# Урок 15.03. Вложенные словари и значение по умолчанию в get

Мы уже посмотрели, что из себя представляют словари. Вообще данные могут быть разные, а еще иногда может складываться ситуация, когда некоторые данные еще не готовы. И давайте рассмотрим одну из таких.

Задача «Структура данных».

Не редко программистам приходится создавать структуру данных «на лету», т.е. в режиме реального времени. И вот нам для проекта необходимо создать такую структуру:

{

'server': {

'host': '127.0.0.1',

'port': '10'

},

'configuration': {

'ssh': {

'access': 'true',

'login': 'ivan',

'password': 'qwerty'

}

}

}

На самом деле примерно такую структуру данных можно встретить в реальных проектах. В программе эта структура так и будет выглядеть:

data = {

'server': {

'host': '127.0.0.1',

'port': '10'

},

'configuration': {

'ssh': {

'access': 'true',

'login': 'ivan',

'password': 'qwerty'

}

}

}

И вот мы создали эту структуру, но создали программно. Однако в программах такие структуры создаются не просто так, а, чтобы проверять какие-либо свойства или использовать в качестве настроечных параметров. Например, нам нужно вывести номер порта, по которому работает наш сервер, тогда программа будет выглядеть следующим образом:

data = {

'server': {

'host': '127.0.0.1',

'port': '10'

},

'configuration': {

'ssh': {

'access': 'true',

'login': 'ivan',

'password': 'qwerty'

}

}

}

print(data['server']['port'])

Получилось. Теперь давайте получим имя логина и попробуем поменять его на «Vova».

data = {

'server': {

'host': '127.0.0.1',

'port': '10'

},

'configuration': {

'ssh': {

'access': 'true',

'login': 'ivan',

'password': 'qwerty'

}

}

}

print(data['server']['port'])

data['configuration']['ssh']['login'] = 'vova'

print(data)

Ага. Получилось. Думаю, вы уже догадались как работать со всеми этими полями.

Как уже говорилось, некоторые поля к моменту работы программы могут быть еще не заполнены. Давайте представим такую ситуацию, что у нас есть данные сервера, но нет еще сохраненных логинов и паролей в конфигурации, т.е. вот так:

data = {

'server': {

'host': '127.0.0.1',

'port': '10'

},

}

print(data)

И мы хотим проверить есть у нас в структуре данных логины, и для этого напишем такую проверку:

data = {

'server': {

'host': '127.0.0.1',

'port': '10'

},

}

if data.get('configuration').get('ssh').get('login'):

print('В структуре уже есть логин')

Но наша программа выйдет с ошибкой «AttributeError: 'NoneType' object has no attribute 'get'», которая говорит, что у нас нет метода get для значения None, а это получилось потому что ***data.get('configuration'***) вернуло нам ***None***, а дальше мы пытаемся использовать метод ***.get('ssh')*** у ***None***, что конечно же не правильно. Для таких случаев в запросе придуман механизм возвращения значения по умолчанию, т.е. это значение, которое будет получаться, если метод ***get*** не найдет поле, по которому обращаются. И так как мы ничего не нашли в нашем словаре, то и нужно возвращать пустой словарь:

data = {

'server': {

'host': '127.0.0.1',

'port': '10'

},

}

if data.get('configuration', {}).get('ssh', {}).get('login'):

print('В структуре уже есть логин')

Отлично. Такая программа уже работает.

Давайте рассмотри вот такую задачу.

Задача «Игроки».

Есть готовый словарь игроков, у каждого игрока есть имя, команда, в которой он играет, а также его текущий статус, в котором указано, отдыхает он, тренируется или путешествует:

players\_dict = {

1: {'name': 'Vanya', 'team': 'A', 'status': 'Rest'},

2: {'name': 'Lena', 'team': 'B', 'status': 'Training'},

3: {'name': 'Maxim', 'team': 'C', 'status': 'Travel'},

4: {'name': 'Egor', 'team': 'C', 'status': 'Rest'},

5: {'name': 'Andrei', 'team': 'A', 'status': 'Training'},

6: {'name': 'Sasha', 'team': 'A', 'status': 'Rest'},

7: {'name': 'Alina', 'team': 'B', 'status': 'Rest'},

8: {'name': 'Masha', 'team': 'C', 'status': 'Travel'}

}

Нам нужно написать программу, которая выводит на экран список всех членов команды А, которые, которые сейчас отдыхают.

Предлагаю использовать list comprehension, и такая программа будет выглядеть вот так:

players\_dict = {

1: {'name': 'Vanya', 'team': 'A', 'status': 'Rest'},

2: {'name': 'Lena', 'team': 'B', 'status': 'Training'},

3: {'name': 'Maxim', 'team': 'C', 'status': 'Travel'},

4: {'name': 'Egor', 'team': 'C', 'status': 'Rest'},

5: {'name': 'Andrei', 'team': 'A', 'status': 'Training'},

6: {'name': 'Sasha', 'team': 'A', 'status': 'Rest'},

7: {'name': 'Alina', 'team': 'B', 'status': 'Rest'},

8: {'name': 'Masha', 'team': 'C', 'status': 'Travel'}

}

team\_a\_member = [

player.get('name')

for player in players\_dict.values()

if player.get('team') == 'A' and player.get('status') == 'Rest'

]

print(team\_a\_member)

Вот так просто получилась наша программа.