Week 1	**Introduction to Data Structures, operations, classification.** **Primitive types – primitive data structures, python examples. Non primitive types – Non primitive data structures, python examples. Linear and nonlinear data structures – with python examples.** **Introduction, Abstract Data Types, An Example of Abstract Data Type (Student, Date, Employee), Defining the ADT, Using the ADT, Implementing the ADT.**

Data Structures

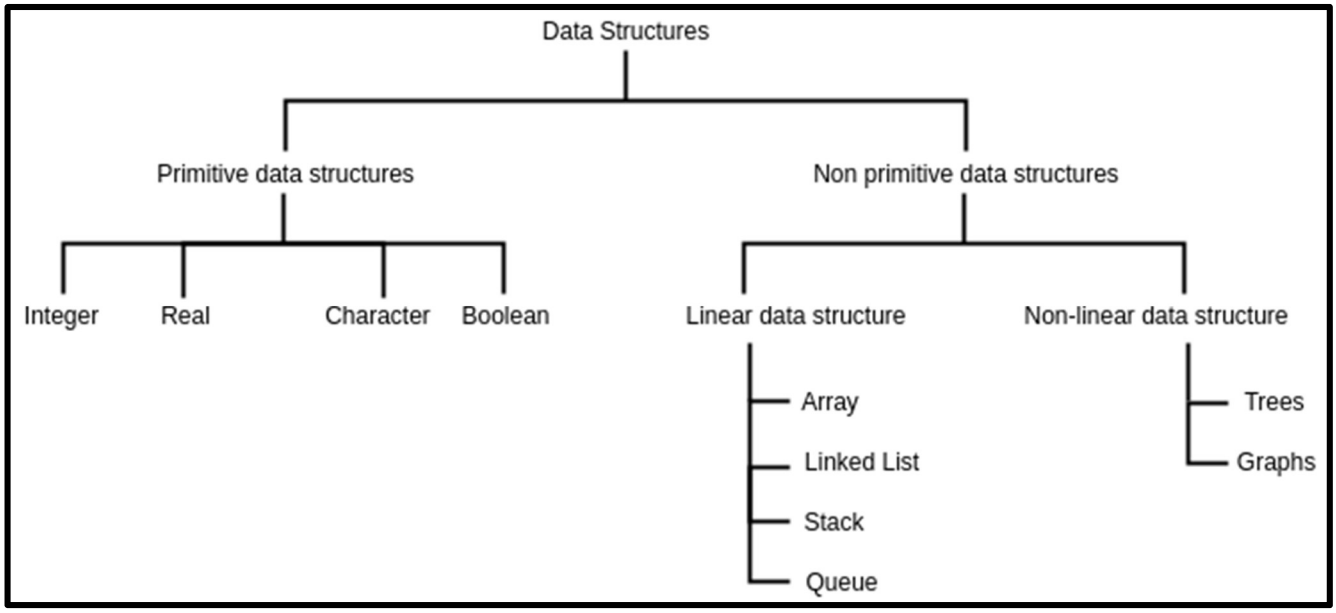
Data Structures is defined as "Systematic way of organizing group of related data items" in such a way that:

- Relationship among the data items is preserved
- Operations on it becomes easy

Operations on Data Structures

- ✓ Searching
- ✓ Insertion
- ✓ Deletion
- ✓ Traversal
- ✓ Sorting

Classification of data structures



Data Structures can be classified into two types,

1. **Primitive Data Types (Primitive Data Structure)**
2. **Non-Primitive Data Types (Non-Primitive Data Structure)**

Primitive Data Structures: Primitive Data Types are capable of holding single value. Python Examples include Integers, Floating Point Numbers, Strings, Boolean data types.

Non-Primitive Data Structures: A data structure which is derived from primitive data structures and capable of holding group of values is referred as Non-Primitive Data Structures. It is further classified into two types,

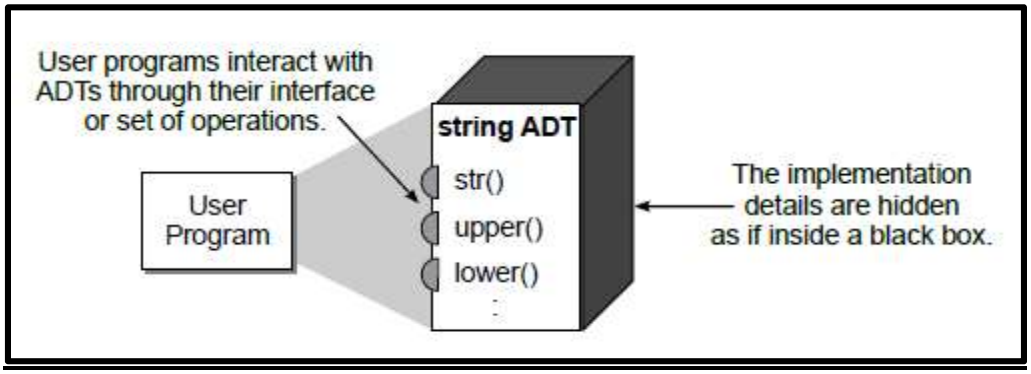
1. **Linear Data Structure:** In linear data structure, data items are arranged in sequence or linear fashion. Python Examples includes arrays, Lists, sets, tuples, dictionary, stack, queue, linked lists etc.
2. **Non-Linear Data Structure:** In Non-Linear data structure, data items are not arranged in sequence or linear fashion. Python Examples includes trees, graphs etc.

Linear Vs. Non-Linear Data Structures

Linear data Structure	Non-Linear Data Structure
Data items are arranged in sequence	Data Items are not arranged in sequence
Implementation of linear data structures is easy	Implementation of non-linear data structures is difficult
Each data item in linear data structure related to its previous and next data item	Each data item in non-linear data structure related to several data items.
Python Examples: Arrays, Sets, Tuples, Dictionary, Lists, Stack, Queue and Linked Lists.	Python Examples: Trees, Graphs.

Abstract Data Type

An abstract data type (or ADT) is a user defined data type that contains a set of data values and a collection of operations that can be performed on those values. Fig below shows pictorial representation of string ADT:



The Date Abstract Data Type

- The date abstract data type for representing a date and contains its operations.

Define: Date ADT

Date(dd, mm, yy): Creates a new Date which must be valid.
getDay( ): Returns the day number of this date.
getMonth( ): Returns the month number of this date.
getYear( ): Returns the year of this date.
isLeapYear( ): Determines if this date is leap year or not.

Implementation: Date ADT

```
Class Date:
    def __init__(self, dd, mm, yy):
        self.day = dd
        self.month = mm
        self.year = yy

    def getDay(self):
        return self.day

    def getMonth(self):
        return self.month

    def getYear(self):
        return self.year

    def isLeapyear(self):
        if self.year % 4 == 0:
            return True
        else:
            return False
```

How to Use Date ADT:

```
D = Date(21,06,2022)
print("Day:". D.getDay())
print("Month:". D.getMonth())
print("Year:". D.getYear())
print("Leap year:". D.isLeapYear())
```

The Student Abstract Data Type

- The Student abstract data type for representing a student (contains student information and its operations).

Define: Student ADT

Student(name, sem, age): Creates a new student object.
displayDetails( ): displays complete information of a student.
getSem( ): Returns semester of the student.
updateAge( ): updates student age.

Implementation: Student ADT

```
Class Student:
    def  _ _init_ _(self, nm, semester, age):
            self.name = nm
            self.sem = semester
            self.age = age

    def displayDetails(self):
            print("Name:",self.name)
            print("Semester:",self.sem)
            print("Age:",self.age)

    def getSem(self):
            return self.sem

    def updateAge(self,value):
            self.age=value
```

How to Use Student ADT:

```
st = Student("ABC", 4, 20)
st.displayDetails( )
print("Semester:". st.getSem( ))
st.updateAge(22)
st.displayDetails( )
```

The Employee Abstract Data Type

- The Employee abstract data type for representing an employee (contains employee information and its operations).

Define: Employee ADT

Employee(ID,name, salary): Creates a new employee object.
displayIDName( ): displays ID and name of an employee.
getSalary( ): Returns salary of the employee.
displayDetails( ): displays complete information of an employee.

Implementation: Employee ADT

```
Class Employee:
    def  __init__(self, id, nm, sal):
            self.ID = id
            self.name = nm
            self.salary = sal

    def displayDetails(self):
            print("ID:",self.ID)
            print("Name:",self.name)
            print("Salary:",self.salary)

    def displayIDName(self):
            print("ID:",self.ID)
            print("Name:",self.name)

    def getSalary(self):
            return self.salary
```

How to Use Student ADT:

```
emp = Employee(11,"ABC", 40000)
emp.displayIDName( )
emp.displayDetails( )
print("Salary:". st.getSalary( ))
```

Activity #1

1. **Demonstrate Python IDE: Anaconda**