**Regular Dictionary vs Ordered Dictionary in Python**

[Dictionary](https://www.geeksforgeeks.org/python-dictionary/) in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds key:value pair. Key-value is provided in the dictionary to make it more optimized. A regular dictionary type does not track the insertion order of the (key, value) pairs and thus iterates through the keys based on how they are stored in the hash table which in turn is based on random values so as to reduce collisions.  
In contrast to this Python provides the [OrderedDict](https://www.geeksforgeeks.org/ordereddict-in-python/) type which remembers the insertion order of (key, value) pairs in the dictionary and thus preserves the order. OrderedDict consumes more memory than a regular dictionary in Python because of the underlying Doubly LinkedList implementation to preserving the order.

**Example:**

|  |
| --- |
| # A Python program to demonstrate  # the difference between regular  # and ordered dictionary.      import collections      # Creating a regular dictionary  print('Regular dictionary:')  d = {chr(k):k for k in range(ord('a'), ord('g'))}    for k, v in d.items():      print(k, v)    # Creating an Ordered dictionary  print('\nOrderedDict:')  d = collections.OrderedDict()  [d.setdefault(chr(k), k) for k in range(ord('a'), ord('g'))]    for k, v in d.items():      print(k, v) |

**Output :**

Regular dictionary:

('a', 97)

('c', 99)

('b', 98)

('e', 101)

('d', 100)

('f', 102)

OrderedDict:

('a', 97)

('b', 98)

('c', 99)

('d', 100)

('e', 101)

('f', 102)

**Note:** Starting from Python 3.7, insertion order of Python dictionaries is guaranteed.

**Deletion and Re-Inserting:**   
Deleting and re-inserting the same key will push it to the back as OrderedDict however maintains the order of insertion.

**Example:**

|  |
| --- |
| # A Python program to demonstrate  # working of deletion and re-insertion in  # regular and OrderedDict      from collections import OrderedDict    print("Before deleting:\n")    d = {}  print("Regular dictionary:")  d['a'] = 1  d['b'] = 2  d['c'] = 3  d['d'] = 4  for key, value in d.items():      print(key, value)    od = OrderedDict()  print("\nOrdered dictionary:")  od['a'] = 1  od['b'] = 2  od['c'] = 3  od['d'] = 4  for key, value in od.items():      print(key, value)      print("\nAfter deleting:\n")      print("Regular dictionary:")  d.pop('c')  for key, value in d.items():      print(key, value)    print("\nOrdered dictionary:")  od.pop('c')  for key, value in od.items():      print(key, value)      print("\nAfter re-inserting:\n")      print("Regular dictionary:")  d['c'] = 3  for key, value in d.items():      print(key, value)    print("\nOrdered dictionary:")  od['c'] = 3  for key, value in od.items():      print(key, value) |

**Output:**

Before deleting:

Regular dictionary:

('a', 1)

('c', 3)

('b', 2)

('d', 4)

Ordered dictionary:

('a', 1)

('b', 2)

('c', 3)

('d', 4)

After deleting:

Regular dictionary:

('a', 1)

('b', 2)

('d', 4)

Ordered dictionary:

('a', 1)

('b', 2)

('d', 4)

After re-inserting:

Regular dictionary:

('a', 1)

('c', 3)

('b', 2)

('d', 4)

Ordered dictionary:

('a', 1)

('b', 2)

('d', 4)

('c', 3)