**Data Structure**

Data Structures are the way to store and organize the data so that it can be access effectively and also it can be used and manipulated easily, Data structure is the basic and important concept of any programming language.

**Python Data Structure**

**Built in User define**

1. List 1. Stack
2. Tuple 2. Queue
3. Set 3. Linked List
4. Dictionary 4. Tree

5. Graph

**List and Tuple**

**List:**

List is the collection of element separated by commas and enclosed the square bracket or List is the group of element which are separated by commas and enclosed with square bracket.

For example:

List1=[10,20,30,50,65]

**The Characteristics of the list:**

1. **Mutable:** List is mutable. We can change the value after creating a list. We can add element, we can replace element, we can delete element from a list through list operation.

**For example**:

List1=[10,20,30,50,65]

List1.append(10)

Now List1 is changed, the value of List1 is [10,20,30,50,65,10].

1. **Linear Data Structure:** List is a linear data Structure. Because the element of the list is arrange in linear order. The element are arranged such a way that is the first element, that is the second element that is the third element. So List is the linear data structure.

**For example**:

List1=[10,20,30,50,65]

10 is the first element of list1, 20 is the second element of List1 etc.

1. **Mixed type element:** List can contain mixed type data, in a list you can store string, number, list, tuple etc.

**For example**:

List1=[10,”Nahid“,30,50.20,65,(10,25,300)]

Here List1 is a valid list.

1. **Zero base indexing:** List is a zero base indexing. Because the first element index in a list is 0. If we access the first element of a list we should mention the index as 0.

**For example**:

List1=[10,20,30,50,65]

X=List1[0] # program for access the first element

Output of X is : 10

1. **List is nested:** We can use a list as an element of another list.

**For example**:

List1=[[10,20],[30,50],[65,60]]

**List Operation**

1. **Replace:** If we need to change or replace an element of a list. We need to replace use replace operation.

**For example**:

List1=[10,20,30,50,65]

List1[2]=”apple” # program for replace 3rd element**(index 2)** of List1.

Output of List1=[10,20,‘apple’,50,65]

1. **Insert:** If we need to add an item at anywhere of a list we can use insert operation.

**For example**:

List1=[10,20,30,50,65]

List1.insert(2,”element”) # here insert is the method, 2 is index number

Output of List1=[10,20,”element”,50,65]

Look at the output we used 2 as an index of list1 and used “element” as an element of List1 so in the output we can get the “element” at the index of 2.

1. **Sort operation:** we can use sort operation when the element of list will be same category. When the list contain number only or string only then we can use sort operation.

**For example**:

**Example1**

List1=[10,65,20,50,30]

List1.sort() # program for sort a list

Output of List1: [10,20,30,50,65]

**Example2**

List2=[“aa”, ”cc”,”bb”, ee”, dd”]

List2.sort() # program for sort a list

Output of List2: [“aa”,”bb”,”cc”,”dd”,”ee”]

1. **Delete:** If we need to delete an item at anywhere of a list we can use delete operation.

**For example**:

List1=[10,20,30,50,65]

Del List1[3] # here del is the keyword, 3 is index number

Output of List1=[10,20,30,65]

Look at the output 50 is deleted. Because 50 was 3 number index.

1. **Append:** If we need to add an element at the end of a list we can use append operation.

**For example**:

List1=[10,20,30,50,65]

List1.append(50) # here append is the method and 50 is the element.

Output of List1=[10,20,30,50,65,50]

Look at the output 50 is added at the end of the list.

1. **Reverse operation:** If we want to reverse a list we can use reverse method.,

**For example**:

List1=[10,20,30,50,65]

List1.reverse() # reverses operation.

Output of List1=[65,50,30,20,10]

Look at the output List is reversed.

**Tuple:**

Tuple is a build in data structure in python. Tuple is similar to the list.

For example:

Tuple1=(10,20,30,50,65)

**The Characteristics of the Tuple:**

1. **Immutable:** Tuple is immutable. We can’t change the value after creating a Tuple.

**For example**:

Tuple1=[10,20,30,50,65]

Tuple1.append(10) # program will give you error.

1. **Linear Data Structure:** Tuple is a linear data Structure. Because the element of the Tuple is arrange in linear order. The element are arranged such a way that is the first element, that is the second element that is the third element. So Tuple is the linear data structure.

**For example**:

Tuple1=(10,20,30,50,65)

10 is the first element of list1, 20 is the second element of List1 etc.

1. **Mixed type element:** Tuple can contain mixed type data, in a list you can store string, number, list, tuple etc.

**For example**:

Tuple1=(10,”Nahid“,30,50.20,65,(10,25,300))

Here List1 is a valid list.

1. **Zero base indexing:** Tuple is a zero base indexing. Because the first element index in a Tuple is 0. If we access the first element of a Tuple we should mention the index as 0.

**For example**:

Tuple1=(10,20,30,50,65)

X=List1[0] # program for access the first element

Output of X is : 10

1. **List is nested:** We can use a list as an element of another list.

**For example**:

List1=((10,20),(30,50),(65,60))

**Sequences in python**

List, Tuple, String which data structure is index base or linear data are called as sequences.

**Sequences operation**

1. **Length operation:** If we need to know the length of sequence like list, tuple or string we need to use len method.

**For example**:

List1=[10,20,30,50,65]

X=len(List1) # program for getting the length of sequences.

Output of X=5

List1 has five elements so the output of X is 5. We can use len method into a string or tuple too.

1. **Select operation:** If we need to select a specific element and work with it of a sequence like tuple, list or string we use indexing to perform select operation of a sequences.

**For example**:

Str=”I will be a Programmer”

X=Str[2] # program for getting the specific element of a sequences.

Output of X=w

X output is w, because w is setting at the index of 2 of Str variable.

1. **Slice operation:** If we need get some specific element form a sequence we need to perform the slice operation.

**For example**:

Str=”I will be a Programmer”

X=Str[10:20] # here 10 is included index and 20 is excluded index.

Output of X=Programmer

X output is programmer, because the index of p is 10 and the index of r is 19.

1. **Membership operation:** If we need check that an element is present or not, in the sequence m, we can perform the membership operation.

**For example**:

**Example1**

Tuple1=(10,20,30,50,50)

X=20 in Tuple1 # here 20 is element which we check.

Output of X=True

X output is True, because 20 is present in the tuple.

**Example2**

List1=[10,20,30,50,50]

X=100 not in List1 # here 100 is element which we check.

Output of X=True

X output is True, because 100 is not present in the tuple.

1. **Count operation:** If we need to get the number of time the elements present in the sequences we need to perform count operation by count function.

**For example**:

**Example1**

Tuple1=(10,20,30,50,50)

X=Tuple1.count(50) # here 50 is element which we check.

Output of X=2

X output is 2, because 50 is present in the Tuple1 2 time.

**Example2**

**Str**=”Md Nahid Islam”

X=Str.count(“a”) # here “a” is element which we check.

Output of X=2

X output is 2, because “a” is present in the **Str** 2 time.

1. **Concatenation:** If we need add to sequence concatenating is the best way.

**For example**:

**Example1**

Tuple1=(10,20,30,50,50)

Tuple2=(10,20,30,50,50)

X=Tuple1+Tuple2 # program for concatenating tuple

Output of X= (10, 20, 30, 50, 50, 10, 20, 30, 50, 50)

1. **Max & Min:** If we need to know the maximum or minimum value from a sequence we use the max or min function.

**For example**:

**Example1**

Tuple1=(10,20,30,50,50)

X=max(Tuple1) # program for Get the maximum value from a tuple

Output of X: 50

X=min(Tuple1) # program for Get the minimum value from a tuple

Output of Y: 10

**[NB: Here All operation will be valid in every sequence like List, String, Tuple]**

**Sorting in python**

When data arrange into ascending or designing order it’s called sorted data. We can sort Data in many ways.

Sort() # build in List method

Sorted() # Build in function for iterable.

**For Example:**

**Example1:**

**List1=[10,25,36,12,52,12,85]**

**List1.sort()**

Now List one is arrange as [10, 12, 12, 25, 36, 52, 85].Because by default it sorted the list into ascending order.

**List2=[10,25,36,12,52,12,85]**

**List2.sort(reverse=True)**

Now List2 one is arrange as [85, 52, 36, 25, 12, 12, 10].Because we mansion reverse argument as True.

**sorted methods in python**

sorted(iterable, /, \*, key=None, reverse=False) #Syntax of sorted methods

Return a new list containing all items from the iterable in ascending order. Reverse flag can be set to request the result in descending order.

**Example1:**

**Tuple1=(“aaaa”,”bbbbb”,”dddd”,Abid”,rohit”)**

**X =sorted(Tuple1,reverse=True)**

Now Tuple one is arrange as ['rohit', 'dddd', 'bbbbb', 'aaaa', 'Abid']. If we need to sort List we can use sort() method or we need to sort other iterable we can use sorted() function for sorted.

**Dictionary Data structure**

dictionary is a built-in data type that allows you to store and retrieve data in key-value pairs.

**For example:**

# Creating a dictionary

my\_dict = {

'name': 'John',

'age': 30,

'city': 'Thakurgaon'

}

**Here name, age, city are the keys and john, age, Thakurgaon are the value.**

**Dictionary Operation**

**Accessing part of Dictionary**

I we need to access a value of a dictionary. We need to use Access operation.

**For example**

**Example1:**

my\_dict = {

'name': 'John',

'age': 30,

'city': 'Thakurgaon'

}

print(my\_dict['name']) # Output: John

print(my\_dict.get(age)) # Output: 30

**Change the value of a dictionary**

I we need to change a value of a dictionary.

**For example**

**Example1:**

my\_dict = {

'name': 'John',

'age': 30,

'city': 'Thakurgaon'

}

**my\_dict['name']=”Nahid Islam”** # Operation for change element of a dictionary

print(my\_dict['name']) # Output: Nahid Islam

**Adding a new key-value pair**

I we need to add new key to the dictionary.

**For example**

**Example1:**

my\_dict = {

'name': 'Nahid Islam',

'age': 30,

'city': 'Thakurgaon'

}

**my\_dict['Occupation']=”Engineer”** # Operation for add new element to a dictionary

print(my\_dict['Occupation']) # Output: Engineer

**Membership operation**

I we need to check a particular item is present or not in the dictionary. We need to perform membership operation by in and not in operators.

**For example**

**Example1:**

my\_dict = {

'name': 'Nahid Islam',

'age': 30,

'city': 'Thakurgaon'

}

X=city in my\_dict**”** # Membership operation

print(X) # Output: True

here output is True because city is present in my\_dict.

**Delete opetaion**

I we need to delete or remove an item from a dictionary, We need to perform Delete operation by **pop** methods or **del** keyword.

**For example**

**Example1:**

my\_dict = {

'name': 'Nahid Islam',

'age': 30,

'city': 'Thakurgaon'

}

my\_dict.pop()# delete operation

here city will remove from the dictionary

**Example2:**

my\_dict = {

'name': 'Nahid Islam',

'age': 30,

'city': 'Thakurgaon'

}

del my\_dict[“name”] **# delete operation**

Here name will remove from the dictionary

**Getting all keys and values**

If we want to get all the keys of a dictionary we use keys and values method for it.

**For example**

**Example1:**

my\_dict = {

'name': 'Nahid Islam',

'age': 30,

'city': 'Thakurgaon'

}

X=my\_dict.keys()# getting the keys

Output of X: dict\_keys(['name', 'age', 'city'])

Y=my\_dict.values()# getting the values

Output of Y: dict\_values(['Nahid Islam', 30, 'Thakurgaon'])

**Dictionary update methods**

Update methods is very important method for a dictionary because if we want to concatenate a dictionary or we add element easily to the dictionary we should use it.

**For example**

**Example1:**

my\_dict = {

'name': 'Nahid Islam',

'age': 30,

'city': 'Thakurgaon'

}

my\_dict2 = {

'father': 'Md. Zakaria Islam',

'agef': 50,

'cityf': 'Thakurgaon'

}

my\_dict.update(my\_dict2) # concatenating two dictionary

print(my\_dict["agef"]) # output 50

Now my\_dict2 is the element of my\_dict so the output of my\_dict[agef] : 50.

**Example2:**

my\_dict = {

'name': 'Nahid Islam',

'age': 30,

'city': 'Thakurgaon'

}

My\_dict.update(“Father”=”Md. Zakaria Islam”, mother=”Mst. Lovely Begum”) # add element by update methods

Print(my\_dict[“Father”]) # output : Md. Zakaria Islam

**Set Data structure**

Set is a mutable data type with none duplicated unordered values. It is a associated data structure like dictionary. It means it hasn’t the first element, second element and third element. It is unordered.

In Python, a set is an unordered collection of unique elements. Sets are commonly used to eliminate duplicate values from a list, perform set operations like union, intersection, and difference, and check for membership.

**For example:**

# Creating a sety

my\_dict = {values, values, values} # Same value is not suitable

**Set Operation**

As we know set is a mutable data structure, we can change it.

**Membership operation**

I we need to check a particular item is present or not in the Set. We need to perform membership operation by in and not in operators.

**For example**

**Example1:**

my\_set = { 'name', 'Nahid Islam', 'age', 30, 'city', 'Thakurgaon'}

X=city not in my\_dict**”** # Membership operation

print(X) # Output: True

Here output is False because city is present in my\_dict.

**Add operation**

I we need to add a new element into a set. We use add operation.

**For example**

**Example1:**

my\_set= { 'name', 'Nahid Islam', 'age', 30, 'city', 'Thakurgaon'}

my\_set.add(10)

Now 10 is added to my\_set.

**Remove operation**

I we need to remove element from a set. We use remove operation.

**For example**

**Example1:**

my\_set= { 'name', 'Nahid Islam', 'age', 30, 'city', 'Thakurgaon'}

my\_remove.add(“city”)

Now “city” is remove from the set.

**Union and intersection Operation**

**Example**

**Example1: Union**

my\_set= { 'name', 'Nahid Islam', 'age', 30, 'city', 'Thakurgaon'}

my\_set2= { 20, 'Nahid Islam', 'age', 30, 'city', 'Dinajpur'}

X=my\_set | my\_set2 # Union Operation

Output of X is: {'Thakurgaon', 'Nahid Islam', 'name', 'age', 'city', 20, 'Dinajpur', 30}

**Example2: Intersection**

my\_set= { 'name', 'Nahid Islam', 'age', 30, 'city', 'Thakurgaon'}

my\_set2= { 20, 'Nahid Islam', 'age', 30, 'city', 'Dinajpur'}

X=my\_set ^ my\_set2 # Intersection Operation

Output of X is: {20, 'Dinajpur', 'Thakurgaon', 'name'}

**Clear Operation**

If we need to clear all the element entire a set. we use clear operation.

**Example**

my\_set= { 'name', 'Nahid Islam', 'age', 30, 'city', 'Thakurgaon'}

X=my\_set.clear()

Output of X is: {}