1. What does an empty dictionarys code look like?

In [ ]:

```
Ans:-
    In Python, an empty dictionary is represented using curly braces {}.
```

In [1]:

```
empty_dict = {}
print(empty_dict)
```

```
{}
```

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

```
Ans:-
    In Python, dictionaries store data in key-value pairs. To access the value
associated with the key 'foo' in the dictionary, you would simply refer to it using
the key 'foo'.
    For Example:-
```

In [2]:

```
my_dict = {'foo': 42}
value_of_foo = my_dict['foo']
print(value_of_foo)
```

```
42
```

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

```
Ans:-
    (1) Data structure:

List: A list is an ordered collection of elements. It allows you to store multiple
items in a specific sequence, and each item in the list has an index associated with
it, starting from 0 for the first element, 1 for the second element, and so on. Lists
are denoted by square brackets [_] in Python.
Dictionary: A dictionary is an unordered collection of key-value pairs. Instead of
using numerical indexes like lists, dictionaries use unique keys to access their
corresponding values. Keys can be of any immutable data type, such as strings or
numbers, and values can be of any data type, including other lists or dictionaries.
Dictionaries are denoted by curly braces { } in Python.

    (2)Accessing elements:

List: To access elements in a list, you use their numerical index. For example, to
access the second element of a list named my_list, you use my_list[1].
```

```
Dictionary: To access elements in a dictionary, you use their corresponding keys. For
example, to access the value associated with the key "age" in a dictionary named
my_dict, you use my_dict["age"].
    (3)Ordering:

List: Lists maintain the order of elements based on their position in the list. The
first element in the list will always be retrieved with index 0, the second with index
1, and so on.
Dictionary: Dictionaries do not maintain any specific order for their elements. The
order in which key-value pairs are added to the dictionary may not be the same as the
order in which they are accessed. In Python 3.7 and later, dictionaries do preserve
insertion order, but it is not something you can rely on in all versions or for all
use cases.
    (4)Mutability:

List: Lists are mutable, meaning you can change their elements, add new elements, or
remove existing elements.
Dictionary: Dictionaries are also mutable, allowing you to modify the values
associated with existing keys, add new key-value pairs, or delete existing key-value
pairs.
```

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

```
Ans:-
    If We try to access spam['foo'] and spam is { 'bar': 100 }, you will get a
KeyError. The KeyError is raised because the key 'foo' does not exist in the spam
dictionary.
```

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

```
Ans-
'cat' in spam: This expression checks whether the key 'cat' exists in the dictionary
spam. It returns True if the key 'cat' is present as a key in the dictionary, and
False otherwise. This is a membership test for dictionary keys.

'cat' in spam.keys(): This expression checks whether the string 'cat' exists in the
list of keys of the dictionary spam. The method keys() returns a view object that
shows a list of all the keys in the dictionary. Similar to the previous case, it
returns True if the key 'cat' is present in the list of keys, and False otherwise.
    For Example:-
```

In [3]:

```
# Example dictionary
spam = {'cat': 1, 'dog': 2, 'bird': 3}

# Checking membership of 'cat' in the dictionary
print('cat' in spam)         # True

# Checking membership of 'cat' in the keys of the dictionary
print('cat' in spam.keys())  # True

# Checking membership of 'elephant' in the dictionary
print('elephant' in spam)    # False

# Checking membership of 'elephant' in the keys of the dictionary
print('elephant' in spam.keys())  # False
```

```
True
True
False
False
```

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

```
Ans:-
    cat' in spam: This expression checks whether the key 'cat' exists in the
dictionary spam. It checks for the presence of the key in the dictionary's keys, not
its values. If the key 'cat' is present in spam, it will return True, otherwise False.

'cat' in spam.values(): This expression checks whether the value 'cat' exists in the
dictionary spam. It checks for the presence of the value in the dictionary's values,
not its keys. If the value 'cat' is found in any of the values of the spam dictionary,
it will return True, otherwise False.
For Example:-
```

In [4]:

```
spam = {'name': 'Alice', 'age': 25, 'pet': 'cat'}
print('cat' in spam)
```

```
False
```

7. What is a shortcut for the following code?

if 'color' not in spam:

spam['color'] = 'black'

```
Ans:-
```

```
    The shortcut for the given code is to use the Python dictionary method
setdefault().The setdefault() method checks if the key 'color' exists in the spam
dictionary. If it does, it returns the value associated with the key. If it doesn't,
it adds the key 'color' with the value 'black' to the dictionary. This way, you
achieve the same result as the original code but in a more concise manner.
    For Example:-
    spam.setdefault('color', 'black')
```

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

```
Ans:-
    To "pretty print" dictionary values in Python, you can use the pprint module,
which stands for "pretty print." The pprint module provides a function called pprint()
that helps to display dictionary values in a more readable and formatted manner. It's
particularly useful when dealing with nested and complex data structures like
dictionaries.
    For Example:-
```

In [8]:

```python
import pprint

my_dict = {'name': 'John Doe','age': 30,'email': 'john.doe@example.com','address':
           {'street': '123 Main Street','city': 'Anytown','zipcode': '12345'}}

pprint.pprint(my_dict)
```

```
{'address': {'city': 'Anytown',
             'street': '123 Main Street',
             'zipcode': '12345'},
 'age': 30,
 'email': 'john.doe@example.com',
 'name': 'John Doe'}
```

In [ ]: