Dictionary Methods

This notebook introduces some dictionary methods. They are:

1. `keys()` : used to retrieve the keys in the dictionary. The method returns a list of keys containing the keys in the dictonary.
2. `get()` : returns the value associated with the key.

```
    If the key does not exist and the optional value is not provided, the get() method re
turns the reserved word None
    If the key does not exist and the optional value is provided, the get() method will r
eturn the value.
```

3. `values()` : used to retrieve the values in the dictionary. The method returns a list of the values in the dictionary.
4. `items()` : used to retrieve the values in the dictionary. The method returns a list containing each key-value pair as a tuple

Methods of an object are accessed using the `objectname.methodname()`. So to use dictionary methods, we will use `dictionaryname.methodname()`

The `keys()` method is used to retrieve the keys in the dictionary. The method returns a list.

In [1]:

```
sales = { 'apple': 2, 'orange': 3, 'grapes': 4, 'persimmon': 1, 'mango': 8 }
print(sales.keys())
```

```
dict_keys(['apple', 'orange', 'grapes', 'persimmon', 'mango'])
```

The key in a dictionary be traversed by using the for construct together with the `keys()` methods.

In [2]:

```
sales = { 'apple': 2, 'orange': 3, 'grapes': 4, 'persimmon': 1, 'mango': 8 }
for x in sales.keys():
    print(x)
```

```
apple
orange
grapes
persimmon
mango
```

The `values()` method is used to retrieve the values in the dictionary. The method returns a list.

In [3]:

```
sales = { 'apple': 2, 'orange': 3, 'grapes': 3, 'persimmon': 1, 'mango': 8 }
print(sales.values())
```

```
dict_values([2, 3, 3, 1, 8])
```

The values in a dictionary be traversed by using the for construct together with the `values()` method.

In [4]:

```
sales = { 'apple': 2, 'orange': 3, 'grapes': 4, 'persimmon': 1, 'mango': 8 }
for value in sales.values():
    print(value)
```

```
2
3
4
1
```

```
8
```

The `items()` method is used to retrieve the keys together with the corresponding values in the dictionary. The method returns a list containing each key-value pair as a tuple.

In [5]:

```
sales = { 'apple': 2, 'orange': 3, 'grapes': 4, 'persimmon': 1, 'mango': 8 }
print(sales.items())
l = sales.items()
print(type(l))
```

```
dict_items([('apple', 2), ('orange', 3), ('grapes', 4), ('persimmon', 1), ('mango', 8)])
<class 'dict_items'>
```

In [6]:

```
'''
Each key-value pair can be printed out as a tuple using the for construct.
'''
sales = { 'apple': 2, 'orange': 3, 'grapes': 4, 'persimmon': 1, 'mango': 8 }
for item in sales.items():
    print(item)
```

```
('apple', 2)
('orange', 3)
('grapes', 4)
('persimmon', 1)
('mango', 8)
```

In [7]:

```
'''
The key from each key-value pair can be accessed by using index 0.
'''
sales = { 'apple': 2, 'orange': 3, 'grapes': 4, 'persimmon': 1, 'mango': 8 }
for item in sales.items():
    print(item[0])
```

```
apple
orange
grapes
persimmon
mango
```

In [8]:

```
'''
The value from each key-value pair can be accessed by using index 1.
'''
sales = { 'apple': 2, 'orange': 3, 'grapes': 4, 'persimmon': 1, 'mango': 8 }
for item in sales.items():
    print(item[1])
```

```
2
3
4
1
8
```

We can use two iterator variables to access both the key and the value in one iteration. This is called unpacking.

In [9]:

```
sales = { 'apple': 2, 'orange': 3, 'grapes': 4, 'persimmon': 1, 'mango': 8 }
for x, y in sales.items():
    print(x, y)
```

```
apple 2
orange 3
grapes 4
persimmon 1
mango 8
```

The syntax for the `get()` method is `dictname.get(key, [value])`. This method returns the value associated with the key. If the key does not exist and the optional value is not provided, the `get()` method returns the reserved word `None`. If the key does not exist and the optional value is provided, the `get()` method will return the value.

In [10]:

```
sales = {'apple': 2, 'orange': 3, 'grapes': 4, 'persimmon': 1, 'mango': 8 }
print(sales.get('persimmon'))
print(sales.get('papaya'))
print(sales.get('papaya', 8))
```

```
1
None
8
```

As discussed earlier, we can also look up values associated with a key by using the `dictionaryname[key]` construct which also returns the value associated with the key. However, if the key does not exist, it returns an `KeyError`.
The `get()` method on the other hand does not return an error.

In [11]:

```
sales = { 'apple': 2, 'orange': 3, 'grapes': 4, 'persimmon': 1, 'mango': 8 }
print(sales['persimmon'])
print(sales['papaya'])
```

```
1

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-11-23007f8b0645> in <module>
      1 sales = { 'apple': 2, 'orange': 3, 'grapes': 4, 'persimmon':1, 'mango':8 }
      2 print(sales['persimmon'])
----> 3 print(sales['papaya'])

KeyError: 'papaya'
```