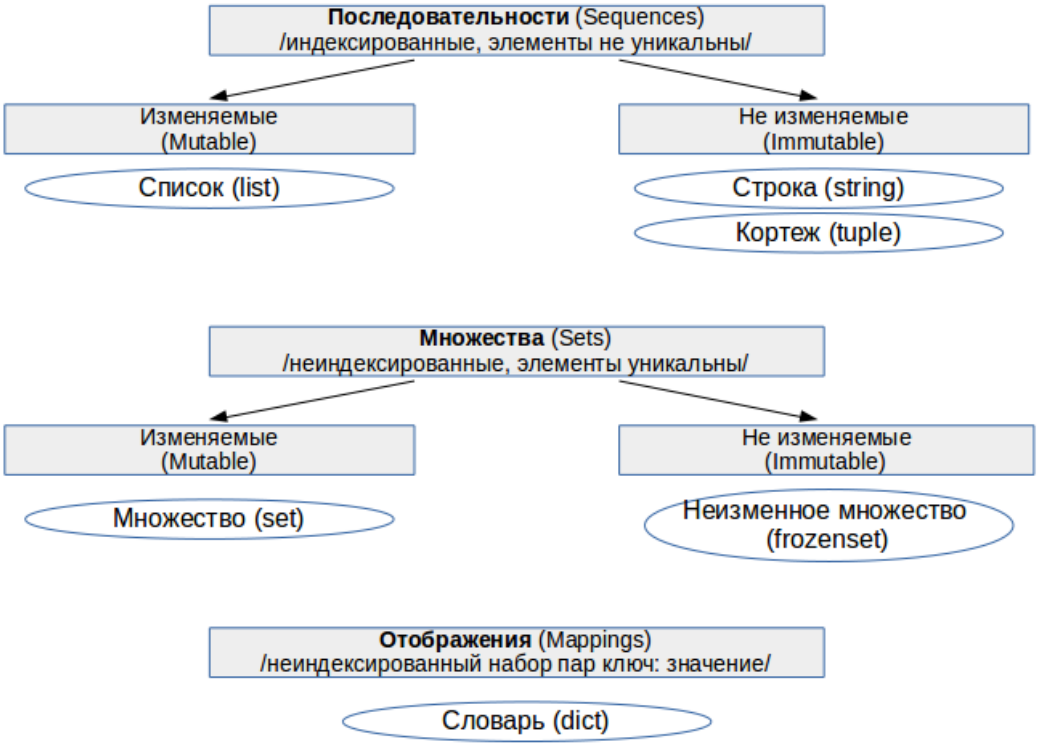
20. Кортежи в Python. Отличия кортежей от списков. Распаковка и частичная распаковка кортежей.



Кортежи в python, они же tuple - это индексированная последовательность, неизменяемая, элементы не уникальны.
Могут хранить данные разных типов, также, как и в list, например:
T = (“5”, 5.3,)

Кортежи могут задаваться несколькими способами:
— Перечисление по типу a = 3, 5. Чтоб создать не int, а tuple, то можно добавить ,

```
a = 1 # a is the value 1
a = 1, # a is the tuple (1,)
```

— Конструктор tuple без аргументов и с аргументами:

```
a = tuple()
b = tuple([3,6])
```

— Использование range и конструктора tuple с аргументами

```
a = tuple(range(5))
```

— С помощью генератора

```
e = tuple(e**2 for e in range(10) if e % 2 == 0)
```

Основное отличия кортежей от списков
— Кортежи (tuple) не изменяются, в отличии от list
— Скорость обработки интерпретатором. Является следствием того, что они не изменяются

— Может использоваться как ключ в словарях dict. Тоже следствие.

Как определить что использовать?
Если программа не предполагает изменения коллекции, то рациональнее использовать tuple, иначе - list. Но это только если вам нужна индексированная коллекция с не уникальными элементами

Распаковка и частичная распаковка кортежей.

Полная запаковка/распаковка tuple работает не только с tuple, но и с list!
Запаковка
Интерпретатор Python по-умолчанию создает tuple, если происходит перечисление элементов коллекции, например:

```
a = 2
b = 3
c = a, b
print(type(c))
#<class 'tuple'>
```

Часто такая штука используется при необходимости возврата нескольких значений от функции, но также часто и используется распаковка, операция, обратная запаковке (ОГО!):

```
def check():
    return 2,5
a, b = check()
```

Частичная распаковка tuple'ов
Во-первых можно использовать _ и __ для заглушки при распаковке:

```
a = 1, 2, 3, 4
_, x, y, _ = a
# x == 2
# y == 3
```

Во-вторых можно использовать *

```
a, *b_list, c = (1, 2, 3, 4, 5)
print(a) #1
print(b_list) #[2,3,4]
print(c) #5
```

В-третьих можно комбинировать это веселье:

```
a, _ , b, *c_list = (1, 2, 3, 4, 5)
print(a) #1
print(b) #3
print(c_list) #[4, 5]
```

Так то распаковка в Python широго используется в том же enumirate

```
a = (4,6,6,3,7,3)
for index, value in enumerate(a):
    print(index,value)
```

или dic.items():

```
```

```
d = dict(sin="cat", s=1)
for key, value in d.items():
    print(key, value)
```