Dictionary

Kittikun Jitpairod

Introduction to Dictionaries

A dictionary in Python is an unordered collection of key-value pairs. It is also referred to as an associative array, hash table, or hash map in other programming languages. Dictionaries are defined by curly braces { } and contain key:value pairs.

The key features of dictionaries are:

- They are mutable (can be modified after creation)
- Keys must be unique and immutable (strings, numbers, or tuples)
- Values can be of any type
- They are unordered (in Python versions < 3.7)

Creating Dictionaries

There are multiple methods to create dictionaries in Python. Consider the following examples:

- Empty dictionary

```
empty_dict = {}
print("Empty dictionary:", empty_dict)
```

```
Empty dictionary: {}
```

- Dictionary with initial key-value pairs

```
person = {"name": "Alice", "age": 30, "city": "New York"}
print("Person dictionary:", person)
```

```
Person dictionary: {'name': 'Alice', 'age': 30, 'city': 'New York'}
```

- Using `dict()` constructor

```
another_person = dict(name="Bob", age=25, city="San Francisco")
print("Another person dictionary:", another_person)
```

```
Another person dictionary: {'name': 'Bob', 'age': 25, 'city': 'San Francisco'}
```

- Creating a dictionary from two lists using `zip()`

```
keys = ["a", "b", "c"]
values = [1, 2, 3]
combined = dict(zip(keys, values))
print("Combined dictionary:", combined)
```

```
Combined dictionary: {'a': 1, 'b': 2, 'c': 3}
```

Accessing Dictionary Elements

The contents of dictionaries can be access using `[key]` or `get()` method.

```
person = {"name": "Alice", "age": 30, "city": "New York"}

# Using square bracket notation
print("Name:", person["name"])

# Using get() method
print("Age:", person.get("age"))

# Using get() with a default value
print("Country:", person.get("country", "Unknown"))
```

```
Name: Alice
Age: 30
Country: Unknown
```

The get() method is particularly useful as it allows for the specification of a default value if the key does not exist.

Modifying Dictionary Elements

These operations illustrate the dynamic nature of dictionaries.

```
person = {"name": "Alice", "age": 30, "city": "New York"}

# Changing a value
person["age"] = 31
print("Updated age:", person["age"])

# Adding a new key-value pair
person["job"] = "Engineer"
print("Updated dictionary:", person)

# Updating multiple key-value pairs
person.update({"age": 32, "country": "USA"})
print("Dictionary after update:", person)
```

```
Updated age: 31
Updated dictionary: {'name': 'Alice', 'age': 31, 'city': 'New York', 'job': 'Engineer'}
Dictionary after update: {'name': 'Alice', 'age': 32, 'city': 'New York', 'job': 'Engineer', 'cou
```

These operations illustrate the dynamic nature of dictionaries.

```
# Removing a key-value pair
del person["city"]
print("Dictionary after deletion:", person)

# Removing and returning a value
job = person.pop("job")
print("Removed job:", job)
print("Dictionary after pop:", person)
```

```
Dictionary after deletion: {'name': 'Alice', 'age': 32, 'job': 'Engineer', 'country': 'USA'}
Removed job: Engineer
Dictionary after pop: {'name': 'Alice', 'age': 32, 'country': 'USA'}
```

Dictionary Methods

Python provides several useful methods for working with dictionaries:

```
person = {"name": "Alice", "age": 30, "city": "New York"}

# Get all keys
print("Keys:", person.keys())

# Get all values
```

```python
print("Values:", person.values())

# Get all key-value pairs
print("Items:", person.items())

# Clear the dictionary
print("After clear:", person.clear())
```

```
Keys: dict_keys(['name', 'age', 'city'])
Values: dict_values(['Alice', 30, 'New York'])
Items: dict_items([('name', 'Alice'), ('age', 30), ('city', 'New York')])
After clear: None
```

To copy a dictionary:

```python
# Create a new reference
original = {"x": 1, "y": 2}
reference = original
reference["z"] = 3
print("Original:", original)
print("Reference:", reference)
```

```
Original: {'x': 1, 'y': 2, 'z': 3}
Reference: {'x': 1, 'y': 2, 'z': 3}
```

```python
# Create a copy
original = {"x": 1, "y": 2}
copy = original.copy()
copy["z"] = 3
print("Original:", original)
print("Copy:", copy)
```

```
Original: {'x': 1, 'y': 2}
Copy: {'x': 1, 'y': 2, 'z': 3}
```