Dictionaries in Python

Dictionaries are unordered collections of key-value pairs. They are mutable and indexed by keys, which must be unique and immutable.

Why Use Dictionaries?

- **Key-Value Mapping**: Dictionaries provide a way to map unique keys to values.
- **Efficient Lookup**: Dictionaries offer fast lookups, insertions, and deletions.
- **Flexible Data Structure**: Dictionaries can store heterogeneous data types and nested structures.

Creating Dictionaries

You can create a dictionary by enclosing key-value pairs in curly braces `{}` or using the `dict()` function.

```
# Empty dictionary
empty_dict = {}

# Dictionary with integer keys
int_key_dict = {1: "apple", 2: "banana"}

# Dictionary with string keys
str_key_dict = {"name": "Alice", "age": 25}

# Mixed type dictionary
mixed_dict = {1: "apple", "name": "Alice", 3.14: "pi"}

print(empty_dict)
print(int_key_dict)
print(str_key_dict)
print(mixed_dict)
```

Accessing and Modifying Dictionary Elements

You can access and modify elements in a dictionary using keys.

```
person = {"name": "Alice", "age": 25}

# Accessing elements
print(person["name"])  # Output: Alice
print(person.get("age"))  # Output: 25

# Modifying elements
```

```
person["age"] = 26
print(person)  # Output: {'name': 'Alice', 'age': 26}

# Adding new key-value pairs
person["city"] = "New York"
print(person)  # Output: {'name': 'Alice', 'age': 26, 'city': 'New York'}
```

Removing Elements

You can remove elements from a dictionary using the `del` statement, `pop()`, and `popitem()` methods.

```
person = {"name": "Alice", "age": 25, "city": "New York"}

# Remove a specific element
del person["age"]
print(person)  # Output: {'name': 'Alice', 'city': 'New York'}

# Remove and return an element
city = person.pop("city")
print(city)  # Output: New York
print(person)  # Output: {'name': 'Alice'}

# Remove and return an arbitrary element
item = person.popitem()
print(item)  # Output: ('name', 'Alice')
print(person)  # Output: {}
```

Common Dictionary Methods

keys()

Returns a view object containing the dictionary's keys.

```
person = {"name": "Alice", "age": 25}
print(person.keys())  # Output: dict_keys(['name', 'age'])
```

values()

Returns a view object containing the dictionary's values.

```
person = {"name": "Alice", "age": 25}
print(person.values())  # Output: dict_values(['Alice', 25])
```

items()

Returns a view object containing the dictionary's key-value pairs.

```
person = {"name": "Alice", "age": 25}
print(person.items())  # Output: dict_items([('name', 'Alice'), ('age', 25)])
```

update()

Updates the dictionary with elements from another dictionary or iterable of key-value pairs.

```
person = {"name": "Alice", "age": 25}
updates = {"age": 26, "city": "New York"}
person.update(updates)
print(person)  # Output: {'name': 'Alice', 'age': 26, 'city': 'New York'}
```

clear()

Removes all elements from the dictionary.

```
person = {"name": "Alice", "age": 25}
person.clear()
print(person)  # Output: {}
```

Practical Applications

Counting Frequency of Elements

Dictionaries are useful for counting the frequency of elements in a collection.

```
sentence = "apple banana apple strawberry banana apple"
words = sentence.split()
word_count = {}

for word in words:
    if word in word_count:
        word_count[word] += 1
    else:
        word_count[word] = 1

print(word_count)  # Output: {'apple': 3, 'banana': 2, 'strawberry': 1}
```

Storing Nested Data

Dictionaries can store nested data structures.

```
student = {
    "name": "Alice",
    "age": 25,
    "courses": {
        "math": 90,
        "science": 95
    }
}

print(student)  # Output: {'name': 'Alice', 'age': 25, 'courses': {'math': 90, 'science': 9
```

Conclusion

Dictionaries are a powerful data structure for storing and managing key-value pairs. They provide efficient lookup, insertion, and deletion operations and are versatile for various applications, such as counting frequencies and storing nested data.

Stay Updated

Be sure to this repository to stay updated with new examples and enhancements!

License

Contact

Panagiotis Moschos - pan.moschos86@gmail.com

Note: This is a Python script and requires a Python interpreter to run.

Happy Coding

Made with by Panagiotis Moschos (https://github.com/pmoschos)