Sets in Python

Sets are an unordered collection of unique elements. They are mutable and do not allow duplicate values.

Why Use Sets?

- **Uniqueness**: Sets automatically handle duplicates, making them useful for storing unique items.
- **Efficient Membership Testing**: Sets are optimized for checking whether an item is part of the set.
- **Set Operations**: Sets support mathematical set operations like union, intersection, difference, and symmetric difference.

Creating Sets

You can create a set by enclosing items in curly braces `{}` or using the `set()` function.

```
# Empty set
empty_set = set()

# Set of integers
int_set = {1, 2, 3, 4, 5}

# Set of strings
str_set = {"apple", "banana", "cherry"}

# Mixed type set
mixed_set = {1, "hello", 3.14, True}

print(empty_set)
print(int_set)
print(str_set)
print(mixed_set)
```

Adding and Removing Elements

You can add and remove elements from a set using the `add()`, `remove()`, `discard()`, and `pop()` methods.

```
fruits = {"apple", "banana"}

# Add an element
fruits.add("cherry")
print(fruits)  # Output: {'apple', 'banana', 'cherry'}
```

```python
# Remove an element
fruits.remove("banana")
print(fruits)  # Output: {'apple', 'cherry'}

# Discard an element (does not raise an error if element is not found)
fruits.discard("banana")
print(fruits)  # Output: {'apple', 'cherry'}

# Pop an element (removes and returns an arbitrary element)
popped_fruit = fruits.pop()
print(popped_fruit)  # Output: apple or cherry (depends on internal order)
print(fruits)  # Output: {'cherry'} or {'apple'}
```

Set Operations

Union

Combines elements from both sets.

```python
set1 = {1, 2, 3}
set2 = {3, 4, 5}
union_set = set1.union(set2)
print(union_set)  # Output: {1, 2, 3, 4, 5}
```

Intersection

Returns elements common to both sets.

```python
set1 = {1, 2, 3}
set2 = {3, 4, 5}
intersection_set = set1.intersection(set2)
print(intersection_set)  # Output: {3}
```

Difference

Returns elements in the first set but not in the second.

```python
set1 = {1, 2, 3}
set2 = {3, 4, 5}
difference_set = set1.difference(set2)
print(difference_set)  # Output: {1, 2}
```

Symmetric Difference

Returns elements in either set, but not in both.

```python
set1 = {1, 2, 3}
set2 = {3, 4, 5}
```

```
symmetric_difference_set = set1.symmetric_difference(set2)
print(symmetric_difference_set)  # Output: {1, 2, 4, 5}
```

Common Set Methods

issubset()

Checks if a set is a subset of another set.

```
set1 = {1, 2}
set2 = {1, 2, 3, 4}
print(set1.issubset(set2))  # Output: True
```

issuperset()

Checks if a set is a superset of another set.

```
set1 = {1, 2, 3, 4}
set2 = {1, 2}
print(set1.issuperset(set2))  # Output: True
```

isdisjoint()

Checks if two sets have no elements in common.

```
set1 = {1, 2, 3}
set2 = {4, 5, 6}
print(set1.isdisjoint(set2))  # Output: True
```

Conclusion

Sets are a powerful data type for storing unique elements and performing mathematical set operations. They are useful for tasks that involve membership testing, removing duplicates, and working with multiple sets.

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)