```python
"""
Dictionaries are collection of key-value pairs.
When there is association between two values, we use dictionaries.

Characteristics of Dictionaries:
1. Dictionaries are mutable.
2. Dictionaries are ordered.
3. Dictionary keys should be unique.
4. Dictionary keys should be immutable.
"""

# Creation of Dictionary:
# Empty Dictionary
a = {}
print(type(a))
a = dict()
print(type(a))

# Dictionary with elements
# All the key-value pairs in a dictionary are separated by comma.
# Key and value must be separated by a colon(:)
# Keys should be unique
# Keys should be immutable.
# Syntax:
# <dict_variable_name> = {<key1>: <value1>, <key2>: <value2>,....}
a = {1: "One", 2: "Two", 3: "Three", 4: "Four", 5: "Five"}
print(type(a))
print(a)

# If the same key exists,
# the value will be overridden with the latest one.
a = {1: "One", 2: "Two", 3: "Three", 1: "New One"}
print(a)

# a = {[1, 2, 3]: "One, Two and Three", [4, 5, 6]: "Four, Five and Six"}
# print(a)


# Accessing the values from a Dictionary
# <dict_variable_name>[<key>]
a = {"language": "Python", "framework": "Django", "Role": "Full Stack"}
print(a["language"])
print(a["framework"])
print(a["Role"])

# using get method
# print(a["Technology"])
print(a.get("language"))
print(a.get("framework"))
print(a.get("Role"))
print(a.get("Technology"))
print(a.get("Technology", "Key Doesn't exist"))

a = {"key1": "value1", "key2": "value1", "key3": "value1"}
print(a)
a = {"key1": [1, 2, 3], "key2": [4, 5, 6], "key3": [7, 8, 9]}
print(a)
```

```python
# Adding Values to a Dictionary
# Syntax:
# <dict_variable_name>[<new_key>] = <new_value>
a = {"language": "Python", "framework": "Django", "Role": "Full Stack"}
a["IDE"] = "PyCharm"
print(a)

# Updating Values in a Dictionary
# Syntax:
# <dict_variable_name>[<existing_key>] = <new_value>
a["language"] = "Java"
a["framework"] = "SpringBoot"
print(a)


# Removing Key value pair from a Dictionary
a = {"language": "Python", "framework": "Django", "Role": "Full Stack"}
removed_value = a.pop("Role")
print(a)
print("Removed Value = ", removed_value)
removed_key_value_pair = a.popitem()
print(a)
print("Removed Key Value Pair = ", removed_key_value_pair)
del a["language"]
print(a)
a = {"language": "Python", "framework": "Django", "Role": "Full Stack"}
a.clear()
print(a)

"""
Dictionary Methods:
1. get() - retrieves the value asso
2. keys()
3. values()
4. items()
5. pop()
6. popitem()
7. clear()
8. update()
9. fromkeys()
10. setdefault()
11. copy()
"""
a = {"language": "Python", "framework": "Django", "Role": "Full Stack"}
print(a.keys())
print(a.values())
print(a.items())

print(list(a.keys()))
print(list(a.values()))
print(list(a.items()))

a = {1: "One", 2: "Two", 3: "Three", 4: "Four"}
b = {5: "Five", 6: "Six", 7: "Seven", 8: "Eight"}
a.update(b)
print(a)

a = {1: "One", 2: "Two", 3: "Three", 4: "Four"}
b = {1: "Updated One", 4: "Updated Four", 5: "Five", 6: "Six", 7: "Seven", 8:
```

```
"Eight"}
a.update(b)
print(a)


x = [1, 2, 3]
y = "Number"
a = dict.fromkeys(x, y)
print(a)

a = {1: "One", 2: "Two", 3: "Three"}
y = "Number"
a = a.fromkeys(x, y)
print(a)

a = {"1": "One", 2: "Two", 3: "Three"}
print(a.get(1, "No Key Found")) #
print(a.setdefault(2, "Updated Two"))
print(a)
print(a.setdefault(4, "Four"))
print(a)

a = {1: 1, 2: 2, 3: 3}
b = a
b[4] = 4
print("b = ", b) #
print("a = ", a) #
print(id(a))
print(id(b))
print(a is b)
print(a == b)

a = {1: 1, 2: 2, 3: 3}
b = a.copy()
print("-----------")
print(id(a))
print(id(b))
print(a is b)
print(a == b)
print("----------")

b[4] = 4
print("b = ", b) #
print("a = ", a) #
print(id(a))
print(id(b))
print(a is b)
print(a == b)


# Iterating through a dictionary:
a = {1: "One", 2: "Two", 3: "Three", 4: "Four"}
for key in a.keys():
    print(f"Key = {key} and Value = {a[key]}")
print()

for key, value in a.items():
    print(f"Key = {key} and Value = {value}")
print()
```

```
# Dict Comprehension
# An easy and elegant way to create Dictionaries
# Syntax:
# <dict_variable_name> = {<key>: <value> for <loop_variable_name> in
range([start,]stop[,step]) / <expression>}
# Example
squares = {i: i ** 2 for i in range(11)}
print(squares)

a = [1, 2, 3, 4, 5]
b = {item: str(item) for item in a}
print(b)


"""
Functions used with all the datatypes
1. len()
2. reversed()
3. sorted()
4. max()
5. min()
6. sum()
"""
a = [1, 2, 3, 4, 5]
print(len(a))
a = (1, 2, 3, 4, 5)
print(len(a))
a = {1, 2, 3, 4, 5}
print(len(a))
a = {1: "One", 2: "Two", 3: "Three", 4: "Four"}
print(len(a))
a = "String"
print(len(a))

a = [1, 2, 3, 4, 5]
print(max(a))
a = (1, 2, 3, 4, 5)
print(max(a))
a = {1, 2, 3, 4, 5}
print(max(a))
a = {1: "One", 2: "Two", 3: "Three", 4: "Four"}
print(max(a))
a = "String" # Ascii - A - 65, a - 97
print(max(a))

print("Min............")
a = [1, 2, 3, 4, 5]
print(min(a))
a = (1, 2, 3, 4, 5)
print(min(a))
a = {1, 2, 3, 4, 5}
print(min(a))
a = {1: "One", 2: "Two", 3: "Three", 4: "Four"}
print(min(a))
a = "String" # Ascii - A - 65, a - 97
print(min(a))
print("Sum............")
a = [1, 2, 3, 4, 5]
print(sum(a))
a = (1, 2, 3, 4, 5)
print(sum(a))
a = {1, 2, 3, 4, 5}
print(sum(a))
a = {1: "One", 2: "Two", 3: "Three", 4: "Four"}
print(sum(a))


print("Sorted............")
a = [5, 3, 1, 4, 2]
print(sorted(a))
a = (5, 3, 1, 4, 2)
print(sorted(a))
a = {5, 3, 1, 4, 2}
print(sorted(a))
a = {3: "Three", 1: "One", 2: "Two", 4: "Four"}
print(sorted(a))
a = "String" # Ascii - A - 65, a - 97
print(sorted(a))

print("Reversed............")
a = [5, 3, 1, 4, 2]
print(list(reversed(a)))
a = (5, 3, 1, 4, 2)
print(tuple(reversed(a)))
a = {5, 3, 1, 4, 2}
# print(reversed(a))
a = {3: "Three", 1: "One", 2: "Two", 4: "Four"}
print(list(reversed(a)))
a = "String" # Ascii - A - 65, a - 97
print(list(reversed(a)))
```