Dictionary

Dictionaries in Python are unordered collections of elements where each item consists of a key-value pair.

Lets see different ways to iterater over dictionary in upcoming examples.

Iterating over Keys

In [ ]:
```python
# Iterating over Keys
my_dict = {'name': 'John', 'age': 25, 'city': 'New York'}

for key in my_dict:
    print(key)
```

```
name
age
city
```

Iterating over Values

In [ ]:
```python
#  Iterating over Values
for value in my_dict.values():
    print(value)
```

```
John
25
New York
```

Iterating over Key-Value Pairs

In [ ]:
```python
#  Iterating over Key-Value Pairs
for key, value in my_dict.items():
    print(f'{key}: {value}')
```

```
name: John
age: 25
city: New York
```

Iteration using list compresion

In [ ]:
```python
# Iterating over Keys
keys = [key for key in my_dict]
keys
```

Out[ ]:
```
['name', 'age', 'city']
```

In [ ]:
```
# Iterating over Values
values = [value for value in my_dict.values()]
values
```

Out[ ]:
```
['John', 25, 'New York']
```

In [ ]:
```
# Iterating over Key-Value Pairs
key_value_pairs = [(key, value) for key, value in my_dict.items()]
key_value_pairs
```

Out[ ]:
```
[('name', 'John'), ('age', 25), ('city', 'New York')]
```

Dictionary operations

1. Creating Dictionary
2. Accesing dictionary elements
3. Modify dictionary elements
4. Adding new element: key-value pair
5. Deleting dictionary element
6. Membership checking
7. Getting keys and values
8. Copying dictionary
9. Removes all items from dictionary
10. Creating nested dictionary

1. Creating Dictionary

In [ ]:
```
my_dict = {'name': 'John', 'age': 25, 'city': 'New York'}
```

2. Accesing dictionary elements

In [ ]:
```
# Accessing value by key
print(my_dict['name'])  # Output: John
```

```
John
```

3. Modify dictionary elements

In [ ]:
```
# Updating the value of a key
my_dict['age'] = 26
print(my_dict)  # Output: {'name': 'John', 'age': 26, 'city': 'New York'}
```

```
{'name': 'John', 'age': 26, 'city': 'New York'}
```

4. Adding new element: key-value pair

In [ ]:
```
# Adding a new key-value pair
my_dict['occupation'] = 'Engineer'
print(my_dict)
```

```
{'name': 'John', 'age': 26, 'city': 'New York', 'occupation': 'Engineer'}
```

5. Deleting dictionary element

In [ ]:
```
# Deletion using 'del'
del my_dict['city']  # Removes the key-value pair with the key 'city'
print(my_dict)
```

```
{'name': 'John', 'age': 26, 'occupation': 'Engineer'}
```

In [ ]:
```
# Deletion using 'pop()'
age = my_dict.pop('age')  # Removes the key-value pair and returns the value
print(my_dict)
print(age)
```

```
{'name': 'John', 'occupation': 'Engineer'}
26
```

6. Membership checking

In [ ]:
```
print('name' in my_dict)  # Output: True
```

```
True
```

7. Getting keys and values

In [ ]:
```
# Getting all keys
keys = my_dict.keys()
print(keys)

# Getting all values
values = my_dict.values()
print(values)

# Getting key-value pairs as tuples
items = my_dict.items()
print(items)
```

```
dict_keys(['name', 'occupation'])
dict_values(['John', 'Engineer'])
dict_items([('name', 'John'), ('occupation', 'Engineer')])
```

8. Copying dictionary

In [ ]:
```
# Shallow copy ==> A shallow copy creates a new object, but does not create new obj
#                  Instead, it copies references to the objects. Changes made to th
new_dict = my_dict.copy()
print(new_dict)
```

```
{'name': 'John', 'age': 25, 'city': 'New York'}
```

In [ ]:
```
new_dict['name'] = 'Nikhil'
```

In [ ]:
```
my_dict
```

Out[ ]:
```
{'name': 'John', 'age': 25, 'city': 'New York'}
```

In [ ]:
```
new_dict
```

Out[ ]:
```
{'name': 'Nikhil', 'age': 25, 'city': 'New York'}
```

In [ ]:
```
# Deep copy (requires importing the copy module) ==> A deep copy creates a new obje
#                                                     Changes made to the nested ele
import copy
deep_copy = copy.deepcopy(my_dict)
print(deep_copy)
```

```
{'name': 'John', 'age': 25, 'city': 'New York'}
```

In [ ]:
```
deep_copy['age'] = 35
```

In [ ]:
```
my_dict
```

Out[ ]:
```
{'name': 'John', 'age': 25, 'city': 'New York'}
```

In [ ]:
```
deep_copy
```

Out[ ]:
```
{'name': 'John', 'age': 35, 'city': 'New York'}
```

9. Removes all items from dictionary

In [ ]:
```
my_dict.clear()  # Removes all items from the dictionary
print(my_dict)
```

```
{}
```

10. Creating nested dictionary

In [ ]:
```
employee = {
    'name': 'John',
    'age': 30,
```

```
    'address': {
        'street': '123 Main St',
        'city': 'Cityville',
        'zip': '12345'
    }
}
```