---------- Lists ----------

A list is a collection of items in a particular order. You can make a list that includes the letters of the alphabet, the digits from 0–9, or the names of all the people in your family. You can put anything you want into a list the items in your list don't have to be related in any particular way. Because a list usually contains more than one element, it's a good idea to make the name of your list plural, such as letters, digits, or names. In Python, square brackets ([]) indicate a list, and individual elements in the list are separated by commas.

Creating and accessing Lists

In [1]:

```
family = ['liz', 'emma', 'mom', 'dad']
height = [1.73, 1.68, 1.71, 1.89 ]

print ("family[0]: ", family[0], end="\n\n")
print ("height: ", height[1:3])
```

```
family[0]:  liz

height:  [1.68, 1.71]
```

Iterating over a list

In [2]:

```
for i in family:
    print(i)
```

```
liz
emma
mom
dad
```

List of Lists

In [3]:

```
fam = [['liz', 1.73], ['emma', 1.68], ['mom', 1.71], ['dad', 1.89]]
print(fam)
```

```
[['liz', 1.73], ['emma', 1.68], ['mom', 1.71], ['dad', 1.89]]
```

In [4]:

```
for i in fam :
    for j in i :
        print(j,end=" ")
    else:
        print()
```

```
liz 1.73
emma 1.68
mom 1.71
dad 1.89
```

Indexing and slicing

In [5]:

```
fam = ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]

print("Print 1-3 index elements --> nos[1:4] : ", fam[1:4], end="\n\n")
print("Print all after index 2 --> nos[2:] :", fam[2:], end="\n\n")
print("Print all before 4 --> nos[:4] :", fam[:4], end="\n\n")
```

```
Print 1-3 index elements --> nos[1:4] :  [1.73, 'emma', 1.68]

Print all after index 2 --> nos[2:] : ['emma', 1.68, 'mom', 1.71, 'dad', 1.8
9]

Print all before 4 --> nos[:4] : ['liz', 1.73, 'emma', 1.68]
```

reverse indexing

In [6]:

```
fam = ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]

print("Print last no in the list --> nos[-1] : ", fam[-1], end="\n\n")
print("Print last 4 elements in the list --> nos[-4:] :", fam[-4:], end="\n\n")
```

```
Print last no in the list --> nos[-1] :  1.89

Print last 4 elements in the list --> nos[-4:] : ['mom', 1.71, 'dad', 1.89]
```

Updating Lists

In [7]:

```
fam = ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
print ("Printing the original list : ",fam,end="\n\n")

fam[5] = 1.70
print ("Printing the updated list : ",fam)
```

```
Printing the original list :  ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'da
d', 1.89]

Printing the updated list :  ['liz', 1.73, 'emma', 1.68, 'mom', 1.7, 'dad',
1.89]
```

In [8]:

```
fam = ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]

print(fam[2], type(fam[2])) # returns a single value

print(fam[:2], type(fam[:2])) # always returns a list
```

```
emma <class 'str'>
['liz', 1.73] <class 'list'>
```

Delete List Elements

In [8]:

```
fam = ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
print ("Printing the original list : ",fam,end="\n\n")

del fam[2]
del fam[:2]
del fam
print ("After deleting value at index 1 : ", fam)
```

```
Printing the original list :  ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'da
d', 1.89]

After deleting value at index 1 :  ['emma', 1.68, 'mom', 1.71, 'dad', 1.89]
```

Built-in Python functions

In [17]:

```
mylist = [10, 20, 30, 15]
print(len(mylist)," -- Gives the total length of the list.")

 # Applicable to numeric datatypes only:
print(max(mylist)," -- Returns item from the list with max value.")
print(min(mylist)," -- Returns item from the list with min value.")
print(sum(mylist)," -- Returns the summation of all nunmerical elements in the list")
```

```
4  -- Gives the total length of the list.
30  -- Returns item from the list with max value.
10  -- Returns item from the list with min value.
75  -- Returns the summation of all nunmerical elements in the list
```

List Methods

In [12]:

```
mylist = [40, 10, 20, 30, 15, 10]

print ("Printing the original list : ",mylist,end="\n\n")

# Returns count of how many times obj occurs in list
print("count of object 10 the list : ", mylist.count(10))

 # Returns the lowest index in list for that obj appears
print("Lowest index of object 10 : ", mylist.index(10))
```

```
Printing the original list :  [40, 10, 20, 30, 15, 10]

count of object 10 the list :  2
Lowest index of object 10 :  1
```

Append object to list

In [16]:

```
mylist = [40, 10, 20, 30, 15, 10]
print("Original list : ", mylist, end="\n\n")

mylist.append(70)
mylist.append("Hello")
mylist.append([1,2,3,4])
mylist[-1].append("abc")

print("List after appending 70 : ", mylist)
```

```
Original list :  [40, 10, 20, 30, 15, 10]

List after appending 70 :  [40, 10, 20, 30, 15, 10, 70, 'Hello', [1, 2, 3,
4, 'abc']]
```

Extending Lists

In [18]:

```
fam_ext = fam + ["me", 1.79]
print(fam_ext)
```

```
['liz', 1.73, 'emma', 1.7, 'mom', 1.71, 'dad', 1.89, 'me', 1.79]
```

Append contents of seq to list

In [18]:

```
mylist = [40, 10, 20]
print("Original list : ", mylist, end="\n\n")

# mylist.extend(70)
mylist.extend([1,2,5])
mylist.extend("Hello")
mylist.extend(range(50,100,10))
print( mylist)
```

```
Original list :  [40, 10, 20]

[40, 10, 20, 1, 2, 5, 'H', 'e', 'l', 'l', 'o', 50, 60, 70, 80, 90]
```

Inserts object obj into list at offset index

In [13]:

```
mylist = [40, 10, 20, 30, 15, 10]
print("Original list : ", mylist, end="\n\n")

# mylist[4]=85 # this will replace element at index 4

# insert() will add element at index 4
mylist.insert(4, 85)
print("List after inserting 85 at index 4 : ", mylist)
```

```
Original list :  [40, 10, 20, 30, 15, 10]

List after inserting 85 at index 4 :  [40, 10, 20, 30, 85, 15, 10]
```

Removinng element from lists

In [14]:

```
mylist = [40, 10, 20, 30, 15, 10]
print("Original list : ", mylist, end="\n\n")

x = mylist.pop()  # Removes and returns last object
print("List after pop : ", mylist, end="\n\n")

y = mylist.pop(2)  # Removes the object at given index from list
print("List after pop value at index 2 : ", mylist, end="\n\n")

mylist.remove(30)  # Removes object obj from list
print("List after removing 20 : ", mylist)
```

```
Original list :  [40, 10, 20, 30, 15, 10]

List after pop :  [40, 10, 20, 30, 15]

List after pop value at index 2 :  [40, 10, 30, 15]

List after removing 20 :  [40, 10, 15]
```

Organizing a List

Often, your lists will be created in an unpredictable order, because you can't always control the order in which your users provide their data. Although this is unavoidable in most circumstances, you'll frequently want to present your information in a particular order. Sometimes you'll want to preserve the original order of your list, and other times you'll want to change the original order. Python provides a number of different ways to organize your lists, depending on the situation.

Sorting and Reversing using reverse() and sort() will make changes to the original list object.

Sort and Reverse

In [20]:

```
mylist = [40, 10, 20, 30, 15, 10]
print("Original list : ", mylist, end="\n\n")

mylist.reverse()  # Reverses objects of list in place
print("List after reversing : ", mylist, end="\n\n")

mylist.sort(reverse=True)  # Sorts objects of list, use compare func if given
print("List after sort : ", mylist, end="\n\n")
```

```
Original list :  [40, 10, 20, 30, 15, 10]

List after reversing :  [10, 15, 30, 20, 10, 40]

List after sort :  [40, 30, 20, 15, 10, 10]
```

Sorting a List Temporarily with the sorted() Function

To maintain the original order of a list but present it in a sorted order, you can use the sorted() function. The sorted() function lets you display your list in a particular order but doesn't affect the actual order of the list.

Sorting and Reversing using reversed() and sorted() will return new list objects without making changes to the original list object.

In [21]:

```
cars = ['bmw', 'audi', 'toyota', 'subaru']

print("Original list : ", cars,"\nSorted list : ",sorted(cars))

print("\nOriginal list : ", cars,"\nReversed list : ",list(reversed(cars)))
```

```
Original list :  ['bmw', 'audi', 'toyota', 'subaru']
Sorted list :  ['audi', 'bmw', 'subaru', 'toyota']

Original list :  ['bmw', 'audi', 'toyota', 'subaru']
Reversed list :  ['subaru', 'toyota', 'audi', 'bmw']
```

List works as a object reference

In [23]:

```
list1 = [1, 2, 3]
list2 = list1
print("List1 = ", list1," List2 = ", list2, end="\n\n")

list1[0] = 5

print("List1 = ", list1," List2 = ", list2, end="\n\n")
```

```
List1 =  [1, 2, 3]  List2 =  [1, 2, 3]

List1 =  [5, 2, 3]  List2 =  [5, 2, 3]
```

To make to separate copy lists (Create a shadow copy)

In [45]:

```
list1 = [1, 2, 3]
list2 = list1[:]
print("List1 = ", list1, "List2 = ", list2, end="\n\n")
list1[0] = 5
print("List1 = ", list1, "List2 = ", list2, end="\n\n")
```

```
List1 =  [1, 2, 3] List2 =  [1, 2, 3]

List1 =  [5, 2, 3] List2 =  [1, 2, 3]
```

List Comprehension

Example: Print list of squares of first 10 natural numbers

In [2]:

```
l = [i**2 for i in range(1,11)]
print(l)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Example: checking whether elements in a list are even or odd and printing both the lists together

In [23]:

```
l = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]

# Creating a list of only even numbers:
l1 = ["even" for i in l if i % 2 == 0]
print(l1,end="\n\n")

# checking whether elements in a list are even or odd and creating a new list
l2 = ["even" if i%2 ==0 else "odd" for i in l]

# Iterating over both the lists together:
for i,j in zip(l,l2):
    print(i,j,sep=" - ",end = ", ")
```

```
['even', 'even', 'even', 'even', 'even']

1 - odd, 2 - even, 3 - odd, 4 - even, 5 - odd, 6 - even, 7 - odd, 8 - even,
9 - odd, 0 - even,
```

Example simulating use of else-if in list comprehension

In [52]:

```
l = [1,5,3,2,7,1,6,2]
l1 = ['yes' if x==1 else 'no' if x==2 else 'idle' for x in l]
print(l1)
```

```
['yes', 'idle', 'idle', 'no', 'idle', 'yes', 'idle', 'no']
```

Example :

Prints all the vowels present in the word entered by user.

In [7]:

```
print(", ".join(sorted({j for j in (input("Enter a word : ")).lower() if j in "aeiou"})))
```

```
Enter a word : Aeroplane
a, e, o
```