The "toc" dictionary represents the table of contents for a book. Fill in the blanks to do the following: 1) Add an entry for Epilogue on page 39. 2) Change the page number for Chapter 3 to 24. 3) Display the new dictionary contents. 4) Display True if there is Chapter 5, False if there isn't.

toc = {"Introduction":1, "Chapter 1":4, "Chapter 2":11, "Chapter 3":25, "Chapter 4":30}

toc["Epilogue"]=39 # Epilogue starts on page 39

toc["Chapter 3"]=24 # Chapter 3 now starts on page 24

print(toc) # What are the current contents of the dictionary?

print("Chapter 5" in toc) # Is there a Chapter 5?

RunReset

Here is your output:

{'Introduction': 1, 'Chapter 1': 4, 'Chapter 2': 11, 'Chapter 3': 24, 'Chapter 4': 30, 'Epilogue': 39}

False

Great work! You've made the changes to the dictionary

exactly as requested.

Dictionaries Defined

Dictionaries are another data structure in Python. They’re similar to a list in that they can be used to organize data into collections. However, data in a dictionary isn't accessed based on its position. Data in a dictionary is organized into pairs of keys and values. You use the key to access the corresponding value. Where a list index is always a number, a dictionary key can be a different data type, like a string, integer, float, or even tuples.

When creating a dictionary, you use curly brackets: **{}**. When storing values in a dictionary, the key is specified first, followed by the corresponding value, separated by a colon. For example, **animals = { "bears":10, "lions":1, "tigers":2 }** creates a dictionary with three key value pairs, stored in the variable animals. The key "bears" points to the integer value 10, while the key "lions" points to the integer value 1, and "tigers" points to the integer 2. You can access the values by referencing the key, like this: **animals["bears"]**. This would return the integer 10, since that’s the corresponding value for this key.

Now, it's your turn! Have a go at iterating over a dictionary!

Complete the code to iterate through the keys and values of the cool\_beasts dictionary. Remember that the items method returns a tuple of key, value for each element in the dictionary.

cool\_beasts = {"octopuses":"tentacles", "dolphins":"fins", "rhinos":"horns"}

for key,value in cool\_beasts.items():

    print("{} have {}".format(key,value))

RunReset

Here is your output:

octopuses have tentacles

dolphins have fins

rhinos have horns

Nice job! Your dictionary skills are getting stronger and

stronger!

You can also check if a key is contained in a dictionary using the **in** keyword. Just like other uses of this keyword, it will return True if the key is found in the dictionary; otherwise it will return False.

Dictionaries are mutable, meaning they can be modified by adding, removing, and replacing elements in a dictionary, similar to lists. You can add a new key value pair to a dictionary by assigning a value to the key, like this: **animals["zebras"] = 2**. This creates the new key in the animal dictionary called zebras, and stores the value 2. You can modify the value of an existing key by doing the same thing. So **animals["bears"] = 11** would change the value stored in the bears key from 10 to 11. Lastly, you can remove elements from a dictionary by using the **del**keyword. By doing **del animals["lions"]** you would remove the key value pair from the animals dictionary.

Iterating Over Dictionaries

You can iterate over dictionaries using a *for* loop, just like with strings, lists, and tuples. This will iterate over the sequence of keys in the dictionary. If you want to access the corresponding values associated with the keys, you could use the keys as indexes. Or you can use the **items** method on the dictionary, like **dictionary.items()**. This method returns a tuple for each element in the dictionary, where the first element in the tuple is the key and the second is the value.

If you only wanted to access the keys in a dictionary, you could use the **keys()** method on the dictionary: **dictionary.keys()**. If you only wanted the values, you could use the **values()**method: **dictionary.values()**.

In Python, a dictionary can only hold a single value for a given key. To workaround this, our single value can be a list containing multiple values. Here we have a dictionary called "wardrobe" with items of clothing and their colors. Fill in the blanks to print a line for each item of clothing with each color, for example: "red shirt", "blue shirt", and so on.

wardrobe = {"shirt":["red","blue","white"], "jeans":["blue","black"]}

for dresses , colors in wardrobe.items():

    for color in colors:

        print("{} {}".format(color,dresses))

RunReset

Here is your output:

red shirt

blue shirt

white shirt

blue jeans

black jeans

Woohoo! You're really mastering the Python dictionaries and

lists!

## Dictionary Methods Cheat Sheet

## Dictionary Methods Cheat Sheet

**Definition**

x = {key1:value1, key2:value2}

**Operations**

* len(dictionary) - Returns the number of items in the dictionary
* for key in dictionary - Iterates over each key in the dictionary
* for key, value in dictionary.items() - Iterates over each key,value pair in the dictionary
* if key in dictionary - Checks whether the key is in the dictionary
* dictionary[key] - Accesses the item with key key of the dictionary
* dictionary[key] = value - Sets the value associated with key
* del dictionary[key] - Removes the item with key key from the dictionary

**Methods**

* dict.get(key, default) - Returns the element corresponding to key, or default if it's not present
* dict.keys() - Returns a sequence containing the keys in the dictionary
* dict.values() - Returns a sequence containing the values in the dictionary
* dict.update(other\_dictionary) - Updates the dictionary with the items coming from the other dictionary. Existing entries will be replaced; new entries will be added.
* dict.clear() - Removes all the items of the dictionary

Check out the [official documentation for dictionary operations and methods](https://docs.python.org/3/library/stdtypes.html#mapping-types-dict).