# Урок 16.03. Составные ключи. Функция zip

Мы с вами увидели, что в python работа с кортежами происходит не прямо, а через какие-нибудь функции или методы, например, через items или enumerate. Однако не редко кортежи связаны и с другими структурами, например, со словарями.

И давайте рассмотрим такой пример. Возьмем наш старый пример с телефонной книгой. Там у на сбыла такая структура

phonebook = {

 'Ваня': 89123456789,

 'Аня': 89014569834,

 'Игорь': 89786549846,

}

И в таком словаре у нам имеются ключи, которые являются уникальными, и двух Вань или Игорей у нас не может быть. Но тут может сложиться странная ситуация – мы не можем добавить двух Иванов (с одним именем «Иван»). В жизни двух Иванов мы можем различить по фамилии, может, и мы можем это как то использовать. Давайте добавим.

phonebook = {

 'Иванов Ваня': 89123456789,

 'Петрова Аня': 89014569834,

 'Сидоров Игорь': 89786549846,

}

Такое решение можно принять, однако в таком случае возникает много вопросов с обработкой таких данных (поиск, сортировка, для всего этого мы должны дополнительно проводить обработку данной строки). И тут мы могли бы разделить имя и фамилию на две строки и объединить их в один список, но, как мы знаем, ключом словаря должен быть не изменяемый тип (immutable), а это точно не про списки. И как вы уже догадались, здесь нам могут помочь кортежи. Они относятся к не изменяемым типам и могут хранить несколько раздельных данных. Давайте попробуем.

phonebook = {

 ('Иванов', 'Ваня'): 89123456789,

 ('Петрова', 'Аня'): 89014569834,

 ('Сидоров', 'Игорь'): 89786549846,

}

print(phonebook)

Python ошибок нам не выдает, а значит запустим программу. Теперь всё работает. И теперь мы имеем больше возможности по обработки данных, и в дальнейшем можем сэкономить время на обработке данных и тем самым повысить скорость работы программы. Такое разделение ключа на несколько составляющих на языке баз данных называется составной ключ, т.е. уникальность ключа определяется не одним параметров, а несколькими.

И давайте попробуем в действие удобство обработки, дополнив нашу задачу тем, что мы должны вывести имя и номер телефона всех людей, фамилия которых «Сидоров».

phonebook = {

 ('Иванов', 'Ваня'): 89123456789,

 ('Петрова', 'Аня'): 89014569834,

 ('Сидоров', 'Игорь'): 89786549846,

}

print(phonebook)

for i\_person in phonebook:

 if 'Сидоров' in i\_person:

 print(i\_person[1], phonebook[i\_person])

Запустим и проверим программу.

А теперь давайте проверим можно ли в качестве ключа использовать кортеж, который содержит список, ведь, мы помним, что кортеж может содержать список. Напишем такую программу и запустим её.

my\_dict = {

 ('Иванов', 'Иван', []): 123123543,

}

print(my\_dict)

Проверьте самостоятельно.

**Функция zip.**

Мы уже знаем довольно большое количество функций и методов, которые сильно упрощают жизнь программисту. И перед выполнением итоговых заданий модулю давайте познакомимся с новой очень полезной функцией python.

Рассмотрим на такую задачу. У нас есть два списка: в одном имена людей, а во втором их возраст. И нам нужно их объединить по парам «имя, возраст». Такую задачу мы с вами можем решить через переменную счетчика. Однако мы не будем это придумывать, а сразу воспользуемся новой функцией, которая позволяет нам объединять итерируемые объект вместе, и такая функция называется zip. И вот как она работает.

names = ['Tom', 'Bob', 'Alice']

ages = [12, 15, 20]

people = zip(names, ages)

print(people)

Запустим и… Мы получили, что-то не то, что хотели. На самом деле это лишь нам говорит, что python экономит память, а точнее он не занимается копированием данных из этих двух списков в новую структуру. Но как с ней работать, а очень просто, чаще всего такой типа данных нужно использовать в циклах как объект по который мы будем перебирать.

names = ['Tom', 'Bob', 'Alice']

ages = [12, 15, 20]

people = zip(names, ages)

print(people)

for i\_person in people:

 print(i\_person)

Запустим и увидим, что мы получили кортеж, которые объединяет в себе элементы этих списков.

На самом деле получить объединенный список можно получить вот так

names = ['Tom', 'Bob', 'Alice']

ages = [12, 15, 20]

people = list(zip(names, ages))

print(zip)

Правда, получится список кортежей. На самом деле можно сделать и словарь.

names = ['Tom', 'Bob', 'Alice']

ages = [12, 15, 20]

people = dict(zip(names, ages))

print(zip)

Так же можно использовать zip и в list comprehension. Представим, что нам нужен не возраст, а год рождения. Тогда нам нужно из текущего года вычесть их возраст, и всё это будет словарем, где имя – ключ.

names = ['Tom', 'Bob', 'Alice']

ages = [12, 15, 20]

people = {name: 2024 - age for name, age in zip(names, ages)}

print(people)

Вот так легко и просто можно объединять перечисляемые (iterable) объекты.

А что будет, если в одном из списков объектов будет больше, чем в другом? Проверьте самостоятельно.