Sets In Python

In Python, a set is an unordered collection of unique elements. Sets are used to store multiple items, but unlike lists and tuples, they do not allow duplicate values. The elements in a set are enclosed in curly braces '{}' and separated by commas.

Characterstics:

- Unordered
- Mutable
- No Duplicates
- Can't contain mutable data types

Creating Sets

In [ ]:
```
# empty set
s = set()
print(s)
print(type(s))

# 1D and 2D
s1 = {1,2,3}
print(s1)

s2 = {1,2,3,(4,5)}
print(s2)

# Homo and hetro
s3 = {1,'hello',4.5,(1,2,3)}
print(s3)

# using type conversion
s4 = set([1,2,3])
print(s4)

#duplicated not allowed
s5 = {1,1,2,2,3}
print(s5)
```

```
set()
<class 'set'>
{1, 2, 3}
{3, 1, (4, 5), 2}
{1, (1, 2, 3), 'hello', 4.5}
{1, 2, 3}
{1, 2, 3}
```

In [ ]:
```
#sets can't have a mutable items
#s6 ={1,2,3,[3,4]}
#print(s6)
# it thows error
```

In [ ]:
```
# True or false
s1 = {1,2,3}
s2 = {3,2,1}
```

```
print(s1 == s2)
```

```
True
```

Accessing Items

- In Python sets, you cannot access items by indexing or slicing because sets are unordered collections of unique elements, and they do not have a specific order like lists or tuples. Therefore, indexing and slicing operations, which are common with ordered sequences like lists and strings, are not applicable to sets.

Adding Items

In [ ]:
```
#add
s = {1,2,3,4,5}
s.add(5)
print(s)
```

```
{1, 2, 3, 4, 5}
```

In [ ]:
```
#update
s.update([5,6,7])
print(s)
```

```
{1, 2, 3, 4, 5, 6, 7}
```

Deleting Items

Sets Doesn't support item Deletion

In [ ]:
```
# del
s= {1,2,3,4,5}
del(s)
# it complete delete set
```

In [ ]:
```
# Discard
s = {1,2,3,4}
s.discard(3)
print(s)
```

```
{1, 2, 4}
```

In [ ]:
```
#remove
s = {1,2,3,4}
s.remove(4)
print(s)
```

```
{1, 2, 3}
```

In [ ]:
```
# pop
s.pop()
# it select random number from set
```

Out[ ]:
```
2
```

In [ ]:
```
# clear
s.clear()
print(s)
```

```
set()
```

Set opeartion

In [ ]:
```
s1 = {1,2,3,4,5}
s2 = {4,5,6,7,8}

# union
print(s1 | s2)

# intersection
print(s1 & s2)

# Differences
print(s1 - s2)
print(s2 - s1)

# symmetric diffrences
print(s1 ^ s2)

# membership test
print(1 in s1)
print(1 not in s1)

# iteration
for i in s1:
    print(i)
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
{4, 5}
{1, 2, 3}
{8, 6, 7}
{1, 2, 3, 6, 7, 8}
True
False
1
2
3
4
5
```

Set Function

In [ ]:
```
# len/sum/min/max/sorted
s = {3,1,4,5,2,7}

print(len(s))

print(sum(s))

print(min(s))

print(max(s))

print(sorted(s))
```

```
6
22
1
7
[1, 2, 3, 4, 5, 7]
```

In [ ]:
```python
# union/update
s1 = {1,2,3,4,5}
s2 = {4,5,6,7,8}

# s1 | s2
s1.union(s1)

s1.update(s2)
print(s1)
print(s2)
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
{4, 5, 6, 7, 8}
```

In [ ]:
```python
# intersection/intersection_update
s1 = {1,2,3,4,5}
s2 = {4,5,6,7,8}

s1.intersection(s2)

s1.intersection_update(s2)
print(s1)
print(s2)
```

```
{4, 5}
{4, 5, 6, 7, 8}
```

In [ ]:
```python
# difference/difference_update
s1 = {1,2,3,4,5}
s2 = {4,5,6,7,8}

s1.difference(s2)

s1.difference_update(s2)
print(s1)
print(s2)
```

```
{1, 2, 3}
{4, 5, 6, 7, 8}
```

In [ ]:
```python
# symmetric_difference/symmetric_difference_update
s1 = {1,2,3,4,5}
s2 = {4,5,6,7,8}

s1.symmetric_difference(s2)

s1.symmetric_difference_update(s2)
print(s1)
print(s2)
```

```
{1, 2, 3, 6, 7, 8}
{4, 5, 6, 7, 8}
```

In [ ]:
```python
# isdisjoint/issubset/issuperset
s1 = {1,2,3,4}
s2 = {7,8,5,6}

s1.isdisjoint(s2)
```

Out[ ]:
```
True
```

In [ ]:
```
s1 = {1,2,3,4,5}
s2 = {3,4,5}

s1.issuperset(s2)
```

Out[ ]:
```
True
```

In [ ]:
```
# copy
s1 = {1,2,3}
s2 = s1.copy()

print(s1)
print(s2)
```

```
{1, 2, 3}
{1, 2, 3}
```

Frozenset

Frozen set is just an immutable version of a Python set object

In [ ]:
```
# create frozenset
fs1 = frozenset([1,2,3])
fs2 = frozenset([3,4,5])

fs1 | fs2
```

Out[ ]:
```
frozenset({1, 2, 3, 4, 5})
```

Set Comprehension

In [ ]:
```
# examples

{i**2 for i in range(1,11) if i>5}
```

Out[ ]:
```
{36, 49, 64, 81, 100}
```

In [ ]:
```
{i+2 for i in range(1,11) if i>5}
```

Out[ ]:
```
{8, 9, 10, 11, 12}
```