Список в Python – это встроенный тип (класс) составных данных. Является изменяемой последовательностью произвольных элементов, т.е. иметь переменную длину. Список может содержать элементы только одного типа или разных – числа, строки и другие списки. Элементы в списке упорядочены, каждый элемент имеет свой индекс или номер. Индексация начинается с нуля. Список определятся квадратными скобками. Для получения доступа к элементу списка – указывается его имя. Существует индексация с конца. Она начинается с -1: a[-1]. Очистить список можно просто заново его проинициализировав.

Список можно рассматривать как одномерный массив, хотя в базовом Пайтоне такой структуры данных нет, она подключается из модуля numpy.

Создать список можно простым перечислением

Раздаточный материал № 62

```
>>> a = [12, 3.85, "black", -4]

>>> a

[12, 3.85, 'black', -4]
```

Чтобы извлечь конкретный элемент, надо после имени списка указать в квадратных скобках индекс необходимого элемента

Раздаточный материал № 63

```
>>> a[0]
12
>>> a[3]
-4
```

Часто требуется извлечь не один элемент, а так называемый срез – часть списка. В этом случае указывается индекс первого элемента среза и индекс следующего за последним элементом среза

Раздаточный материал № 64

```
>>> a[0:2]
[12, 3.85]
```

Для извлечения среза, включающего в себя последний элемент

Раздаточный материал № 65

```
>>> a[:3]
[12, 3.85, 'black']
>>> a[2:]
['black', -4]
>>> a[:]
[12, 3.85, 'black', -4]
```

Для изменения значения элемента списка

Раздаточный материал № 66

```
>>> a[1] = 4
>>> a
[12, 4, 'black', -4]
```

Создание копии списка
Раздаточный материал № 67

```
>>> a = [1, 3, 5, 7]
>>> b = a[:]
>>> print(a)
[1, 3, 5, 7]
>>> print(b)
[1, 3, 5, 7]
```

Изменять списки и выполнять работу с ними можно с помощью специальных встроенных методов списка

Раздаточный материал № 68 (справочно)

list.append(значение) – добавление нового значения в конец списка.
list.insert(позиция, значение) - добавление нового значения в указанную позицию списка.
list.remove(значение) – удаляет указанное значение из списка, не привязываясь к индексу.
list.pop() – удаляет последний элемент списка и возвращает значение.
list.pop(индекс) – удаляет элемент списка с указанным индексом и возвращает значение.
del a[индекс] - удаляет элемент списка с указанным индексом.
del a[индекс : индекс] - удаляет срез элементов списка с указанными индексами.
list.clear() – удаляет все элементы из списка.

list.index - Возвращает индекс элемента
list.count(x) Возвращает количество вхождений элемента x в список
list.sort(key=None, reverse=False) - сортирует элементы в списке по возрастанию. Для сортировки в обратном порядке используйте флаг reverse=True.
ist.reverse() - изменяет порядок расположения элементов в списке на обратный.
list.copy() - возвращает копию списка.

Цикл For

Цикл for предназначен для перебора элементов структур данных и некоторых других объектов. Сам определит конец структуры, не требует счетчика.

Цикл for является универсальным итератором в Python: он может проходить по элементам в любой упорядоченной последовательности или в другом итерируемом объекте.

> Объект считается итерируемым, если он является либо физически сохраненной последовательностью в памяти, либо объектом, который генерирует по одному элементу за раз в контексте итерационной операции — своего рода “виртуальной” последовательностью.

Оператор for работает на строках, списках, кортежах и прочих встроенных итерируемых объектах, а также на новых объектах, определяемых пользователем.

Цикл for языка Python начинается со строки заголовка, где указывается цель (или цели) присваивания наряду с объектом, по которому нужно совершить проход. После заголовка находится блок операторов (обычно с отступами), который необходимо повторять:

Раздаточный материал № 69

```
for цель in объект:        # Присваивает цели элементы объекта
    операторы              # Повторяемое тело цикла: использует цель
else:                      # Необязательная часть else
    операторы              # Если не встречался оператор break
```

```
spisok = [10, 40, 20, 30]
>>> for element in spisok:
        print(element + 2)
12
42
22
32
```

Когда Python запускает цикл for, он присваивает цели элементы итерируемого объекта по очереди и выполняет для каждого тело цикла. Внутри тела цикла цель присваивания обычно используется для ссылки на текущий элемент в последовательности, как если бы цель была курсором, проходящим через последовательность.

Имя, применяемое как цель присваивания в строке заголовка for, обычно является (возможно, новой) переменной внутри области видимости, где находится оператор for. Имя можно изменять внутри тела цикла, но оно будет автоматически устанавливаться в следующий элемент последовательности, когда управление снова возвратится в начало цикла. После цикла эта переменная, как правило, по-прежнему ссылается на последний посещенный элемент, которым будет последний элемент в последовательности, если только не произошел выход из цикла посредством оператора break.

Оператор for также поддерживает необязательный блок else, который работает точно как в цикле while — он выполняется, если выход из цикла осуществляется без помощи оператора break (т.е. когда были посещены все элементы последовательности). Операторы break и continue в цикле for также работают аналогично циклу while. Полный формат цикла for может быть описан следующим образом:

Раздаточный материал № 70

```
for цель in объект:                  #Присваивает цели элементы объекта
        операторы if проверка: break  #Выход из цикла с пропуском else
        if проверка: continue                #Переход в начало цикла
else:
        операторы                     #Если не встречался оператор break
```

Примеры работы со списками

Фронтальный опрос:

- ✓ Понятие списка.
- ✓ Доступ к элементу списка.
- ✓ Создание списка,
- ✓ Как задать срез списка.
- ✓ Как изменить значение элемента списка.
- ✓ Как создать копию списка.
- ✓ Назначение и возможности цикла for.
- ✓ Формат цикла for.
- ✓ Принцип работы цикла for.
- ✓ Понятие итерируемого объекта.