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:

Unordered
No duplication

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

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

Answer:

```
new_set = set(languages)
```

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

Answer:

A set is mutable.

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

Answer:

A set does not support indexing and slicing operations because it is an unordered collection.

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

Answer:

A frozenset() only allows reading of the data in it.

How many elements would exist in the following set?

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

Answer:

The code would produce an error.

And how many elements would exist in this set?

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

Answer:

There would be 5 elements in this set.

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:

The following type of expression is called set comprehension.

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

Answer:

The ‘&’ 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 ‘-‘ 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:

languages.add("Python")

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

Answer:

The elements placed in a set are not always in the same position.

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

```
languages &= oo_languages
```

Answer:

The following operation is a mutator.

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

Answer:

Key value pair is often used to refer to each pair of elements stored within a dictionary.

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

Answer:

It is not possible to have more than one key with the same value.

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

Answer:

Yes, it is possible for the same value to appear more than once.

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

Answer:

A dictionary is mutable.

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

Answer:

The key values within a dictionary are

How many *elements* exist in the following dictionary?

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

Answer:

The following dictionary contains 3 elements.

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

Answer:

The data-type of the keys is ‘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:

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:
        print(“Yoohoo”)
```

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 i in module_stats.values():
        print i
```

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

Answer:

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

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 ‘**’ prefix causes the dictionary to be unpacked prior to the function call.

Exercises are complete

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