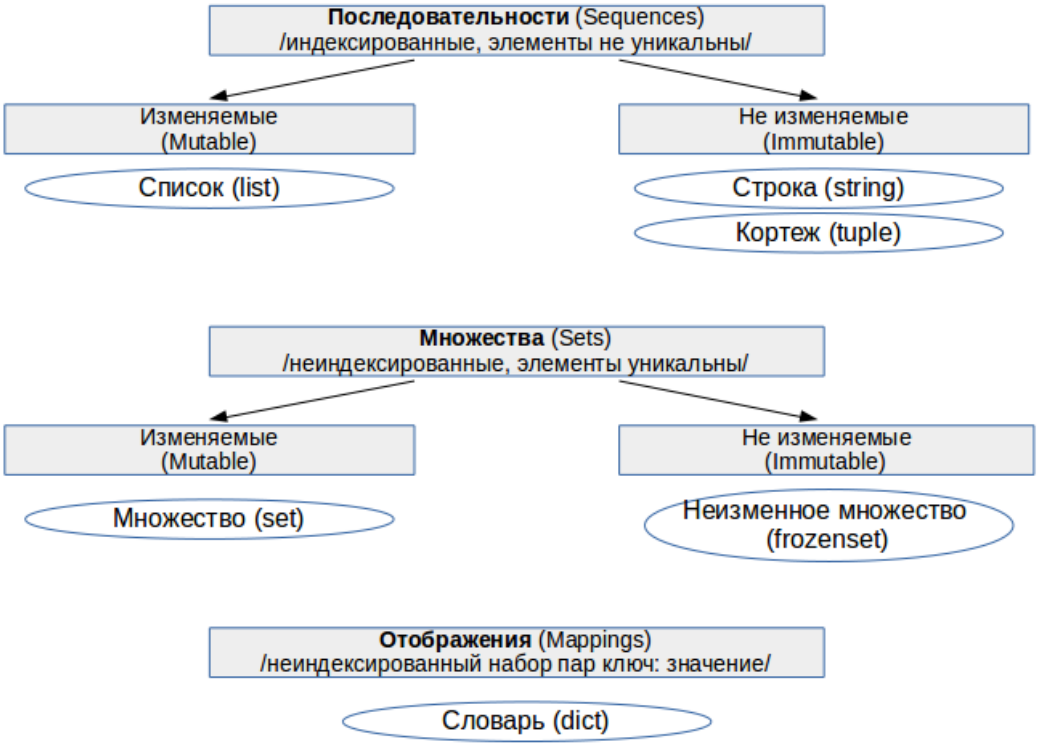
7. Списки в Python. Обращение к элементам списка и создание срезов. Обход списка и поиск элементов в списке.

Списки в Python - упорядоченные изменяемые коллекции объектов произвольных типов.



Чтобы использовать списки, их нужно создать. Создать список можно несколькими способами. Например, можно обработать любой итерируемый объект (например, строку встроенной функцией **list**:

```
x = list("список")
print(x)
#Выдаст ['с', 'п', 'и', 'с', 'о', 'к']
```

Или просто объявив его:
x = []
Или вызывав конструктор списка
x = list()

Список может содержать любое количество любых объектов (в том числе и вложенные списки, tuple, dict и т. д), или не содержать ничего.

У списков достаточно много методов

Метод	Что делает
list.append(x)	Добавляет элемент в конец списка
list.extend(L)	Расширяет список list, добавляя в конец все элементы списка L
list.insert(i, x)	Вставляет на i-ый элемент значение x
list.remove(x)	Удаляет первый элемент в списке, имеющий значение x. ValueError, если такого элемента не существует
list.pop([i])	Удаляет i-ый элемент и возвращает его. Если индекс не указан, удаляется последний элемент
list.index(x, [start [, end]])	Возвращает положение первого элемента со значением x (при этом поиск ведется от start до end)
list.count(x)	Возвращает количество элементов со значением x
list.sort([key=функция])	Сортирует список на основе функции
list.reverse()	Разворачивает список
list.copy()	Поверхностная копия списка
list.clear()	Очищает список

Обращение к элементам списка происходит по индексу. Индексы начинаются с 0
Например:

```
l = [5,7,4,6]
print(l[1])
#Выведет 7
```

При этом индексы могут быть и отрицательные, тогда нумерация будет с конца
Например:

```
l = [5,7,4,6]
print(l[-1])
#Выведет 6
```

В Python, кроме индексов, существуют ещё и **срезы**.

Срезы используются не только в списках, но и в большинстве остальных коллекций

item[START:STOP:STEP] - берёт срез от номера START, до STOP (не включая его), с шагом STEP. По умолчанию START = 0, STOP = длине объекта, STEP = 1. Соответственно, какие-нибудь (а возможно, и все) параметры могут быть опущены.

Пример использования срезов в списках:

```
>>> a = [1, 3, 8, 7]
>>> a[::-1]
[7, 8, 3, 1]
>>> a[:-2]
[1, 3]
>>> a[-2::-1]
[8, 3, 1]
>>> a[1:4:-1]
[]
```

Пример использования срезов в строках:

```
Python 3.7.6 (default, Dec 30 2019, 19:38:28)
[Clang 11.0.0 (clang-1100.0.33.16)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> str = "kotiki_meow"
>>> str[:1]
'k'
>>> str[1:]
'otiki_meow'
>>>
```

Обход списка и поиск элементов в списке.

В цикле for возможно обращаться к элементам списка двумя способами:
№1 - по индексу:

```
mylist = [6,3,8,1]
for i in range(len(mylist)):
    print(mylist[i])
```

№2 - автоматически представляя каждый элемент как переменную

```
mylist = [6,3,8,1]
for e in mylist:
    print(e)
```

Поиск элементов в списке можно осуществить несколькими способами:
1) Банально вручную в цикле перебрать каждый элемент и сравнить с исходным

```
search_element = 4
l = [5,7,4,6]
for i in range(len(l)):
    if l[i] == search_element:
        print("Элемент",l[i],"найден, индекс",i)
```

2) С помощью in

```
search_element = 4
l = [5,7,4,6]
print(search_element in l)
```

3) С помощью генератора списка (считай тот же цикл, только в одну строку)

```
search_element = 4
l = [5,7,4,6]
el = [element for element in l if element == search_element]
if len(el) == 1:
    print("Элемент присутствует")
else:
    print("Элемент отсутствует")
```