Sets in Python

UNDERSTANDING PYTHON COLLECTIONS

What is a Set?

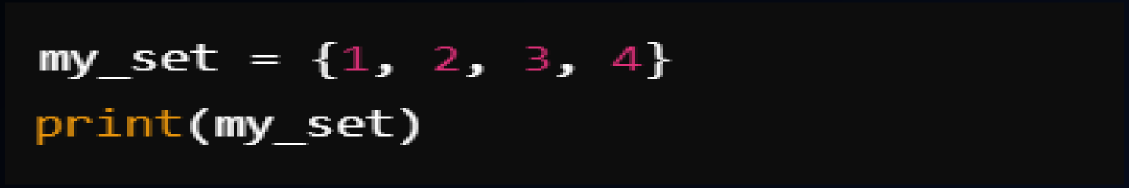
- A set is an unordered collection of unique elements in Python.
- Sets do not allow duplicates and are mutable.
- Syntax :

```
set_name = {element1, element2, element3, ...}
```

Characteristics of Sets

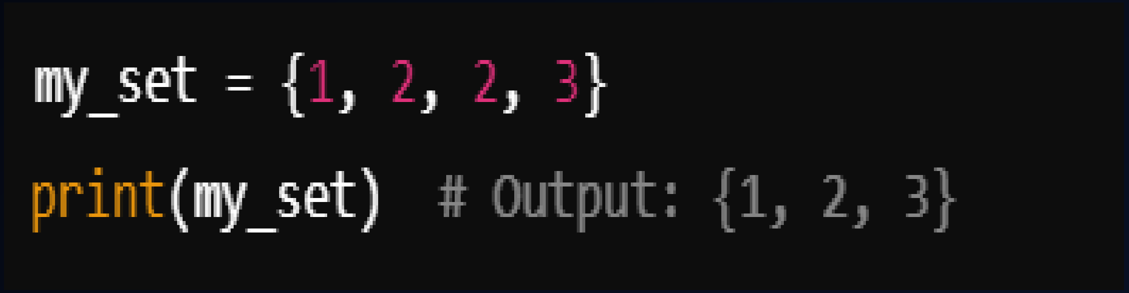
- **Unordered**: No indexing or order to elements in a set.
- **Unique Elements**: No duplicates allowed in sets.
- **Mutable**: Elements can be added or removed, but the set itself is unordered and unique.
- **Heterogeneous**: Can contain different data types.

Creating Sets



```python
my_set = {1, 2, 3, 4}
print(my_set)
```

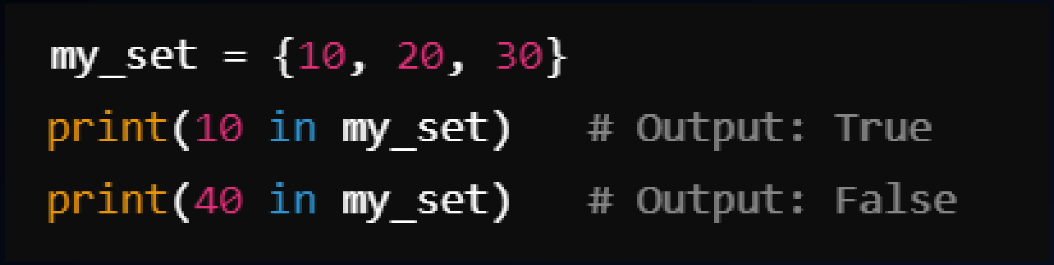
Note: Duplicates are automatically removed.



```python
my_set = {1, 2, 2, 3}
print(my_set)  # Output: {1, 2, 3}
```

Accessing Set Elements

- Since sets are unordered, they don’t support indexing or slicing.

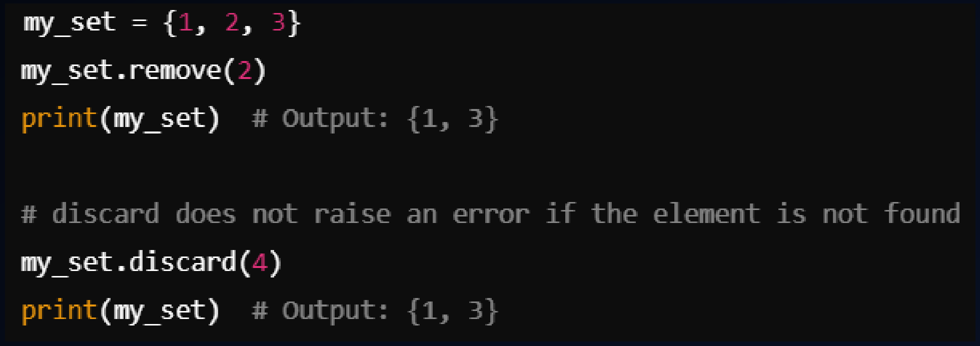


```python
my_set = {10, 20, 30}
print(10 in my_set)   # Output: True
print(40 in my_set)   # Output: False
```

Methods in Python

REMOVING ELEMENTS FROM A SET

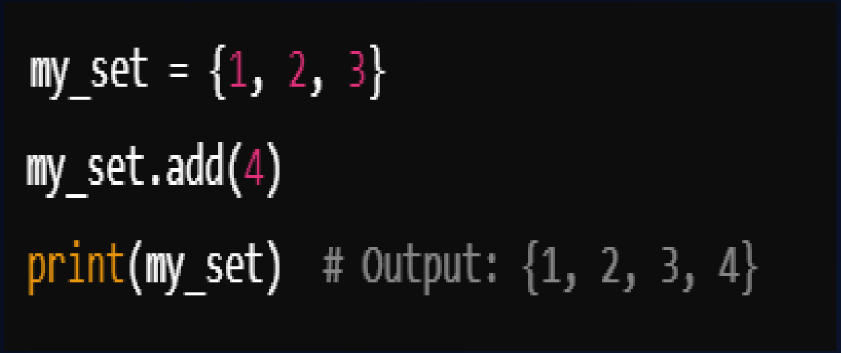
- **Methods**: remove() and discard()



```python
my_set = {1, 2, 3}
my_set.remove(2)
print(my_set)  # Output: {1, 3}

# discard does not raise an error if the element is not found
my_set.discard(4)
print(my_set)  # Output: {1, 3}
```

ADDING ELEMENTS TO A SET

- **Method**: add()



```python
my_set = {1, 2, 3}
my_set.add(4)
print(my_set)  # Output: {1, 2, 3, 4}
```

Set Operations

- **Union**: Combines elements from two sets

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
print(set1 | set2)  # Output: {1, 2, 3, 4, 5}
```

- **Intersection**: Returns only common elements

```
print(set1 & set2)  # Output: {3}
```

- **Difference**: Elements in one set but not in the other

```
print(set1 - set2)  # Output: {1, 2}
```

- **Symmetric Difference**: Elements in either set, but not both

```
print(set1 ^ set2)  # Output: {1, 2, 4, 5}
```

Set Methods

- **add()**: Adds a single element to the set.

```
my_set = {1, 2, 3}
my_set.add(4)
print(my_set)  # Output: {1, 2, 3, 4}
```

- **update():** Adds multiple elements to the set.

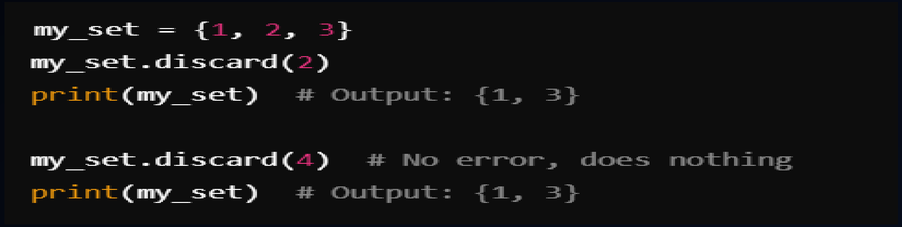
```
my_set = {1, 2, 3}
my_set.update([4, 5])
print(my_set)  # Output: {1, 2, 3, 4, 5}
```

- **remove():** Removes an element, raises an error if it doesn't exist.

```
my_set = {1, 2, 3}
my_set.remove(2)
print(my_set)  # Output: {1, 3}

# If you try to remove an element that doesn't exist
my_set.remove(4)  # Raises KeyError
```

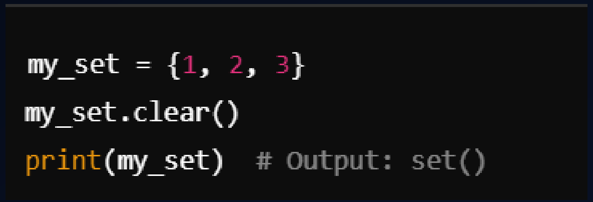
Set Methods

- **discard():** Removes an element, does not raise an error if it doesn't exist.



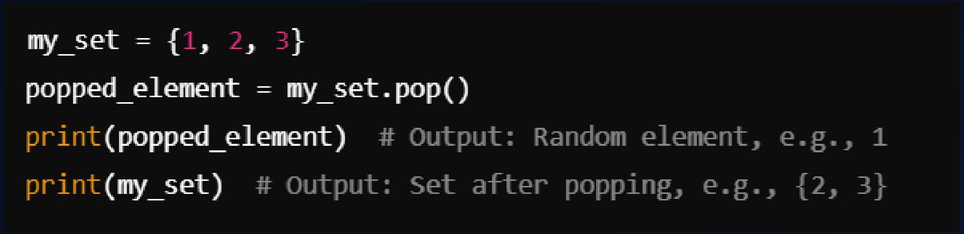
```
my_set = {1, 2, 3}
my_set.discard(2)
print(my_set)   # Output: {1, 3}

my_set.discard(4)   # No error, does nothing
print(my_set)   # Output: {1, 3}
```

- **clear():** Removes all elements from the set.



```
my_set = {1, 2, 3}
my_set.clear()
print(my_set)  # Output: set()
```

- **pop()**: Removes and returns an arbitrary element from the set. Since sets are unordered, you don’t know which element will be popped.



```
my_set = {1, 2, 3}
popped_element = my_set.pop()
print(popped_element)  # Output: Random element, e.g., 1
print(my_set)  # Output: Set after popping, e.g., {2, 3}
```

Why Use Sets?

- Fast membership testing (in and not in operations).
- Automatically removes duplicates.
- Useful for mathematical set operations like union and intersection.

THANK YOU

HAPPY LEARNING!