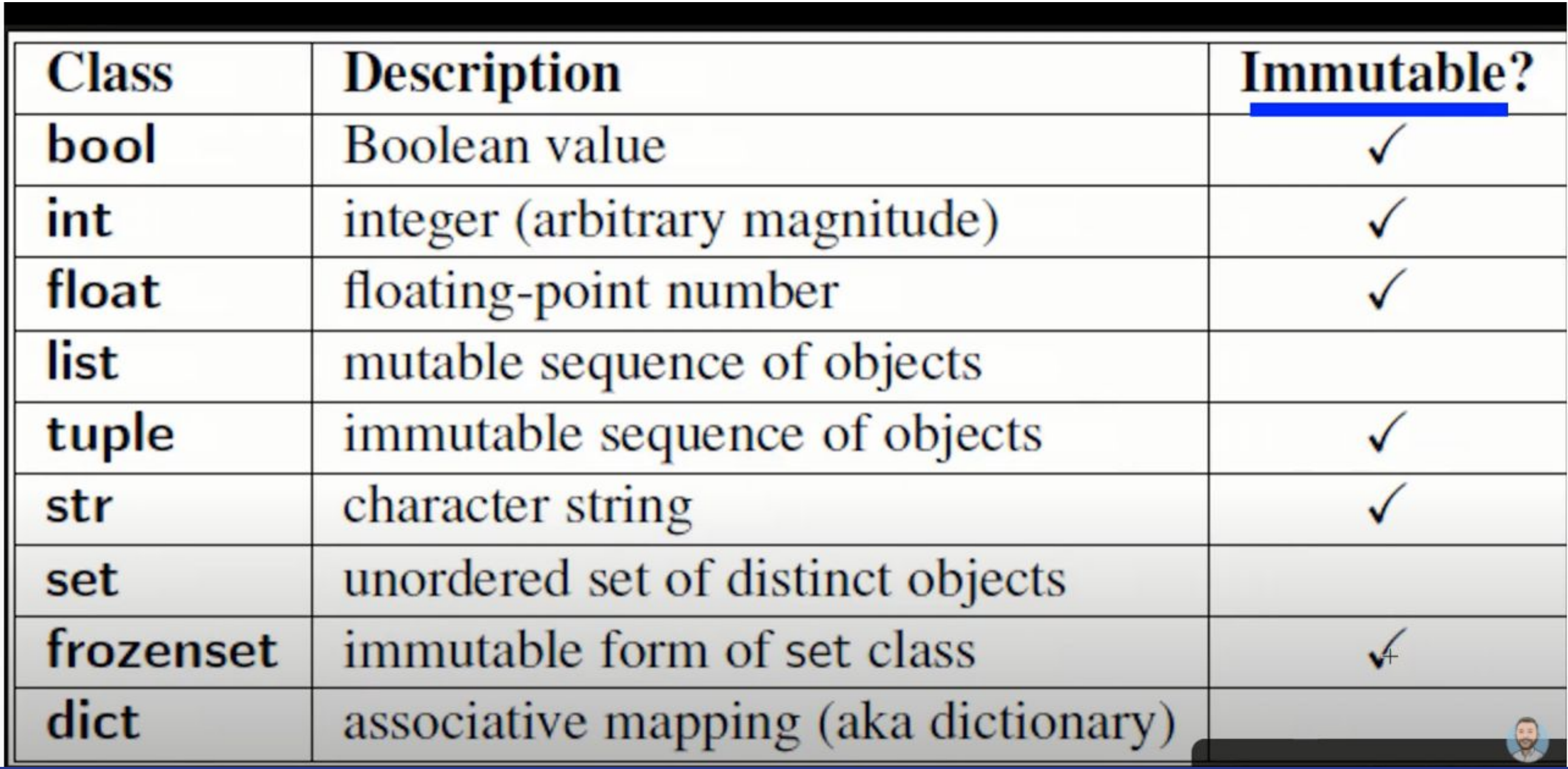
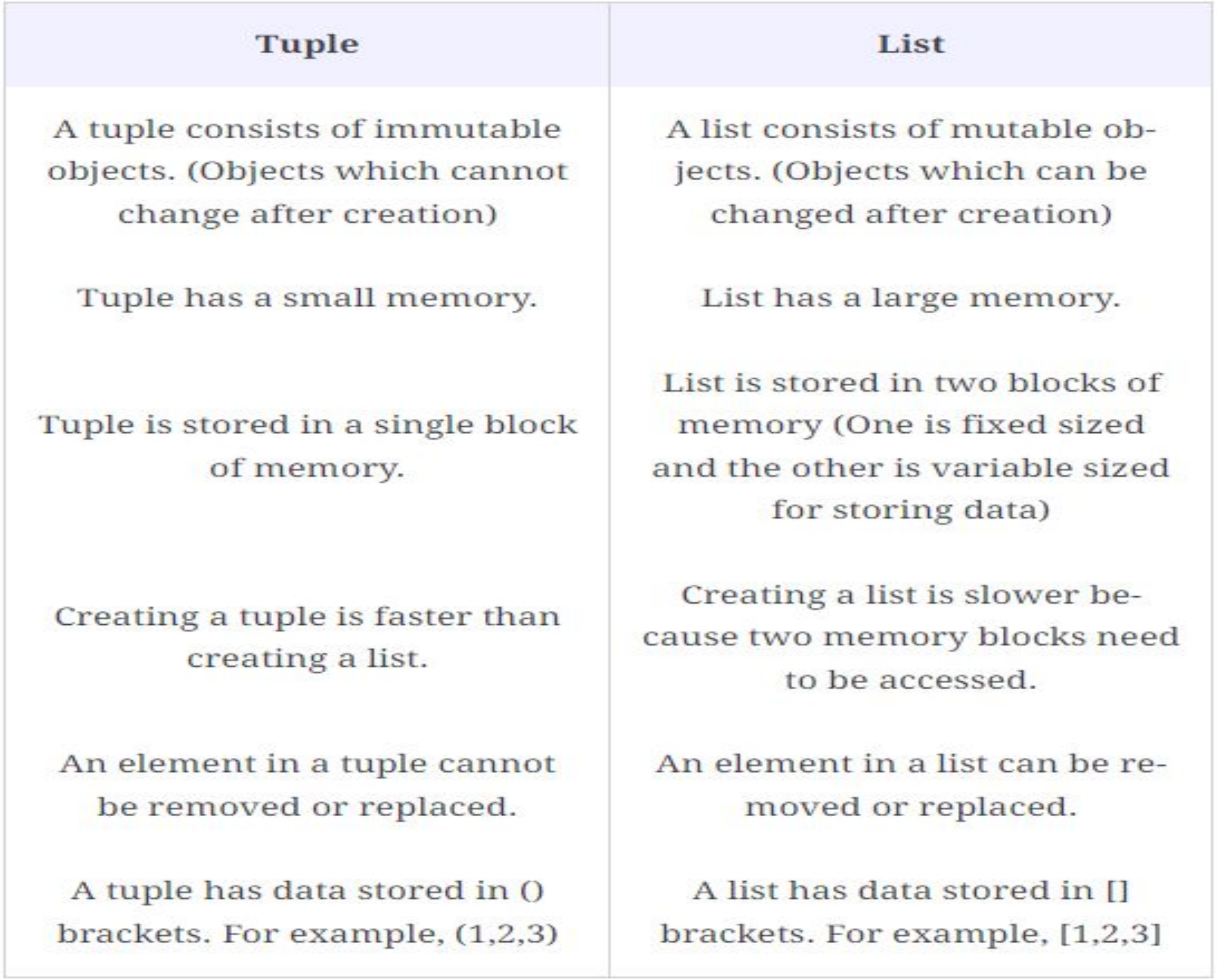
- Python Built in Functions
- Difference Between Tuple and Lists
- Dictionary Concepts Revision
- Dictionary Examples
- Set Introduction

Python Built in Functions

- eval()
- sum()
- exec()
- round()

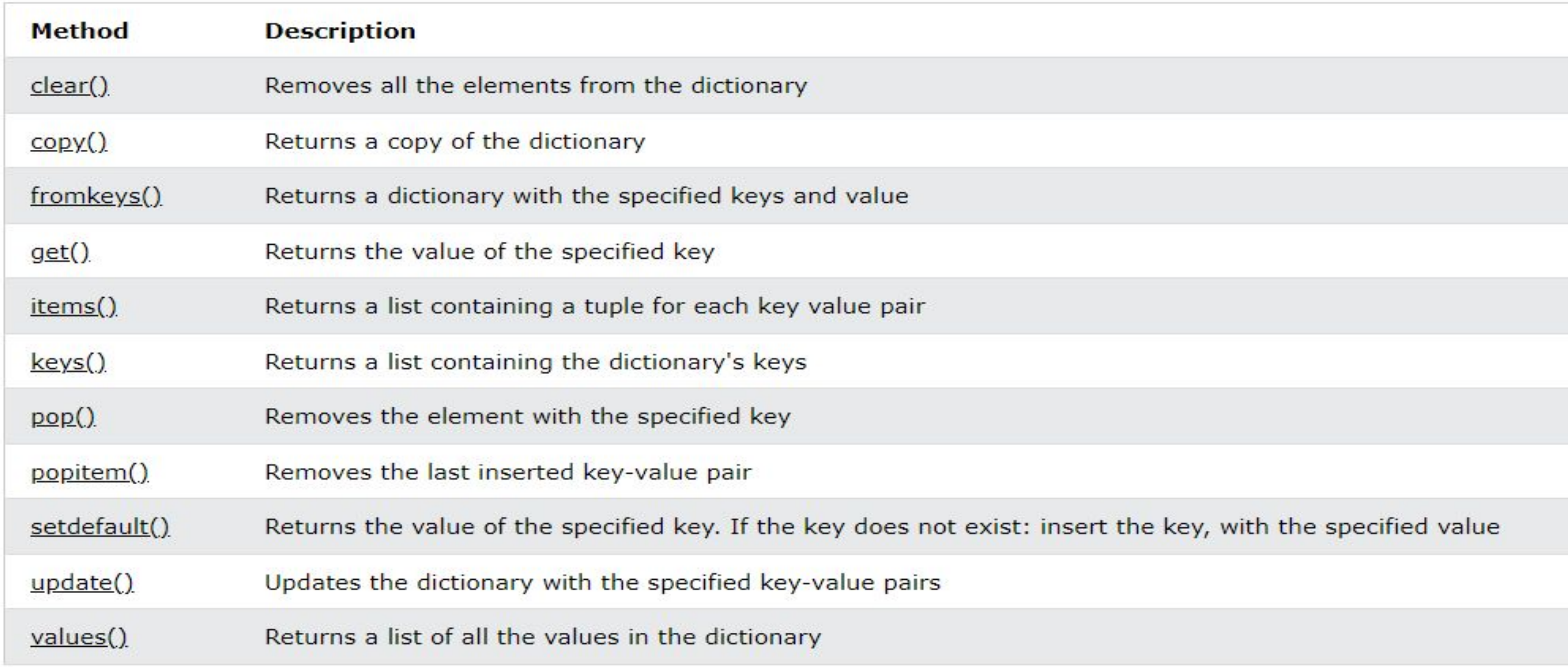


Class	Description	Immutable?
bool	Boolean value	✓
int	integer (arbitrary magnitude)	✓
float	floating-point number	✓
list	mutable sequence of objects	
tuple	immutable sequence of objects	✓
str	character string	✓
set	unordered set of distinct objects	
frozenset	immutable form of set class	✓
dict	associative mapping (aka dictionary)	



Tuple	List
A tuple consists of immutable objects. (Objects which cannot change after creation)	A list consists of mutable objects. (Objects which can be changed after creation)
Tuple has a small memory.	List has a large memory.
Tuple is stored in a single block of memory.	List is stored in two blocks of memory (One is fixed sized and the other is variable sized for storing data)
Creating a tuple is faster than creating a list.	Creating a list is slower because two memory blocks need to be accessed.
An element in a tuple cannot be removed or replaced.	An element in a list can be removed or replaced.
A tuple has data stored in () brackets. For example, (1,2,3)	A list has data stored in [] brackets. For example, [1,2,3]

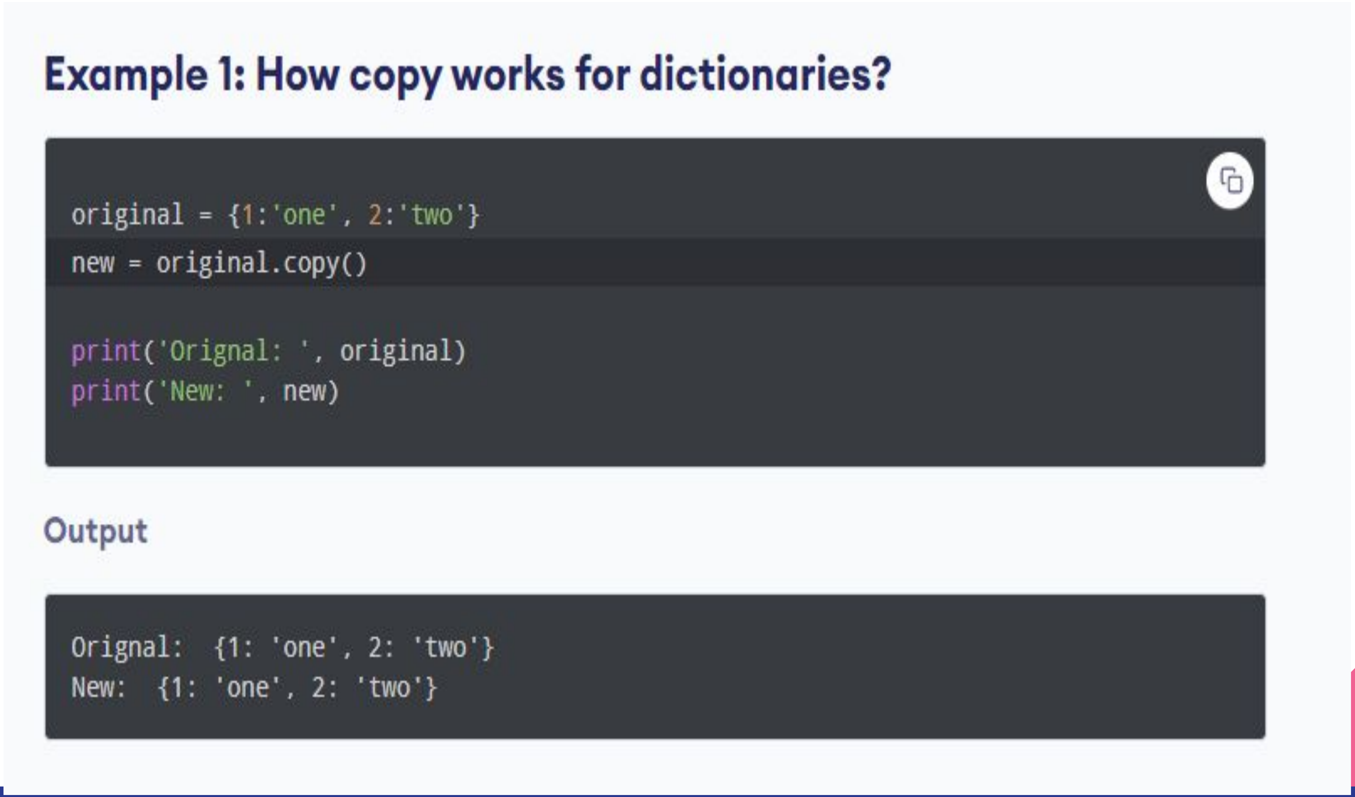
Dictionary Functions



Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Copy of dictionary

dict.copy()

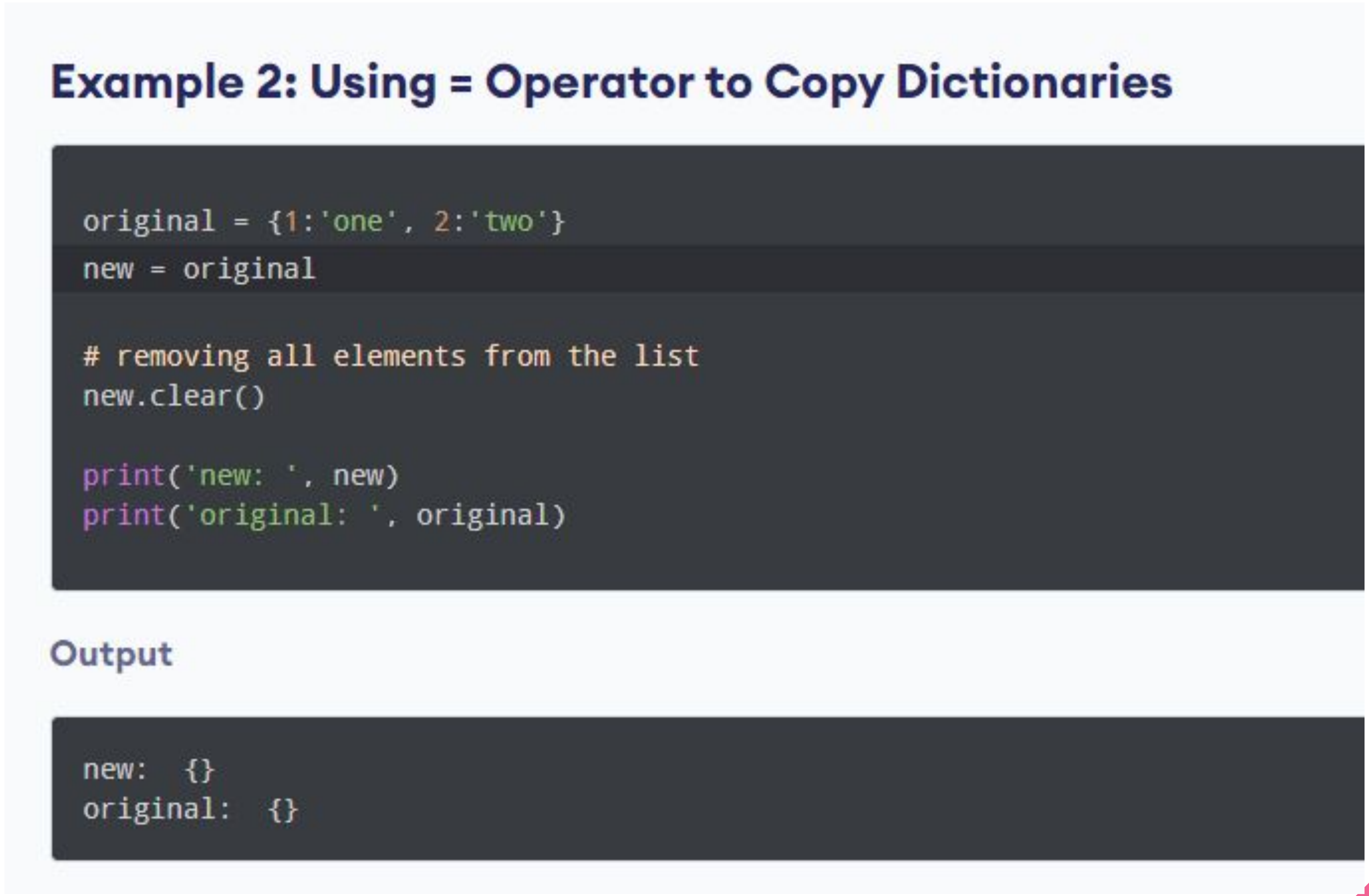


Example 1: How copy works for dictionaries?

```
original = {1:'one', 2:'two'}
new = original.copy()

print('Orignal: ', original)
print('New: ', new)
```

Output

```
Orignal:  {1: 'one', 2: 'two'}
New:  {1: 'one', 2: 'two'}
```

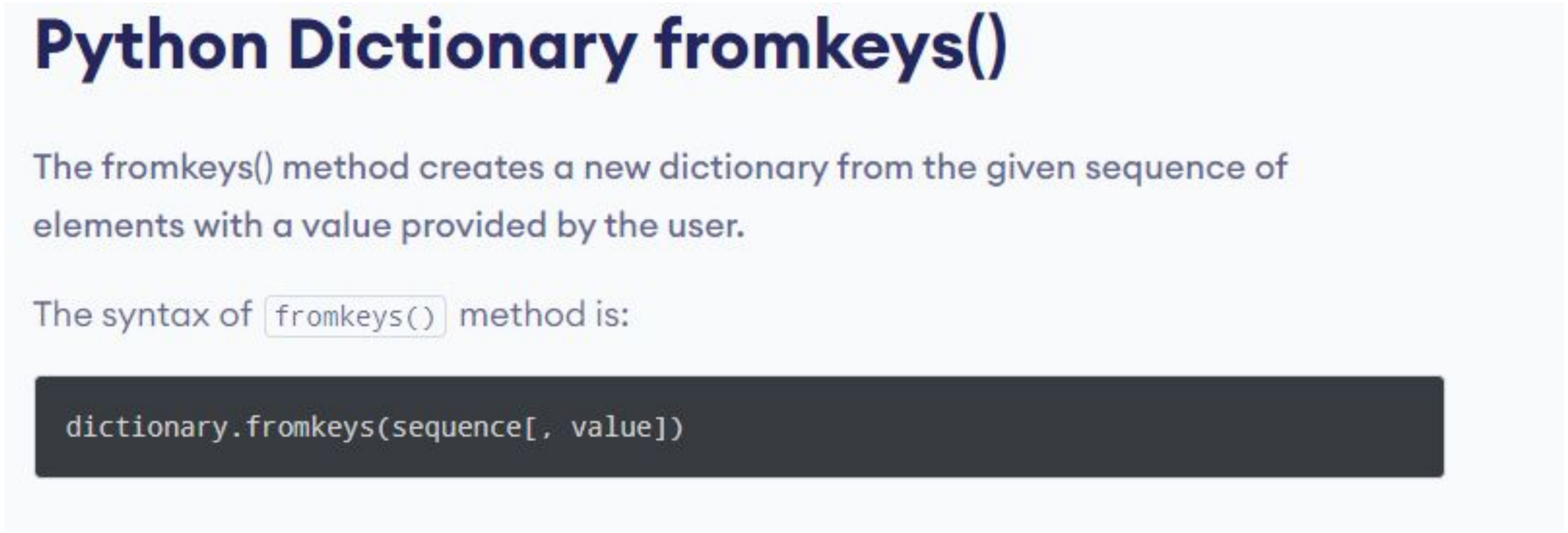


Example 2: Using = Operator to Copy Dictionaries

```python
original = {1:'one', 2:'two'}
new = original

# removing all elements from the list
new.clear()

print('new: ', new)
print('original: ', original)
```

Output

```
new:  {}
original:  {}
```

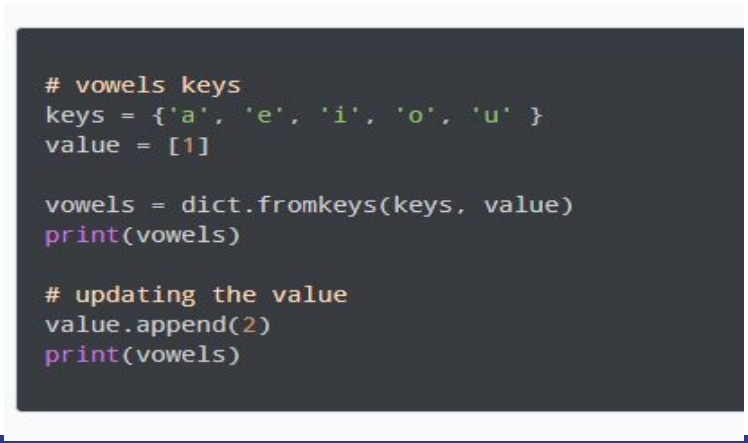


Python Dictionary fromkeys()

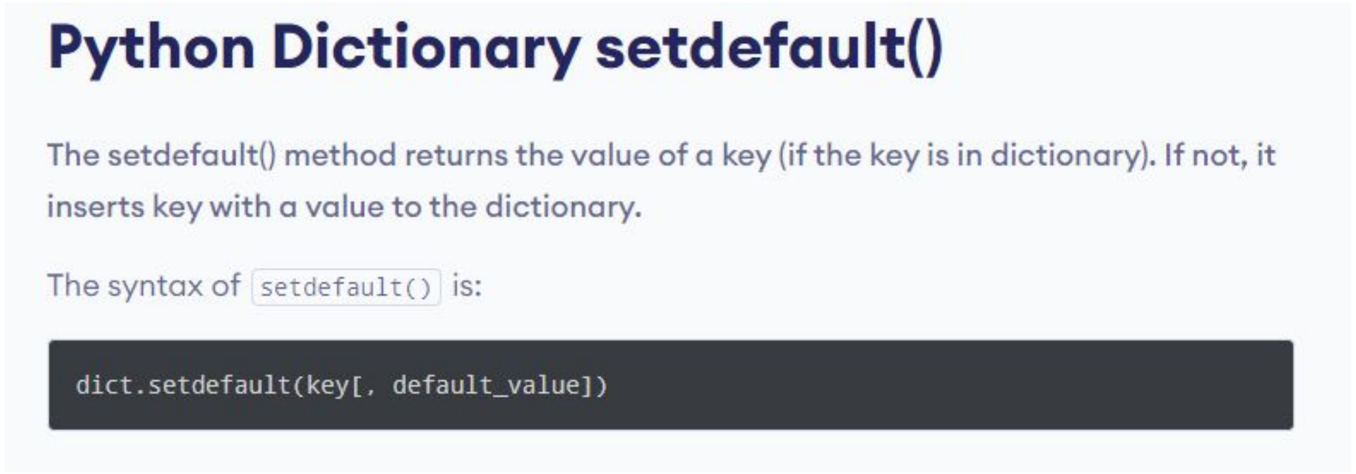
The fromkeys() method creates a new dictionary from the given sequence of elements with a value provided by the user.

The syntax of `fromkeys()` method is:

```
dictionary.fromkeys(sequence[, value])
```



```
# vowels keys
keys = {'a', 'e', 'i', 'o', 'u' }
value = [1]

vowels = dict.fromkeys(keys, value)
print(vowels)

# updating the value
value.append(2)
print(vowels)
```

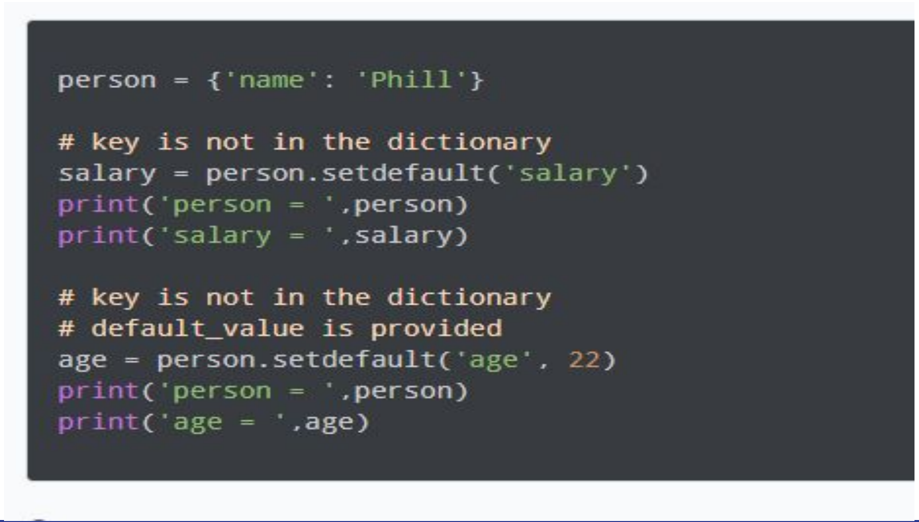


Python Dictionary setdefault()

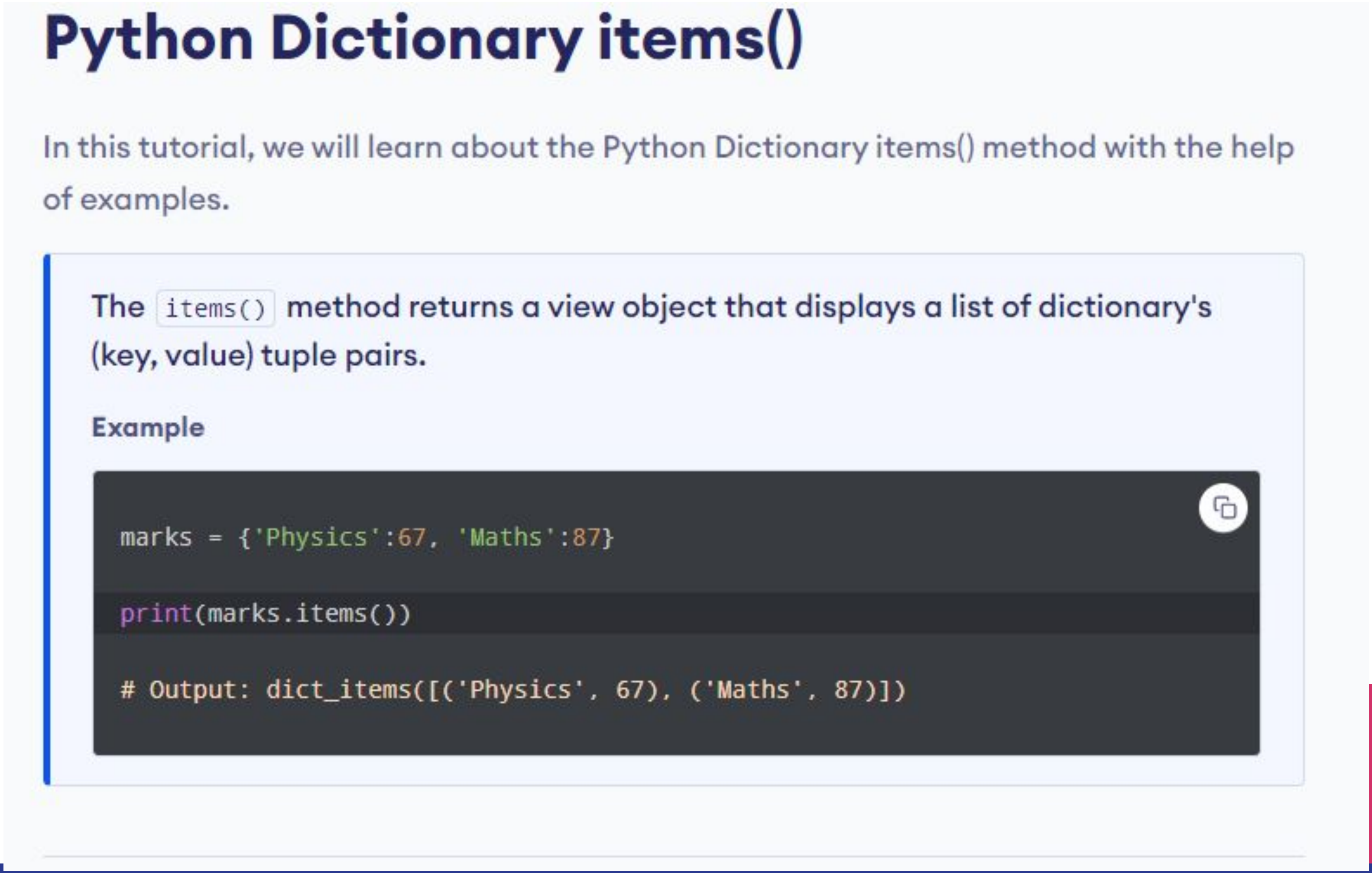
The setdefault() method returns the value of a key (if the key is in dictionary). If not, it inserts key with a value to the dictionary.

The syntax of `setdefault()` is:

```
dict.setdefault(key[, default_value])
```



```
person = {'name': 'Phill'}

# key is not in the dictionary
salary = person.setdefault('salary')
print('person = ',person)
print('salary = ',salary)

# key is not in the dictionary
# default_value is provided
age = person.setdefault('age', 22)
print('person = ',person)
print('age = ',age)
```



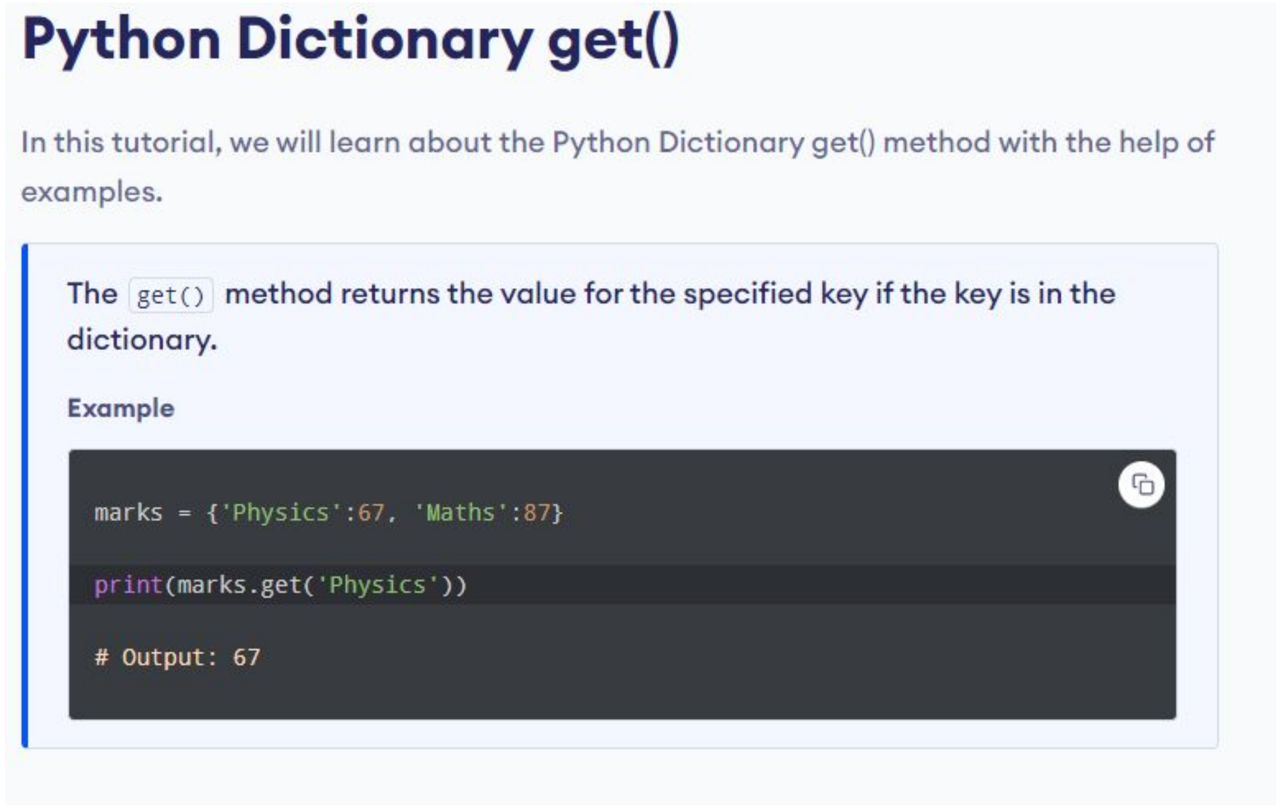
Python Dictionary items()

In this tutorial, we will learn about the Python Dictionary items() method with the help of examples.

The `items()` method returns a view object that displays a list of dictionary's (key, value) tuple pairs.

Example

```
marks = {'Physics':67, 'Maths':87}

print(marks.items())

# Output: dict_items([('Physics', 67), ('Maths', 87)])
```



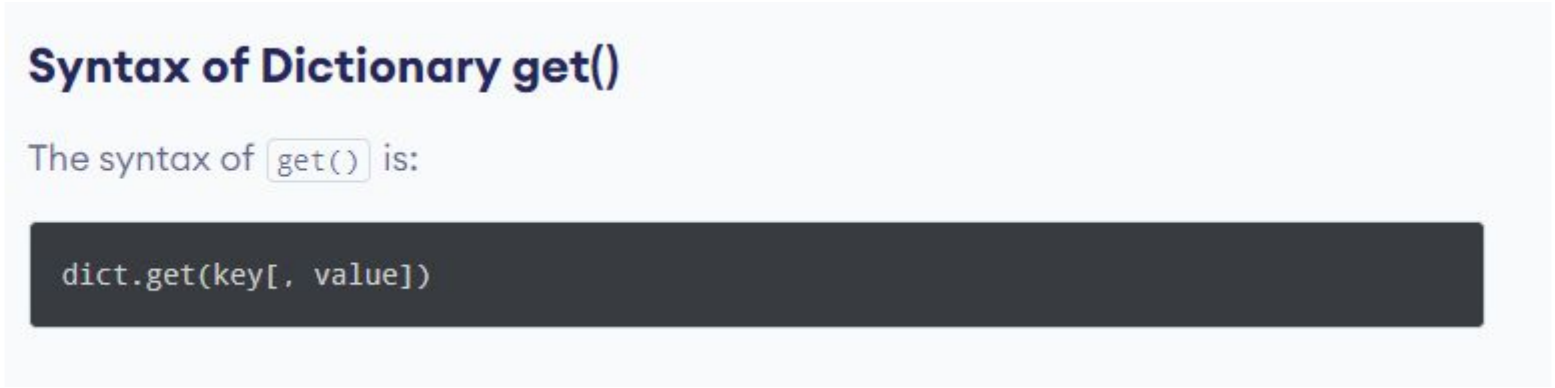
Python Dictionary get()

In this tutorial, we will learn about the Python Dictionary get() method with the help of examples.

The `get()` method returns the value for the specified key if the key is in the dictionary.

Example

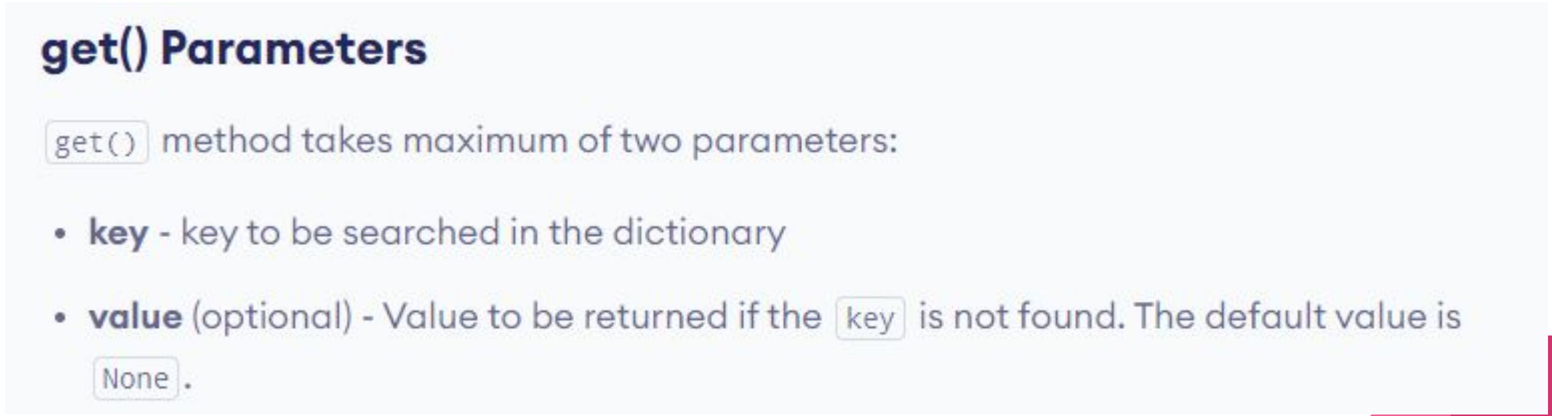
```
marks = {'Physics':67, 'Maths':87}

print(marks.get('Physics'))

# Output: 67
```



Syntax of Dictionary get()

The syntax of `get()` is:

```
dict.get(key[, value])
```

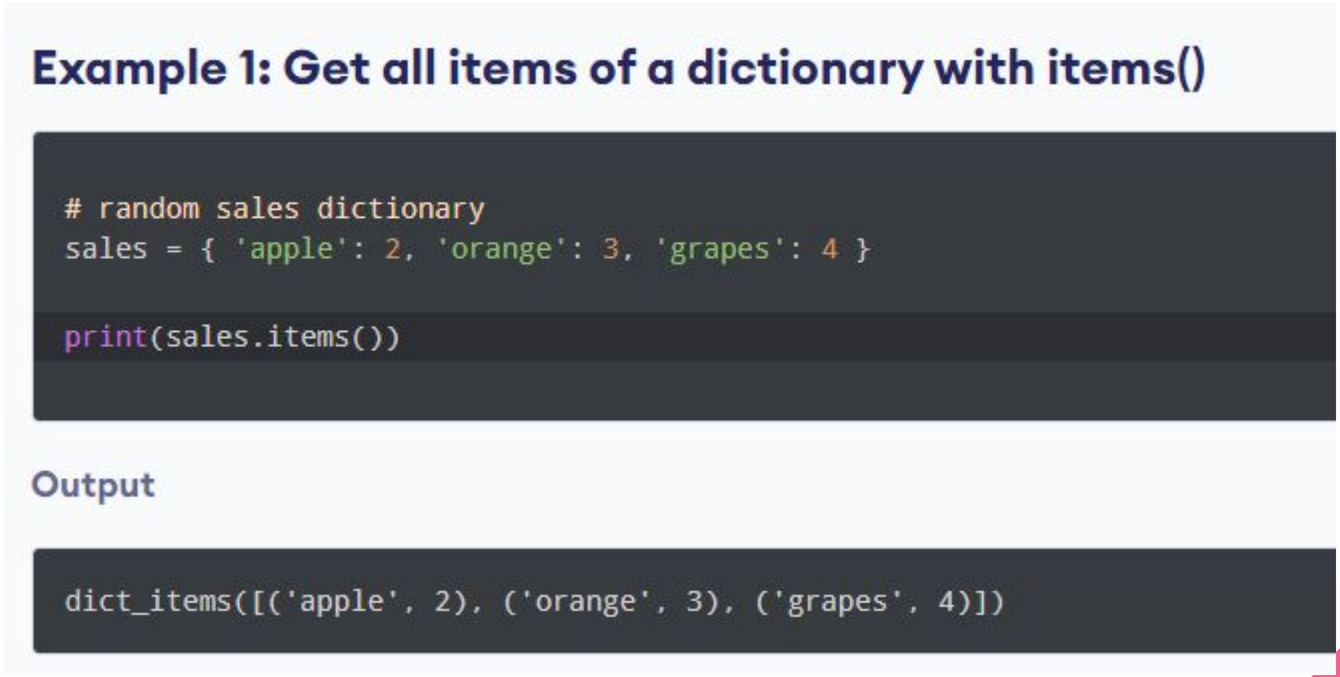


get() Parameters

`get()` method takes maximum of two parameters:

- **key** - key to be searched in the dictionary
- **value** (optional) - Value to be returned if the `key` is not found. The default value is `None`.

items()



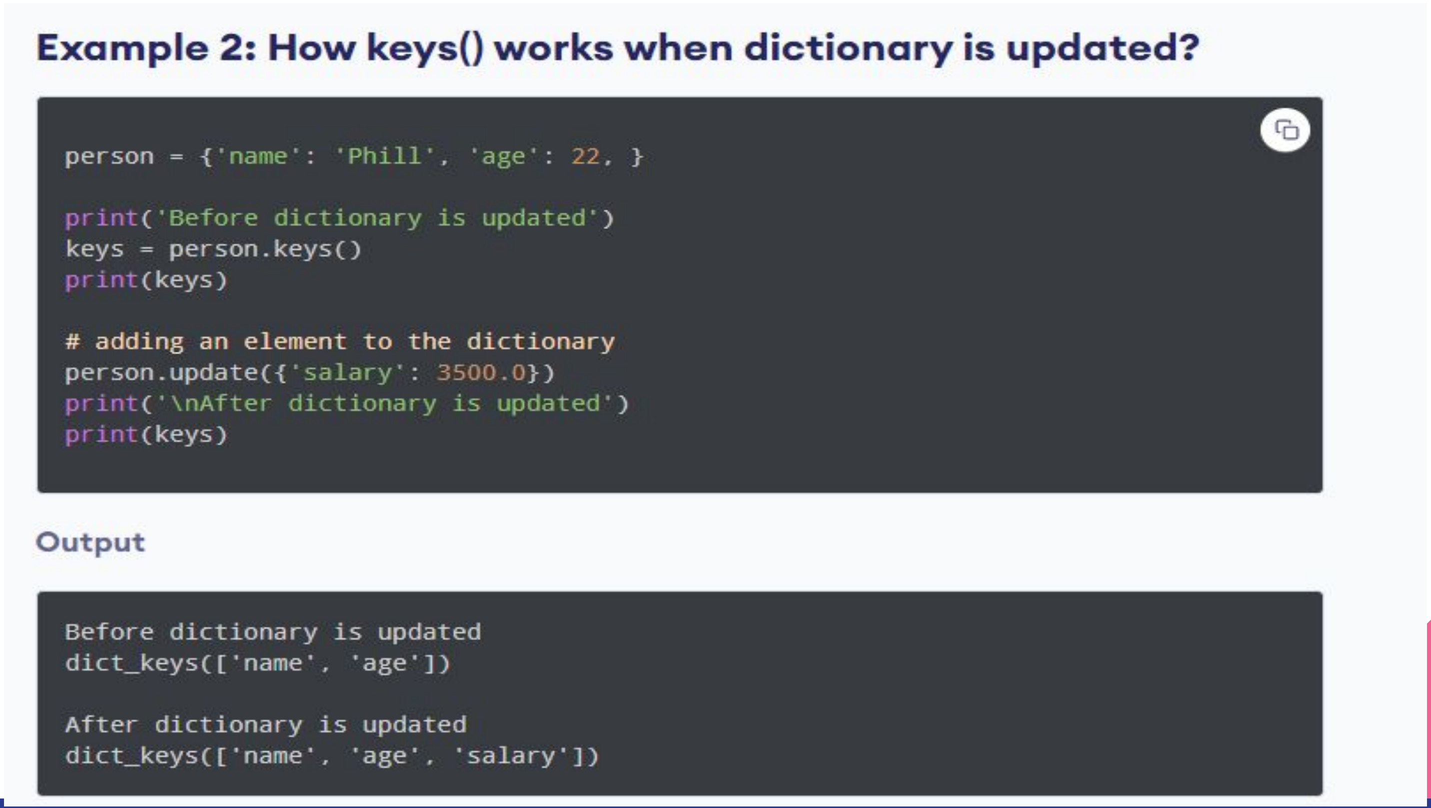
Example 1: Get all items of a dictionary with items()

```
# random sales dictionary
sales = { 'apple': 2, 'orange': 3, 'grapes': 4 }

print(sales.items())
```

Output

```
dict_items([('apple', 2), ('orange', 3), ('grapes', 4)])
```

keys()



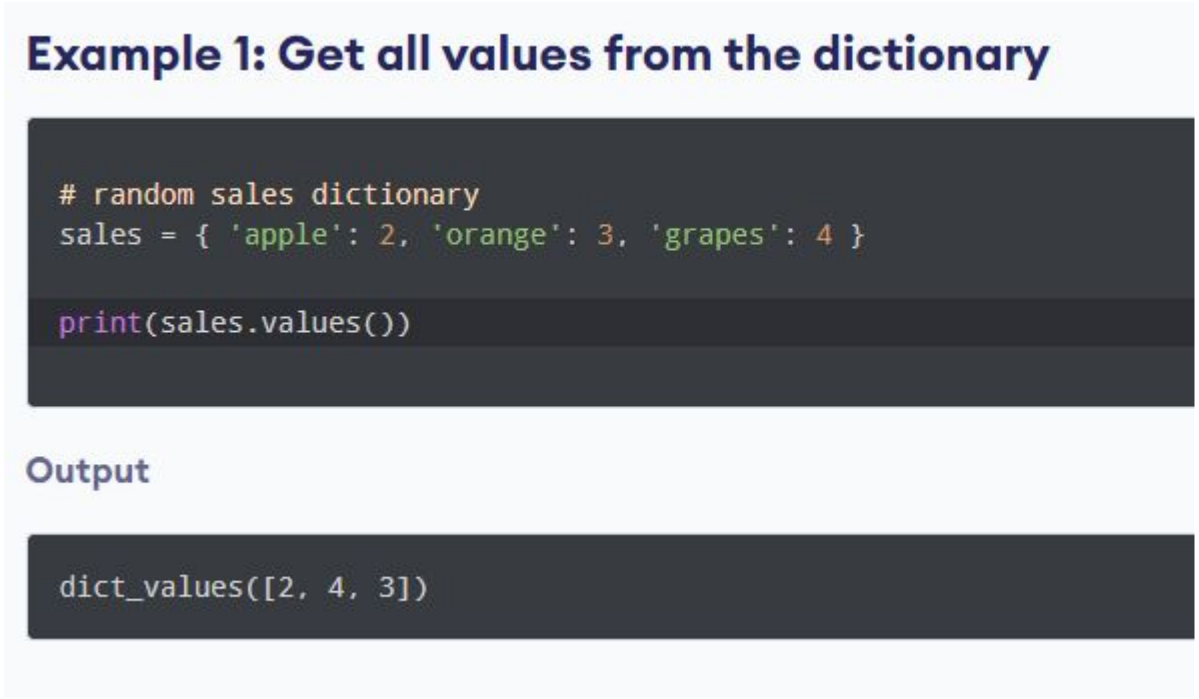
Example 2: How keys() works when dictionary is updated?

```
person = {'name': 'Phill', 'age': 22, }

print('Before dictionary is updated')
keys = person.keys()
print(keys)

# adding an element to the dictionary
person.update({'salary': 3500.0})
print('\nAfter dictionary is updated')
print(keys)
```

Output

```
Before dictionary is updated
dict_keys(['name', 'age'])

After dictionary is updated
dict_keys(['name', 'age', 'salary'])
```

values()



Example 1: Get all values from the dictionary

```
# random sales dictionary
sales = { 'apple': 2, 'orange': 3, 'grapes': 4 }

print(sales.values())
```

Output

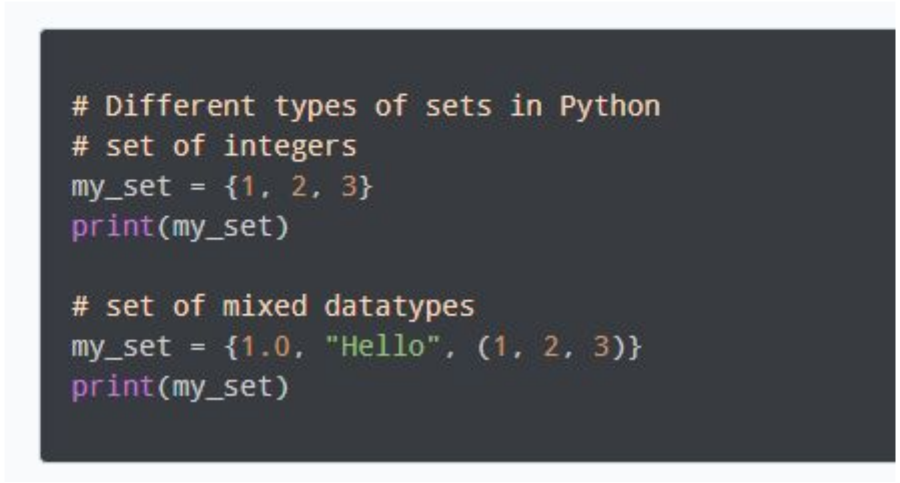
```
dict_values([2, 4, 3])
```

Set Introduction

- A set is an unordered collection of items.
- Every set element is unique (no duplicates) and must be immutable (cannot be changed).
- However, a set itself is mutable. We can add or remove items from it.
- Sets can also be used to perform mathematical set operations like union, intersection, symmetric difference, etc.
-

Creating Python Sets

But a set cannot have mutable elements like lists, sets or dictionaries as its elements.



```
# Different types of sets in Python
# set of integers
my_set = {1, 2, 3}
print(my_set)

# set of mixed datatypes
my_set = {1.0, "Hello", (1, 2, 3)}
print(my_set)
```