MODULE 02 - 064: Python - Adding new Key/Value pairs into Dictionaries

Introduction

Dictionaries in Python are dynamic data structures that allow key-value storage. One of their powerful features is the ability to dynamically add new key-value pairs. This guide explores how to update dictionaries by inserting new elements, including considerations for best practices.

Python Documentation: Dictionaries

Adding New Key-Value Pairs

The process of adding elements to a dictionary in Python is straightforward. It involves assigning a value to a new key using square bracket notation (`dict[key] = value`).

Basic Syntax

```
# Creating a dictionary
teams = {
    "astros": ["Altuve", "Correa", "Bregman"],
    "angels": ["Trout", "Pujols"],
    "yankees": ["Judge", "Stanton"],
}

# Adding a new key-value pair
teams['red sox'] = ['Price', 'Betts']

# Output updated dictionary
print(teams)
```

The new key 'red sox' is added dynamically to the dictionary.

Python Documentation: Dictionary Operations

Updating Dictionaries with `update()`

An alternative way to add multiple key-value pairs at once is by using the `update()` method.

Using `update()` Method

```
teams.update(
{
"dodgers": ["Kershaw", "Bellinger"], "giants": ["Posey", "Crawford"]
}
)
print(teams)
```

This approach is useful when merging multiple values at once.

Python Documentation: dict.update()

Considerations for Dictionary Keys

- **Keys must be unique:** If a key already exists, assigning a new value will overwrite the previous value.
- **Keys can be any immutable type:** Strings, numbers, and tuples (if they contain only immutable elements) can be used as keys.
- **Keys can contain spaces:** Strings used as keys may include spaces or special characters.

Example of overwriting an existing key:

large

Figure 1: large

```
teams["angels"] = ["Ohtani", "Trout"]  # Updates existing key
print(teams)
```

Best Practices

Use concise and meaningful keys to improve readability. **Avoid overwriting keys accidentally** by checking with `dict.get()` before adding new ones. **Use `update()` when adding multiple pairs** to maintain cleaner code. **Ensure keys are immutable types** to prevent unexpected behavior.

Summary

- New key-value pairs can be added using `dict[key] = value`.
- Multiple pairs can be inserted using `update()`.
- Dictionary keys must be unique and immutable.
- Overwriting values happens if an existing key is reassigned.

Video speech Lesson

In this guide, we're going to extend our knowledge on nested collections inside of Python dictionaries and we're going to see how we can add new key-value pairs.

Now, thankfully the syntax for performing this action is relatively straightforward and all we have to do is update the list so if we have a dictionary-like we have right here with teams we can simply add a new key the same way that we would perform a query but instead of performing that query on an item that exists we're going to add a new one.

So, if I come here and I say red sox then to add this key-value pair I can add it and then since I'm adding a list I'm going to just add Price and Betts and add two Red Sox players.

And, now, if I run this you'll see that our team's list which is what I'm printing out now contains the Red Sox:

A part of the reason why I also wanted to use this example was to see that you can break your regular keys into multiple words.

That's one of the nice things about using strings when it comes to implementing keys inside of a dictionary because **they are perfectly fine to be multiple words or even sentences.**

Now, obviously the longer the key is and the harder it is to type out or the harder it is to look up the more that you could potentially be running into bugs where you think that you have the right name for the key.

But you actually have one character off or something like that.

So usually you want to keep your keys as simple and as explicit as possible.

But, I did want to show in this example that they are simply strings.

They can be treated just like any other string in the sense that you can have spaces in them they can contain regular alphanumeric characters they could contain numbers anything like that.

Anything that you'd put in a string that you can put in a key just like we did here.

So that is how you can add new elements to new key-value pairs to dictionaries in Python.

Code

```
teams = {
 "astros": ["Altuve", "Correa", "Bregman"],
 "angels": ["Trout", "Pujols"],
 "yankees": ["Judge", "Stanton"],
}
teams['red sox'] = ['Price', 'Betts']
print(teams)
```