Rex Galaxy Technology

"Innovating the future through technology."

PYTHON DICTIONARY

‘’This is complete guide about python dictionary that will help you to learn and understand all operations of dictionary.’’

10 Aug, 2023

Submitted by:
Chetan.

Rex Galaxy Technology
"Innovating the future through technology."

PYTHON DICTIONARY GUIDE

1. Python Dictionary:
2. Create the Dictionary:
3. Accessing elements of a Dictionary:
4. Accessing an element of a nested dictionary:
5. Deleting Elements using del Keyword:
6. Dictionary methods:

- copy() method:
- clear() method:
- get() method:
- items() method:
- keys() method:
- pop() method:
- update() method:
- values() method :

Rex Galaxy Technology
"Innovating the future through technology."

Python Dictionary

Dictionaries are one of Python's most powerful and important data structures. You may recognize them from other programming languages, in which they can be known as "hashes," "associative arrays," "hash maps," or "hash tables." In a dictionary, we don't enter individual elements, as in a list or tuple. Rather, we enter pairs of data, with the first item known as the "key," and the second item known as the "value."

A dictionary holds indexes with keys that are mapped to certain values. These key-value pairs offer a great way of organizing and storing data in Python. They are mutable, meaning you can change the stored information.

- Dictionaries are unordered mappings of 'Name : Value' associations.
- Comparable to hashes and associative arrays in other languages.
- Intended to approximate how humans remember associations.
- Keys must be a single element
- Value can be any type such as list, tuple, integer, etc.

Note– As of Python version 3.7, dictionaries are ordered. In Python 3.6 and earlier, dictionaries are unordered.

Create the Dictionary:

The dictionary can be created by using multiple key-value pairs enclosed with the curly brackets {}, and each key is separated from its value by the colon (:).The syntax to define the dictionary is given below.

```
Dict1 = {"name": "Rohit","class": "Python", "city": "New York"}
```

Dictionary can also be created by the built-in function dict(). An empty dictionary can be created by just placing to curly braces{}.

```
# Creating an empty Dictionary
Dict = {}
print("Empty Dictionary: ")
print(Dict)
```

Output:

```
Empty Dictionary:
{}
```

```
# Creating a Dictionary
# with dict() method
Dict = dict({1: 'Vinnie', 2: 'and', 3: 'Vasu'})
print("\nDictionary with the use of dict(): ")
print(Dict)
```

Output:

```
Dictionary with the use of dict():
{1: 'Vinnie', 2: 'and', 3: 'Vasu'}
```

```
# Creating a Dictionary
# with each item as a Pair
Dict = dict([(1, 'Vinnie'), (2, 'For')])
print("\nDictionary with each item as a pair: ")
print(Dict)
```

Output:

```
Dictionary with each item as a pair:
{1: 'Vinnie', 2: 'for'}
```

Accessing elements of a Dictionary:

In order to access the items of a dictionary refer to its key name. Key can be used inside square brackets.

```
# Python program to demonstrate
# accessing a element from a Dictionary
# Creating a Dictionary


Dict = {1: 'Roy', 'name': 'and', 3: 'Vani'}


# accessing a element using key
print("Accessing a element using key:")
print(Dict['name'])
```

Output:

Accessing a element using key: and

```
# accessing a element using key
print("Accessing a element using key:")
print(Dict[1])
```

Output:

Accessing a element using key: Roy

There is also a method called get() that will also help in accessing the element from a dictionary. This method accepts key as argument and returns the value.

```
# Creating a Dictionary
Dict = {1: 'Vinni', 'name': 'For', 3: 'Roy'}

# accessing a element using get()
# method

print("Accessing a element using get:")
print(Dict.get(3))
```

Output:

Accessing a element using get: Roy

Accessing an element of a nested dictionary:

In order to access the value of any key in the nested dictionary, use indexing [] syntax.

```
# Creating a Dictionary
Dict = {'Dict1': {1: 'Vinni'},
'Dict2': {'Name': 'Vasu'}}

# Accessing element using key
print(Dict['Dict1'])
print(Dict['Dict1'][1])
print(Dict['Dict2']['Name'])
```

Output:

```
{1: 'Vinni'}
Vinni
Vasu
```

Deleting Elements using del Keyword:

The items of the dictionary can be deleted by using the del keyword as given below.

```
# Python program to demonstrate
# Deleting Elements using del Keyword
# Creating a Dictionary

Dict = {1: 'Vinni', 'name': 'and', 3: 'Vasu'}
print("Dictionary =")
print(Dict)
```

Output:

```
Dictionary ={1: 'Vinni', 'name': 'and', 3: 'Vasu'}
```

```
#Deleting some of the Dictionary data

Dict = {1: 'Vinni', 'name': 'and', 3: 'Vasu'}
del(Dict[1])
print("Data after deletion Dictionary=")
print(Dict)
```

Output:

```
Data after deletion Dictionary={'name': 'and', 3: 'Vasu'}
```

Rex Galaxy Technology
"Innovating the future through technology."

Dictionary methods

dict.copy(): Returns a copy of the dictionary.

```
# copy() method
dict1 = {1: "Python", 2: "Java", 3: "Ruby", 4: "Scala"}
dict2 = dict1.copy()
print(dict2)
```

Output:

```
{1: 'Python', 2: 'Java', 3: 'Ruby', 4: 'Scala'}
```

dict.clear(): Remove all the elements from the dictionary.

```
# clear() method
dict1 = {1: "Python", 2: "Java", 3: "Ruby", 4: "Scala"}
dict1.clear()
print(dict1)
```

Output:

```
{}
```

dict.get(key, default = “None”): Returns the value of specified key.

```
# get() method
dict1 = {1: "Python", 2: "Java", 3: "Ruby", 4: "Scala"}
print(dict1.get(1))
```

Output:

```
Python
```

dict.items(): Returns a list containing a tuple for each key value pair.

```
# items() method
dict1 = {1: "Python", 2: "Java", 3: "Ruby", 4: "Scala"}
print(dict1.items())
```

Output:

```
dict_items([(1, 'Python'), (2, 'Java'), (3, 'Ruby'), (4, 'Scala')])
```

dict.keys(): Returns a list containing dictionary's keys.

```
# keys() method
dict1 = {1: "Python", 2: "Java", 3: "Ruby", 4: "Scala"}
print(dict1.keys())
```

Output:

```
dict_keys([1, 2, 3, 4])
```

pop(): Remove the element with specified key.

```
# pop() method
dict1 = {1: "Python", 2: "Java", 3: "Ruby", 4: "Scala"}
dict1.pop(4)
print(dict2)
```

Output:

```
{1: 'Python', 2: 'Java', 3: 'Ruby'}
```

dict.update(): Updates dictionary with specified key-value pairs.

```
# update() method
dict1 = {1: "Python", 2: "Java", 3: "Ruby", 4: "Scala"}
dict2.update({3: "Scala"})
print(dict2)
```

Output:

```
{1: 'Python', 2: 'Java', 3: 'Scala'}
```

dict.values(): Returns a list of all the values of dictionary.

```
# values() method
dict1 = {1: "Python", 2: "Java", 3: "Ruby", 4: "Scala"}
print(dict1.values())
```

Output:

```
dict_values(['Python', 'Java', 'Ruby', 'Scala'])
```