**Collections Module Notes**

**Syed Mansoor ul Hassan Bukhari**

**Collections Module in Python**

The collections module in Python provides alternatives to Python’s general-purpose built-in containers like dict, list, set, and tuple. Here are some of the key components:

1. **namedtuple()**
   * Factory function for creating tuple subclasses with named fields.
   * Example:

from collections import namedtuple

Point = namedtuple('Point', ['x', 'y'])

p = Point(11, y=22)

print(p.x, p.y) # Output: 11 22

1. **deque**

* List-like container with fast appends and pops on either end.
* Example:

from collections import deque

d = deque(['a', 'b', 'c'])

d.append('d')

d.appendleft('z')

print(d) # Output: deque(['z', 'a', 'b', 'c', 'd'])

1. **ChainMap**
   * Dictionary-like class for creating a single view of multiple mappings.
   * Example:

from collections import ChainMap

dict1 = {'a': 1, 'b': 2}

dict2 = {'b': 3, 'c': 4}

chain = ChainMap(dict1, dict2)

print(chain['b']) # Output: 2

1. **Counter**
   * Dictionary subclass for counting hashable objects.
   * Example:

from collections import Counter

cnt = Counter(['a', 'b', 'c', 'a', 'b', 'b'])

print(cnt) # Output: Counter({'b': 3, 'a': 2, 'c': 1})

1. **OrderedDict**

* Dictionary subclass that remembers the order entries were added.
* Example:

from collections import OrderedDict

od = OrderedDict()

od['a'] = 1

od['b'] = 2

od['c'] = 3

print(od) # Output: OrderedDict([('a', 1), ('b', 2), ('c', 3)])

1. **defaultdict**
   * Dictionary subclass that calls a factory function to supply missing values.
   * Example:

from collections import defaultdict

dd = defaultdict(int)

dd['a'] = 1

dd['b'] += 1

print(dd) # Output: defaultdict(<class 'int'>, {'a': 1, 'b': 1})

These components provide more specialized data structures and functionalities, making it easier to write efficient and readable code.