Journal automation - database structure in python

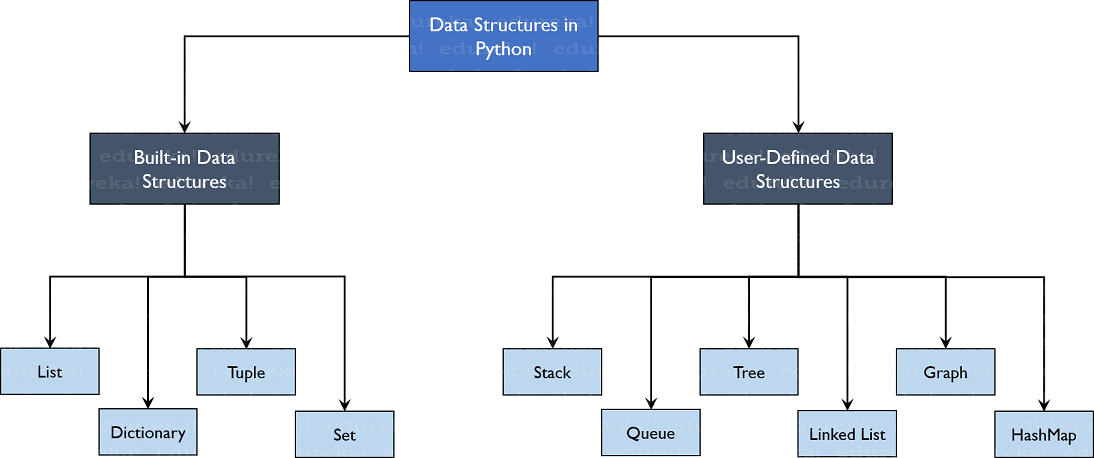
## **What is a Data Structure?**

**Organizing**, **managing** and **storing** data is important as it enables easier access and efficient modifications. Data Structures allows you to organize your data in such a way that enables you to store collections of data, relate them and perform operations on them accordingly.

## **Types of Data Structures in Python**

Python has **implicit** support for Data Structures which enable you to store and access data. These structures are called [List](https://www.edureka.co/blog/lists-in-python/), [Dictionary](https://www.edureka.co/blog/dictionary-in-python/), [Tuple](https://www.edureka.co/blog/tuple-in-python/) and [Set](https://www.edureka.co/blog/sets-in-python/).

Python allows its users to create their own Data Structures enabling them to have **full control** over their [functionality](https://www.edureka.co/blog/python-functions). The most prominent Data Structures are Stack, Queue, Tree, Linked List and so on which are also available to you in other programming languages. So now that you know what are the types available to you, why don’t we move ahead to the Data Structures and implement them using Python.



## **Built-in Data Structures**

As the name suggests, these Data Structures are built-in with [Python which makes programming easier](https://www.edureka.co/blog/python-basics/) and helps programmers use them to obtain solutions faster. Let’s discuss each of them in detail.

### **Lists**

[Lists](https://www.edureka.co/blog/lists-in-python/) are used to store data of different data types in a sequential manner. There are addresses assigned to every element of the list, which is called as Index. The index value starts from 0 and goes on until the last element called the **positive index**. There is also **negative indexing** which starts from -1 enabling you to access elements from the last to first. Let us now understand lists better with the help of an example program.

#### **Creating a list**

To create a list, you use the square brackets and add elements into it accordingly. If you do not pass any elements inside the square brackets, you get an empty list as the output.

|  |  |
| --- | --- |
| 1  2  3  4 | my\_list = [] #create empty list  print(my\_list)  my\_list = [1, 2, 3, 'example', 3.132] #creating list with data  print(my\_list) |

**Output:**  
[]  
[1, 2, 3, ‘example’, 3.132]

#### **Adding Elements**

Adding the elements in the list can be achieved using the append(), extend() and insert() functions.

* The append() function adds all the elements passed to it as a single element.
* The extend() function adds the elements one-by-one into the list.
* The insert() function adds the element passed to the index value and increase the size of the list too.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | my\_list = [1, 2, 3]  print(my\_list)  my\_list.append([555, 12]) #add as a single element  print(my\_list)  my\_list.extend([234, 'more\_example']) #add as different elements  print(my\_list)  my\_list.insert(1, 'insert\_example') #add element i  print(my\_list) |

**Output:**  
[1, 2, 3]  
[1, 2, 3, [555, 12]]  
[1, 2, 3, [555, 12], 234, ‘more\_example’]  
[1, ‘insert\_example’, 2, 3, [555, 12], 234, ‘more\_example’]

#### **Deleting Elements**

* To delete elements, use the del keyword which is built-in into Python but this does not return anything back to us.
* If you want the element back, you use the pop() function which takes the index value.
* To remove an element by its value, you use the remove() function.

**Example:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | my\_list = [1, 2, 3, 'example', 3.132, 10, 30]  del my\_list[5] #delete element at index 5  print(my\_list)  my\_list.remove('example') #remove element with value  print(my\_list)  a = my\_list.pop(1) #pop element from list  print('Popped Element: ', a, ' List remaining: ', my\_list)  my\_list.clear() #empty the list  print(my\_list) |

**Output:**  
[1, 2, 3, ‘example’, 3.132, 30]  
[1, 2, 3, 3.132, 30]  
Popped Element: 2 List remaining: [1, 3, 3.132, 30]