Assignment 5  
  
**Q.1 What does an empty dictionary’s code look like?**Ans. There are two common ways to create an empty dictionary in Python:

1. Using curly braces:  
   empty\_dict = {}
2. Using the dict() function:  
   empty\_dict = dict()

**Q.2 What is the value of a dictionary value with the key "foo" and the value 42?**Ans. If a dictionary has a key "foo" with a value of 42, then the value you would get by accessing that key would be **42**.

Here's an example:  
my\_dict = {"foo": 42}

value = my\_dict["foo"] *# Accessing the value with the key "foo"*

print(value) *# This will print 42***Q.3 What is the most significant distinction between a dictionary and a list?**Ans. The most significant distinction between a dictionary and a list in Python lies in how they store and access data:  
**Ordering:**  
List: Ordered collection. Elements maintain the order they were added in. You can access elements by their position (index), starting from 0.  
Dictionary: Unordered collection. Elements are not guaranteed to be in any specific order. You access elements using keys, which can be any immutable data type (like strings, numbers, or tuples).  
  
**Duplicate Values:**  
List: Can contain duplicate elements. You can have multiple instances of the same value at different positions in the list.  
Dictionary: Cannot contain duplicate keys. Each key must be unique within the dictionary. However, the values associated with the keys can be duplicates.

**Q.4 What happens if you try to access spam ['foo'] if spam is {'bar':100}?  
Ans.**   
If you try to access spam['foo'] and the dictionary spam is defined as {'bar': 100}, you will encounter a KeyError.  
This happens because the dictionary spam only contains one key-value pair: "bar" maps to the value 100. Since there's no key named "foo" in the dictionary, attempting to access it using spam['foo'] results in a KeyError.  
The KeyError essentially signals that the program tried to access a key that doesn't exist within the dictionary.  
  
**Q.5 If a dictionary is stored in spam, what is the difference between the expressions 'cat' in spam and 'cat' in spam.keys()?**  
Ans. There is actually no difference between the expressions 'cat' in spam and 'cat' in spam.keys() in terms of what they check.  
Both expressions use the in operator to verify if a key exists in the dictionary spam. They will return:  
 1. True: If the key "cat" is present in the dictionary spam.  
 2. False: If the key "cat" is not found in spam.  
  
The reason why spam.keys() doesn't make a difference here is because the in operator works directly with dictionaries. It can efficiently check for key existence without needing to explicitly create the keys view.  
  
Explanation :  
'cat' in spam: This directly checks if the key "cat" exists within the dictionary spam.  
'cat' in spam.keys(): Here, spam.keys() creates a view of all the keys in the dictionary spam. Then, the in operator checks if the key "cat" is present within that collection of keys.Since Python's in operator can handle dictionaries directly, both methods achieve the same outcome.  
  
Q.6 If a dictionary is stored in spam, what is the difference between the expressions "cat"in spam and "cat" in spam.values()?  
Ans. The difference between "cat" in spam and "cat" in spam.values() lies in what they check within the dictionary spam:  
Explanation:  
'cat' in spam: This expression utilizes the in operator to search for the key "cat" within the dictionary spam. Dictionaries are collections of key-value pairs, and keys must be unique. So, this check determines if "cat" exists as a key that unlocks a value in the dictionary.

'cat' in spam.values(): Here, spam.values() returns a view of all the values present in the dictionary spam. Then, the in operator searches within this view of values to see if there's an element that matches "cat". This essentially checks if any value in the dictionary, irrespective of its key, is equal to "cat".  
  
Q.7 What is a shortcut for the following code?  
If ‘color’ not in spam:  
spam[‘color’] = ‘black  
Ans. The shortcut for the code you provided is:  
spam.setdefault('color', 'black')  
This uses the setdefault() method of dictionaries in Python. It offers a concise way to set a default value for a key if it's missing from the dictionary.  
How it works:  
spam: This refers to the dictionary you're working with.  
'color': This is the key you want to check and potentially set a default value for.  
'black': This is the default value that will be assigned to the key "color" if it doesn't already exist in the dictionary spam.  
  
Q.8 How do you “pretty print” dictionary values using which module and function?  
Ans. There are two main ways to pretty print dictionary values in Python:  
1. Using the pprint module:

The pprint module provides a function named pprint() that can be used to format various data structures, including dictionaries, in a human-readable way. It adds indentation and line breaks to improve readability.  
Python Code:  
import pprint  
my\_dict = {'name': 'Alice', 'age': 30, 'city': 'New York'}  
pprint.pprint(my\_dict)  
  
2. Using the json module:  
The json module offers a function named dumps() that can be used to convert a Python dictionary into a JSON (JavaScript Object Notation) string. By default, the dumps() function includes indentation, making the JSON output easier to read.  
Python Code:  
import json  
my\_dict = {'name': 'Alice', 'age': 30, 'city': 'New York'}  
print(json.dumps(my\_dict, indent=4)) # You can adjust the indent level as needed