Dictionary in Python

Dictionary in Python

- Python's dictionaries are kind of hash table type. They consist of key-value pairs.
- Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces.
- An empty dictionary without any items is written with just two curly braces, like this: {}.
- Keys are unique within a dictionary while values may not be.
- The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples.

```
mydict = {'name' : 'RCCIIT', 'code' : 117, 'dept' : 'IT'}
print(mydict)              # Print complete dictionary
print(mydict.keys())       # Prints all the keys
print(mydict.values())     # Prints all the values
```

Dictionary in Python

Accessing Values in Tuples

Dictionary values can be assigned and accessed using square braces ([ ]).

```
dict1 = {'Name': 'RCCIIT', 'Code': 117, 'Location': 'Kolkata'}
print(dict1['Name'])          # output   RCCIIT
    print(mydict.keys())      # Prints all the keys
    print(mydict.values())    # Prints all the values
    print(mydict['name'])     # Print RCCIIT
    print(mydict['code'])     # Print 117
```

Unlike lists and tuples, there is no add(), insert(), or append()

Dictionary in Python

Updating Dictionary:

- We can add a new entry, modifying or deleting an existing entry

```
dict1 = {'Name': 'RCCIIT', 'Code': 117, 'Location': 'Kolkata'}
dict1['Code']=123          # update a value
dict1['PIN']=700015        # insert one item
print(dict1)
```

o/p: {'Name': 'RCCIIT', 'Code': 123, 'Location':‘ Kolkata', 'PIN': 700015}

Delete Dictionary Elements

- We can remove individual dictionary elements.
- We can clear the entire contents of a dictionary.
- **del** statement is used to remove an entire dictionary.

```
dict1 = {'Name': 'RCCIIT', 'Code': 117, 'Location': 'Kolkata'}
del dict1['Location']    # remove entry with key 'Location‘
dict1.clear()            # remove all entries in dict1
del dict1                # delete entire dictionary
```

Dictionary in Python

Built-in Dictionary Functions

```
dict1 = {'Name': 'RCCIIT', 'Code': 117, 'Location': 'Kolkata'}

print(len(dict1))          # output  3
```

```
dict1 = {'Name': 'RCCIIT', 'Code': 117, 'Location': 'Kolkata'}

print(type(dict1))          # output  <class 'dict'>
```

```
dict1 = {'Name': 'RCCIIT', 'Code': 117, 'Location': 'Kolkata'}

print('Dictionary is %s' % str(dict1))
```

o/p: Dictionary is {'Name': 'RCCIIT', 'Code': 117, 'Location': 'Kolkata'}

Dictionary in Python

Built-in Dictionary Methods

```
dict1 = {'Name': 'RCCIIT', 'Code': 117, 'Location': 'Kolkata'}

dict1.clear()              # remove all entries in dict1

print(dict1)               # output   { }
```

```
dict1 = {'Name': 'RCCIIT', 'Code': 117, 'Location': 'Kolkata'}

print(dict1.items())      # This method returns a list of tuple pairs.

o/p: dict_items([('Name', 'RCCIIT'), ('Code', 117), ('Location', 'Kolkata')])
```

```
dict1 = {'Name': 'RCCIIT', 'Code': 117, 'Location': 'Kolkata'}

print(dict1.keys())  # returns a list of all the keys

o/p: dict_keys(['Name', 'Code', 'Location'])
```

Dictionary in Python

Built-in Dictionary Methods

```
dict1 = {'Name': 'RCCIIT', 'Code': 117, 'Location': 'Kolkata'}

print(dict1.values())  # returns a list of all the values

o/p: dict_values(['RCCIIT', 117, 'Kolkata'])
```

```
dict1 = {'Name': 'RCCIIT', 'Code': 117, 'Location': 'Kolkata'}

dict1.pop('Code')     # remove Code entries from dict1

print(dict1)          # output  {'Name': 'RCCIIT', 'Location': 'Kolkata'}
```

```
dict1 = {'Name': 'RCCIIT', 'Code': 117, 'Location': 'Kolkata'}

dict1.popitem( )        # remove Location entries from dict1

print(dict1)            # output  {'Name': 'RCCIIT', 'Code': 117}
```

Dictionary in Python

Properties of Dictionary

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects.

There are two important points to remember about dictionary keys −

(a) More than one entry per key is not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins.

```
dict1 = {'Name': 'MAKAUT', 'Code': 117, 'Name': 'RCCIIT'}
print(dict1['Name'])        # output  RCCIIT
```

(b) Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed.

```
dict1 = {['Name']: 'MAKAUT', 'Code': 117}
o/p: TypeError: unhashable type: 'list'
```

THANK YOU