CBSE: Class: XII Computer Science

UNIT-1 CHAPTER-1: Python Dictionary

Python Dictionary

Dictionary is used to implement the key-value pair in python. The dictionary is the data type in python which can simulate the real-life data arrangement where some specific value exists for some particular key.

In other words, we can say that a dictionary is the collection of key-value pairs where the value can be any python object whereas the keys are the immutable python object, i.e., Numbers, string or tuple.

Dictionary simulates Java hash-map in python.

Creating the dictionary

The dictionary can be created by using multiple key-value pairs enclosed with the small brackets () and separated by the colon (:). The collections of the key-value pairs are enclosed within the curly braces {}.

The syntax to define the dictionary is given below.

```
1. Dict = {"Name": "Ayush","Age": 22}
```

In the above dictionary **Dict**, The keys **Name**, and **Age** are the string that is an immutable object.

Let's see an example to create a dictionary and printing its content.

```
1. Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}
2. print(type(Employee))
3. print("printing Employee data .... ")
4. print(Employee)
```

Output

```
<class 'dict'>
printing Employee data ....
{'Age': 29, 'salary': 25000, 'Name': 'John', 'Company': 'GOOGLE'}
```

Accessing the dictionary values

We have discussed how the data can be accessed in the list and tuple by using the indexing.

However, the values can be accessed in the dictionary by using the keys as keys are unique in the dictionary.

The dictionary values can be accessed in the following way.

```
1. Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}
2. print(type(Employee))
```

```
print("printing Employee data .... ")
print("Name : %s" %Employee["Name"])
print("Age : %d" %Employee["Age"])
print("Salary : %d" %Employee["salary"])
print("Company : %s" %Employee["Company"])
```

Output:

```
<class 'dict'>
printing Employee data ....
Name : John
Age : 29
Salary : 25000
Company : GOOGLE
```

Python provides us with an alternative to use the get() method to access the dictionary values. It would give the same result as given by the indexing.

Updating dictionary values

The dictionary is a mutable data type, and its values can be updated by using the specific keys.

Let's see an example to update the dictionary values.

```
Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}
print(type(Employee))
print("printing Employee data .... ")
print(Employee)
print("Enter the details of the new employee....");
Employee["Name"] = input("Name: ");
Employee["Age"] = int(input("Age: "));
Employee["salary"] = int(input("Salary: "));
Employee["Company"] = input("Company:");
print("printing the new data");
print(Employee)
```

Output:

```
<class 'dict'>
printing Employee data ....
{'Name': 'John', 'salary': 25000, 'Company': 'GOOGLE', 'Age': 29}
Enter the details of the new employee....
Name: David
Age: 19
Salary: 8900
Company:JTP
printing the new data
{'Name': 'David', 'salary': 8900, 'Company': 'JTP', 'Age': 19}
```

Deleting elements using del keyword

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

```
Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}
print(type(Employee))
print("printing Employee data .... ")
print(Employee)
print("Deleting some of the employee data")
del Employee["Name"]
del Employee["Company"]
print("printing the modified information ")
print(Employee)
print("Deleting the dictionary: Employee");
del Employee
print("Lets try to print it again ");
print(Employee)
```

Output:

```
<class 'dict'>
printing Employee data ....
{'Age': 29, 'Company': 'GOOGLE', 'Name': 'John', 'salary': 25000}
Deleting some of the employee data
printing the modified information
{'Age': 29, 'salary': 25000}
Deleting the dictionary: Employee
Lets try to print it again
Traceback (most recent call last):
  File "list.py", line 13, in <module>
    print(Employee)
NameError: name 'Employee' is not defined
```

Iterating Dictionary

A dictionary can be iterated using the for loop as given below.

Example 1

 # for loop to print all the keys of a dictionary

```
Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}
for x in Employee:
    print(x);
```

Output:

```
Name
Company
```

```
salary
Age
```

Example 2

#for loop to print all the values of the dictionary

```
1. Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}
2. for x in Employee:
3.     print(Employee[x]);
```

Output:

```
29
GOOGLE
John
25000
```

Example 3

#for loop to print the values of the dictionary by using values() method.

```
1. Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}
2. for x in Employee.values():
3.     print(x);
```

Output:

```
GOOGLE
25000
John
29
```

Example 4

#for loop to print the items of the dictionary by using items() method.

```
1. Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}
2. for x in Employee.items():
3.     print(x);
```

Output:

```
('Name', 'John')
('Age', 29)
('salary', 25000)
('Company', 'GOOGLE')
```

Properties of Dictionary keys

1. In the dictionary, we can not store multiple values for the same keys. If we pass more than one values for a single key, then the value which is last assigned is considered as the value of the key.

Consider the following example.

```
1. Employee = {"Name": "John", "Age": 29, "Salary":25000,"Company":"GOOGLE","Name":"Johnn"}
2. for x,y in Employee.items():
3.     print(x,y)
```

Output:

```
Salary 25000
Company GOOGLE
Name Johnn
Age 29
```

2. In python, the key cannot be any mutable object. We can use numbers, strings, or tuple as the key but we can not use any mutable object like the list as the key in the dictionary.

Consider the following example.

```
1. Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE",[100,201,301]:"Department ID"}
2. for x,y in Employee.items():
3.     print(x,y)
```

Output:

```
Traceback (most recent call last):
  File "list.py", line 1, in
    Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE",[100,201,301]:"Department
ID"}
TypeError: unhashable type: 'list'
```

Built-in Dictionary functions

The built-in python dictionary methods along with the description are given below.

SN	Function	Description
1	cmp(dict1, dict2)	It compares the items of both the dictionary and returns true if the first dictionary values are greater than the second dictionary, otherwise it returns false.
2	len(dict)	It is used to calculate the length of the dictionary.
3	str(dict)	It converts the dictionary into the printable string representation.
4	type(variable)	It is used to print the type of the passed variable.

Built-in Dictionary methods

The built-in python dictionary methods along with the description are given below.

SN	Method	Description

1	dic.clear()	It is used to delete all the items of the dictionary.
2	dict.copy()	It returns a shallow copy of the dictionary.
3	dict.fromkeys(iterable, value = None, /)	Create a new dictionary from the iterable with the values equal to value.
4	dict.get(key, default = "None")	It is used to get the value specified for the passed key.
5	dict.has_key(key)	It returns true if the dictionary contains the specified key.
6	dict.items()	It returns all the key-value pairs as a tuple.
7	dict.keys()	It returns all the keys of the dictionary.
8	dict.setdefault(key,default= "None")	It is used to set the key to the default value if the key is not specified in the dictionary
9	dict.update(dict2)	It updates the dictionary by adding the key-value pair of dict2 to this dictionary.
10	dict.values()	It returns all the values of the dictionary.
11	len()	It returns the total no of items of the dictionary
12	popItem()	It removes specified item from dictionary.
13	pop()	It removes specified/last key:value pair from dictionary and also return the corresponding value.
14	count()	It is used to count total
15	index()	