**PYTHON ASSGN. 5**

1. **What does an empty dictionary’s code look like?**

**ANS.** In Python, an empty dictionary is denoted by a pair of curly braces with nothing inside. Here is an example:

**empty\_dict = {}**

You can also create an empty dictionary using the dict() function, without passing any arguments:

**empty\_dict = dict()**

Both of these approaches will create an empty dictionary that you can later populate with key-value pairs.

1. **What is the value of a dictionary value with the key ‘foo’ and the value 42?**

**Ans.** The value of a dictionary with the key 'foo' and the value 42 is simply 42. In Python, dictionaries consist of key-value pairs, where each key is unique and associated with a particular value. In this case, the value associated with the key 'foo' is 42. To access this value, you can use the key 'foo' to retrieve it from the dictionary. Here is an example of how you can access this value:

**my\_dict = {'foo': 42}**

**value\_of\_foo = my\_dict['foo']**

**print(value\_of\_foo) # This will print 42**

Make sure that the key you are using for retrieval is indeed present in the dictionary; otherwise, it will result in a KeyError.

1. **What is the most significant distinction between a dictionary and a list?**

**Ans.** The most significant distinction between a dictionary and a list lies in their structure and the way they store and retrieve data.

* **Structure**:
  + A list is an ordered collection of items where each item has an index, starting from 0, that represents its position in the list.
  + A dictionary is an unordered collection of data in a key-value pair format. Each element in a dictionary is stored as a key-value pair, where each key is unique and is used to access its corresponding value.
* **Indexing and Accessing Elements**:
  + In a list, elements are accessed by their position (index) in the list. For example, **my\_list[0]** will access the first element in the list.
  + In a dictionary, elements are accessed by their keys. For example, **my\_dict['key']** will access the value associated with the key 'key'.
* **Mutability**:
  + Lists are mutable, which means you can change, add, or remove elements after the list has been created.
  + Dictionaries are also mutable, allowing for changes to the key-value pairs within the dictionary.
* **Use Cases**:
  + Lists are typically used for ordered collections of items where the order or position of the elements matters.
  + Dictionaries are used when you want to associate unique keys with values, creating a mapping between the keys and their corresponding values.

In summary, while both lists and dictionaries are data structures in Python used for storing collections of data, they differ significantly in their structure, access methods, and use cases. Lists are best used when the order of elements is important, while dictionaries are ideal for mapping unique keys to their corresponding values.

1. **What happens if you try to access spam[‘foo’] if spam is {‘bar’: 100}?**

**Ans.** If you try to access **spam['foo']** where **spam** is **{'bar': 100}**, you will encounter a **KeyError**. This is because the dictionary **spam** does not contain the key **'foo'**, so attempting to access it will raise an error.

In Python, accessing a key that does not exist in a dictionary using square brackets will result in a **KeyError**. Dictionaries are key-value pairs, so each key must be unique. In this case, attempting to access the key **'foo'** in the dictionary **spam** will raise an error because **'foo'** is not a key in the dictionary.

1. **If a dictionary is stored in spam, what is the difference between the expressions ‘cat’ in spam and ‘cat’ in spam.keys()?**

**Ans.** In Python, when you use the expression **'cat' in spam**, it checks whether the string **'cat'** is a key in the dictionary **spam**. If **'cat'** is a key in **spam**, it returns **True**; otherwise, it returns **False**.

On the other hand, when you use the expression **'cat' in spam.keys()**, it returns **True** if there exists a key in the dictionary **spam** that matches the string **'cat'**. The **keys()** method in Python returns a view object that displays a list of all the keys in the dictionary.

Therefore, the key difference between the two expressions is that the first one checks directly whether the key **'cat'** exists in the dictionary **spam**, while the second one checks whether the string **'cat'** exists in the list of keys returned by **spam.keys()**. Both expressions can yield the same result if the dictionary **spam** contains the key **'cat'**. However, if there is a possibility that the key might exist as a value instead of a key, the two expressions could yield different results.

1. **What is a shortcut for the following code?**

**if ‘color’ not in spam:**

**spam[‘color’] = ‘black’**

**Ans.** You can use the **setdefault** method for dictionaries. This method checks if a key is in the dictionary. If the key is in the dictionary, it returns its value. If the key is not in the dictionary, it inserts the key with a value to the dictionary. Here's how you can use it for your specific case:

**spam.setdefault('color', 'black')**

This line of code is equivalent to the if statement you provided. If 'color' is not in the dictionary spam, it will add the key 'color' with the value 'black' to the dictionary. If 'color' is already in the dictionary, it will not change the value.

1. **How do you “pretty print” dictionary values using which module and function?**

**Ans.** You can use the json module in Python to pretty print dictionary values. Specifically, you can use the json.dumps() function with the indent parameter to make the output more human-readable. Here's an example:

**import json**

**my\_dict = {"name": "John", "age": 30, "city": "New York"}**

**# Pretty print the dictionary**

**print(json.dumps(my\_dict, indent=4))**

In this example, the indent parameter specifies the number of spaces to use for indentation in the output. You can adjust this value as needed to control the indentation level.