Data Structures are a way of organizing data so that it can be accessed more efficiently depending upon the situation. Data Structures are fundamentals of any programming language around which a program is built. Python helps to learn the fundamental of these data structures in a simpler way as compared to other programming languages.

In this article, we will discuss the Data Structures in the Python Programming Language and how they are related to some specific Python Data Types. We will discuss all the in-built data structures like list tuples, dictionaries, etc. as well as some advanced data structures like trees, graphs, etc.

Lists

Python Lists are just like the arrays, declared in other languages which is an ordered collection of data. It is very flexible as the items in a list do not need to be of the same type.

The implementation of Python List is similar to Vectors in C++ or ArrayList in JAVA. The costly operation is inserting or deleting the element from the beginning of the List as all the elements are needed to be shifted. Insertion and deletion at the end of the list can also become costly in the case where the preallocated memory becomes full.

We can create a list in python as shown below.
Example: Creating Python List

```
List = [1, 2,  3, "GFG", 2.3]
print(List)
```

List elements can be accessed by the assigned index. In python starting index of the list, sequence is 0 and the ending index is (if N elements are there) N-1.

Dictionary

Python dictionary is like hash tables in any other language with the time complexity of O(1). It is an unordered collection of data values, used to store data values like a map, which, unlike other Data Types that hold only a single value as an element, Dictionary holds the key:value pair. Key-value is provided in the dictionary to make it more optimized.

Indexing of Python Dictionary is done with the help of keys. These are of any hashable type i.e. an object whose can never change like strings, numbers, tuples, etc. We can create a dictionary by using curly braces ({}) or dictionary comprehension.

Tuple

Python Tuple is a collection of Python objects much like a list but Tuples are immutable in nature i.e. the elements in the tuple cannot be added or removed once created. Just like a List, a Tuple can also contain elements of various types.

In Python, tuples are created by placing a sequence of values separated by ‘comma’ with or without the use of parentheses for grouping of the data sequence.

Note: Tuples can also be created with a single element, but it is a bit tricky. Having one element in the parentheses is not sufficient, there must be a trailing ‘comma’ to make it a tuple.

Set

Python Set is an unordered collection of data that is mutable and does not allow any duplicate element. Sets are basically used to include membership testing and eliminating duplicate entries. The data structure used in this is Hashing, a popular technique to perform insertion, deletion, and traversal in O(1) on average.

If Multiple values are present at the same index position, then the value is appended to that index position, to form a Linked List. In, CPython Sets are implemented using a dictionary with dummy variables, where key beings the members set with greater optimizations to the time complexity.

Frozen Sets

Frozen sets in Python are immutable objects that only support methods and operators that produce a result without affecting the frozen set or sets to which they are applied. While elements of a set can be modified at any time, elements of the frozen set remain the same after creation.

If no parameters are passed, it returns an empty frozenset.

String

Python Strings are arrays of bytes representing Unicode characters. In simpler terms, a string is an immutable array of characters. Python does not have a character data type, a single character is simply a string with a length of 1.

Note: As strings are immutable, modifying a string will result in creating a new copy.

Bytearray

Python Bytearray gives a mutable sequence of integers in the range 0 <= x < 256.

Till now we have studied all the data structures that come built-in into core Python. Now let dive more deep into Python and see the collections module that provides some containers that are useful in many cases and provide more features than the above-defined functions.

Collections Module

Python collection module was introduced to improve the functionality of the built-in datatypes. It provides various containers let’s see each one of them in detail.

Counters

A counter is a sub-class of the dictionary. It is used to keep the count of the elements in an iterable in the form of an unordered dictionary where the key represents the element in the iterable and value represents the count of that element in the iterable. This is equivalent to a bag or multiset of other languages.

OrderedDict

An OrderedDict is also a sub-class of dictionary but unlike a dictionary, it remembers the order in which the keys were inserted.

DefaultDict

DefaultDict is used to provide some default values for the key that does not exist and never raises a KeyError. Its objects can be initialized using DefaultDict() method by passing the data type as an argument.

Note: default_factory is a function that provides the default value for the dictionary created. If this parameter is absent then the KeyError is raised.

ChainMap

A ChainMap encapsulates many dictionaries into a single unit and returns a list of dictionaries. When a key is needed to be found then all the dictionaries are searched one by one until the key is found.