ASSIGNMENT 5

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

An empty dictionary in Python is represented using curly braces {} without any key-value pairs.

Example

```
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 be **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 data:

Key-Value vs. Ordered Elements:

Dictionary: A dictionary is an unordered collection of data that uses key-value pairs to store and retrieve elements. Each element in a dictionary is associated with a unique key, and you access values by specifying the key. Dictionaries are used when you need to map keys to values for efficient data retrieval.

```
my_dict = {'name': 'John', 'age': 30}

name = my_dict['name']                    # Access value by key
```

List: A list is an ordered collection of elements where each element is accessed by its position (index) in the list. Lists are ordered, meaning elements are stored in a specific sequence, and you access elements by their index.

```
my_list = [1, 2, 3, 4, 5]

value = my_list[2]                        # Access element by index
```

Mutability:

Dictionary: Dictionaries are mutable, which means you can change the values associated with existing keys, add new key-value pairs, or remove key-value pairs from a dictionary.

List: Lists are also mutable. You can modify elements within a list, append new elements, insert elements, or remove elements from a list.

Access and Retrieval Efficiency

Dictionary: Dictionaries are highly efficient for accessing and retrieving values by key. The time complexity for dictionary lookups is typically O(1), meaning it's constant time and doesn't depend on the size of the dictionary.

List: Lists are efficient for accessing elements by index, but the time complexity for accessing elements is O(n) when searching by value since it depends on the list's length.

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

If you try to access spam['foo'] and spam is a dictionary containing {'bar': 100}, you will encounter a KeyError because the key 'foo' does not exist in the dictionary spam

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

the expressions cat in spam and cat in spam.keys() are functionally equivalent when checking if the key cat exists in the dictionary spam. Both expressions check for the presence of the key within the dictionary and return a Boolean value based on whether the key is found.

cat in spam

This expression checks if the key cat exists directly within the dictionary spam. It returns True. if cat is a key in the dictionary spam, and False otherwise.

```
spam = {'cat': 1, 'dog': 2, 'fish': 3}

print('cat' in spam)  # Output: True

print('bird' in spam)  # Output: False
```

cat in spam.keys()

This expression first calls the keys() method on the dictionary spam, which returns a view of the dictionary's keys.It then checks if cat exists within that view of keys.It also returns True if cat is a key in the dictionary spam, and False otherwise.

```
spam = {'cat': 1, 'dog': 2, 'fish': 3}

print('cat' in spam.keys())  # Output: True

print('bird' in spam.keys())  # Output: False
```

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 checks if 'cat' is a key in the dictionary, while 'cat' in spam.values() checks if 'cat' is a value in the dictionary. The former checks keys, and the latter checks values

7. What is a shortcut for the following code?

```
if 'color' not in spam:

spam['color'] = 'black'
```

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

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

To "pretty print" dictionary values in Python by using the `pprint` module (pretty-print module) and its `pprint()` function. The `pprint()` function is part of the `pprint` module, which stands for "pretty-print."

```
import pprint

my_dict = {
   'name': 'John',
   'age': 30,
   'address': {
      'street': '123 Main St',
      'city': 'Exampleville',
      'zip': '12345'
   }
}

pprint.pprint(my_dict)          # Pretty print the dictionary
```

The pprint() function formats the dictionary in a way that's easier to read, especially when dealing with nested dictionaries or complex data structures. It indents nested elements and formats the output to make it more visually appealing.Using the pprint module is particularly useful when need to display nested dictionaries, lists, or other data structures in a clear and organized manner.