1)

In [5]:
```
# We can create empty dictionary in the following ways.

dict1 = dict() # Method 1

dict2 = {} # Method 2

print('1.',dict1)

print('2.',dict2)
```
```
1. {}
2. {}
```

2)

In [12]:
```
dict = {'foo':42}
print('Key:',dict.keys())
print('Values:',dict.values())
```
```
Key: dict_keys(['foo'])
Values: dict_values([42])
```

3)

In [ ]:
```
'''
Data Organization:

List: A list is an ordered collection of elements, and each element is identified by its index.
The index starts from 0 for the first element, 1 for the second element, and so on.
Lists maintain the order of elements based on their insertion, and you access elements using their numeric index.

Dictionary: A dictionary is an unordered collection of key-value pairs, where each element (item)
is identified by a unique key.
Keys in a dictionary must be immutable objects (strings, numbers, or tuples),
and they are used to access their associated values.
Dictionaries do not guarantee any specific order for their items.

Element Access:

List: To access elements in a list, you use numeric indexing.
For example, my_list[0] would access the first element of the list.

Dictionary: To access elements in a dictionary, you use their corresponding keys.
For example, my_dict['name'] would access the value associated with the key 'name'.

'''
```

4)

In [14]:
```
spam = {'bar':100}
spam['foo']

# It shows a KeyError, since "foo" is not present among keys
```
```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[14], line 2
      1 spam = {'bar':100}
----> 2 spam['foo']

KeyError: 'foo'
```

5)

In [ ]:
```
'''
'cat' in spam and 'cat' in spam.keys(): Both expressions check for the presence of the key 'cat'
in the dictionary spam.
They will return True if the key 'cat' exists in the dictionary as a key, and False otherwise.

'''
```

In [25]:
```
spam = {'cat': 1, 'dog': 2}

print('cat' in spam)
print('cat' in spam.keys())

# Here 'cat' is present in key, so the function returned TRUE.
```
```
True
True
```

In [26]:
```
spam = {'key1': 'cat', 'dog': 2}

print('cat' in spam)
print('cat' in spam.keys())

# Here 'cat' is present in values, so the function returned FALSE.
```
```
False
False
```

6)

In [ ]:
```
'''
'cat' in spam: checks for the presence of the key 'cat'.
They will return True if the key 'cat' exists in the dictionary as a key, and False otherwise.

'cat' in spam.values(): checks for the presence of the value 'cat'.
They will return True if the value 'cat' exists in the dictionary as a value, and False otherwise.

'''
```

In [28]:
```
spam = {'cat': 1, 'dog': 2}

print('cat' in spam)
print('cat' in spam.values())
```
```
True
False
```

In [29]:
```
spam = {'key1': 'cat', 'dog': 2}
print('cat' in spam.values())
```
```
True
```

7)

In [37]:
```
if 'color' not in spam:
    spam['color'] = 'black'

# Shortcut
spam.setdefault('color', 'black')
spam
```
Out[37]:
```
{'key1': 'cat', 'dog': 2, 'color': 'black'}
```

8)

In [ ]:
```
'''
To "pretty print" dictionary values in Python, one can use the pprint module (Pretty Print) and its pprint() function.
The pprint module provides a way to display data structures,
such as dictionaries and lists, in a more human-readable and formatted manner.
It is especially useful when dealing with complex nested data structures

'''

import pprint
pprint.pprint(spam)
```

END