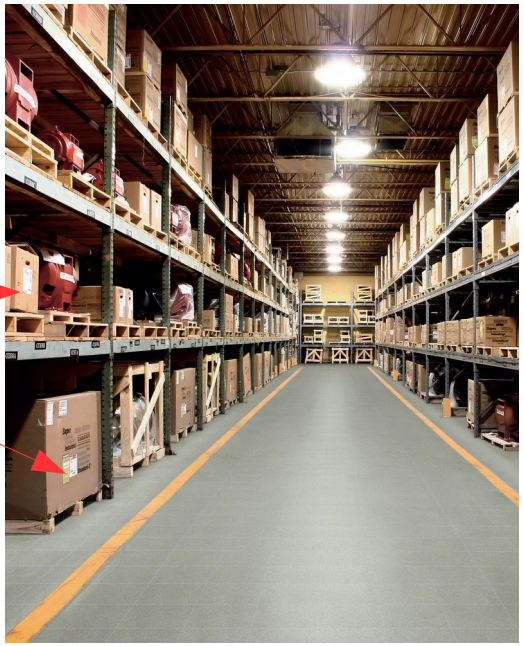
Списки (тип **list**)

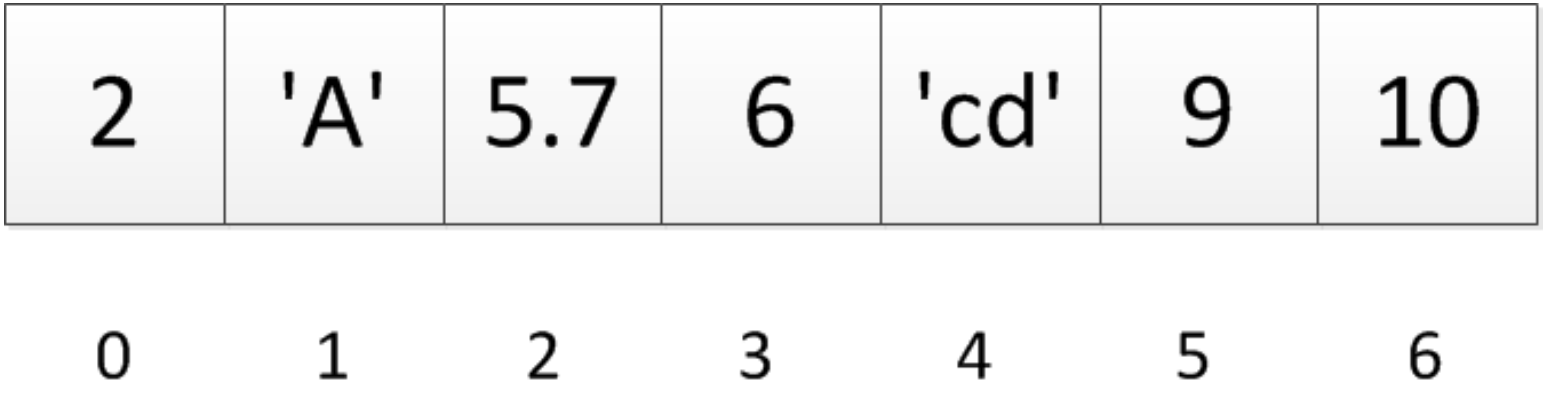
Список – это упорядоченная последовательность из нуля
или более ссылок на объекты



Списки в Python можно сравнить с набором карточек в картотеке, которые указывают на предметы различных типов, которые хранятся в другом месте. Используя картотеку (список) можно извлечь или положить какой то предмет на складе, можно перебирать карточки, можно добавлять или удалять их.

Упрощенное представление списка

Часто для многих задач список можно представить как последовательность пронумерованных ячеек, в которых могут храниться различные данные.



Отличие от массивов в Паскале

- Размер списка может меняться в ходе выполнения программы
- Нумерация элементов **всегда** начинается с нуля
- Элементами списка являются адреса (ссылки) на объекты различных типов. Упрощенно можем рассматривать, что элементами списка являются объекты разного типа

Сходство и различие списков со строками

- Строку можно рассматривать как список из символов (точнее строк одинарной длинны)
- Элементы строк и списков являются **упорядоченными**
- К спискам, как и к строкам, можно применять **срезы**
- Строки и списки являются **итерируемыми** (т.е. их элементы можно последовательно перебрать и выполнить какие-либо действия)
- В отличие от строк, **списки являются изменяемыми**. Т.е. можно изменить элемент списка, добавить новые элементы или удалить существующие

Создание списков

```
L = [ ]   # создание пустого списка
L = [0]*n   # создание списка из нулей длины n
L = [1, 2, 3, 4]  # создание списка перечислением
L = list('Python')  # преобразованием объекта
                    # итерируемого типа
L = input().split()  # из данных введенной строки,
                     # разделенных пробелом
```

С помощью генератора (рассмотрено ниже)

Запись, считывание элемента списка

```
>>>L = [2, 'A', 5.7, 6]
>>>print(L[0])
2
>>> L[1] = 3
>>> print(L)
 [2, 3, 5.7, 6]
>>>print(L[4])  # IndexError: list index out of range
```

Обход элементов списка

Непосредственно:

```
for elem in L:
    print(elem)
```

С помощью индекса:

```
for i in range(len(L)):
    print(L[i])
```

Операции над списками

- Сцепление (склеивание) списков:

```
>>>[1, 2, 3] + [4, 5, 6]
[1, 2, 3, 4, 5, 6]
```

- Умножение списка на число:

```
>>>[1, 2, 3]*3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

- Списки можно сравнивать с помощью операторов сравнения (<, <=, ==, !=, >=, >)
- in (not in) - проверка на вхождение элемента в список:

```
>>> 2 in [1, 2, 3]    # True
```

Методы списков

L.append (x)	Добавляет элемент x в конец списка L
L.count(x)	Возвращает число вхождений элемента x в список L
L.extend(m) L += m	Добавляет в конец списка L все элементы итерируемого объекта m
L.index(x,start, end)	Возвращает индекс самого первого (слева) вхождения элемента x в список L (или в срез start:end списка L), при отсутствии - возбуждает исключение ValueError
L.insert(i, x)	Вставляет элемент x в список L в позицию int i
L.pop()	Удаляет самый последний элемент из списка L и возвращает его в качестве результата

Методы списков (продолжение)

L.pop(i)	Удаляет из списка L элемент с индексом int i и возвращает его в качестве результата
L.remove(x)	Удаляет самый первый (слева) найденный элемент x из списка L или возбуждает исключение ValueError, если элемент x не будет найден
L.reverse()	Переставляет в памяти элементы списка в обратном порядке
L.sort(...)	Сортирует список в памяти.

Функции для работы со списками

sum(L)	Возвращает сумму элементов списка L
min(L)	Возвращает минимум в списке L
max(L)	Возвращает максимум в списке L
len(L)	Возвращает длину списка L
sorted(L)	Возвращает новый список, в котором элементы списка L отсортированы по возрастанию
sorted(L, reverse=True)	Возвращает новый список, в котором элементы списка L отсортированы по убыванию

Генератор списков

Генератор списков – это выражение и цикл с дополнительным условием, заключенное в квадратные скобки, в котором цикл используется для создания элементов списка, а условие используется для исключения нежелательных элементов:

[*выражение* for *элемент* in *итерируемый_объект*]
[*выражение* for *элемент* in *итерируемый_объект* if *условие*]

```
L = [int(el) for el in input().split()]
```