Set

A set is a collection which is unordered and unindexed. In Python sets are written with curly brackets.

Example

Create a Set:

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

Note: Sets are unordered, so you cannot be sure in which order the items will appear.

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:

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

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

Example

Check if "banana" is present in the set:

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

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:

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

myset.add("orange")

print(myset)
```

Example

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

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

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

print(myset)
```

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:

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

Remove Item

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

Example

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

```
myset = {"apple", "banana", "cherry"}
myset.remove("banana")
print(myset)
```

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

Example

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

```
myset = {"apple", "banana", "cherry"}
myset.discard("banana")
print(myset)
```

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

Example

The `clear()` method empties the set:

```
myset = {"apple", "banana", "cherry"}
myset.clear()
print(myset)
```

Example

The `del` keyword will delete the set completely:

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

del myset

print(myset)
```

Join Two Sets

There are several ways to join two or more sets in Python.

You can use the `union()` method that returns a new set containing all items from both sets, or the `update()` method that inserts all the items from one set into another:

Example

The `union()` method returns a new set with all items from both sets:

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}

set3 = set1.union(set2)
print(set3)
```

Example

The `update()` method inserts the items in set2 into set1:

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}

set1.update(set2)
print(set1)
```

Note: Both `union()` and `update()` will exclude any duplicate items.

Dictionary

A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.

Example

Create and print a dictionary:

```
mydict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(mydict)
```

Accessing Items

You can access the items of a dictionary by referring to its key name, inside square brackets:

Example

Get the value of the "model" key:

```
x = mydict["model"]
```

There is also a method called `get()` that will give you the same result:

Example

Get the value of the "model" key:

```
x = mydict.get("model")
```

Change Values

You can change the value of a specific item by referring to its key name:

Example

Change the "year" to 2018:

```
mydict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict["year"] = 2018
```

Loop Through a Dictionary

You can loop through a dictionary by using a `for` loop.

When looping through a dictionary, the return value are the *keys* of the dictionary, but there are methods to return the *values* as well.

Example

Print all key names in the dictionary, one by one:

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

Example

Print all *values* in the dictionary, one by one:

```
for x in mydict:
  print(mydict[x])
```

Example

You can also use the `values()` function to return values of a dictionary:

```
for x in mydict.values():
  print(x)
```

Example

Loop through both *keys* and *values*, by using the `items()` function:

```
for x, y in mydict.items():
  print(x, y)
```

Check if Key Exists

To determine if a specified key is present in a dictionary use the `in` keyword:

Example

Check if "model" is present in the dictionary:

```
mydict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
if "model" in mydict:
  print("Yes, 'model' is one of the keys in the mydict dictionary")
```

Dictionary Length

To determine how many items (key-value pairs) a dictionary has, use the `len()` method.

Example

Print the number of items in the dictionary:

```
print(len(mydict))
```

Adding Items

Adding an item to the dictionary is done by using a new index key and assigning a value to it:

Example

```
mydict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict["color"] = "red"
print(mydict)
```

Removing Items

There are several methods to remove items from a dictionary:

Example

The `pop()` method removes the item with the specified key name:

```
mydict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict.pop("model")
print(mydict)
```

Example

The `del` keyword removes the item with the specified key name:

```
mydict = {
  "brand": "Ford",
```

```
  "model": "Mustang",
  "year": 1964
}
del mydict["model"]
print(mydict)
```

Example

The `del` keyword can also delete the dictionary completely:

```
mydict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del mydict
print(mydict) #this will cause an error because "mydict" no longer exists.
```

Example

The `clear()` keyword empties the dictionary:

```
mydict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict.clear()
print(mydict)
```

Copy a Dictionary

You cannot copy a dictionary simply by typing `dict2 = dict1`.

There are ways to make a copy, one way is to use the built-in Dictionary method `copy()`.

Example

Make a copy of a dictionary with the `copy()` method:

```
Mydict1 = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict2 = mydict1.copy()
print(mydict2)
```

Another way to make a copy is to use the built-in method `dict()`.

Example

Make a copy of a dictionary with the `dict()` method:

```
mydict1 = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict2 = dict(mydict1)
print(mydict2)
```