Python Dictionary

Dictionary in Python is a collection of keys values, used to store data values like a map, which, unlike other data types which hold only a single value as an element.

Example of Dictionary in Python

Dictionary holds **key:value** pair. Key-Value is provided in the dictionary to make it more optimized.

```
Dict = {1: 'Geeks', 2: 'For', 3: 'Geeks'}

print(Dict)
```

```
# Creating a Dictionary

Dict = {1: 'Geeks', 2: 'For', 3: 'Geeks'}

print("\nDictionary with the use of Integer Keys: ")

print(Dict)


# Creating a Dictionary with Mixed keys

Dict = {'Name': 'Geeks', 1: [1, 2, 3, 4]}

print("\nDictionary with the use of Mixed Keys: ")

print(Dict)
```

```
# Creating a Nested Dictionary as shown in the below image

Dict = {1: 'Geeks', 2: 'For',3: {'A': 'Welcome', 'B': 'To', 'C': 'Geeks'}}

print(Dict)
```

Adding elements to a Dictionary

Addition of elements can be done in multiple ways. One value at a time can be added to a Dictionary by defining value along with the key e.g. Dict[Key] = 'Value'. Updating an existing value in a Dictionary can be done by using the built-in **update()** method. Nested key values can also be added to an existing Dictionary.

Note- While adding a value, if the key-value already exists, the value gets updated otherwise a new Key with the value is added to the Dictionary.

```
# Creating an empty Dictionary

Dict = {}

print("Empty Dictionary: ")

print(Dict)

# Adding elements one at a time

Dict[0] = 'Disha'

Dict[2] = 'For'

Dict[3] = 1

print("\nDictionary after adding 3 elements: ")

print(Dict)


# Adding set of values to a single Key

Dict['Value_set'] = 2, 3, 4

print("\nDictionary after adding 3 elements: ")

print(Dict)


# Updating existing Key's Value

Dict[2] = 'Welcome'

print("\nUpdated key value: ")

print(Dict)


# Adding Nested Key value to Dictionary

Dict[5] = {'Nested': {'1': 'Like', '2': Disha}}

print("\nAdding a Nested Key: ")

print(Dict)
```

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: 'Disha', 'name': 'Computer', 3: 'Institute'}

# accessing a element using key

print("Accessing a element using key:")

print(Dict['name'])

# accessing a element using key

print("Accessing a element using key:")

print(Dict[1])
```

```
# Creating a Dictionary

Dict = {1: 'Geeks', 'name': 'For', 3: 'Geeks'}

# accessing a element using get()method

print("Accessing a element using get:")

print(Dict.get(3))
```

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: 'Geeks'},'Dict2': {'Name': 'For'}}

# Accessing element using key

print(Dict['Dict1'])

print(Dict['Dict1'][1])

print(Dict['Dict2']['Name'])
```

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: 'Geeks', 'name': 'For', 3: 'Geeks'}
print("Dictionary =")
print(Dict)
#Deleting some of the Dictionar data
del(Dict[1])
print("Data after deletion Dictionary=")
print(Dict)
```

```
# demo for all dictionary methods
dict1 = {1: "Python", 2: "Java", 3: "Ruby", 4: "Scala"}


# copy() method
dict2 = dict1.copy()
print(dict2)
# clear() method
dict1.clear()
print(dict1)
# get() method
print(dict2.get(1))
# items() method
print(dict2.items())
```

```
# keys() method
print(dict2.keys())

# pop() method
dict2.pop(4)
print(dict2)

# popitem() method
dict2.popitem()
print(dict2)

# update() method
dict2.update({3: "Scala"})
print(dict2)

# values() method
print(dict2.values())
```