DICTIONARY

{ Collection Data Type }

Learning Path With Pankaj Chouhan
@p4n.in

Python Dictionary

The compound data types like lists, tuples, strings and sets have only one value as an element.

Now, we will learn about dictionaries. A dictionary is an unordered collection of items (similar like sets). However, it has a `key: value` pair instead of just values.

Let's learn how to create dictionaries next.

With Pankaj Chouhan
@p4n.in

How to Create a Dictionary?

A dictionary is as simple as placing items inside curly braces `{}`, separated by commas.

An item in a dictionary has a key and the corresponding value expressed as a pair, `key: value`.

While values can be of any data type and can repeat, keys must be of immutable type (string, number or tuple with immutable elements) and must be unique.

```
# empty dictionary
my_dict = {}

# dictionary with integer keys
my_dict = {1: 'apple', 2: 'ball'}

# dictionary with mixed keys
my_dict = {'name': 'John', 1: [2, 4, 3]}
```

LEARNING PATH With Pankaj Chouhan

Creating Dictionaries using dict()

A dictionary can also be created using the built-in method `dict()`.

```
my_dict = dict({1:'apple', 2:'ball'})

# creating dictionary from a sequence
my_dict = dict([(1,'apple'), (2,'ball')])

print(my_dict)
# Output: {1: 'apple', 2: 'ball'}
```

LEARNING PATH
With Pankaj Chouhan
@p4n.in

Access Dictionary Elements

A dictionary has a `key: value` pair as elements. To access the `value` of an element, we use its `key`. Here's how:

```
person = {'name': 'Jack', 'age': 26}

person_name = person['name']
print(person_name)  # Output: Jack

person_age = person['age']
print(person_age)  # Output: 26
```

We have created a dictionary, `person`. The dictionary has two elements.

Here, `person['name']` gives us `'Jack'` and `person['age']` gives us `26`.

If the key is not present, you get an error.

```
>> person['salary']
KeyError: 'salary'
```

LEARNING PATH With Pankaj Chouhan

Access Value using get()

We can also access the value of an element using the `get()` method.

```
my_dict = {'name':'Jack', 'age': 26}

person_name = my_dict.get('name')
print(person_name)   # Output: 'Jack'

person_age = my_dict.get('age')
print(person_age)   # Output: 26
```

The difference while using `get()` is that it returns `None` instead of `KeyError`, if the key is not found.

```
my_dict = {'name':'Jack', 'age': 26}

person_salary = my_dict.get('salary')
print(person_salary)    # Output: None
```

In Python, `None` signifies empty or no value.

Learning Path with Pankaj Chouhan

Change & Add Dictionary Items

Dictionary is mutable. We can add new items or change the value of existing items using the `=` operator.

Add Element to a dictionary

```
person = {'name':'Jack', 'age': 26}

# adding element
person['salary'] = 6000

print(person)
# Output: {'name': 'Jack', 'age': 26,
'salary': 6000}
```

Here, we have added an element to a dictionary. The element's key is `'salary'` and its value is `6000`.

Update element of a dictionary

We can update an element in a similar way to how we add an element.

```
person = {'name':'Jack', 'age': 26}

# updating age
person['age'] = 19

print(person)
# Output: {'name': 'Jack', 'age': 19}
```

LEARNING PATH With Pankaj Chouhan

Remove Dictionary Item

To remove a single item from a dictionary, we can use the `pop()` method. The method removes an item with the provided `key` and returns the value.

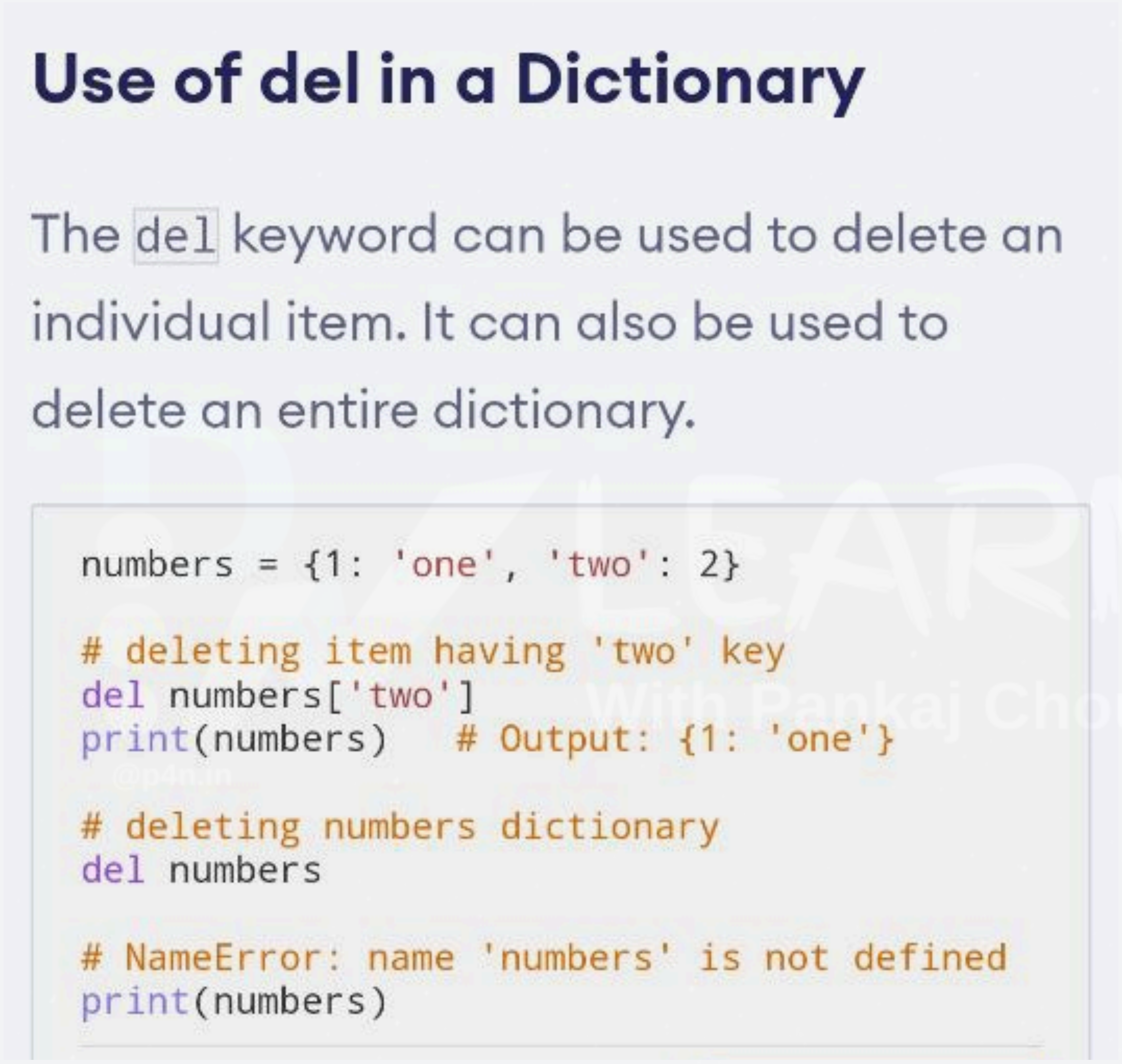
```
person = {'name':'Jack', 'age': 26}

# removing element having key 'name'
value = person.pop('name')

print('Value removed:', value)
print('Updated dict:', person)
```

Output

```
Value removed: Jack
Updated dict: {'age': 26}
```

If we need to remove an arbitrary element from a dictionary, we can use the `popitem()` method.

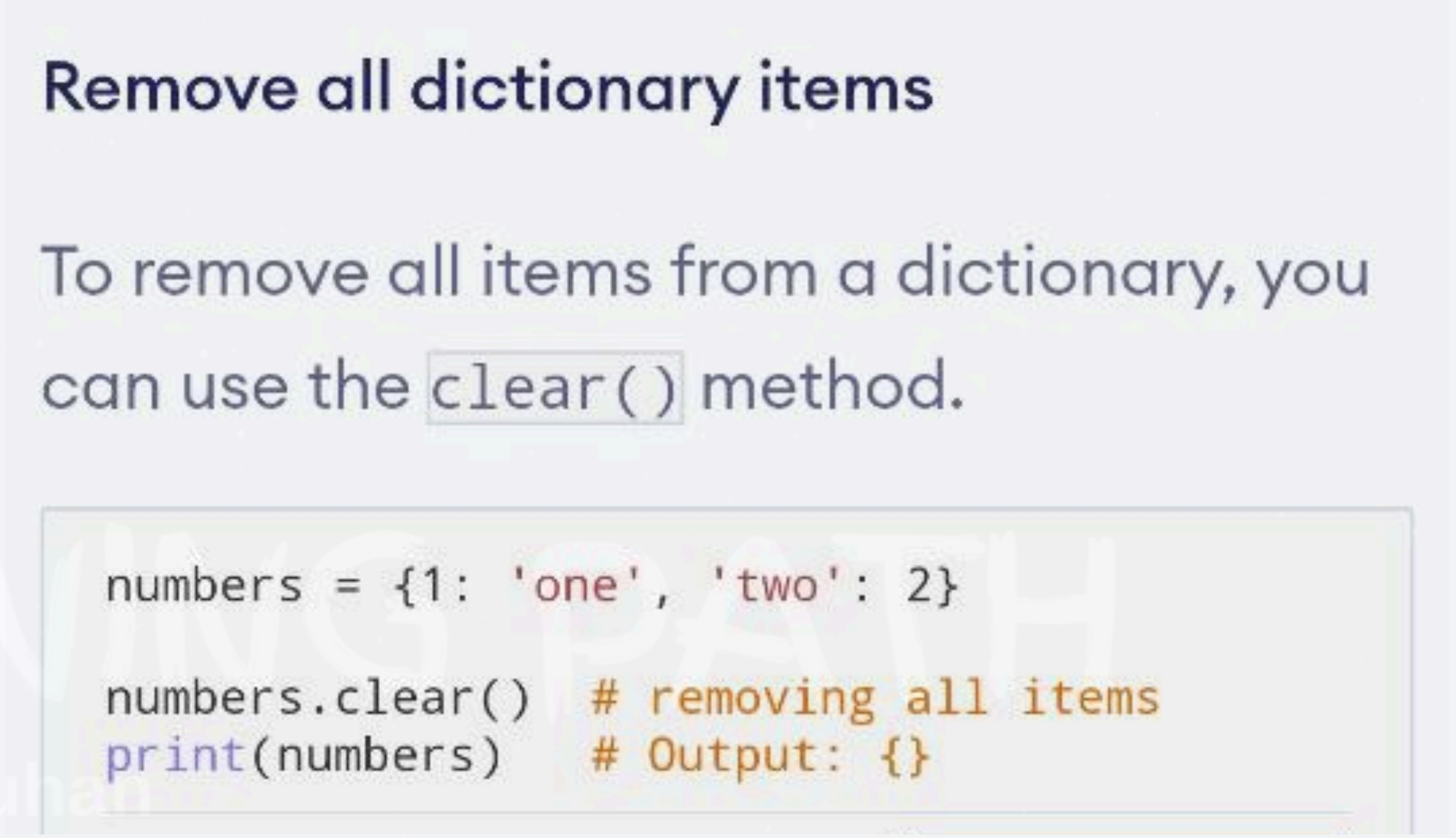
LEARNING PATH
With Pankaj Chouhan



Use of del in a Dictionary

The `del` keyword can be used to delete an individual item. It can also be used to delete an entire dictionary.

```
numbers = {1: 'one', 'two': 2}

# deleting item having 'two' key
del numbers['two']
print(numbers)   # Output: {1: 'one'}

# deleting numbers dictionary
del numbers

# NameError: name 'numbers' is not defined
print(numbers)
```



Remove all dictionary items

To remove all items from a dictionary, you can use the `clear()` method.

```
numbers = {1: 'one', 'two': 2}

numbers.clear()  # removing all items
print(numbers)   # Output: {}
```

LEARNING PATH
With Pankaj Chouhan
@p4n.in

Iterating Through a Dictionary

Using a `for` loop, we can iterate through each key in a dictionary.

```
squares = {1: 1, 3: 9, 5: 25}

for i in squares:
    print(squares[i])
```

Output

```
1
9
25
```

Once you get the key, you can always get its corresponding value.

Check if a key exists

You can check whether a key exists in a dictionary or not using the `in` keyword.

```
squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}

print(1 in squares)      # Output: True
print(2 not in squares)  # Output: True
print(49 in squares)     # Output: False
```

Notice that the membership test is for keys only, not for values.

Python Dictionary Methods

Here's a list of dictionary methods we have learned till now:

- `get()` - returns the value of the key passed
- `pop()` - removes the item of the key passed and returns its value
- `popitem()` - removes an arbitrary item and returns its key
- `clear()` - removes all items from a dictionary

LEARNING PATH
With Pankaj Chouhan