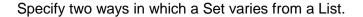# Sets and Dictionaries

Exercises

Week 7

Prior to attempting these exercises ensure you have read the lecture notes and/or viewed the video, and followed the practical. You may wish to use the Python interpreter in interactive mode to help work out the solutions to some of the questions.

Download and store this document within your own filespace, so the contents can be edited. You will be able to refer to it during the test in Week 6.

Enter your answers directly into the highlighted boxes.

For more information about the module delivery, assessment and feedback please refer to the module within the MyBeckett portal.

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

Answer:

They do not support indexing and slicing.
A value can appear at most once within the set, i.e. no duplicates exist

_____

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 = set(["C++", "Java", "C#", "PHP", "JavaScript"])

_____

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

Answer:

mutable

_____

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

Answer:

Because sets do not support the concept of element positioning.

_____

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

Answer:

They are immutable.

_____

How many elements would exist in the following set?

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

Answer:

5

And how many elements would exist in this set?

```
vowels = set("aeiou")
```

Answer:

5

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:

&

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

Answer:

-

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:

True

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

Answer:

False

```
{ "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:

update()

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

Answer:

No

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:

key-value pair

_____

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

Answer:

Yes

_____

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

Answer:

Yes

_____

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

Answer:

> mutable

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

Answer:

> immutable

How many *elements* exist in the following dictionary?

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

Answer:

> 3

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

Answer:

> string

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

```
print(stock["banana"])
```

Answer:

> 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:

> if "Assembly" in lang_gen:
>   print("Assembly is a member of the dictionary")
> else:
>   print("Assembly is not a member of the dictionary")

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:

```
for value in module_stats.values():
    print(value)
```

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

Answer:

```
for key in module_stats.keys():
    print(key)
```

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

Answer:

```
Yes
```

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:

```
**
```

Exercises are complete

Save this logbook with your answers. Then ask your tutor to check your responses to each question.