Python Data Types

◼Everything in Python programming is an object, and each object has its own unique

identity(a type and a value). Identity of an object / variable is nothing but its address

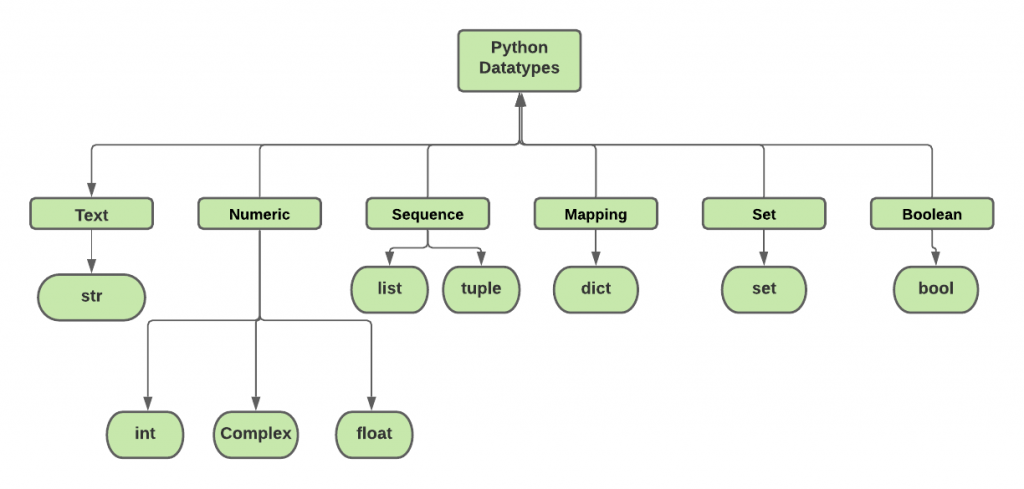
in memory when it is created.

◼ Type or data type indicate a range of values and operations allowed on those values.

◼ Python has five standard data types, named Numbers, None, Sequences, Sets and

Mappings. Python sets the type of variable based on the type of value assigned to it and

it will automatically change the variable type if the variable is set to some other value.



1.Numbers:

◼ Number stores numeric values. Python creates Number objects when a number is

assigned to a variable.

Ex:- a=3, b=5, c=10 # a, b, c are number objects.  
  
◼ Python supports 3 types of numeric data.

(i) int (Ex:- Signed integers 10, 2, 29, etc.)

(ii) float (Ex:- 1.9, 9.902, 15.2, etc.)

(iii) complex (Ex:- 2.14j, 2.0 + 2.3j, etc.)  
  
2.Boolean:

◼ Comparisons in Python can only generate one of two possible

responses: True or False. Python internally represents True as 1 and

False as 0.  
  
3.Sequences:

• In Python, Sequences are the general term for ordered sets. A Sequence is

a data type that represents a group of elements.

• The purpose of any sequence is to store and process a group of elements.

• There are mainly six types of sequences in Python - string, list, tuple,

range, bytes, bytearray.

• Sequences are iterable and has the concept of indexing.

• Sets and dictionaries not comes under sequences.

a.Lists:  
◼ A list is a sequence data type that can be used to store any type and number of variables. It is one of the simplest and most important data structures in Python.

◼ List is ordered, indexed and mutable (i.e its content is changeable). Allows duplicate members.

◼ Lists are enclosed in square brackets [ ] and each item is separated by a comma.

◼ Lists are very flexible and have many built-in control functions.

Syntax: list\_var = [item\_1, item\_2, .....item\_n]

Program:

# Creating a List with the use of multiple strings

colors = [“red", “green", “yellow“, “blue]

print(colors[0])

print(colors[2])

output:

red  
yellow

# Adding elements to the List using Iterator

List1=[]

for i in range(1, 10):

List1.append(i)

print(List1)

List2 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

# Removing elements from List using Remove() method

List2.remove(5)

List2.remove(6)

print(“List2: ”,List2)

o/p:

List2: [1,2,3,4,7,8,9,10,11,12]

# Removing element at a specific location with the pop() method

List2.pop(2)

Print(List2)

o/p:[1,2,4,7,8,9,10,11,12]  
  
b.Tuples:  
  
◼ A tuple is a sequence of immutable Python objects. Items of the

tuple cannot be updated. Tuples are Index based, duplicates are

allowed.

◼ The differences between tuples and lists are, the tuples cannot be

changed unlike lists and tuples use parentheses, whereas lists use

square brackets.

◼ Creating Tuples

tup1 = (‘phone', ‘laptop', 1997, 2000)

tup2 = (1, 2, 3, 4, 5 )

tup3 = ("a", "b", "c", "d")  
Programs:  
◼ Accessing Values in Tuples

print ("tup1[0]: ", tup1[0]) o/p:phone

print ("tup2[1:5]: ", tup2[1:5]) o/p:(2,3,4,5)   
  
◼ Updating Tuples

Tuples are immutable which means you cannot update or change the

values of tuple elements.

tup1[0]=100; #invalid

◼ Concatenation of tuples

tup3=tup1+tup2

print(tup3) output: (‘phone', ‘laptop', 1997, 2000,1,2,3,4,5)

◼Slicing Tuples

tup1=(0,1,2,3,4,5)

print(tup1[2:]) output: (2, 3, 4, 5)

print(tup1[::-1]) output: (5, 4, 3, 2, 1, 0)

4.SETS:

◼ A set is a collection which is unordered and unindexed. In

Python sets are written with curly brackets { }.

◼ The elements in the set cannot be duplicates.

◼ There is no index attached to any element in a python set. So

they do not support any indexing or slicing operation.

◼ The sets in python are typically used for mathematical

operations like union, intersection, difference and

complement etc

◼ We cannot access items in a set by referring to an index,

since sets are unordered & the items has no index.

◼But we can loop through the set items using a for loop.  
  
◼A set is created by using

a) set() function

b) placing all the elements within a pair of curly braces.

Example :-

Days=set(["Mon","Tue","Wed","Thu","Fri","Sat","Sun"])

Months={"Jan","Feb","Mar"}

Dates={21,22,17}

print(Days) (or) for d in Days: print(d)

print(Months)

Output:-

{'Sat', 'Wed', 'Fri', 'Mon', 'Sun', 'Tue', 'Thu'}

{'Mar', 'Feb', 'Jan'}

#Adding Items to a Set

Ex:- Days=set(["Mon","Tue","Wed","Thu","Fri","Sat“])

Days.add("Sun")

print(Days)

Output: {'Fri', 'Sat', 'Wed', 'Thu', 'Tue', 'Mon', 'Sun'}

#Removing Items from a Set

Ex:- Days.discard("Sun")

print(Days)

Output: {'Fri', 'Sat', 'Wed', 'Thu', 'Tue', 'Mon'}

#Union of Sets

The union operation on two sets produces a new set containing all the distinct elements

from both the sets.

Ex:- DaysA = set(["Mon","Tue","Wed"])

DaysB = set(["Wed","Thu","Fri","Sat","Sun"])

AllDays = DaysA|DaysB

print(AllDays)   
Output: {'Fri', 'Sat', 'Wed', 'Thu', 'Tue', 'Mon', 'Sun'}  
  
#Intersection of Sets

The intersection operation on two sets produces a new set containing only the

common elements from both the sets.

Ex:- DaysA = set(["Mon","Tue","Wed"])

DaysB = set(["Wed","Thu","Fri","Sat","Sun"])

AllDays = DaysA & DaysB

print(AllDays)

Output: {'Wed'}

#Difference of Sets

The difference operation on two sets produces a new set containing only the elements

from the first set and none from the second set.

Ex:- DaysA = set(["Mon","Tue","Wed"])

DaysB = set(["Wed","Thu","Fri","Sat","Sun"])

AllDays = DaysA - DaysB

print(AllDays)

Output: {'Mon', 'Tue'}  
  
5.DICTIONARY:

Intersection of Sets

The intersection operation on two sets produces a new set containing only the

common elements from both the sets.

Ex:- DaysA = set(["Mon","Tue","Wed"])

DaysB = set(["Wed","Thu","Fri","Sat","Sun"])

AllDays = DaysA & DaysB

print(AllDays)

Output: {'Wed'}

Difference of Sets

The difference operation on two sets produces a new set containing only the elements

from the first set and none from the second set.

Ex:- DaysA = set(["Mon","Tue","Wed"])

DaysB = set(["Wed","Thu","Fri","Sat","Sun"])

AllDays = DaysA - DaysB

print(AllDays)

Output: {'Mon', 'Tue'}  
  
◼ Accessing Values in a Dictionary

Ex:- dict = {'Name': ‘Rama', 'Age': 6, 'Class': 'First'}

print("Name:", dict['Name'])

print("Age:", dict['Age'])

Output: Name: Rama

Age: 6

◼ Updating Dictionary

We can update a dictionary by adding a new entry or a key-value pair, modifying

an existing entry.

Ex:- dict['Age'] = 8; # update existing entry

dict['School'] = "DPS School"; # Add new entry

print(dict)

Output: {'Name': ‘Rama', 'Age': 8, 'Class': 'First‘, ‘School': "DPS School"}