{название метода()}` .

Например, давайте проверим содержит ли строка упоминание университета.

In [5]:
```
'В ВШЭ стартовала новая программа по Data Science'.find('ВШЭ')
```

Out[5]:
```
2
```

Метод .find() берет один аргумент - подстроку, которую ищет в строке. Возвращает метод индекс первого символа подстроки, если ее удалось найти.

Если подстрока не была найдена, метод вернет -1 (на следующем занятии мы будем использовать это свойство, когда разберемся с условным оператором).

In [32]:
```
'В ВШЭ стартовала новая программа по Data Science'.find('вшэ') # обратите внимание,
```

Out[32]:
```
-1
```

.find() иногда используется в парсинге веб-страниц. Зная индекс первого элемента, мы можем достать интересующую нас информацию.

Например, мы скачали с сайта информацию о цене нового планшета и хотим достать оттуда собственно цену. Мы знаем, что цена идет после подстроки "ЦЕНА:" и что после самой цены идет постфикс "руб.". Давайте попробуем достать цену и посчитать, сколько стоит два таких планшета.

In [41]:
```
info = 'iPad 64 GB ЦЕНА: 39 990 руб. Скидка: 5%'
print(info.find('ЦЕНА:')) # нашли индекс Ц - начала подстроки "ЦЕНА:"
print(info.find('руб.')) # нашли индекс р
price = info[info.find('ЦЕНА:')+6:info.find('руб.')-1] # вывели срез от от начала до
                                                       # слогаемых 6 и -1 откорретировали ин
print(price)
```

```
11
24
39 990
```

Почти готово, но теперь мешается пробел. Кстати, это очень частая проблема, что числа в интернете оформлены с разделителями и перед конвертацией их приходится еще и приводить к стандартному виду, который можно скормить функции int(). Пока мы не знаем метода, который может заменять символы, поэтому давайте попробуем почистить цену с помощью .find() и срезов.

In [42]:
```
price_clean = price[:price.find(' ')]+price[price.find(' ')+1:]
print(price_clean)
```

```
39990
```

Теперь с этим можно работать!

In [44]:
```
print(int(price_clean) * 2)
```

```
79980
```

Если подстрока входит в строку несколько раз, то find() вернет индекс только для первого вхождения.

In [48]:
```
price.find('9')
```

Out[48]:
```
3
```

Есть модификация метода find(): rfind(substring) - возвращает позицию самого правого вхождения подстроки substring в строку string или -1, если подстрока не найдена.

In [47]:
```
price.rfind('9')
```

Out[47]:
```
4
```