ASSIGNMENT-5(PYTHON)

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

An empty dictionary in Python is represented by a pair of curly braces {} with no key-value pairs inside. Here's what it looks like:

```
empty_dict = {}
```

This creates an empty dictionary named empty_dict. It contains no key-value pairs and is effectively an empty container ready to store data.

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.

In Python, dictionaries are collections of key-value pairs, where each key is associated with a corresponding value. To access the value associated with a specific key in a dictionary, you use the key within square brackets ([]) after the dictionary variable.

Here's how you would access the value '42' using the key 'foo':

```
my_dict = {'foo': 42}
value = my_dict['foo']
print(value)  # Output: 42
```

So, the value of the dictionary with the key 'foo' and the value 42 is 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 organize and access their elements:

1. Organization:
    - Lists are ordered collections of elements, where each element is accessed by its position (index) in the list. The order of elements in a list is preserved.

- Dictionaries are unordered collections of key-value pairs, where each element (value) is accessed by its associated key. The order of elements in a dictionary is not guaranteed.

2. Accessing Elements:
   - In a list, elements are accessed by their integer index positions, starting from 0. You can use indexing and slicing to access individual elements or sublists.
   - In a dictionary, elements (values) are accessed by their keys. Each key in a dictionary must be unique, and it is used to retrieve the corresponding value.

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

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

In Python, when you attempt to access a key in a dictionary that does not exist, it raises a KeyError exception. This occurs because dictionaries use keys to access their corresponding values, and attempting to access a non-existent key results in an error.

Here's what happens if you try to access spam['foo']:

```
spam = {'bar': 100}

value = spam['foo']  # KeyError: 'foo'
```

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

'cat' in spam:

This expression checks if the key 'cat' exists directly in the dictionary spam.

It evaluates to True if the key 'cat' exists in spam, regardless of its associated value.

Example:

```
spam = {'cat': 10, 'dog': 20}

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

'cat' in spam.keys():

This expression first retrieves all the keys of the dictionary spam using the keys() method, and then checks if 'cat' exists among those keys.

It also evaluates to True if the key 'cat' exists in spam, but it explicitly checks among the keys of the dictionary.

Example:

```
spam = {'cat': 10, 'dog': 20}

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

In summary, **'cat' in spam** checks for the presence of the key **'cat'** directly within the dictionary **spam**, while **'cat' in spam.keys()** retrieves all keys of **spam** and checks if **'cat'** is among them. Both expressions return **True** if **'cat'** is a key in **spam**, but the latter explicitly involves accessing the dictionary's keys.

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 value 'cat' exists directly as a key in the dictionary spam.

It evaluates to True if there exists a key in spam that equals 'cat'.

Example:

```
spam = {'cat': 10, 'dog': 20}

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

'cat' in spam.values():

This expression retrieves all the values of the dictionary spam using the values() method and checks if 'cat' exists among those values.

It evaluates to True if there exists a value in spam that equals 'cat'.

Example:

```
spam = {'cat': 10, 'dog': 20}

print('cat' in spam.values())  # Output: False (assuming 'cat' is not a value in spam)
```

In summary, 'cat' in spam checks if 'cat' exists as a key in spam, while 'cat' in spam.values() checks if 'cat' exists as a value in spam.

7. What is a shortcut for the following code?

```
if 'color' not in spam:

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

A shortcut for the given code can be achieved using the dict.setdefault() method. This method sets the value of a key in a dictionary if the key does not exist, and returns the value associated with the key. Here's how you can use it:

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

This line of code checks if the key 'color' exists in the dictionary spam. If it does not exist, it sets the value of the key 'color' to 'black'. If the key already exists, it does nothing.

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

"pretty print" dictionary values using the pprint module and its pprint() function (pretty print function). The pprint() function provides a more human-readable and formatted output for dictionaries and other data structures. Here's how you can use it:

```
import pprint

my_dict = {'name': 'John', 'age': 30, 'city': 'New York'}

# Pretty print the dictionary

pprint.pprint(my_dict)
```

This will print the dictionary my_dict in a more structured and readable format. The pprint() function automatically formats the output with indentation and line breaks, making it easier to read complex data structures like dictionaries.