**Python Assignment 5**

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

An empty dictionary in Python is represented by a pair of curly braces {} with nothing inside. Here's what it looks like:

empty\_dict = {}

2. What is the value of a dictionary value with the key 'foo' and the value 42?

The value of a dictionary with the key 'foo' and the value 42 would look like this:

{'foo': 42}

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

The most significant distinction between a dictionary and a list in Python is how they store and retrieve elements:

Storage and Access Mechanism:

Dictionary: A dictionary stores elements as key-value pairs, where each key is unique and maps to a specific value. Keys are used to access the corresponding values quickly. Dictionaries are useful for associating and looking up values based on their unique keys.

List: A list stores elements in an ordered sequence, and each element is assigned an index based on its position in the list. You access list elements by their index, starting from 0. Lists are useful for maintaining an ordered collection of items.

While both dictionaries and lists are versatile data structures, dictionaries are particularly useful when you need to associate values with specific keys, while lists are more suitable for maintaining an ordered collection of elements that you access by their positions.

4. What happens if you try to access spam['foo'] if spam is {'bar': 100}?

The KeyError occurs when you try to access a key that is not present in the dictionary.

5. If a dictionary is stored in spam, what is the difference between the expressions 'cat' in spam and

'cat' in spam.keys()?

If a dictionary is stored in the variable spam, the expressions 'cat' in spam and 'cat' in spam.keys() both check for the presence of the key 'cat' in the dictionary. However, they have slightly different behaviors:

'cat' in spam:

This expression checks if the key 'cat' exists in the dictionary spam. It searches through the keys of the dictionary and returns True if the key is found, and False otherwise. This expression is equivalent to checking for the existence of a key in the dictionary.

'cat' in spam.keys():

This expression explicitly checks if the key 'cat' exists in the list of keys of the dictionary spam. It retrieves the list of keys using the keys() method and then checks if 'cat' is present in that list. Like the previous expression, it returns True if the key is found and False otherwise.

6. If a dictionary is stored in spam, what is the difference between the expressions 'cat' in spam and

'cat' in spam.values()?

'cat' in spam:

This expression checks if the string 'cat' exists as a key in the dictionary spam. It searches for the key 'cat' among the keys of the dictionary and returns True if the key is found, and False otherwise. It does not check the values of the dictionary.

'cat' in spam.values():

This expression checks if the string 'cat' exists as a value in the dictionary spam. It searches for the value 'cat' among the values associated with the dictionary's keys and returns True if the value is found, and False otherwise.

7. What is a shortcut for the following code?

if 'color' not in spam:

spam['color'] = 'black'

Ans :- spam.setdefault('color', 'black')

8. How do you "pretty print" dictionary values using which module and function?

To "pretty print" dictionary values in Python, you can use the json module's dumps() function with the indent parameter. This will format the dictionary in a more readable and organized manner. Here's how you can do it:

import json

my\_dict = {

'name': 'Alice',

'age': 30,

'city': 'New York',

'interests': ['reading', 'hiking', 'cooking']

}

# Use json.dumps() with the indent parameter to pretty print the dictionary

pretty\_json = json.dumps(my\_dict, indent=4)

print(pretty\_json)

In this example, the indent parameter is set to 4, which means the JSON output will be indented with four spaces for each level of nesting. This makes the structure of the dictionary more visually appealing and easier to read.

Remember to import the json module before using the dumps() function.

If you want to pretty print the dictionary to a file, you can use the dump() function with the indent parameter:

with open('output.json', 'w') as f:

json.dump(my\_dict, f, indent=4)

This will write the pretty printed JSON representation of the dictionary to the output.json file.