Python Sets

Riya Jacob K
Assistant Professor on Contract
Dept of Computer Applications

Set

- A set is a collection which is unordered and unindexed.
- In Python sets are written with curly brackets.
- # Note: the set list is unordered, meaning: the items will appear in a random order.

Example

<u>Create a Set:</u>

```
thisset = {"apple", "banana", "cherry"}
print(thisset)
```

<u>Output</u>

```
{'cherry', 'banana', 'apple'}
```

Access Items

You cannot access items in a set by referring to an index, since sets are unordered the items has no index.

But you can loop through the set items using a `for` loop, or ask if a specified value is present in a set, by using the `in` keyword.

Example

Loop through the set, and print the values:

```
thisset = {"apple", "banana", "cherry"}

for x in thisset:
  print(x)
```

- <u>Output</u>

  apple
  banana
  cherry

Example

Check if "banana" is present in the set:

```
thisset = {"apple", "banana", "cherry"}

print("banana" in thisset)
```

- <u>Output</u>

  True

Change Items

Once a set is created, you cannot change its items, but you can add new items.

Add Items

To add one item to a set use the `add()` method.

To add more than one item to a set use the `update()` method.

Example

Add an item to a set, using the `add()` method:

```
thisset = {"apple", "banana", "cherry"}

thisset.add("orange")

print(thisset)
```

- <u>Output</u>

  {'orange', 'apple', 'banana', 'cherry'}

Example

Add multiple items to a set, using the `update()` method:

```
thisset = {"apple", "banana", "cherry"}

thisset.update(["orange", "mango", "grapes"])

print(thisset)
```

- Output

  {'apple', 'grapes', 'mango', 'banana', 'cherry', 'orange'}

Get the Length of a Set

To determine how many items a set has, use the `len()` method.

Example

Get the number of items in a set:

```
thisset = {"apple", "banana", "cherry"}

print(len(thisset))
```

- Output

  3

Remove Item

To remove an item in a set, use the `remove()`, or the `discard()` method.

Example

Remove "banana" by using the `remove()` method:

```
thisset = {"apple", "banana", "cherry"}

thisset.remove("banana")

print(thisset)
```

Note: If the item to remove does not exist, `remove()` will raise an error.

- Output

  {'cherry', 'apple'}

Example

Remove "banana" by using the `discard()` method:

```
thisset = {"apple", "banana", "cherry"}

thisset.discard("banana")

print(thisset)
```

Note: If the item to remove does not exist, `discard()` will **NOT** raise an error.

- Output

  {'cherry', 'apple'}

You can also use the `pop()`, method to remove an item, but this method will remove the *last* item. Remember that sets are unordered, so you will not know what item that gets removed.

The return value of the `pop()` method is the removed item.

Note: Sets are *unordered*, so when using the `pop()` method, you will not know which item that gets removed.

Example

Remove the last item by using the `pop()` method:

```
thisset = {"apple", "banana", "cherry"}

x = thisset.pop()

print(x)

print(thisset)
```

- <u>Output</u>
  banana
  {'apple', 'cherry'}

Example

The `clear()` method empties the set:

```
thisset = {"apple", "banana", "cherry"}

thisset.clear()

print(thisset)
```

- <u>Output</u>

  set()

Example

The `del` keyword will delete the set completely:

```
thisset = {"apple", "banana", "cherry"}

del thisset

print(thisset)
```

- **<u>Output</u>**

Traceback (most recent call last):
File "demo_set_del.py", line 5, in <module>
print(thisset) #this will raise an error because the set no longer exists
NameError: name 'thisset' is not defined