Самое важное

Кортежи:
Кортежи — ещё одна структура данных, она так же, как и строки, неизменяема. В ней можно хранить разнородные данные, но зачастую кортежи используют для хранения однородных данных.

Создать кортеж можно при помощи круглых скобок:
numbers = (1, 2, 3)
или через функцию tuple():
x = [1, 2, 3]
tuple(x) — в этом случае вам нужен будет какой-то итерируемый объект, из которого можно извлечь элементы, например список.

Кортежи очень похожи на списки, но есть различия:

1) кортежи используют для безопасного хранения данных, так как элементы кортежей нельзя изменять;
2) кортежи занимают меньше памяти на компьютере;
3) кортежи работают быстрее.

Схожесть со списками: к кортежам можно применять все те же правила работы с индексами и срезами. Можно применять метод index() для поиска индекса внутри кортежа. Мы даже можем хранить внутри кортежа другой кортеж или список.

Различия: кортежи можно создавать, даже не замечая этого.
Вспомните операцию *a,b = b,a*. Казалось бы, при чём тут кортежи?
Но для Python эта операция разделяется на несколько своих:

1) создаётся кортеж (b,a);
2) происходит распаковка a,b = кортеж с b,a;
3) переменным a,b присваются первый и второй элемент из кортежа.

То есть мы можем перечислять элементы через запятую, а Python сам соберёт их в кортеж.

x = 1, 2, 3
print(x, type(x)) → (1, 2, 3) <class 'tuple'>

Функция enumerate
Эта функция позволяет «пронумеровать» элементы любого итерируемого объекта. Чаще всего она используется в циклах:

```
x = [3,2,1,2,3,2,3]
for i in x:
    print(i)
```

Такой простой цикл позволит нам перебрать элементы списка. Но что, если нам нужно будет узнать индекс определённого элемента? Например, мы хотим узнать индексы всех элементов, которые равны 2.
index() не подойдёт, так как он всегда будет выдавать индекс только первого элемента. Изменять список «на ходу», удаляя из него каждую найденную цифру, — это слишком сложно. И тут нам как раз поможет enumerate():

for index, element in enumerate(x): — изначально enumerate(x) будет возвращать нам кортежи с двумя значениями (номер, элемент из списка);
    print(index, element) — но если мы используем в качестве переменных цикла две переменные, то кортеж будет «распакован» между ними: в первую переменную попадёт номер, а во вторую — элемент из списка.

На каждой итерации номер будет увеличиваться (начиная с 0). Из-за этой особенности этот номер будет совпадать с индексами элементов списка:
0 3 — первое число будет index, второе — element.
1 2
2 1
3 2
4 3
5 2
6 3

Таким же образом можно работать и со словарём:
словарь.items() — этот метод вернёт пары «ключ — значение», которые будут записаны в кортежи.

print(exam_book.items()) → dict_items([('Вася', 3), ('Петя', 4), ('Катя', 5)])

Удобно будет и тут использовать цикл с двумя переменными:
for key, value in exam_book.items() — тогда в первую переменную будут попадать все ключи, а во вторую — все значения.

Составные ключи словарей
Как уже известно, ключи словарей не могут быть изменяемыми типами объектов. По этой причине мы, например, не могли использовать список в качестве ключа. Однако кортеж — это неизменяемый тип данных, а значит, его мы можем использовать в качестве ключа.

Это позволяет создавать ключи, состоящие из нескольких вложенных в кортеж объектов. Так, например, можно записать ФИО. в кортеж и при этом не собирать их в одну сплошную строку, а хранить в строгом порядке, отдельно друг от друга:

```
exam_book = {("Сидорова", "Елена", "Валерьевна"): 5,
                    ("Петров", "Игорь", "Максимович"): 4}
```

Функция zip
Эта функция позволяет соединить элементы нескольких итерируемых объектов по парам:
x = [1, 2, 3]
y = [4, 5, 6]
zipped = zip(x, y) — если мы передадим параметрами два списка.
list(zipped) — изначально мы получим zip-object, который нельзя просто так обработать, но элементы этого объекта можно получить при помощи функции list (либо через цикл по этому объекту).
[(1, 4), (2, 5), (3, 6)] — на выходе мы получим список с кортежами-парами из элементов этих двух списков.
При этом они будут совпадать по порядку, то есть первый элемент первого списка будет соединен с первым элементом второго и так далее.

При помощи этой функции удобно будет создавать новые словари:
print(dict(zipped)) → {1: 4, 2: 5, 3: 6}

Не допускай следующих ошибок!

При создании кортежа из одного элемента добавляйте запятую:

x = (1) — так Python создаст обычное число типа int,
x = (1,) — а вот так Python создаст кортеж с одним элементом.

Не забывайте, что даже при возможности использования кортежей в качестве ключей словаря нельзя использовать кортеж со вложенным списком в качестве ключа.

Помните о том, что если в функцию zip подаются объекты разной длины, то количество пар будет соответствовать наименьшей из двух длин объектов, то есть часть элементов вы можете просто потерять.