In [1]:

```
#List
# If we want to represent a group of indivdual objects as a single entity where insertion order is preserved and
duplicates
# are allowed, then we should go for list.
# 1.Insertion order is preserved.
# 2.duplicates are allowed
# 3.heterogenous objects are allowed
# 4.list is dynamic becasue based on our requirement we can increase and decrease the size.
# 5.In list elements will be placed within square brackets and with comma seperator.
# 6.We can differentiate duplicate elements by using index and we can preserve insertion order by using index.Hen
ce index
#   place important role in list.
# 7.python supports both positive and negative indexes. +ve index means left to right. -ve index means right to l
eft.
# 8. list objects are mutable. we can change the content.
```

In [2]:

```
#creating list of employees and print the employees
employees=['aravind','rahul','jake','sunny']
print(employees)
```

```
['aravind', 'rahul', 'jake', 'sunny']
```

In [3]:

```
# To get length of list
print(len(employees))
```

```
4
```

In [6]:

```
#To access the elements of list.
# 1. By using index
print(employees[0])#index value starts with 0
print(employees[3])
print(employees[-1] ) #we can last item of list using negative index -1
```

```
aravind
sunny
sunny
```

In [11]:

```
# to access the elements of list
# 2. By using slice operator
# syntax:list2=list1[start:stop:step]
# 1.start ==> It indicates the index where slice has to start. default value is 0
# 2.stop ==> It indicates the index where slice has to stop. default value is max allowed index of list ie length
of list
# 3.step ==> increment value. default value is 1.
employees=['aravind','rahul','jake','sunny']
print(employees[0:2]) # start index is 0 and stop index is 2 where 2 is exclusive.
print(employees[0:3:2])# start index is 0 and stop index is 3 and increment value is 2
print(employees[:2]) # it will start from index 0 and stop index is 2
print(employees[2:]) #start index is 2 and stop index is upto last value in list.
```

```
['aravind', 'rahul']
['aravind', 'jake']
['aravind', 'rahul']
['jake', 'sunny']
```

In [12]:

```
#to add employee to the employeee list
# use append() function - we can use append() function to add item at end of list.
employees=['aravind','rahul','jake','sunny']
employees.append('mike')
print(employees)
```

```
['aravind', 'rahul', 'jake', 'sunny', 'mike']
```

In [13]:

```
#insert() -to insert item at specified index position.
employees.insert(2,'kohli') #to add the employee kohli in 2nd index position.
print(employees)
```

```
['aravind', 'rahul', 'kohli', 'jake', 'sunny', 'mike']
```

In [14]:

```
#extend() - To add multiple values in the list or To add all items of one list to another list.
employees=['aravind', 'rahul', 'kohli', 'jake', 'sunny', 'mike']
employee_1=['rohith','ronaldo','chrishemsworth']
employees.extend(employee_1)
print(employees)
```

```
['aravind', 'rahul', 'kohli', 'jake', 'sunny', 'mike', 'rohith', 'ronaldo', 'chrishemsworth']
```

In [15]:

```
#remove function() - we can use remove function to remove specified item from list.
# If the item present multiple times then only first occurrence will be removed.
employees.remove('aravind')
print(employees)

#note: If the specifief item is not present in list. we will get ValueError.
```

```
['rahul', 'kohli', 'jake', 'sunny', 'mike', 'rohith', 'ronaldo', 'chrishemsworth']
```

In [16]:

```
#pop() function - It removes and returns last element of list.
#This is the only function which manipulates the list and returns some element.
print(employees.pop())

#Note: If the list is empty the pop() function raises IndexError
```

```
chrishemsworth
```

In [18]:

```
#reverse() -we  can use to reverse() order of employees in a list.
employees=['aravind', 'rahul', 'kohli', 'jake', 'sunny', 'mike']
employees.reverse()
print(employees)
```

```
['mike', 'sunny', 'jake', 'kohli', 'rahul', 'aravind']
```

In [21]:

```
#sort() function - In list by default insertion order is preserved.
# If want to sort the elements of list according to default natural sorting order then we should go for sort() me
thod
# for numbers ==> default natural sorting order is Ascending order.
# for strings ==> default natural sorting order is Alphabetic order.

# Note: To use sort() function, compulsory list should contain homogenous elements. otherwise we will get TypeErr
or.
employees=['aravind', 'rahul', 'kohli', 'jake', 'sunny', 'mike']
employees.sort()
print(employees)

#To sort in reverse of default natural sorting order
employees.sort(reverse=True)
print(employees)

#sorted
sorted_employees=sorted(employees)
print(sorted_employees) #It will not alter the original list that is the advantage of sorted()
```

```
['aravind', 'jake', 'kohli', 'mike', 'rahul', 'sunny']
['sunny', 'rahul', 'mike', 'kohli', 'jake', 'aravind']
['aravind', 'jake', 'kohli', 'mike', 'rahul', 'sunny']
```

In [27]:

```
# to get minimum value in the list
nums=[1,24,6,7,8,9]
print(min(nums))

# to get maximum value in the list
print(max(nums))

# count() - It returns the no of occurrences of specified item in the list.
print(nums.count(1))

#index() - returns the index of first occurrence of the specified item
print(nums.index( 7))

#sum - to get the sum of values
print(sum(nums))
```

```
1
24
1
3
55
```

In [29]:

```
#Note if the specified item not present in the list then we will get ValueError.
# Hence before index() method we have to check whether element present in the list or not by using in operator
# in operator
employees=['aravind', 'rahul', 'kohli', 'jake', 'sunny', 'mike']
print('rahul' in employees)
print('rahull' not in employees)
```

```
True
True
```

In [35]:

```
#print each employee in the list by using for loop
employees=['aravind', 'rahul', 'kohli', 'jake', 'sunny', 'mike']
for member in employees:
    print(member)
# we can access the index and the value by using enumerate funciton()
# enumerate function returns two values it returns the index and the value.
for index, member in enumerate(employees):
    print(index, member)
for index, member in enumerate(employees,start=1): #start the index value with 1
    print(index, member)
```

```
aravind
rahul
kohli
jake
sunny
mike
0 aravind
1 rahul
2 kohli
3 jake
4 sunny
5 mike
1 aravind
2 rahul
3 kohli
4 jake
5 sunny
6 mike
```

In [37]:

```
# we will want to turn our list to strings seperated by a certain value now we are going to
# use string method called join and we are going to pass our list as the argument.
employees=['aravind', 'rahul', 'kohli', 'jake', 'sunny', 'mike']
employees_str=', '.join(employees)
print(employees_str)
```

```
aravind, rahul, kohli, jake, sunny, mike
```

In [40]:

```
#mathematical operations of list objects
# concatenation operator - we can use + to concatenate 2 lists into a single list.
a=[10,44,6]
b=[45,64,7]
c=a+b
print(c)
# Repetetion operator - we can use repetetion operator * to repeat elements of list specified no of times
x=[10,23,56]
y=x*3
print(y)
```

```
[10, 44, 6, 45, 64, 7]
[10, 23, 56, 10, 23, 56, 10, 23, 56]
```

In [41]:

```
#clear function - we can use clear() function to remove all elements of list
n=[10,20,4,6]
n.clear()
print(n)
```

```
[]
```

In [43]:

```
#comparing list objects
# We can use comparison operators for list objects
x=['shyam','ram']
y=['krish','shakesphere']
print(x==y)
print(x!=y)
```

```
False
True
```

In [44]:

```
#nested lists
#Sometimes we can take one list inside another list. such type of lists are called nested lists.
n=[10,20,[30,40]]
print(n)
print(n[0])
print(n[2][0])
print(n[2][1])
```

```
[10, 20, [30, 40]]
10
30
40
```

In [49]:

```
# Aliasing and cloning of objects
# Aliasing -> The process of giving another reference variable to existing list.
# Cloning -> The process f creating exactly duplicated independent object.
# We can implement cloning by using slice operator or by using copy() function.

#aliasing
x=[10,20,30,40]
y=x
print(id(x))
print(id(y))
y[1]=888
print(x)

#cloning - By using slice operator
x=[10,20,30,40]
y=x[:]
y[1]=888
print(x)
print(y)

#cloning - By using copy() function
x=[10,20,30,40]
y=x.copy()
y[1]=888
print(x)
print(y)

#difference between = operator and copy() function?
# = operator meant for aliasing.
# copy() function meant for cloning.
```

```
2210446685896
2210446685896
[10, 888, 30, 40]
[10, 20, 30, 40]
[10, 888, 30, 40]
[10, 20, 30, 40]
[10, 888, 30, 40]
```