Module 2: Understanding Dictionaries

Introduction

Dictionaries in Python are versatile data structures used to store collections of items. Unlike sequences such as lists or tuples, dictionaries are unordered collections of key-value pairs, offering fast lookup and retrieval operations. This document provides an overview of dictionary basics, common operations, and examples to illustrate their usage.

Basics

Definition: A dictionary in Python is defined using curly braces `{}`. Each item in the dictionary consists of a key-value pair separated by a colon `:`.

```
my_dict = {'key1': 'value1', 'key2': 'value2'}
```

Accessing Values: Values in a dictionary are accessed by specifying their corresponding keys in square brackets `[]`.

```
print(my_dict['key1']) # Output: value1
```

Dictionary Keys: Keys in a dictionary must be unique and immutable types such as strings, numbers, or tuples. Values can be of any data type.

Dictionary Operations

Adding or Updating Items: Use square brackets `[]` with the key to add or update items in a dictionary.

```
my_dict['new_key'] = 'new_value' # Adding a new item
my_dict['key1'] = 'updated_value' # Updating an existing item
```

Removing Items: Use the `del` keyword or the `pop()` method to remove items from a dictionary.

```
del my_dict['key1'] # Remove a specific item
value = my_dict.pop('key2') # Remove and return the value of a specific item
```

Checking for Key Existence: Use the `in` keyword to check if a key exists in a dictionary.

```
if 'key1' in my_dict:
 print("Key exists!")
```

Example:

Let's consider a dictionary representing information about a person:

```
person = {
 'name': 'John Doe',
 'age': 30,
 'city': 'New York'
}
```

Operations

Accessing values:

```
print(person['name']) # Output: John Doe
```

Adding or updating items:

```
person['occupation'] = 'Engineer' # Adding a new item
person['age'] = 35 # Updating an existing item
```

Removing items:

```
del person['city'] # Removing a specific item
occupation = person.pop('occupation') # Removing and returning the value of a
specific item
```

Checking for key existence:

```
if 'age' in person:
 print("Age exists!")
```

Conclusion

Dictionaries are powerful and flexible data structures in Python, offering efficient storage and retrieval of key-value pairs. Understanding their basics and common operations is essential for effective data manipulation and algorithm design in Python.

Exercises and Answers for Dictionaries

Exercise 1:
Create a dictionary named `student` representing a student's information. Include keys for `"name"`, `"age"`, `"grade"`, and `"subjects"`. The value of `"subjects"` should be a list of subjects the student is taking.

Answer 1:

```
student = {
 "name": "Wun",
 "age": 18,
 "grade": 12,
 "subjects": ["Math", "Science", "English"]
}
```

Exercise 2:
Add a new subject `"History"` to the list of subjects in the `student` dictionary.

Answer 2:

```
student["subjects"].append("History")
```

Exercise 3:
Remove the `"grade"` key-value pair from the `student` dictionary.

Answer 3:

```
del student["grade"]
```

Exercise 4:
Check if the `"age"` key exists in the `student` dictionary.

Answer 4:

```
if "age" in student:
 print("Age exists!")
```

Exercise 5:
Update the value of the `"name"` key in the `student` dictionary to `"Bob"`.

Answer 5:

```
student["name"] = "Du"
```