Practice 1 (print List)

In [26]:

```
employee=["Asad",102,"USA"]
Dep1=["CS",10]
Dep2=["IT",11]
HOD_CS=[1,"MR Ahmed"]
HOD_IT=[2,"Mr Nisar"]
print("Employee Data")
print(" Name  :  %s,   ID   :  %d,   Country   :    %s"  %(employee[0],  employee[1],  employee[2]))

print("Printing Departments   ")

print("Department 1:\nName:%s , ID:  %d\nDepartment 2:\n  Name : %s , ID:  %d"%(Dep1[0],Dep1[1], Dep2[0],Dep2[1]))

print("HOD DETAILS")
print(" CS HOD NAME: %s  , CS HOD ID: %d  "%(HOD_CS[1],HOD_CS[0]))

print("IT HOD Name: %s , IT HOD ID: %d "%(HOD_IT[1], HOD_IT[0]))

print(type(employee),type(Dep1),type(Dep2),type(HOD_CS),type(HOD_IT))

print("employee  Country: ", employee[2])

print("Name of IT HOD: ", HOD_IT[1])
```

```
Employee Data
 Name  :  Asad,   ID   :  102,   Country   :    USA
Printing Departments
Department 1:
Name:CS , ID:  10
Department 2:
  Name : IT , ID:  11
HOD DETAILS
 CS HOD NAME: MR Ahmed  , CS HOD ID: 1
IT HOD Name: Mr Nisar , IT HOD ID: 2
<class 'list'> <class 'list'> <class 'list'> <class 'list'> <class 'list'>
employee  Country:  USA
Name of IT HOD:  Mr Nisar
```

Practice 2 (slicing)

In [30]:

```
list = [1,2,3,4,5,6,7,8,9,10]
print(list [1:4])
print(list [1::4])#i didnt undertstand this
print(list [5])
print(list [-7])
print(list [-7:-2])#print 4_8 using minus
print(list [-2:])
print(list [::2])#print odd
print(list [1:2])
print(list [1::2])#print even

print(list[0:4])
print(list[2:5])
print(list[2:7:2]) #???
print("Print all the odd number from the list ", list[1::2])
print("print Even number:  ",  list[0::2])
list2=[1,2,3,4]
print(list2[-1])
print(list2[-3])
print(list2[-4::-1])#????

list1=[1,2,3,4,5,6,7,8,9,10]
print(list1)
list1[2]=11;
print(list1)

#adding multiple elements:
list1[1:3]=[12,13,14]
print(list1)

list1[-4]=25;
print(list1)

list1[-1]=15;
print(list1)
```

```
[2, 3, 4]
[2, 6, 10]
6
4
[4, 5, 6, 7, 8]
[9, 10]
[1, 3, 5, 7, 9]
[2]
[2, 4, 6, 8, 10]
[1, 2, 3, 4]
[3, 4, 5]
[3, 5, 7]
Print all the odd number from the list  [2, 4, 6, 8, 10]
print Even number:   [1, 3, 5, 7, 9]
4
2
[1]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 11, 4, 5, 6, 7, 8, 9, 10]
[1, 12, 13, 14, 4, 5, 6, 7, 8, 9, 10]
[1, 12, 13, 14, 4, 5, 6, 25, 8, 9, 10]
[1, 12, 13, 14, 4, 5, 6, 25, 8, 9, 15]
```

In [35]:

```
list1=["aima",123,"21313422","ABC sector"]
print (list1)

list1.insert(0,11) # use inset function , here 0 is the index no and the second value is to be added in the list
print(list1)

list1.insert(3, "Pak Army")
print(list1)

list1.remove(123)# remove method use to delete  by value
print(list1)

list1.pop(3)#pop method use to delete by using index number
print(list1)

list1.pop()
print(list1)

del list1[0]# delete (del) is keyword use delete the value using index
print(list1)

list1.clear()
print(list1)
#To append elements from another list to the current list, use the extend() method.
fruitbasket =["Banana " ,"Apple"]
buyFruits =["Grapes" ,"Watermelon"]
fruitbasket.extend(buyFruits);
print(fruitbasket);


#print all items one by one using loops
listt = [1,2,3,4,5]
for x in listt:
    print(x)

for y in range(len(listt)):
   print(listt[y])
```

```
['aima', 123, '21313422', 'ABC sector']
[11, 'aima', 123, '21313422', 'ABC sector']
[11, 'aima', 123, 'Pak Army', '21313422', 'ABC sector']
[11, 'aima', 'Pak Army', '21313422', 'ABC sector']
[11, 'aima', 'Pak Army', 'ABC sector']
[11, 'aima', 'Pak Army']
['aima', 'Pak Army']
[]
['Banana ', 'Apple', 'Grapes', 'Watermelon']
1
2
3
4
5
```

Practice 3 (Tuple)

In [2]:

```
#There are four collection data types in the Python programming language:
#List is a collection which is ordered and changeable. Allows duplicate members.

#Tuple is a collection which is ordered and unchangeable. Allows duplicate members
collectionintupple = ("ab" ,"cd" ,"Ef" ,"gh" , "hi", "jk", "lm")
print(collectionintupple)
#When we say that tuples are ordered, it means that the items have a defined order, and that order will not change.
#Since tuples are indexed, they can have items with the same value so it can have duplicate values:
thistuple = ("apple", "banana", "cherry", "apple", "cherry")
print(thistuple)
#Print the number of items in the tuple:
print(len(collectionintupple))
#To create a tuple with only one item, you have to add a comma after the item, otherwise Python will not recognize it as a tuple.
n1=(1)
print(n1)
print(type(n1))

n2=(2, )
print(n2)
print(type(n2))
#String, int and boolean data types also different data type:
tuple1 = ("apple", "banana", "cherry")
tuple2 = (1, 5, 7, 9, 3)
tuple3 = (True, False, False)
tuple4 = ("abc", 34, True, 40, "male")
print(tuple1,tuple2 , tuple3 ,tuple4)
#It is also possible to use the tuple() constructor to make a tuple.
n5=tuple((12,"AB" ))
print(n5)
print(type(n5))

#acces value from tupple
thistuple = ("apple", "banana", "cherry")
print(thistuple[1])
print(thistuple[-1])

thistuple1 = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple1[2:5])
print(thistuple1[2::])
print(thistuple1[:4])
print(thistuple1[2:])
print(thistuple1[-4:-1])

#Check if "apple" is present in the tuple:
thistuple3 = ("apple", "banana", "cherry")
if "lolo" in thistuple:
  print("Yes, 'apple' is in the fruits tuple")
else:
  print("No it doesn't exist")

#update tupple
#Once a tuple is created, you cannot change its values. Tuples are unchangeable, or immutable as it also is called.
#But there is a workaround. You can convert the tuple into a list, change the list, and convert the list back into a tuple.
a=(1,2,3,4,5)
b=list(a)
b[1]=9
print(b)
print(type(b))

#Since tuples are immutable, they do not have a built-in append() method, but there are other ways to add items to a tuple.

#1. Convert into a list: Just like the workaround for changing a tuple, you can convert it into a list, add your item(s), and convert it back into a tuple.
number=(1,2,3,4,5,6,7,8)
print(number)
mynumlist=list(number)
mynumlist.append(9)
number=tuple(mynumlist)
print(number)

#2. Add tuple to a tuple. You are allowed to add tuples to tuples, so if you want to add one item, (or many), create a new tuple with the item(s), and add it to the existing tuple:
n1=(1,2,3)
print(n1)
n2=(3,4,5)
print(n2)
n1+=n2
print(n1)
#delete in tupple is noty possible again this is how we can perform deletation
n3=(10,12,14,16,18,20)
print(n3)
n4=list(n3)
n4.remove(12)
#del n4[0]
#print((n4))
n3=tuple(n4)
print(n3)

#unpacking tupple

list1=(100,200,300,400)
(green,yellow,red,purple)=list1
print(green)
print(yellow)
print(red)
print(purple)
#If the number of variables is less than the number of values, you can add an * to the variable name and the values will be assigned to the variable as a list:
fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")
(ab,cd, *ef)=fruits
print(ab)
print(cd)
print(ef)

(ab, *cd, ef)=fruits
print(ab)
print(cd)
print(ef)

#loop through a tuple
print("printing the tupple using loop")
for x in fruits:
  print(x)

#using loop through index number:
print("Using loop through index number")
for y in range(len(fruits)):
  print(fruits[y])

print("Printing using while loop")
i=0
while i<len(fruits):
  print(fruits[i])
  i+=1
#join tupple
print("Join tupple")
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2
print(tuple3)

print("Multiply tuple: ")
fruits = ("apple", "banana", "cherry")
mytuple = fruits * 2
print(mytuple)

#method 1 count repetation
thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)

x = thistuple.count(5)

print(x)
#method 2 index tells the position of value
x=thistuple.index(8)
print(x)
#Set is a collection which is unordered, unchangeable*, and unindexed. No duplicate members.
#Dictionary is a collection which is ordered** and changeable. No duplicate members.
```

```
('ab', 'cd', 'Ef', 'gh', 'hi', 'jk', 'lm')
('apple', 'banana', 'cherry', 'apple', 'cherry')
7
1
<class 'int'>
(2,)
<class 'tuple'>
('apple', 'banana', 'cherry') (1, 5, 7, 9, 3) (True, False, False) ('abc', 34, True, 40, 'male')
(12, 'AB')
<class 'tuple'>
banana
cherry
('cherry', 'orange', 'kiwi')
('cherry', 'orange', 'kiwi', 'melon', 'mango')
('apple', 'banana', 'cherry', 'orange')
('cherry', 'orange', 'kiwi', 'melon', 'mango')
('orange', 'kiwi', 'melon')
No it doesn't exist
[1, 9, 3, 4, 5]
<class 'list'>
(1, 2, 3, 4, 5, 6, 7, 8)
(1, 2, 3, 4, 5, 6, 7, 8, 9)
(1, 2, 3)
(3, 4, 5)
(1, 2, 3, 3, 4, 5)
(10, 12, 14, 16, 18, 20)
(10, 14, 16, 18, 20)
100
200
300
400
apple
banana
['cherry', 'strawberry', 'raspberry']
apple
['banana', 'cherry', 'strawberry']
raspberry
printing the tupple using loop
apple
banana
cherry
strawberry
raspberry
Using loop through index number
apple
banana
cherry
strawberry
raspberry
Printing using while loop
apple
banana
cherry
strawberry
raspberry
Join tupple
('a', 'b', 'c', 1, 2, 3)
Multiply tuple:
('apple', 'banana', 'cherry', 'apple', 'banana', 'cherry')
2
3
```

In [1]:

```
#Tuple unchangeable,ordered,indexed.....
mytuple=("Arham",123)
print(mytuple)
print(mytuple[0])
print(type(mytuple))

#unpacking Tuple
mytuple2=("ab","cd","ef","gh")
(green,*yellow,red)=mytuple2;
print(green,yellow,red)

#Tuple is index it means it can have dupilicate values
tuple1=(1,2,3,4,5,6,7,7,2)

temp=list(tuple1)
print(temp)
temp.remove(7);
print(temp)
tuple1=tuple(temp)
print(tuple1)
```

```
('Arham', 123)
Arham
<class 'tuple'>
ab ['cd', 'ef'] gh
[1, 2, 3, 4, 5, 6, 7, 7, 2]
[1, 2, 3, 4, 5, 6, 7, 2]
(1, 2, 3, 4, 5, 6, 7, 2)
```

Practice 4(Set)

In [4]:

```
#Set is a collection which is unordered, unchangeable*, and unindexed. No duplicate members.
set1={1,3}
print(set1)
print(type(set1))
#Sets cannot have two items with the same value.
set2={12,12,13,1,4}
print(set2)
# The values True and 1 are considered the same value in sets, and are treated as duplicates:
set3={1,2,True , 14,19}
print(set3)
#Note: The values False and 0 are considered the same value in sets, and are treated as duplicates:
set4={2,False , 14,19,0}
print(set4)

#To determine how many items a set has, use the len() function.
print(len(set4))
#A set can contain different data types:
set5 = {"apple", "banana", "cherry"}
set6 = {1, 5, 7, 9, 3}
set7 = {True, False, False}
set8 = {"abc", 34, True, 40, "male"}

print(set5,set6,set7,set8)
#It is also possible to use the set() constructor to make a set.
ab=set((1,2))
print(ab)

#acces of set
print("Get value using for loop")
for x in set8:
  print(x)
print("check whether male exist in list or not")
if "male" in set8:
  print("yes")
else:
  print("no")

print(34 in set8)

print('banana' not in set8)

#Once a set is created, you cannot change its items, but you can add new items.
set8.add("ABC sector")
print(set8)
#To add items from another set into the current set, use the update() method.
set8.update(set7)
print(set8)

#methods for delete in set If the item to remove does not exist, remove() will raise an error
#1 Remove
set8.remove("male")
print(set8)

#2 discard  If the item to remove does not exist, discard() will NOT raise an error.
set8.discard(40)
print(set8)
#3 pop() Note: Sets are unordered, so when using the pop() method, you do not know which item that gets removed.
set8.pop()
print(set8)
#4 clear()
set8.clear()
print(set8)

#join
#The union() and update() methods joins all items from both sets.
s1 = {"a", "b", "c"}
s2 = {1, 2, 3}

s3 = s1.union(s2)
print(s3)
#You can use the | operator instead of the union() method, and you will get the same result.

output = s1| s2
print(output)
#Join multiple sets with the union() method:

s3 = {"John", "Elena"}
s4 = {"apple", "bananas", "cherry"}

myset = s1.union(s2, s3, s4)
print(myset)

#or
myset1=s1|s2|s3|s4
print(myset1)

#you can join set and tuple using union
x = {"a", "b", "c"}
y = (1, 2, 3)

z = x.union(y)
print(z)

#The update() method inserts all items from one set into another.
#The update() changes the original set, and does not return a new set.
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}

set1.update(set2)
print(set1)
#The intersection() method will return a new set, that only contains the items that are present in both sets.
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}

set3 = set1.intersection(set2)
print(set3)
#You can use the & operator instead of the intersection() method, and you will get the same result.
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}

set3 = set1 & set2
print(set3)

#The intersection_update() method will also keep ONLY the duplicates, but it will change the original set instead of returning a new set.
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}

set1.intersection_update(set2)

print(set1)

set1 = {"apple", 1,  "banana", 0, "cherry"}
set2 = {False, "google", 1, "apple", 2, True}

set3 = set1.intersection(set2)

print(set3)

#The difference() method will return a new set that will contain only the items from the first set that are not present in the other set.
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}

set3 = set1.difference(set2)

print(set3)

#or
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}

set3 = set1 - set2
print(set3)

#or
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}

set1.difference_update(set2)

print(set1)
#The symmetric_difference() method will keep only the elements that are NOT present in both sets.
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}

set3 = set1.symmetric_difference(set2)

print(set3)

#You can use the ^ operator instead of the symmetric_difference() method, and you will get the same result.
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}

set3 = set1 ^ set2
print(set3)
#The symmetric_difference_update() method will also keep all but the duplicates, but it will change the original set instead of returning a new set.
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}

set1.symmetric_difference_update(set2)

print(set1)
```

```
{1, 3}
<class 'set'>
{1, 13, 4, 12}
{1, 2, 19, 14}
{False, 2, 19, 14}
4
{'cherry', 'banana', 'apple'} {1, 3, 5, 7, 9} {False, True} {True, 34, 'male', 40, 'abc'}
{1, 2}
Get value using for loop
True
34
male
40
abc
check whether male exist in list or not
yes
True
True
{True, 34, 'ABC sector', 'male', 40, 'abc'}
{False, True, 34, 'ABC sector', 'male', 40, 'abc'}
{False, True, 34, 'ABC sector', 40, 'abc'}
{False, True, 34, 'ABC sector', 'abc'}
{True, 34, 'ABC sector', 'abc'}
set()
{1, 2, 'c', 3, 'a', 'b'}
{1, 2, 'c', 3, 'a', 'b'}
{1, 2, 3, 'Elena', 'a', 'b', 'cherry', 'c', 'John', 'apple', 'bananas'}
{1, 2, 3, 'Elena', 'a', 'b', 'cherry', 'c', 'John', 'apple', 'bananas'}
{1, 2, 3, 'a', 'b', 'c'}
{1, 2, 'c', 3, 'a', 'b'}
{'apple'}
{'apple'}
{'apple'}
{False, 1, 'apple'}
{'cherry', 'banana'}
{'cherry', 'banana'}
{'cherry', 'banana'}
{'microsoft', 'google', 'cherry', 'banana'}
```

```
{'microsoft', 'google', 'cherry', 'banana'}
{'microsoft', 'google', 'cherry', 'banana'}
```