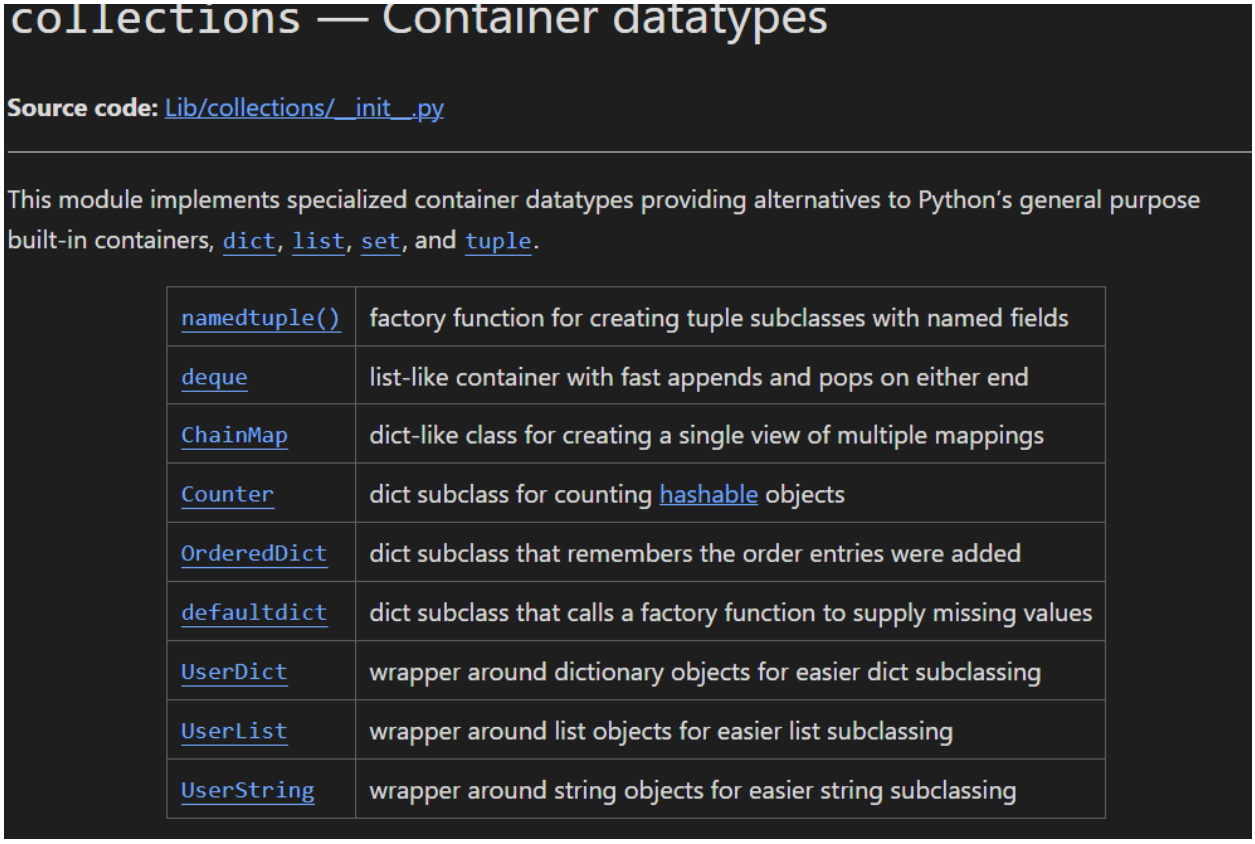
# Python Notes For L&T Crop

## Module 10: Collections

In Python, "Collections" refers to the collections module, which provides specialized container datatypes beyond the built-in types like list, tuple, dict, and set. These specialized data types provide alternatives to the general-purpose containers and are designed to improve efficiency and readability in various scenarios. Here's a summary of the primary collections provided by the collections module.



collections — Container datatypes

**Source code:** Lib/collections/__init__.py

This module implements specialized container datatypes providing alternatives to Python's general purpose built-in containers, `dict`, `list`, `set`, and `tuple`.

| | |
|---|---|
| `namedtuple()` | factory function for creating tuple subclasses with named fields |
| `deque` | list-like container with fast appends and pops on either end |
| `ChainMap` | dict-like class for creating a single view of multiple mappings |
| `Counter` | dict subclass for counting hashable objects |
| `OrderedDict` | dict subclass that remembers the order entries were added |
| `defaultdict` | dict subclass that calls a factory function to supply missing values |
| `UserDict` | wrapper around dictionary objects for easier dict subclassing |
| `UserList` | wrapper around list objects for easier list subclassing |
| `UserString` | wrapper around string objects for easier string subclassing |

Source image : collections — Container datatypes — Python 3.12.3 documentation

## Collection

### 1.namedtuple in Python

The namedtuple is a factory function provided by the collections module to create tuple subclasses with named fields. This provides a way to create simple, lightweight, and immutable objects where the fields can be accessed by name rather than by index, making the code more readable and self-documenting.

# Python Notes For L&T Crop

Key Features
Immutability: Like tuples, namedtuple instances are immutable.
Named Fields: Fields can be accessed using dot notation, which makes the code more readable.
Lightweight: It is a memory-efficient alternative to defining a class manually.
Creating a namedtuple
To create a namedtuple, you use the namedtuple factory function from the collections module.

Example : go to vs code

2.**A deque**
(pronounced "deck") is a double-ended queue that allows you to append and pop elements from either the left or the right side with O(1) time complexity. It is part of the collections module in Python.

Key Operations:

append(x): Add x to the right end.
appendleft(x): Add x to the left end.
pop(): Remove and return an element from the right end.
popleft(): Remove and return an element from the left end.
extend(iterable): Extend the right end by appending elements from the iterable.
extendleft(iterable): Extend the left end by appending elements from the iterable (in reverse order).
rotate(n): Rotate the deque n steps to the right. If n is negative, rotate to the left.

Example :- go to vs code

3. ChainMap
ChainMap groups multiple dictionaries (or other mappings) into a single view, allowing you to treat them as a unified dictionary. It searches through these mappings one by one until it finds the key.

**Priority Order**: The order of the dictionaries in the ChainMap matters; keys found in earlier dictionaries will take precedence over keys in later dictionaries.

**Mutability**: Modifications to the ChainMap will affect the first dictionary in the list. This means that updates, insertions, or deletions only apply to the first dictionary.

Example :- go to vs code

**4.`Counter`:**
The Counter class in Python is part of the collections module. It is a subclass of the dictionary object and is used to count hashable objects. Essentially, Counter is a specialized dictionary for counting items.

Key Features of Counter:

- Initialization: Can be initialized with a sequence, dictionary, or another Counter object.

Nagendra Yadav

- Elements Method: Returns an iterator over elements repeating each as many times as its count.
- Most Common Method: Returns a list of the n most common elements and their counts from the most common to the least. Arithmetic and Set Operations: Supports addition, subtraction, intersection, and union.
- Example :- go to vs code

### 5. OrderedDict

The OrderedDict class in Python is part of the collections module. It is a dictionary subclass that remembers the order in which its contents are added. This can be particularly useful for tasks where the order of items is important.

Key Features of OrderedDict:

- Preserves Order: Remembers the order of insertion.
- Methods: Supports methods like move_to_end and popitem which are specific to OrderedDict.
- Compatibility: Can be used in place of a regular dictionary for most applications.
- Example :- go to vs code

### 6.defaultdict

defaultdict is a subclass of the built-in dict class in Python. It is part of the collections module and provides a default value for a nonexistent key. When accessing a key that doesn't exist in the dictionary, instead of raising a KeyError, a default value is provided, and the key is automatically added to the dictionary with this default value.

The defaultdict requires a factory function to specify the default value for nonexistent keys. This factory function is called without arguments to produce the default value.

### 7.UserDict

The UserDict class acts as a wrapper around the standard dictionary, allowing you to create dictionary-like objects that can be customized by subclassing. Here are some key points:

Inheritance: UserDict is a base class for creating a custom dictionary. It is a simpler way to create a custom dictionary by overriding its methods without affecting the original dictionary implementation.
Customization: By subclassing UserDict, you can override dictionary methods such as __getitem__, __setitem__, __delitem__, and others to modify their behavior.
Compatibility: Since it is a wrapper around the built-in dictionary, it provides all standard dictionary methods and behaves similarly.
Attribute Access: The actual dictionary is stored in the data attribute, which can be accessed or modified directly if needed.

### 8. UserString

**UserString** class in Python's collections module is a wrapper class that acts as a proxy for string objects, providing a way to create string-like objects with extended or modified behavior.

- Inheritance: UserString is a subclass of MutableSequence, a built-in class for sequences that can be modified. This means it inherits all the behaviors and methods of MutableSequence, including methods like __getitem__, __setitem__, __delitem__, __len__, etc.
- Composition: UserString wraps around a string object (usually an instance of str). The actual string data is stored within the data attribute of the UserString instance

# Python Notes For L&T Crop

Nagendra Yadav