# Sets and Dictionaries

## Exercises

### Week 7

Specify two ways in which a Set varies from a List.

*Answer:*

List is a ordered collection whereas set is an unordered collection. Similarly, list can have duplicate elements but set do not allow duplicate elements.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Write a Python statement that uses the set() *constructor* to produce the same Set as the following -

languages = { "C++", "Java", "C#", "PHP", "JavaScript" }

*Answer:*

languages = {"C++", "Java", "C#", "PHP", "JavaScript"}

new\_set = set(languages)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Is a Set **mutable** or **immutable**?

*Answer:*

Mutable

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Why does a Set not support *indexing* and *slicing* type operations?

*Answer:*

A Set does not support indexing and slicing type operations because it is an unordered collection of unique elements.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Why is a frozenset() different from a regular set?

*Answer:*

frozenset () is different from a regular set on the basis of mutability.

set () is mutable whereas frozenset () is immutable.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

How many elements would exist in the following set?

names = set("John", "Eric", "Terry", "Michael", "Graham", "Terry")

*Answer:*

The set “names” would contain 5 elements.

And how many elements would exist in this set?

vowels = set("aeiou")

*Answer:*

The set “vowels” would contain 5 elements.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

What is the name given to the following type of expression which can be used to programmatically populate a set?

chars = {chr(n) for n in range(32, 128)}

*Answer:*

Set comprehension

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

What **operator** can be used to calculate the intersection (common elements) between two sets?

*Answer:*

Ampersand Operator (&) can be used to calculate the intersection between two sets.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

What **operator** can be used to calculate the difference between two sets?

*Answer:*

The minus (-) operator can be used to calculate the difference between two sets.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

What would be the result of each of the following expressions?

{ "x", "y", "z" } < { "z" , "u", "t", "y", "w", "x" }

*Answer:*

True

{ "x", "y", "z" } < { "z", "y", "x" }

*Answer:*

False

{ "x", "y", "z" } <= { "y", "z", "x" }

*Answer:*

True

{ "x" } > { "x" }

*Answer:*

False

{ "x", "y" } > { "x" }

*Answer:*

True

{ "x", "y" } == { "y", "x" }

*Answer:*

True

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Write a Python statement that uses a **method** to perform the equivalent of the following operation -

languages = languages | { "Python" }

*Answer:*

To add an element on existing set you can also use add method.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Do the elements which are placed into a set always remain in the same position?

*Answer:*

No, the elements in sets do not have a fixed position or order.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Is the following operation a **mutator** or an **accessor**?

languages &= oo\_languages

*Answer:*

Mutator

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

What term is often used to refer to each *pair* of elements stored within a **dictionary**?

*Answer:*

The term commonly used to refer to each pair of elements stored within a dictionary is “key-value pair”.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Is it possible for a dictionary to have more than one **key** with the same value?

*Answer:*

Yes, it is possible.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Is it possible for a dictionary to have the same **value** appear more than once?

*Answer:*

Yes, it is posibble.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Is a Dictionary **mutable** or **immutable**?

*Answer:*

Mutable

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Are the **key** values within a dictionary **mutable** or **immutable**?

*Answer:*

They are immutable.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

How many *elements* exist in the following dictionary?

stock = {"apple":10, "banana":15, "orange":11}

*Answer:*

Three

And, what is the data-type of the **keys**?

*Answer:*

The data-type of the keys are string.

And, what output would be displayed by executing the following statement -

print(stock["banana"])

*Answer:*

15 gets printed because the value assigned to it is 15.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Write a Python statement that uses the dictionary() *constructor* to produce the same dictionary as the following -

lang\_gen = { "Java":3, "Assembly":2, "Machine Code":1 }

*Answer:*

lang\_gen = dict({"Java": 3, "Assembly": 2, "Machine Code": 1})

Now write a simple expression that tests whether the word "Assembly" is a member of the dictionary.

*Answer:*

print("Assembly" in lang\_gen)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Write some Python code that uses a for statement to iterate over a dictionary called module\_stats and print only its **values** (i.e. do not output any keys) -

*Answer:*

module\_stats = {"Java": 3, "Assembly": 2, "Machine Code": 1}

for value in module\_stats.values():

print(value)

Now write another loop which prints the only the **keys** -

*Answer:*

module\_stats = {"Java": 3, "Assembly": 2, "Machine Code": 1}

for key in module\_stats:

print(key)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Is it possible to construct a dictionary using a **comprehension** style expression, as supported by lists and sets?

*Answer:*

Yes, it is possible to construct a dictionary using a comprehension style expression.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

When a Dictionary type value is being passed as an argument to a function, what characters can be used as a prefix to force the dictionary to be **unpacked** prior to the call being made?

*Answer:*

The \*\* character can be used as a prefix to force the dictionary to be unpacked prior to

the call being made.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**