

Python dictionary is an ordered collection (starting from **Python 3.7**) of items. It stores elements in **key/value** pairs. Here, **keys** are unique identifiers that are associated with each **value**.

Let's see an example,

If we want to store information about countries and their capitals, we can create a dictionary with country names as **keys** and capitals as **values**.

Keys	Values
Nepal	Kathmandu
Italy	Rome
England	London

## Create a dictionary in Python

Here's how we can create a dictionary in Python.



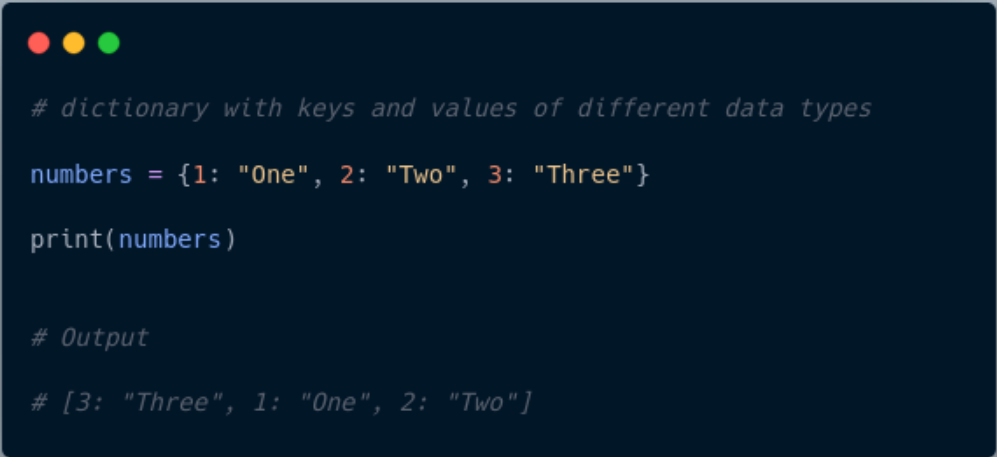
```
capital_city = {"Nepal": "Kathmandu", "Italy": "Rome",  
"England": "London"}  
  
print(capital_city)  
  
# Output  
  
# {'Nepal': 'Kathmandu', 'Italy': 'Rome', 'England': 'London'}
```

In the above example, we have created a dictionary named `capital_city`. Here,

1. **Keys** are "Nepal", "Italy", "England"
2. **Values** are "Kathmandu", "Rome", "London"

**Note:** Here, **keys** and **values** both are of string type. We can also have **keys** and **values** of different data types.

## Example 1: Python Dictionary



```
# dictionary with keys and values of different data types  
numbers = {1: "One", 2: "Two", 3: "Three"}  
print(numbers)  
  
# Output  
# [3: "Three", 1: "One", 2: "Two"]
```

In the above example, we have created a dictionary named numbers. Here, **keys** are of integer type and **values** are of string type.

## Add Elements to a Python Dictionary

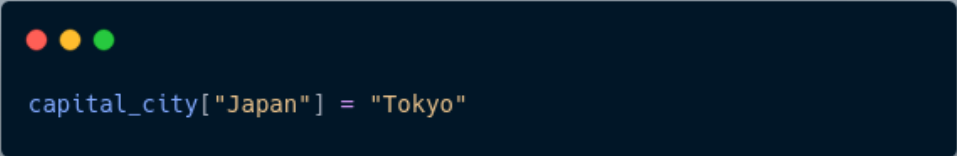
We can add elements to a dictionary using the name of the dictionary with [].

For example,



```
capital_city = {"Nepal": "Kathmandu", "England": "London"}  
print("Initial Dictionary: ",capital_city)  
  
capital_city["Japan"] = "Tokyo"  
print("Updated Dictionary: ",capital_city)  
  
# Output  
  
# Initial Dictionary: {'Nepal': 'Kathmandu', 'England':  
'London'}  
  
# Updated Dictionary: {'Nepal': 'Kathmandu', 'England':  
'London', 'Japan': 'Tokyo'}
```

In the above example, we have created a dictionary named `capital_city`. Notice the line,

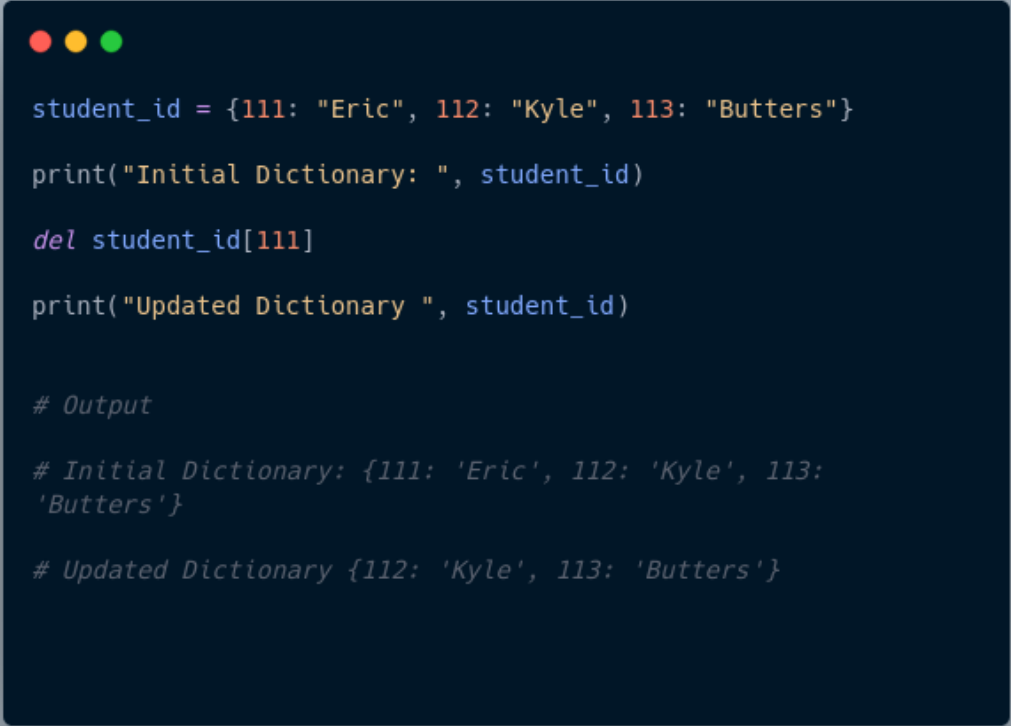


```
capital_city["Japan"] = "Tokyo"
```

Here, we have added a new element to `capital_city` with **key**: Japan and **value**: Tokyo.

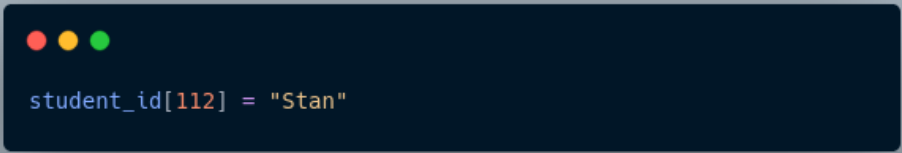
## Change Value of Dictionary

We can also use `[]` to change the value associated with a particular key. For example,



```
student_id = {111: "Eric", 112: "Kyle", 113: "Butters"}  
print("Initial Dictionary: ", student_id)  
  
del student_id[111]  
print("Updated Dictionary ", student_id)  
  
# Output  
  
# Initial Dictionary: {111: 'Eric', 112: 'Kyle', 113: 'Butters'}  
  
# Updated Dictionary {112: 'Kyle', 113: 'Butters'}
```

In the above example, we have created a dictionary named `student_id`. Initially, the value associated with the key 112 is "Kyle". Now, notice the line,




```
student_id[112] = "Stan"
```

Here, we have changed the value associated with the key 112 to "Stan".

## Accessing Elements from Dictionary

In Python, we use the keys to access their corresponding values. For example,

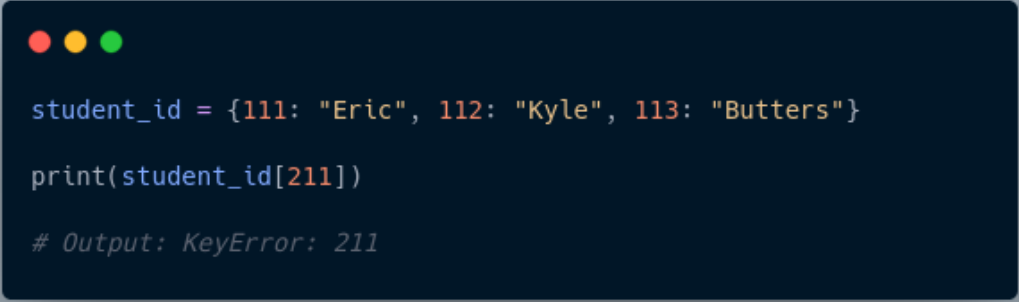
A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains Python code that defines a dictionary and prints values using keys.

```
student_id = {111: "Eric", 112: "Kyle", 113: "Butters"}  
print(student_id[111]) # prints Eric  
print(student_id[113]) # prints Butters
```

Here, we have used the keys to access their corresponding values.

If we try to access the value of a key that doesn't exist, we'll get an error.

For example,

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains Python code that attempts to access a non-existent key in a dictionary, resulting in a KeyError.

```
student_id = {111: "Eric", 112: "Kyle", 113: "Butters"}  
print(student_id[211])  
  
# Output: KeyError: 211
```

## Removing elements from Dictionary

We use the `del` statement to remove an element from the dictionary. For example,



```
student_id = {111: "Eric", 112: "Kyle", 113: "Butters"}
print("Initial Dictionary: ", student_id)

del student_id[111]

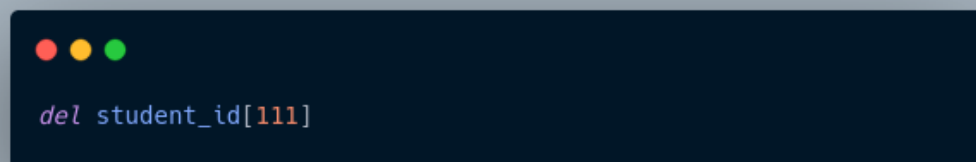
print("Updated Dictionary ", student_id)

# Output

# Initial Dictionary:  {111: 'Eric', 112: 'Kyle', 113: 'Butters'}

# Updated Dictionary  {112: 'Kyle', 113: 'Butters'}
```

Here, we have created a dictionary named `student_id`. Notice the code,



```
del student_id[111]
```

The `del` statement removes the element associated with the key 111.

We can also delete the whole dictionary using the `del` statement,



```
student_id = {111: "Eric", 112: "Kyle", 113: "Butters"}  
  
# delete student_id dictionary  
  
del student_id  
  
print(student_id)  
  
# Output: NameError: name 'student_id' is not defined
```

We are getting an error message because we have deleted the `student_id` dictionary and `student_id` doesn't exist anymore.

## Python Dictionary Methods


Methods that are available with a dictionary are tabulated below. Some of them have already been used in the above examples.



Function	Description
<code>all()</code>	Return <code>True</code> if all keys of the dictionary are True (or if the dictionary is empty).
<code>any()</code>	Return <code>True</code> if any key of the dictionary is true. If the dictionary is empty, return <code>False</code> .
<code>len()</code>	Return the length (the number of items) in the dictionary.
<code>sorted()</code>	Return a new sorted list of keys in the dictionary.
<code>clear()</code>	Removes all items from the dictionary.
<code>keys()</code>	Returns a new object of the dictionary's keys.
<code>values()</code>	Returns a new object of the dictionary's values

## Dictionary Membership Test

We can test if a key is in a dictionary or not using the keyword `in`. Notice that the membership test is only for the keys and not for the values.



```
# Membership Test for Dictionary Keys

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

# Output: True

print(1 in squares) # prints True

print(2 not in squares) # prints True

# membership tests for key only not value

print(49 in squares) # prints false

# Output

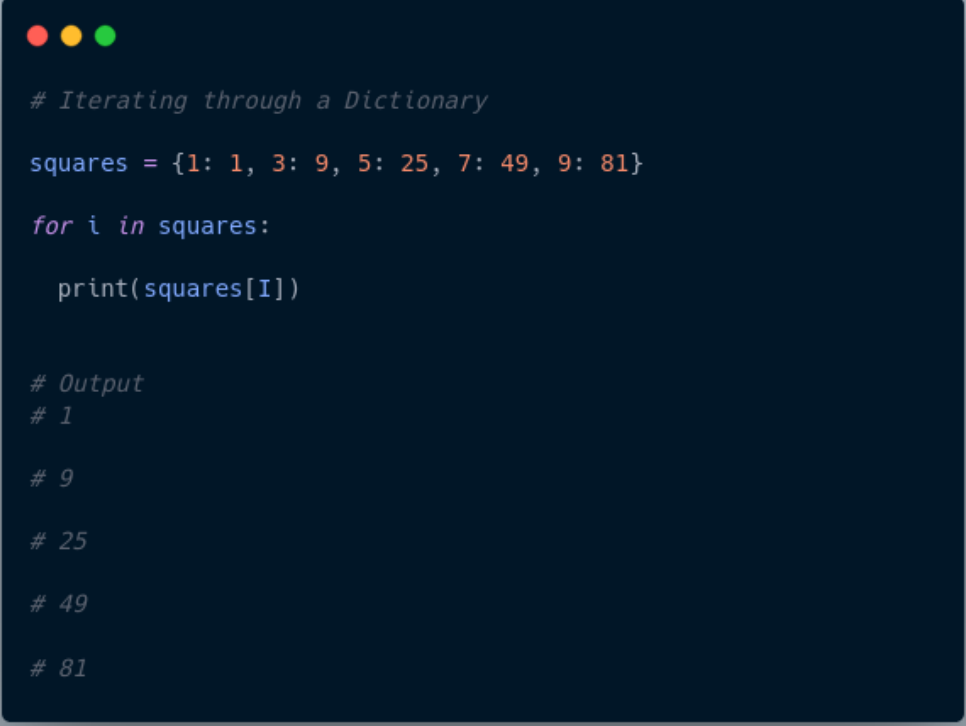
# True

# True

# False
```

## Iterating Through a Dictionary

We can iterate through each key in a dictionary using a loop.



```
# Iterating through a Dictionary

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

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

# Output
# 1

# 9

# 25

# 49

# 81
```

Here, we have iterated through each **key** in the squares dictionary using the for loop.

More Resources:

1. [https://www.w3schools.com/python/python\\_dictionaries.asp](https://www.w3schools.com/python/python_dictionaries.asp)
2. <https://realpython.com/python-dicts/>