Dictionary with the use of Integer Keys:

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

Dictionary with the use of Mixed Keys:

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

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

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| # 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:

{}

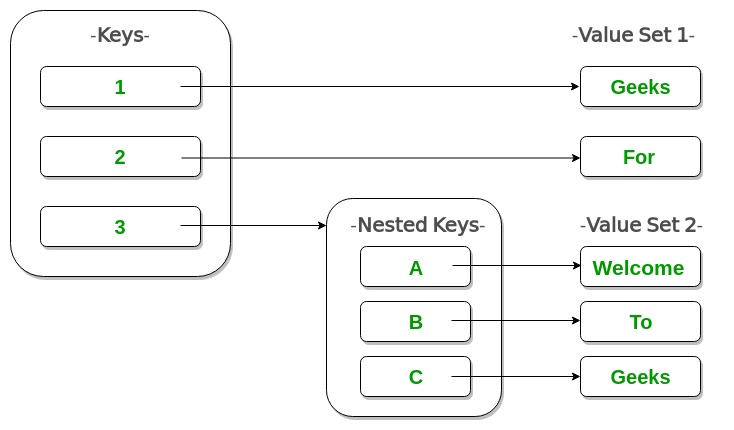
Dictionary with the use of dict():

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

Dictionary with each item as a pair:

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

**Nested Dictionary:**



filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| # 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**

In Python 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.

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| # 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, 5: {'Nested': {'1': 'Life', '2': 'Geeks'}}, 'Value\_set': (2, 3, 4)}

**Accessing elements from a Dictionary**

In order to access the items of a dictionary refer to its key name.Key can be used inside square brackets.

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| # 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 acessing the element from a dictionary.

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| # 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 element of a nested dictionary**

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

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| # 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

**Removing Elements from Dictionary**

**Using del keyword**

In Python Dictionary, deletion of keys can be done by using the **del**keyword. Using del keyword, specific values from a dictionary as well as whole dictionary can be deleted. Items in a Nested dictionary can also be deleted by using del keyword and providing specific nested key and particular key to be deleted from that nested Dictionary.

**Note- del Dict** will delete the entire dictionary and hence printing it after deletion will raise an Error.

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| # Initial Dictionary  Dict = { 5 : 'Welcome', 6 : 'To', 7 : 'Geeks',          'A' : {1 : 'Geeks', 2 : 'For', 3 : 'Geeks'},          'B' : {1 : 'Geeks', 2 : 'Life'}}  print("Initial Dictionary: ")  print(Dict)    # Deleting a Key value  del Dict[6]  print("\nDeleting a specific key: ")  print(Dict)    # Deleting a Key from  # Nested Dictionary  del Dict['A'][2]  print("\nDeleting a key from Nested Dictionary: ")  print(Dict) |

**Output:**

Initial Dictionary:

{'A': {1: 'Geeks', 2: 'For', 3: 'Geeks'}, 'B': {1: 'Geeks', 2: 'Life'}, 5: 'Welcome', 6: 'To', 7: 'Geeks'}

Deleting a specific key:

{'A': {1: 'Geeks', 2: 'For', 3: 'Geeks'}, 'B': {1: 'Geeks', 2: 'Life'}, 5: 'Welcome', 7: 'Geeks'}

Deleting a key from Nested Dictionary:

{'A': {1: 'Geeks', 3: 'Geeks'}, 'B': {1: 'Geeks', 2: 'Life'}, 5: 'Welcome', 7: 'Geeks'}

**Using pop() method**

[Pop(](https://www.geeksforgeeks.org/python-dictionary-pop-method/)) method is used to return and delete the value of the key specified.

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| # Creating a Dictionary  Dict = {1: 'Geeks', 'name': 'For', 3: 'Geeks'}    # Deleting a key  # using pop() method  pop\_ele = Dict.pop(1)  print('\nDictionary after deletion: ' + str(Dict))  print('Value associated to poped key is: ' + str(pop\_ele)) |

**Output:**

Dictionary after deletion: {3: 'Geeks', 'name': 'For'}

Value associated to poped key is: Geeks

**Using popitem() method**

The popitem() returns and removes an arbitrary element (key, value) pair from the dictionary.

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| # Creating Dictionary  Dict = {1: 'Geeks', 'name': 'For', 3: 'Geeks'}    # Deleting an arbitrary key  # using popitem() function  pop\_ele = Dict.popitem()  print("\nDictionary after deletion: " + str(Dict))  print("The arbitrary pair returned is: " + str(pop\_ele)) |

**Output:**

Dictionary after deletion: {3: 'Geeks', 'name': 'For'}

The arbitrary pair returned is: (1, 'Geeks')

**Using clear() method**

All the items from a dictionary can be deleted at once by using **clear()** method.

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| # Creating a Dictionary  Dict = {1: 'Geeks', 'name': 'For', 3: 'Geeks'}      # Deleting entire Dictionary  Dict.clear()  print("\nDeleting Entire Dictionary: ")  print(Dict) |

**Output:**

Deleting Entire Dictionary:

{}

**Dictionary Methods**

|  |  |
| --- | --- |
| **METHODS** | **DESCRIPTION** |
| [copy()](https://www.geeksforgeeks.org/python-dictionary-copy/) | They copy() method returns a shallow copy of the dictionary. |
| [clear()](https://www.geeksforgeeks.org/python-dictionary-clear/) | The clear() method removes all items from the dictionary. |
| [pop()](https://www.geeksforgeeks.org/python-dictionary-pop-method/) | Removes and returns an element from a dictionary having the given key. |
| [popitem()](https://www.geeksforgeeks.org/python-dictionary-popitem-method/) | Removes the arbitrary key-value pair from the dictionary and returns it as tuple. |
| [get()](https://www.geeksforgeeks.org/get-method-dictionaries-python/) | It is a conventional method to access a value for a key. |
| [dictionary\_name.values()](https://www.geeksforgeeks.org/python-dictionary-values/) | returns a list of all the values available in a given dictionary. |
| str() | Produces a printable string representation of a dictionary. |
| [update()](https://www.geeksforgeeks.org/python-dictionary-update-method/) | Adds dictionary dict2’s key-values pairs to dict |
| [setdefault()](https://www.geeksforgeeks.org/python-dictionary-setdefault-method/) | Set dict[key]=default if key is not already in dict |
| [keys()](https://www.geeksforgeeks.org/python-dictionary-keys-method/) | Returns list of dictionary dict’s keys |
| [items()](https://www.geeksforgeeks.org/python-dictionary-items-method/) | Returns a list of dict’s (key, value) tuple pairs |
| [has\_key()](https://www.geeksforgeeks.org/python-dictionary-has_key/) | Returns true if key in dictionary dict, false otherwise |
| [fromkeys()](https://www.geeksforgeeks.org/python-dictionary-fromkeys-method/) | Create a new dictionary with keys from seq and values set to value. |
| [type()](https://www.geeksforgeeks.org/python-type-function/) | Returns the type of the passed variable. |
| [cmp()](https://www.geeksforgeeks.org/dictionary-methods-in-python-set-1-cmp-len-items/) | Compares elements of both dict. |