Lecture 10

Learning outcomes:

- *Python Data Structures, In- and out*

Data Structures

- Data Structures are a way of organizing data so that it can be accessed more efficiently depending upon the situation.

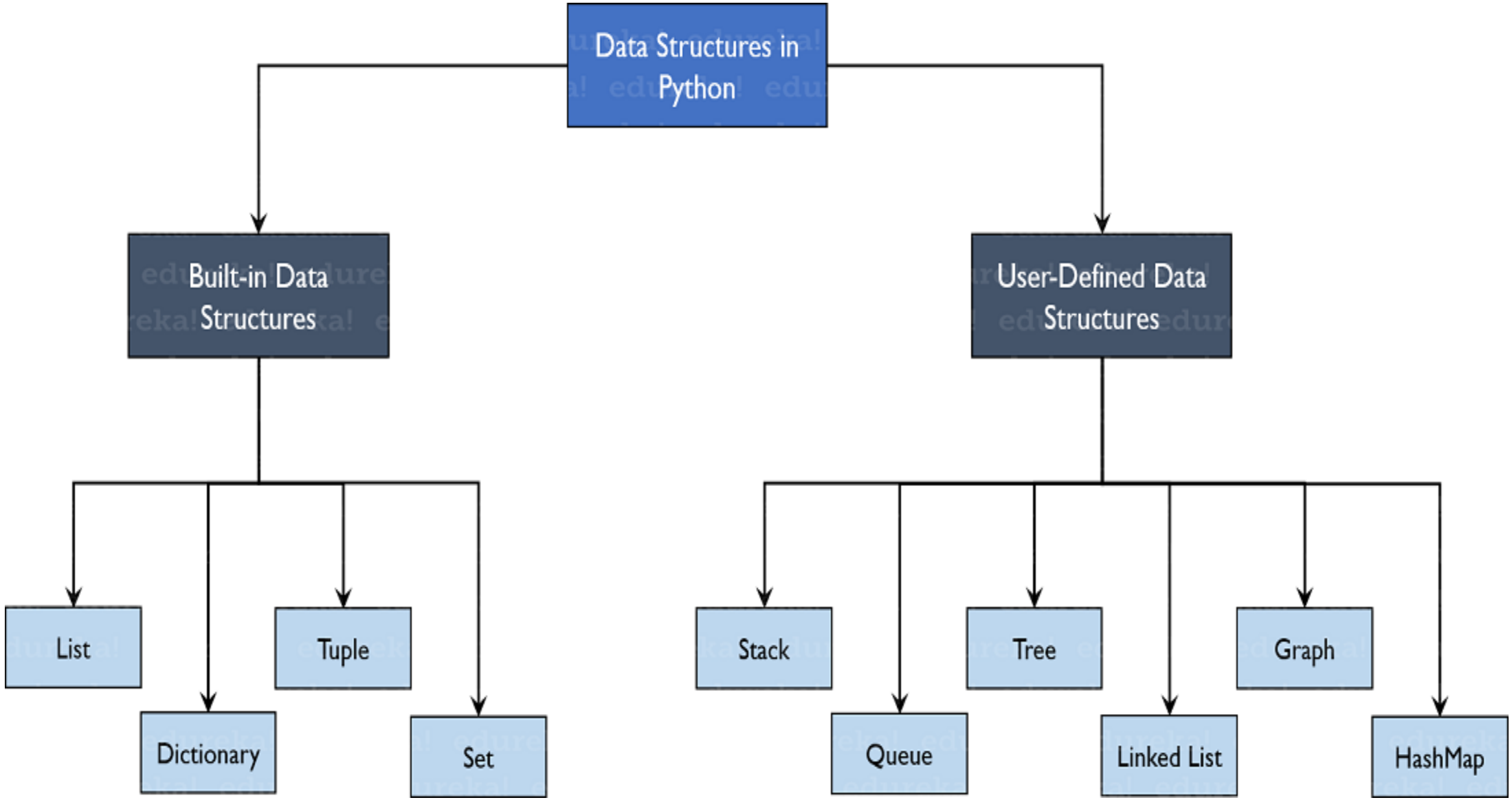
- Data Structures are fundamentals of any programming language around which a program is built.

- Python helps to learn the fundamental of these data structures in a simpler way as compared to other programming languages.

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 **Built-in data structures**.
- **Built-in data structures** in Python can be divided into two broad categories: mutable and immutable. Mutable (from Latin mutabilis, "changeable") data structures are those which we can modify **--** for example, by adding, removing, or changing their elements. Python has three mutable data structures: ***lists, dictionaries, and sets***. Immutable data structures, on the other hand, are those that we cannot modify after their creation. The only basic built-in immutable data structure in Python is a ***tuple.***

- Python allows its users to create their own Data Structures (**User-Defined Data Structures**) enabling them to have full control over their functionality. The most prominent Data Structures are ***Stack, Queue, Tree, Linked List*** and so on which are also available to you in other programming languages.


Data Structures in Python
Built-in Data Structures
User-Defined Data Structures
List
Dictionary
Tuple
Set
Stack
Queue
Tree
Linked List
Graph
HashMap

Built-in Data Structures

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.

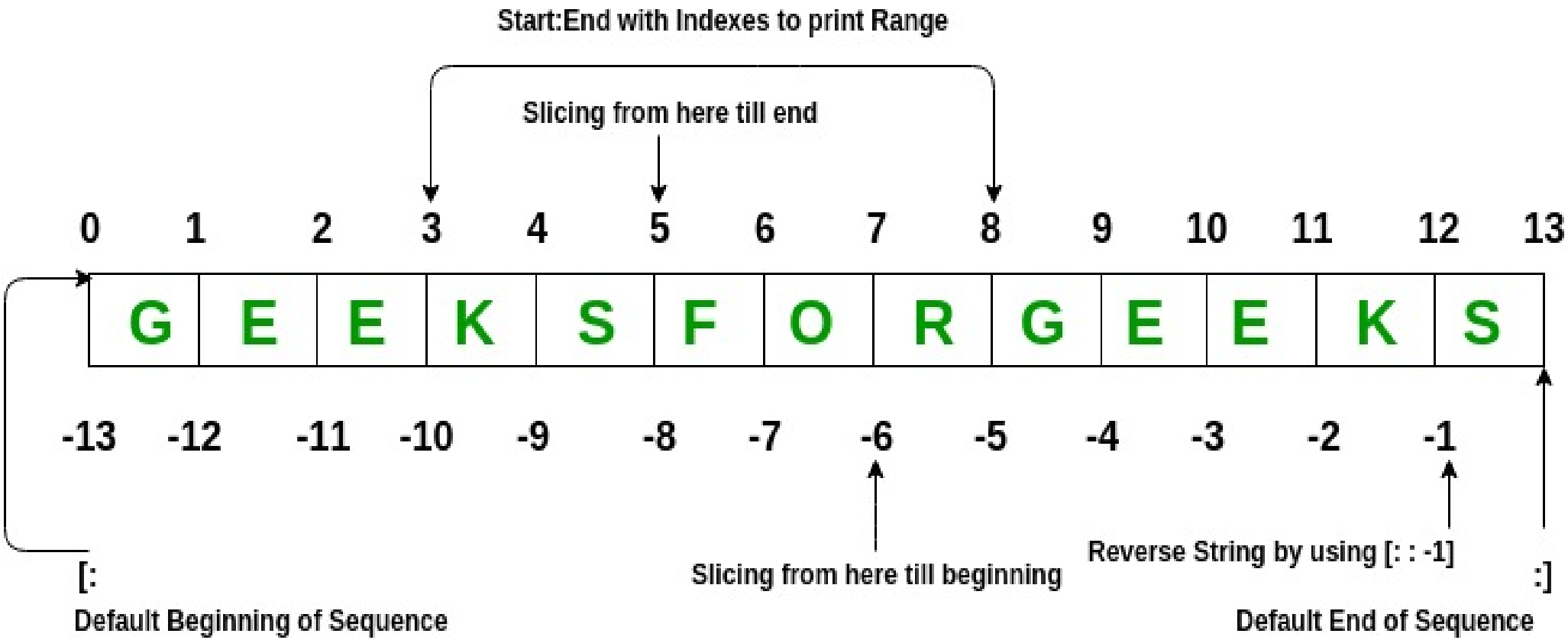
- Example: Creating Python List:

```
List1 = ["Geeks", "For", "Geeks"]
print(List1)
```

✓*Output*

```
['Geeks', 'For', 'Geeks']
```

- 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.
- Example: Python List Operations



 # Creating a List with

➢ ***the use of multiple values:***

✓Input

```
List2 = [1, 2,  3, "GFG", 2.3]
print("\nList containing multiple values: ")
print(List2)
```

✓Output

```
[1, 2, 3, 'GFG', 2.3]
```

 # Creating a Multi-Dimensional List

➢ ***By Nesting a list inside a List:***

```
List3 = [['Geeks', 'For'], ['Geeks']]
print("\nMulti-Dimensional List: ")
print(List3)
```

✓*Output*

```
Multi-Dimensional List:
[['Geeks', 'For'], ['Geeks']]
```

 # accessing a element from the list

➢ ***using positive indexing:***

```
print("Accessing element from the list")
print(List1[0])
print(List1[2])
```

✓*Output*

```
Accessing element from the list
Geeks
Geeks
```

 # accessing a element using

➢ ***using negative indexing***

```
print("Accessing element using negative indexing")
print the last element of list
print(List1[-1])
print the third last element of list
print(List1[-3])
```

✓*Output*

```
Accessing element using negative indexing
Geeks
Geeks
```

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 ({}).

Example: Python Dictionary Operations

➢ ***Creating a Dictionary***

```
Dict = {'Name': 'Geeks', 'Number': 5}
print("Creating Dictionary: ")
print(Dict)
```

✓*Output*

```
Creating Dictionary:
{'Name': 'Geeks', 'Number': 5}
```

- ***accessing a element using key***

```
print("Accessing a element using key:")
print(Dict['Name'])
```

✓*Output*

```
Accessing a element using key:
Geeks
```

- ***accessing a element using get()***

```
print("Accessing a element using get:")
print(Dict.get('Number'))
```

✓*Output*

```
Accessing a element using get:
5
```

 # creation using Dictionary comprehension

```
myDict = {x: x**2 for x in [1,2,3,4,5]}
print(myDict)
```

✓*Output*

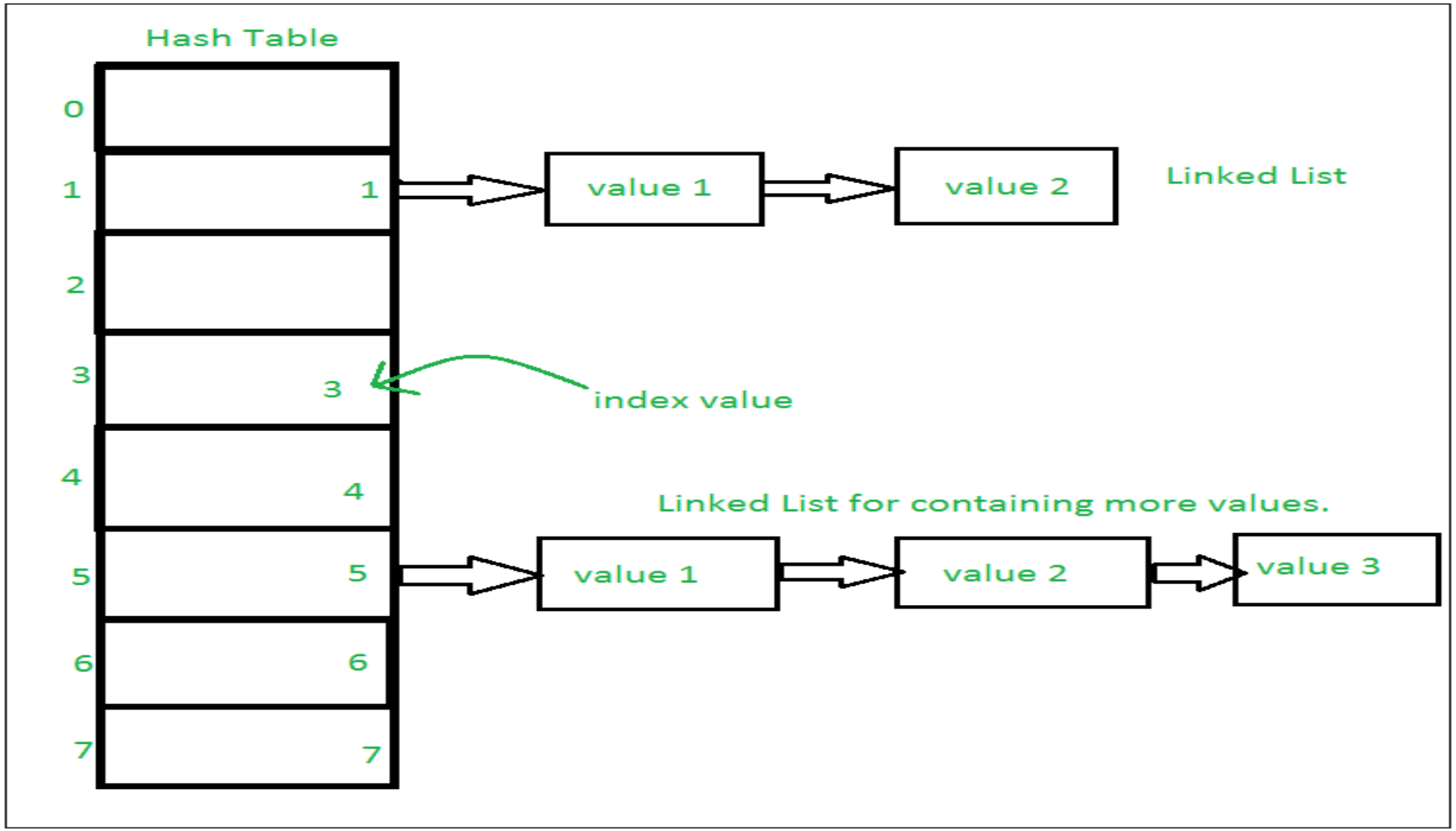
```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Sets

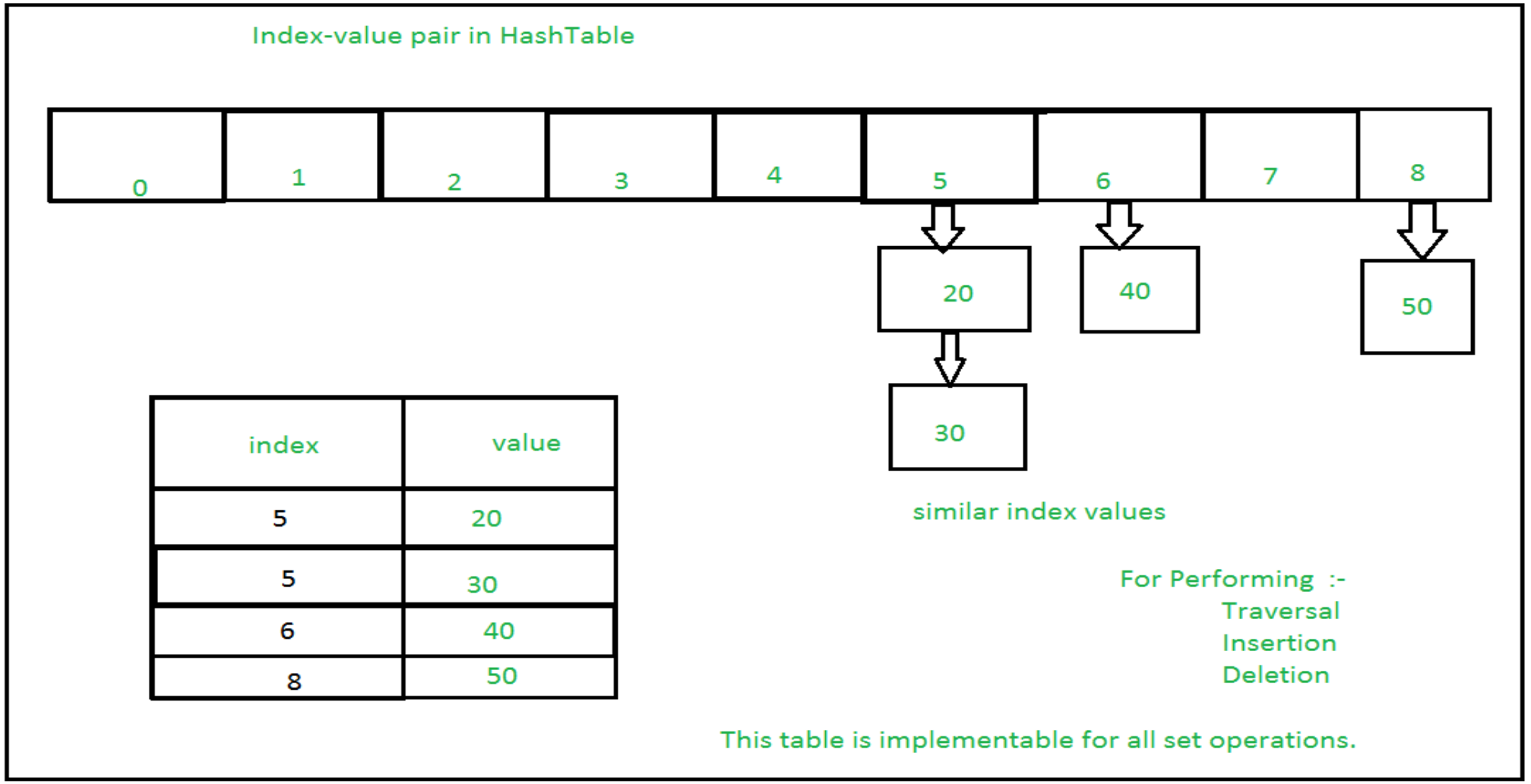
- Python Set is an unordered collection of data that is mutable and does not allow any duplicate element. Sets are basically used to include membership testing and eliminating duplicate entries. The data structure used in this is Hashing, a popular technique to perform insertion, deletion, and traversal 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, Python Sets are implemented using a dictionary with dummy variables, where key beings the members set with greater optimizations to the time complexity.

Set Implementation:



Sets with Numerous operations on a single HashTable:



index	value
5	20
5	30
6	40
8	50

Example: Python Set Operations

➢ ***Creating a Set with***

```
# a mixed type of values
# (Having numbers and strings)
Set = set([1, 2, 'Geeks', 4, 'For', 6, 'Geeks'])
print("\nSet with the use of Mixed Values")
print(Set)
```

✓*Output*

```
Set with the use of Mixed Values
{1, 'Geeks', 2, 4, 6, 'For'}
```

➢ ***Accessing element using***

```
# for loop
print("\nElements of set: ")
for i in Set:
    print(i, end =" ")
print()
# Checking the element
# using in keyword
print("Geeks" in Set)
```

✓*Output*

```
Elements of set:
1 Geeks 2 4 6 For
True
```

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.

Example: Python Tuple Operations

➢ ***Creating a Tuple with***

▪ The use of Strings

```
Tuple = ('Geeks', 'For')
print("\nTuple with the use of String: ")
print(Tuple)
```

✓*Output*

```
Tuple with the use of String:
('Geeks', 'For')
```

➢ ***Creating a Tuple with***

- The use of list

```
list4 = [1, 2, 4, 5, 6]
print("\nTuple using List: ")
Tuple = tuple(list4)
print(Tuple)
```

✓*Output*

```
Tuple using List:
(1, 2, 4, 5, 6)
```

➢***Accessing elements using positive indexing***

```
print("First element of tuple")
print(Tuple[0])
```

✓*Output*

```
First element of tuple
1
```

➢***Accessing elements from last (negative indexing)***

```
print("\nLast element of tuple")
print(Tuple[-1])
print("\nThird last element of tuple")
print(Tuple[-3])
```

✓*Output*

```
Last element of tuple
6
Third last element of tuple
4
```

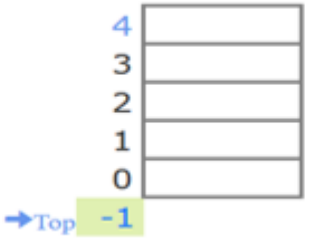
Arrays vs. Lists

- Arrays and lists are the same structure with one difference.

- Lists allow heterogeneous data element storage whereas Arrays allow only homogenous elements to be stored within them.

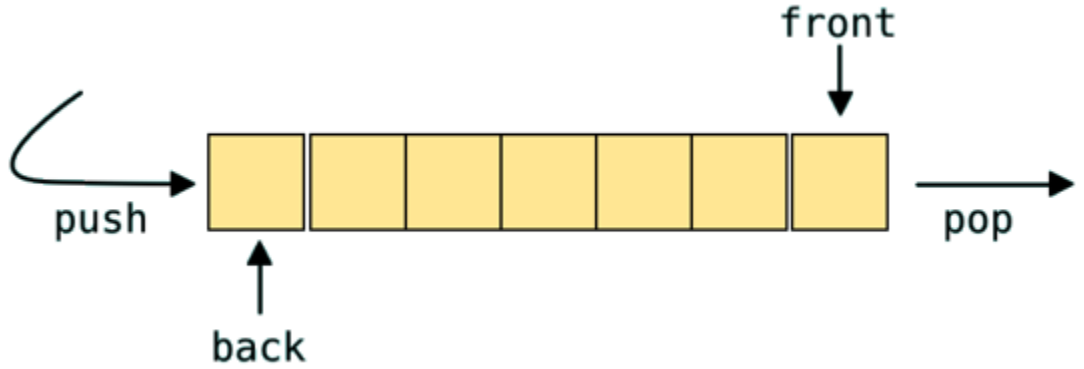
User-Defined Data Structures

Stack



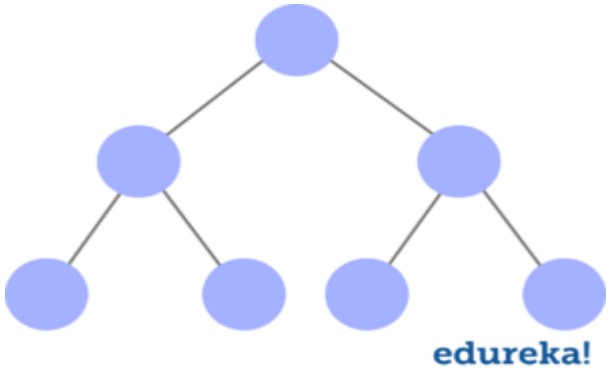
- Stacks are linear Data Structures which are based on the principle of Last-In-First-Out (LIFO) where data which is entered last will be the first to get accessed.
- It is built using the array structure and has operations namely, pushing (adding) elements, popping (deleting) elements and accessing elements only from one point in the stack called as the TOP. This TOP is the pointer to the current position of the stack.
- Stacks are prominently used in applications such as Recursive Programming, reversing words, undo mechanisms in word editors and so forth.

Queue



- A queue is also a linear data structure which is based on the principle of First-In-First-Out (FIFO) where the data entered first will be accessed first.
- It is built using the array structure and has operations which can be performed from both ends of the Queue, that is, head-tail or front-back.
- Operations such as adding and deleting elements are called En-Queue and De-Queue and accessing the elements can be performed.
- Queues are used as Network Buffers for traffic congestion management, used in Operating Systems for Job Scheduling and many more.

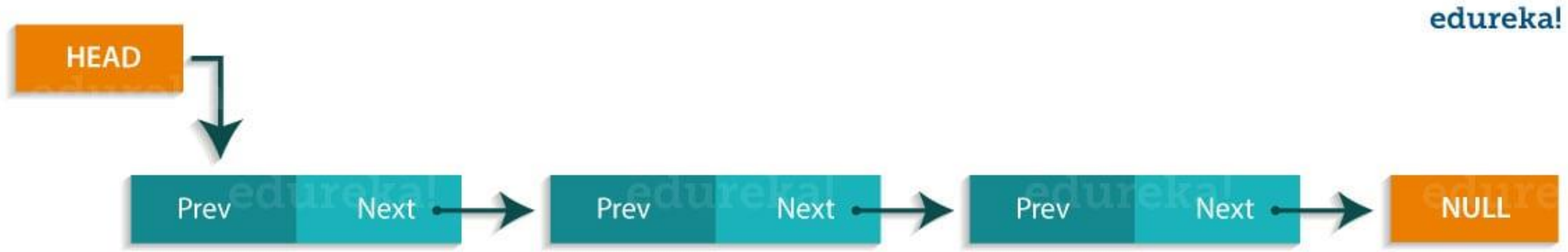
Trees



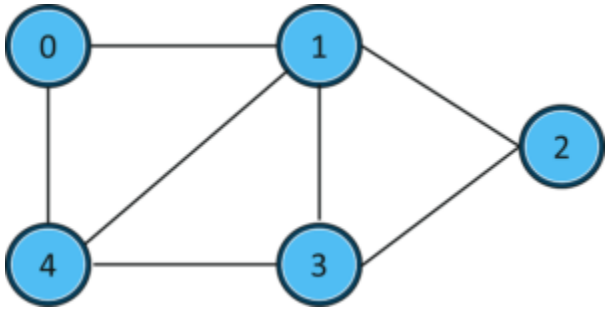
- Trees are non-linear Data Structures which have root and nodes. The root is the node from where the data originates and the nodes are the other data points that are available to us. The node that precedes is the parent and the node after is called the child.
- There are levels a tree has to show the depth of information. The last nodes are called the leaves.
- Trees create a hierarchy which can be used in a lot of real-world applications such as the HTML pages use trees to distinguish which tag comes under which block. It is also efficient in searching purposes and much more.

Linked List

- Linked lists are linear Data Structures which are not stored consequently but are linked with each other using pointers.
- The node of a linked list is composed of data and a pointer called next. These structures are most widely used in image viewing applications, music player applications and so forth.



Graph



- Graphs are used to store data collection of points called vertices (nodes) and edges (edges).
- Graphs can be called as the most accurate representation of a real-world map.
- They are used to find the various cost-to-distance between the various data points called as the nodes and hence find the least path.
- Many applications such as Google Maps, Uber, and many more use Graphs to find the least distance and increase profits in the best ways.

HashMaps

- HashMaps are the same as what dictionaries are in Python. They can be used to implement applications such as phonebooks, populate data according to the lists and much more.