# Строки, списки, кортежи

# Вы уже знаете, что…

Строка – это то, что в кавычках

```
print('Виват, Лицей!’)                          # Виват, Лицей!
```

Строку можно не только выводить, но и вводить

```
name = input('Ваше имя: ')                       # Ваше имя: Саша
```

А ещё, вы знаете f-строки.

```
print(f'С Новым годом, {name}!')                 # С Новым годом, Саша!
print(f'До Нового года осталось {31 - 7} дня.') # До Нового года осталось 24 дня.
```

# Вы уже знаете, что…

Строки можно складывать

```
print('Виват, ' + 'Лицей!')          # Виват, Лицей!
```

И умножать на число

```
print('Виват, Лицей! ' * 3)              # Виват, Лицей! Виват, Лицей! Виват, Лицей!
```

# Вы уже знаете, что…

Строки можно преобразовывать в другие типы и наоборот

```
a = '1'
b = '2'
print(a + b)            # 12

a = int(a)
b = int(b)
print(a + b)            # 3

a = str(a)
b = str(b)
print(2 * a + b * 3)    # 11222
```

# Вы уже знаете, что…

Строки можно сравнивать

```
print('a' == 'b')       # False
print('a' != 'b')       # True
print('a' < 'b')        # True
print('a' <= 'b')       # True

print('aab' < 'abc')    # True
print('ABC' < 'aab')    # True
```

# Вы уже знаете, что…

Для проверки условия наличия подстроки в строке используется оператор `in`. Например, проверим, что во введённой строке встречается корень «добр» (для слов «добрый», «доброе» и подобных):

```
text = "Красота без доброты умирает невостребованной. (Бенджамин Франклин)"
if "добр" in text:
    print("Встретилось 'доброе' слово.")
else:
    print("Добрых слов не найдено.")

# Встретилось 'доброе' слово.
```

# Вы уже знаете, что…

По строкам тоже можно итерироваться.
В качестве диапазона можно поставить строку, переменная будет шагать по символам.

```
for e in 'aba caba':
    print(e, end=', ')   # a, b, a,  , c, a, b, a,
print()

for e in 'aba caba':
    if e in ' a':
        continue

    print(e, end=', ')   # b, c, b,
print()

for e in 'QsWuEmRmTaY UcIuOmP AlSaDuFdGeH':
    if e.isupper():
        continue

    print(e, end='') # summa cum laude
print()
```

# Строка – это упорядоченная коллекция

Когда в Python какой-то тип хранит в себе несколько чего-нибудь, то его называют **коллекцией.** Например – строка.

Если элементы коллекции стоят в каком-то определённом порядке, то такую коллекцию называют **упорядоченной.** Например – строка.

У упорядоченной коллекции каждый элемент имеет свой номер – **индекс.** С помощью индекса можно обращаться к элементам коллекции.

```
s = 'Виват, Лицей!'
print(s[0], s[3], s[5], s[-1])  # В а , !
print(s[99])                    # IndexError: string index out of range
```

# Строка – это упорядоченная коллекция

Итерироваться можно не только по элементам строки, а, например, по индексам

```
s = 'Строка'
for i in range(len(s)):
    print(i, s[i], end=',  ')   # 0 С,  1 т,  2 р,  3 о,  4 к,  5 а,
print()
```

А можно итерироваться одновременно по индексам и элементам строки с помощью функции enumerate()

```
for i, e in enumerate(s):
    print(i, e, end=',  ')  # 0 С,  1 т,  2 р,  3 о,  4 к,  5 а,
print()
```

# У строк бывают срезы

В квадратных скобках можно указать начальный и конечный индексы. А ещё можно указать шаг.

```
text = "Виват, Лицей!"
print(text[8:11])   # ице
print(text[:6])     # Виват,
print(text[8:])     # ицей!
print(text[:])      # Виват, Лицей!
print(text[::2])    # Ввт ие!
print(text[::-1])   # !йециЛ ,тавиВ
```

# У строк бывают срезы

Нельзя просто так взять и поменять символ строки

```
s = 'Some random number is 1 - this is some random number'
s[22] = '8'     # TypeError: 'str' object does not support item assignment
```

но - если очень нужно - можно написать вот такую кракозябру из двух срезов

```
s = s[:22] + '8' + s[23:]
print(s)     # Some random number is 8 - this is some random number
```

**Чуть позже** рассмотрим пример, как лучше поступить, если в строке нужно модифицировать отдельные символы.

# Чуть позже:

Рассмотрим пример, как лучше поступить, если в строке нужно модифицировать отдельные символы. Можно:

```
s = 'Some random number is 1 - this is some random number'

# 1. превратить строку в список символов:
s = list(s)

# 2. поделать с ним все необходимые дела:
s[22] = '8'

# 3. превратить обратно в строку:
s = ''.join(s)
print(s)    # Some random number is 8 - this is some random number
```

# Методы строк

У разных объектов в питоне бывают **методы.** Здесь список (неполный) методов у строк

Вызывается так: <строка>.<метод>()
Или так: <строка>.<метод>(<аргументы>)

**Методы строк не меняют строку, а возвращают новую.**

| Метод | Описание |
|---|---|
| **s.capitalize()** | Возвращает копию строки, у которой первая буква заглавная, а остальные строчные |
| **s.count(sub)** | Возвращает количество неперекрывающихся вхождений подстроки **sub** |
| **s.endswith(suffix)** | Возвращает **True**, если строка оканчивается на подстроку **suffix**. Иначе возвращает **False**. |
| **s.find(sub)** | Возвращает индекс первого вхождения подстроки **sub**. Если подстрока не найдена, то возвращает -1 |
| **s.index(sub)** | Возвращает индекс первого вхождения подстроки **sub**. Вызывает исключение **ValueError**, если подстрока не найдена. |
| **s.isalnum()** | Возвращает **True**, если все символы строки являются буквами и цифрами. |
| **s.isalpha()** | Возвращает **True**, если все символы строки являются буквами. |
| **s.isdigit()** | Возвращает **True**, если все символы строки являются цифрами. |
| **s.islower()** | Возвращает **True**, если все буквы в строке маленькие. |
| **s.isupper()** | Возвращает **True**, если все буквы в строке большие. |
| **s.join(str_col)** | Возвращает строку, полученную конкатенацией (сложением) строк — элементов коллекции **str_col** |
| **s.split(sep)** | Возвращает список строк по разделителю **sep**. По умолчанию **sep** — любое количество пробельных символов |
| **s.upper()** | Возвращает копию строки, у которой все буквы приведены к верхнему регистру |
| **s.replace(a, b)** | Возвращает копию строки, у которой в вхождения подстроки **a** заменены на подстроку **b**. |

# Методы строк: пример

```
s = "hello, World!"
print(s.capitalize())           # Hello, world!
print(s.count("l"))             # 3
print(s.startswith("Hello"))    # False
print(s.endswith("World!"))     # True
print(s.find("o"))              # 4
print(s.isdigit())              # False
print(s.lower())                # hello, world!
print(s.upper())                # HELLO, WORLD!
print(s.split())                # ['hello,', 'World!’]

print(', '.join(['lol', 'kek', 'cheburek']))     # lol, kek, cheburek
print('7 12 23'.split())                         # ['7', '12', '23']
```

# Список – это тоже упорядоченная коллекция

- В качестве элементов может иметь значения любого типа (обычно какого-нибудь одного).
- По нему тоже можно итерироваться, как и по строке, и у него тоже есть срезы
- Можно менять отдельные элементы
- Можно добавлять или удалять элементы
- Строка – это тип `str`, а список – это тип `list`

```
numbers = [10, 20, 30, 40, 50]
print(numbers[0])       # 10
print(numbers[-1])      # 50
print(numbers[1:3])     # [20, 30]
print(numbers[::-1])    # [50, 40, 30, 20, 10]

numbers[2] = 'Биофак'
print(numbers)  # [10, 20, 'Биофак', 40, 50]
```

# Добавление и удаление элементов списка

Для добавления в конец списка можно использовать метод append().

Для удаления (нужно знать индекс) можно использовать метод pop().

```
lyceum = ['Визитки', 'Столовка', 'Базы данных', 'Биофак']

lyceum.pop(0)
lyceum.pop(1)
print(lyceum)   # ['Столовка', 'Биофак’]

lyceum.append('Новый год!')
print(lyceum)   # ['Столовка', 'Биофак', 'Новый год!']
```

# Методы списков

У списков свои методы в отличие от строк.
В отличие от строк, **некоторые** методы могут изменять существующий список, не создавая новый.

| Метод или функция | Описание |
|---|---|
| x in s | Возвращает True, если в списке s есть элемент x. Иначе False |
| x not in s | Возвращает False, если в списке s есть элемент x. Иначе True |
| s + t | Возвращает список, полученный конкатенацией списков s и t |
| s * n (n * s) | Возвращает список, полученный дублированием n раз списка s |
| len(s) | Возвращает длину списка s |
| min(s) | Возвращает минимальный элемент списка |
| max(s) | Возвращает максимальный элемент списка |
| s.index(x) | Возвращает индекс первого найденного элемента x. |
| s.count(x) | Возвращает количество элементов x |
| s.append(x) | Добавляет элемент x в конец списка |
| s.clear() | Удаляет все элементы списка |
| s.copy() | Возвращает копию списка |
| s.insert(i, x) | Вставляет элемент x в список по индексу i |
| s.pop(i) | Возвращает и удаляет элемент с индексом i. |
| s.remove(x) | Удаляет первый элемент со значением x |
| sorted(s) | Возвращает отсортированный по возрастанию список, не меняя исходный. |
| s.reverse() | Переворачивает список |

# Кортеж – тоже упорядоченная коллекция

- Кортеж – это tuple
- В отличие от списка, чтобы создать кортеж, нужно использовать не квадратные скобки, а круглые
- В отличие от списка, в кортеже обычно хранят не очень много объектов и часто разных типов
- В отличие от списка, добавлять, удалять, менять элементы кортежа нельзя

```
numbers = (1, 2, 3, 4, 5)
print(numbers)      # (1, 2, 3, 4, 5)
one_number = (1, )
print(one_number)   # (1, )
```

# Пример использования кортежа

```
a, b = 1, 2
(a, b) = (b, a)
print(f"a = {a}, b = {b}")  # a = 2, b = 1
```

# Преобразования между строкой, списком и кортежем

```
text = "Виват, Лицей!“

list_symbols = list(text)
tuple_symbols = tuple(text)
text_from_list = str(list_symbols)

print(list_symbols)     # ['В', 'и', 'в', 'а', 'т', ',', ' ', 'Л', 'и', 'ц', 'е', 'й', '!']
print(tuple_symbols)    # ('В', 'и', 'в', 'а', 'т', ',', ' ', 'Л', 'и', 'ц', 'е', 'й', '!')
print(text_from_list)   # ['В', 'и', 'в', 'а', 'т', ',', ' ', 'Л', 'и', 'ц', 'е', 'й', '!']

text_from_list = ''.join(list_symbols)
print(text_from_list)   # Виват, Лицей!
```