Chapter 5 – Dictionaries and Sets

Dictionaries

A dictionary in Python is a collection of key-value pairs. Each key is unique and maps to a value.

Syntax:

```python
a = {
    "key": "value",
    "shivam": "code",
    "marks": 100,
    "list": [1, 2, 9]
}

print(a["key"])  # Output: "value"
print(a["list"])  # Output: [1, 2, 9]
```

Properties of Python Dictionaries:

1. **Unordered**: The items are not stored in a specific order.
2. **Mutable**: We can change the value of existing keys.
3. **Indexed**: We can access items using their keys.
4. **Unique Keys**: Keys must be unique; duplicate keys are not allowed.

Dictionary Methods:

Consider the following dictionary:

```python
a = {
    "name": "Shivam",
    "from": "India",
    "marks": [92, 98, 96]
}
```

`a.items()`: Returns a list of `(key, value)` tuples.

```
print(a.items())  # Output: [('name', 'Shivam'), ('from', 'India'), ('marks', [92, 98
```

`a.keys()`: Returns a list containing the dictionary's keys.

```
print(a.keys())  # Output: ['name', 'from', 'marks']
```

`a.update({"friends": "Yes"})`: Updates the dictionary with supplied key-value pairs.

```
a.update({"friends": "Yes"})
print(a)  # Output: {'name': 'Shivam', 'from': 'India', 'marks': [92, 98, 96], 'frienc
```

`a.get("name")`: Returns the value of the specified key.

```
print(a.get("name"))  # Output: "Shivam"
```

Sets

A set in Python is a collection of non-repetitive elements.

Syntax:

```
s = set()  # Creates an empty set
s.add(1)
s.add(2)  # Now s = {1, 2}
```

If you are new to programming and don't know much about mathematical operations on sets, you can think of sets in Python as data types that contain unique values.

Properties of Sets:

1. **Unordered**: The elements are not stored in a specific order.
2. **Unindexed**: You cannot access elements by index.
3. **Immutable Elements**: Once an element is added to a set, it cannot be changed.
4. **Unique Values**: Sets cannot contain duplicate values.

Operations on Sets:

Consider the following set:

```
s = {1, 8, 2, 3}
```

`len(s)`: Returns the length of the set.

```
print(len(s))  # Output: 4
```

`s.remove(8)`: Removes the element 8 from the set.

```
s.remove(8)
print(s)  # Output: {1, 2, 3}
```

`s.pop()`: Removes and returns an arbitrary element from the set.

```
removed_element = s.pop()
print(removed_element)  # Output: (any element from the set)
print(s)  # Output: The set without the removed element
```

`s.clear()`: Empties the set.

```
s.clear()
print(s)  # Output: set()
```

`s.union({8, 11})`: Returns a new set with all items from both sets.

```
union_set = s.union({8, 11})
print(union_set)  # Output: {1, 2, 3, 8, 11}
```

`s.intersection({8, 11})`: Returns a set which contains only items present in both sets.

```
intersection_set = s.intersection({8, 11})
print(intersection_set)  # Output: {8}
```