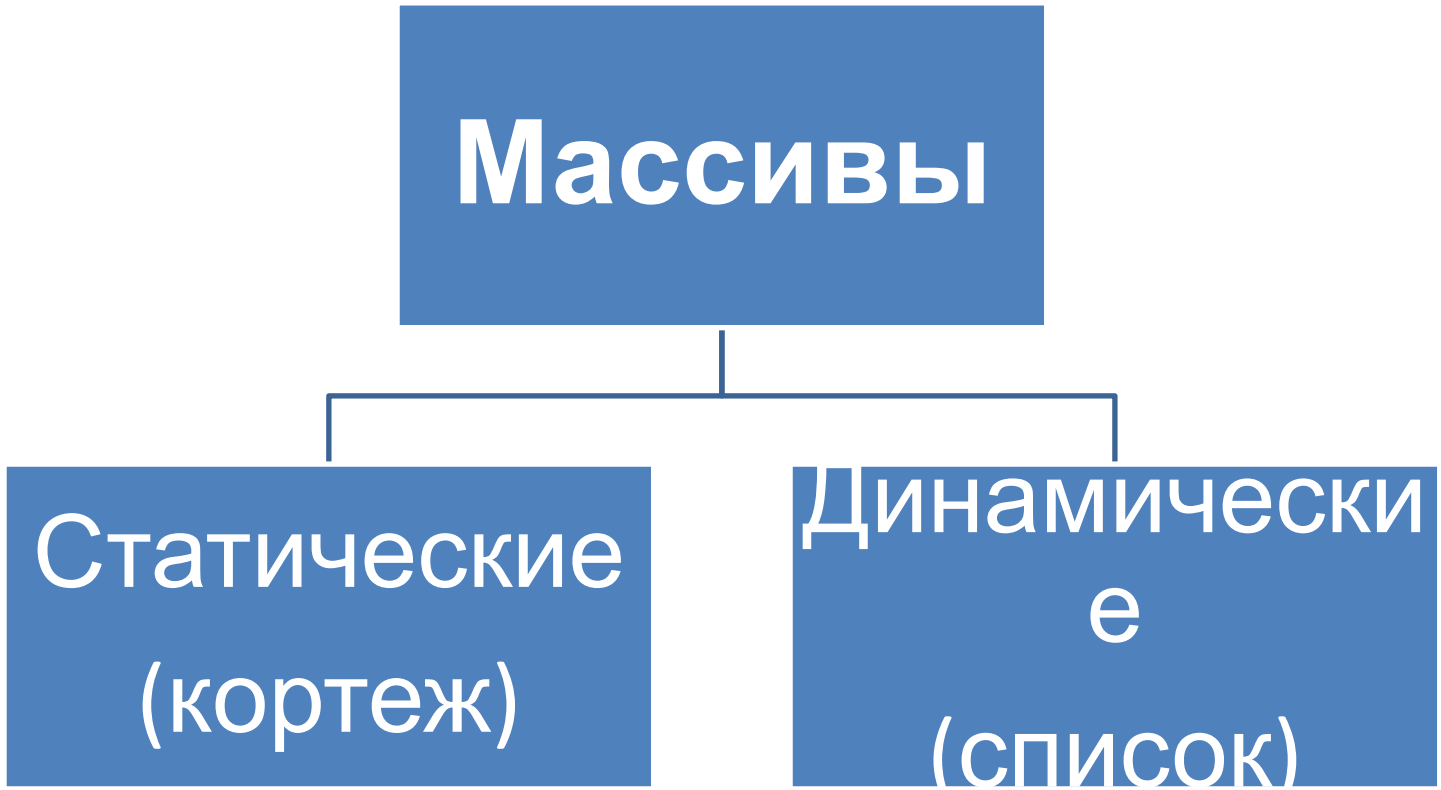
Основные типы данных в Python. Массивы

- Массив – **набор** данных **одного типа,** имеющих одно и то же имя, и различающихся по номеру (индексу).
- Индекс элемента – порядковый номер элемента массива

$$a_1 \quad a_2 \quad a_3 \quad \dots \quad a_n$$



Статический тип

- **Место** для всех данных **отводится сразу,** во время программы не изменяется.
- Используется при объявлении переменных.

Динамический тип

- **Количество элементов заранее неизвестно**, меняется во время исполнения программы.
- Память отводится и освобождается самой программой.

Списки

- Списки — это изменяемые последовательности.

- **Список** в Python — это упорядоченный набор объектов, в список могут одновременно входить объекты разных типов (числа, строки и другие структуры, в частности, он может содержать другие списки и кортежи). Объекты, входящие в список – это элементы списка.

```
>>> numbers = [1, 2, 3, 4, 5]
```

Создание пустого списка

- numbers = []

- С помощью функции list():

```
numbers = list()
```

Функция — это фрагмент кода, который выполняет какую-то задачу.

Для обращения к элементам списка надо использовать индексы, которые представляют **номер** элемента в списке. Индексы начинаются с нуля. То есть второй элемент будет иметь индекс 1.

Для обращения к элементам с конца можно использовать отрицательные индексы, начиная с -1. То есть у последнего элемента будет индекс -1, у предпоследнего -2 и так далее.

```
numbers =  [1, 2, 3, 4, 5]
print(numbers[0])   # 1
print(numbers[2])   # 3
print(numbers[-3])  # 3
```

Способы заполнения списка

- С заранее заданным набором данных:

```
numbers = [1, 2, 3, 4, 5]
```

- Также, список можно заполнить вручную **с помощью цикла, метода append и команды input**

```
mas = []
for i in range(20): //ввод 20 элементов в цикле
    mas.append(int(input())) // добавление целого числа, введенного с клавиатуры, в конец списка
```

Способы заполнения списка

- Случайными числами:

Функция **random** позволяет генерировать случайные числа. Прежде чем использовать **random**, необходимо подключить ее с помощью инструкции: **import random .**

random.randint(a, b) — возвращает псевдослучайное целое число в отрезке от a до b

```
import random                    //подключение модуля случайных чисел
random mas=[]                    // объявление пустого списка
for i in range(0, 10):
    mas.append(random.randint(0, 100))
 //заполнение списка 10-ю случайными числами в диапазоне от 0 до 100
print(mas)                       // вывод списка
```

Способы заполнения списка

- Создание списка случайных чисел с помощью генератора списка

```
import random
mas = [random.randint(0, 100) for i in range(0, 10)]
```

Метод randint(0, 100) генерирует случайное целое число в диапазоне от 0 до 100.

mas = [i for i in range(1,15)]

range(end): создается набор чисел от 0 до числа end

range(start, end): создается набор чисел от числа start до числа end

range(start, end, step): создается набор чисел от числа start до числа end с шагом step

Практическая часть

1. Создайте два любых списка и свяжите их с переменными.

2. Извлеките из первого списка второй элемент.

3. Измените во втором списке последний объект. Выведите список на экран.

4. Соедините оба списка в один, присвоив результат новой переменной. Выведите получившийся список на экран.

1
```
>>> spisok1 = [45, 2, 8, 97, 34]
>>> spisok2 = [65, 23, 10]
```

2
```
>>> spisok1 [1]
2
```

3
```
>>> spisok2 [-1] = 12
>>> spisok2
[65, 23, 12]
```

4
```
>>> big_spisok = spisok1 + spisok2
>>> big_spisok
[45, 2, 8, 97, 34, 65, 23, 12]
```

Функции

- **append(item):** добавляет элемент item в конец списка
- **insert(index, item):** добавляет элемент item в список по индексу index
- **remove(item):** удаляет элемент item. Удаляется только первое вхождение элемента. Если элемент не найден, генерирует исключение ValueError
- **clear():** удаление всех элементов из списка
- **index(item)**: возвращает индекс элемента item. Если элемент не найден, генерирует исключение ValueError
- **pop([index]):** удаляет и возвращает элемент по индексу index. Если индекс не передан, то просто удаляет последний элемент.

Функции

- **count(item):** возвращает количество вхождений элемента item в список
- **sort([key])**: сортирует элементы. По умолчанию сортирует по возрастанию. Но с помощью параметра key мы можем передать функцию сортировки.
- **reverse():** расставляет все элементы в списке в обратном порядке

Кроме того, Python предоставляет ряд встроенных функций для работы со списками:

- **len(list):** возвращает длину списка
- **sorted(list, [key]):** возвращает отсортированный список
- **min(list):** возвращает наименьший элемент списка
- **max(list):** возвращает наибольший элемент списка

Задачи.

1. Создать список и заполнить его случайными числами двумя способами.

2. Найти максимальный элемент в списке **без использования функции max().**

3. Вывести номера элементов списка, если эти элементы больше своих соседей.

4*. Найти произведение элементов списка, кратных 6 и оканчивающихся на 8. Если таких элементов нет, то сообщить об этом.