**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.

* Python3

|  |
| --- |
| Dict = {1: 'Geeks', 2: 'For', 3: 'Geeks'}  print(Dict) |

**Output:**

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

**Creating a Dictionary**

In [Python](https://www.geeksforgeeks.org/python-programming-language/), a dictionary can be created by placing a sequence of elements within curly **{}** braces, separated by ‘comma’. Dictionary holds pairs of values, one being the Key and the other corresponding pair element being its **Key:value**. Values in a dictionary can be of any data type and can be duplicated, whereas keys can’t be repeated and must be *immutable*.

**Note –**Dictionary keys are case sensitive, the same name but different cases of Key will be treated distinctly.

* Python3

|  |
| --- |
| # Creating a Dictionary  # with Integer Keys  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) |

**Output:**

Dictionary with the use of Integer Keys:

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

Dictionary with the use of Mixed Keys:

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

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

* Python3

|  |
| --- |
| # Creating an empty Dictionary  Dict = {}  print("Empty Dictionary: ")  print(Dict)    # Creating a Dictionary  # with dict() method  Dict = dict({1: 'Geeks', 2: 'For', 3: 'Geeks'})  print("\nDictionary with the use of dict(): ")  print(Dict)    # Creating a Dictionary  # with each item as a Pair  Dict = dict([(1, 'Geeks'), (2, 'For')])  print("\nDictionary with each item as a pair: ")  print(Dict) |

**Output:**

Empty Dictionary:

{}

Dictionary with the use of dict():

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

Dictionary with each item as a pair:

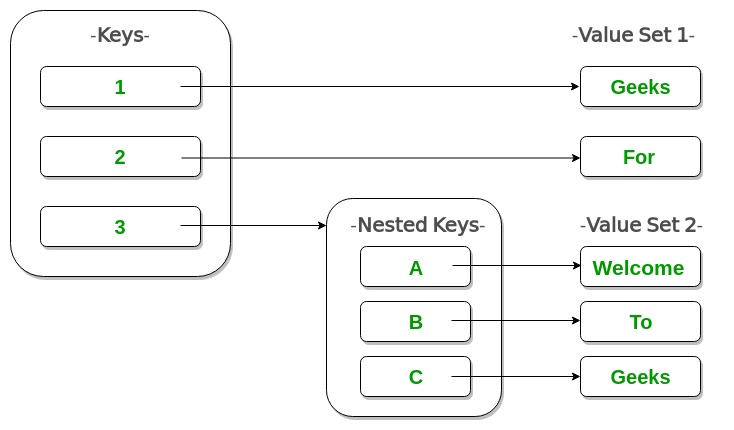
{1: 'Geeks', 2: 'For'}

**Complexities for Creating a Dictionary:**

***Time complexity:****O(len(dict))*

***Space complexity:****O(n)*

**Nested Dictionary**



* Python3

|  |
| --- |
| # Creating a Nested Dictionary  # as shown in the below image  Dict = {1: 'Geeks', 2: 'For',          3: {'A': 'Welcome', 'B': 'To', 'C': 'Geeks'}}    print(Dict) |

**Output:**

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

**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.

* Python3

|  |
| --- |
| # Creating an empty Dictionary  Dict = {}  print("Empty Dictionary: ")  print(Dict)    # Adding elements one at a time  Dict[0] = 'Geeks'  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': 'Life', '2': 'Geeks'}}  print("\nAdding a Nested Key: ")  print(Dict) |

**Output:**

Empty Dictionary:

{}

Dictionary after adding 3 elements:

{0: 'Geeks', 2: 'For', 3: 1}

Dictionary after adding 3 elements:

{0: 'Geeks', 2: 'For', 3: 1, 'Value\_set': (2, 3, 4)}

Updated key value:

{0: 'Geeks', 2: 'Welcome', 3: 1, 'Value\_set': (2, 3, 4)}

Adding a Nested Key:

{0: 'Geeks', 2: 'Welcome', 3: 1, 'Value\_set': (2, 3, 4), 5:

{'Nested': {'1': 'Life', '2': 'Geeks'}}}

**Complexities for Adding elements in a Dictionary:**

**Time complexity:** O(1)/O(n)

**Space complexity:** O(1)

**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.

* Python3

|  |
| --- |
| # Python program to demonstrate  # accessing a element from a Dictionary    # Creating a Dictionary  Dict = {1: 'Geeks', 'name': 'For', 3: 'Geeks'}    # 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]) |

**Output:**

Accessing a element using key:

For

Accessing a element using key:

Geeks

There is also a method called [**get()**](https://www.geeksforgeeks.org/get-method-dictionaries-python/) that will also help in accessing the element from a dictionary.This method accepts key as argument and returns the value.

**Complexities for Accessing elements in a Dictionary:**

**Time complexity:** O(1)

**Space complexity:** O(1)

* Python3

|  |
| --- |
| # 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)) |

**Output:**

Accessing a element using get:

Geeks

**Accessing an element of a nested dictionary**

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

* Python3

|  |
| --- |
| # 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']) |

**Output:**

{1: 'Geeks'}

Geeks

For

**Dictionary methods**

* [**clear()**](https://www.geeksforgeeks.org/python-dictionary-clear/)**–**Remove all the elements from the dictionary
* [**copy()**](https://www.geeksforgeeks.org/python-dictionary-copy/)**–**Returns a copy of the dictionary
* [**get()**](https://www.geeksforgeeks.org/get-method-dictionaries-python/)**–**Returns the value of specified key
* [**items()**](https://www.geeksforgeeks.org/python-dictionary-items-method/)**–**Returns a list containing a tuple for each key value pair
* [**keys()**](https://www.geeksforgeeks.org/python-dictionary-keys-method/)**–**Returns a list containing dictionary’s keys
* [**pop()**](https://www.geeksforgeeks.org/python-dictionary-pop-method/)**–** Remove the element with specified key
* [**popitem()**](https://www.geeksforgeeks.org/python-dictionary-popitem-method/)**–**Removes the last inserted key-value pair
* [**update()**](https://www.geeksforgeeks.org/python-dictionary-update-method/)**–**Updates dictionary with specified key-value pairs
* [**values()**](https://www.geeksforgeeks.org/python-dictionary-values/)**–** Returns a list of all the values of dictionary
* Python3

|  |
| --- |
| # 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()) |

**Output:**

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

{}

Python

dict\_items([(1, 'Python'), (2, 'Java'), (3, 'Ruby'), (4, 'Scala')])

dict\_keys([1, 2, 3, 4])

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

{1: 'Python', 2: 'Java'}

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

dict\_values(['Python', 'Java', 'Scala'])