Dictionary

Python Dictionary is used to store the data in a key-value pair format.

The dictionary is defined into element Keys and values.

- Keys must be a single element
- Value can be any type such as list, tuple, integer, etc.

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 (:)

Dictionaries are order collections of unique values stored in (Key-Value)

- **ordered**: The items in dictionaries are stored without any index value, which is typically a range of numbers. They are stored as Key-Value pairs, and the keys are their index, which will not be in any sequence.
- **Unique:** As mentioned above, each value has a Key; the Keys in Dictionaries should be unique. If we store any value with a Key that already exists, then the most recent value will replace the old value.
- **Mutable:** The dictionaries are collections that are changeable, which implies that we can add or remove items after the creation.

Creating a dictionary

- **Using curly brackets**: The dictionaries are created by enclosing the comma-separated Key: Value pairs inside the {} curly brackets. The colon ':' is used to separate the key and value in a pair.
- **Using dict() constructor**: Create a dictionary by passing the comma-separated key: value pairs inside the dict().
- **Using sequence** having each item as a pair (key-value)

create a dictionary using {}

```
s1={"name":"jaydip","number":"897867","course":"web design"}
print(s1)
```

create a dictionary using dict()

```
s1=dict({"name":"jaydip","number":"897867","course":"web design"})
print(s1)
```

create a dictionary from sequence having each item as a pair

```
s1=dict([("name","jaydip"),("number","897867"),("course","web design")])
print(s1)
```

create dictionary with value as a list

```
s1={"name":"jaydip","number":[1234,4556,678]}
print(s1)
```

- A dictionary value can be of any type, and duplicates are allowed in that.
- Keys in the dictionary must be unique and of immutable types like string, numbers, or tuples.

Accessing elements of a dictionary

1. Retrieve value using the key name inside the `[]` square brackets
2. Retrieve value by passing key name as a parameter to the `get()` method of a dictionary.

```
s1={"name":"jaydip","number":[1234,4556,678]}
print(s1["name"])
```

```
s1={"name":"jaydip","number":[1234,4556,678]}
print(s1.get('name'))
```

Dictionary Method

Method	Description
keys()	Returns the list of all keys present in the dictionary.
values()	Returns the list of all values present in the dictionary
items()	Returns all the items present in the dictionary. Each item will be inside a tuple as a key-value pair.

```
s1={"name":"jaydip","number":[1234,4556,678]}
print(s1.keys())
print(s1.values())

# get all key-pair item

print(s1.items())
```

Iterating a dictionary

```
s1={"name":"jaydip","number":[1234,4556,678]}

for i in s1:
    print(i,s1[i])
```

using items() method

```
s1={"name":"jaydip","number":[1234,4556,678],"course":"web design"}


for i in s1.items():
    print(i[0],i[1])
```

Adding Items to the dictionary

- **Using key-value assignment:** Using a simple assignment statement where value can be assigned directly to the new key.
- **Using update() Method:** In this method, the item passed inside the update() method will be inserted into the dictionary. The item can be another dictionary or any iterable like a tuple of key-value pairs.

```
s1={"name":"jaydip","number":[1234,4556,678],"course":"web design"}

s1['fees']=35000
s1.update({"job":"yes"})

print(s1)
```

Note: We can also add more than one key using the update() method.

```
s1={"name":"jaydip","number":[1234,4556,678],"course":"web design"}

s1.update({"weight": 50, "height": 6})

print(s1)
```

# pass new keys as as list of tuple

```
s1={"name":"jaydip","number":[1234,4556,678],"course":"web design"}

s1.update([("weight", 50), ("height", 6)])

print(s1)
```

Update the state name using update()

```
s1={"name":"jaydip","number":3245546,"course":"web design"}

s1.update([("weight", 50), ("height", 6)])

s1.update({'state':'raj'})

for i,j in s1.items():
    print(i,":",j)
```

Removing items from the dictionary

Method	Description
pop(key[,d])	Return and removes the item with the key and return its value. If the key is not found, it raises KeyError.
popitem()	Return and removes the last inserted item from the dictionary. If the dictionary is empty, it raises KeyError.
del key	The del keyword will delete the item with the key that is passed
clear()	Removes all items from the dictionary. Empty the dictionary
del dict_name	Delete the entire dictionary

```
s1={"name":"jaydip","number":3245546,"course":"web design"}

# using pop('key')
s1.pop('name')
print(s1)

s1.update({'fees':12234})
print(s1)

# using popitem(),it removes last inserted item

s1.popitem()

print(s1)

del s1['number']
```

```
print(s1)

s1.clear()
print(s1)

del s1
print(s1)
```

Checking if a key exists

In order to check whether a particular key exists in a dictionary, we can use the `keys()` method

```
s1={"name":"jaydip","number":3245546,"course":"python",'fees':50000}

keyname="name"

if keyname in s1.keys():
    print('name is ',s1[keyname])
else:
    print('key not found')
```

Join two dictionary

We can add two dictionaries using the `update()` method or unpacking arbitrary keywords operator `**`.

Using UPDATE()

```
s1={"name":"jaydip","number":3245546,"course":"python",'fees':50000}
s2={'education':'bca','result':'pass'}

s1.update(s2)
print(s1)
```

Using `**kwargs` to unpack

```
s1={"name":"jaydip"}
s2={'education':'bca'}
s3={'result':'pass'}

s={**s1,**s2,**s3}
print(s)
```

Join two dictionaries having few items in common

Note: One thing to note here is that if both the dictionaries have a common key then the first dictionary value will be overridden with the second dictionary value.

```
s1={"name":"jaydip",'number':54656}
s2={'education':'bca','number':60000}

s1.update(s2)
print(s1)
```

Copy a Dictionary

We can create a copy of a dictionary using the following two ways

- Using `copy()` method.
- Using the `dict()` constructor

```
s1={"name":"jaydip","number":3245546,"course":"python",'fees':50000}

s2=s1.copy()

print(s2)
```

also used = operator to copy

Sort dictionary

The built-in method sorted() will sort the keys in the dictionary and returns a sorted list.

```
s1={"name":"jaydip","number":3245546,"course":"python",'fees':50000}

# sorting dictionary by keys

print(sorted(s1.items()))
```

In case we want to sort the values we can first get the values using the `values()` and then sort them.

```
s1={"name":50,"number":30,"course":20,'fees':10}

# sorting dictionary by keys

print(sorted(s1.values()))
```