# Урок 16.02. Функция enumerate. Метод items.

Мы уже посмотрели что такое кортежи, для чего они могут понадобиться и как с ними работать. И теперь самое время вернуть немного в прошлое, во времена, когда мы работали с циклом for, и посмотреть его еще немного. И как всегда на примере.

У нас есть список очков пяти игроков, и нам нужно вывести информацию по парам: номер игрока – количество очков. Вот решение:

scores = [54, 67, 99, 27]

for i\_player in range(len(scores)):

print(i\_player, scores[i\_player])

Всё верно, и, казалось бы, что тут могло быть не так. На самом деле, такой подход (пронумеровать список) встречается очень часто, а значит должен быть способ его упростить и сделать python’ic (пайтоник) – сделать более читаемым, проще. И для этого в python есть специальная функция enumerate, а в скобках этой функции мы указываем наш итерируемый объект.

scores = [54, 67, 99, 27]

for i\_player, i\_score in enumerate(scores):

print(i\_player, i\_score)

Так мы увидели, что enumerate возвращает нам кортеж состоящий из номера и итерируемого объекта. Вот так всё просто. Однако следует понимать, что через переменную i\_score изменить количество очков не получится.

Метод items.

Давайте еще немного вернемся назад, к словарям. Не смотря, на то, что тот модуль был насыщен множеством новых знаний, кое-что мы пропустили, но не просто так. Нам, просто, нужно было немного узнать про кортежи, и вот теперь с этим знанием расширим способы работы со словарями. И конечно же на пример.

Возьмем пример из предыдущего задания, но сделаем его словарем и добавим к очкам имя. И задача всё та же, вывести имя и очки рядом.

scores\_dict = {

'Vova': 12,

'Alex': 25,

'Ira': 32

}

for i\_name in scores\_dict:

print(i\_name, scores\_dict[i\_name])

И вот наша задача решена, но настоящие питонисты так задачу решать не будут. Как вы уже могли заметить, у нас получилось аналогично предыдущему первому решению. И по традиции узнаем, как же стоит решать эту задачу. В этот раз нам поможет метод словаря items. Данный метод возвращает пару *ключ‑значение* словаря, перебирая словарь.

scores\_dict = {

'Vova': 12,

'Alex': 25,

'Ira': 32

}

for i\_name, i\_score in scores\_dict.items():

print(i\_name, i\_score)

Вот и всё.

А теперь попробуйте самостоятельно изучить, что получится, если сделать перебор в цикле enumerate(scores\_dict).

Далее рассмотрим задачу, которая нам уже попадалась в предыдущем модуле.

Задача «Товары».

В базе данных магазина вся необходимая информация по товарам делится на два словаря: первый отвечает за коды товаров, второй — за списки количества разнообразных товаров на складе:

goods = {

'Лампа': '12345',

'Стол': '23456',

'Диван': '34567',

'Стул': '45678',

}

store = {

'12345': [

{'quantity': 27, 'price': 42},

],

'23456': [

{'quantity': 22, 'price': 510},

{'quantity': 32, 'price': 520},

],

'34567': [

{'quantity': 2, 'price': 1200},

{'quantity': 1, 'price': 1150},

],

'45678': [

{'quantity': 50, 'price': 100},

{'quantity': 12, 'price': 95},

{'quantity': 43, 'price': 97},

],

}

Каждая запись второго словаря отображает, сколько и по какой цене закупалось товаров (цена указана за одну штуку).

Напишите программу, которая рассчитывает, на какую сумму лежит каждого товара на складе, и выводит эту информацию на экран.

Давайте решим данную задачу, используя метод items. Для этого мы пойдем по первому словарю получая название товара и его код, а дальше во втором словаре будем искать товар по коду (через указание кода в качестве индекса), и считать сколько сумму товара лежащего на складе. Вот так будет выглядеть вся программа:

goods = {

'Лампа': '12345',

'Стол': '23456',

'Диван': '34567',

'Стул': '45678',

}

store = {

'12345': [

{'quantity': 27, 'price': 42},

],

'23456': [

{'quantity': 22, 'price': 510},

{'quantity': 32, 'price': 520},

],

'34567': [

{'quantity': 2, 'price': 1200},

{'quantity': 1, 'price': 1150},

],

'45678': [

{'quantity': 50, 'price': 100},

{'quantity': 12, 'price': 95},

{'quantity': 43, 'price': 97},

],

}

for i\_title, i\_code in goods.items():

total\_quantity = 0

total\_cost = 0

for j\_good in store.get(i\_code, {}):

total\_quantity += j\_good['quantity']

total\_cost += j\_good['quantity'] \* j\_good['price']

print('{title} - {quantity} шт, {cost} руб'.format(

title=i\_title,

quantity=total\_quantity,

cost=total\_cost

))

Конечно, данная задача решается и без применения метода items, как вы и должны были сделать в предыдущем модуле. Однако, использование данного метода позволяет сделать задачу более читаемой.

На этом всё.