```

Variable: can store only one value or one datatype

List: is a data structure to hold sequential collection of values in a single variable
    -Can have number of items
    -Dynamic: do not have to specify any fixed size, can easily add or remove elements
    -Heterogeneous: The elements of list can have mixed type of datas; even lists inside a list.
    -Each item in the list has an index value
    -Indexes range from 0-(n-1) where n=length of the list

```

In [1]:
```
#A list is respresented and created by enclosing
#All the elements inside a SQUARE bracket [] seperated by commas ','

#Creating empty list
create_list1 = []
print(create_list1)

#Creating list_of_integers
create_list2 = [1,2,3]
print(create_list2)

#Creating list of mixed data type
create_list3 = ['CSE110', 3.90, 21, True]
print(create_list3)

#Nested list
create_list4 = ['CSE110', 3.90, [1,3,3.95]]
print(create_list4)
#Note: A list can also contain lists

```

```
[]
[1, 2, 3]
['CSE110', 3.9, 21, True]
['CSE110', 3.9, [1, 3, 3.95]]
```

In [4]:

```
#Accessing list elements

#just like strings, list can also be accessed using indexing method
#Range: 0 to (n-1)
#Type: Index must be an integer
#IndexError: Index out of range. Different type will give Type error.

#Two indexing technique: Positive and Negative.
#Example:

alphabets_list=['A','B','C','D']

#positive index: A=0 B=1 C=2 D=3
#ends at (n-1)
#length=4 so, (n-1)=4-1=3

#negative index: A=-4 B=-3 C=-2 D=-1
#starts at -(Length of the list)
#ends at (-1)

print(alphabets_list[0])
print(alphabets_list[-1])

#The below lines will give error
#print(alphabets_list[5]) #-> list index out of range
#print(alphabets_list[1.0]) #-> index = float not possible

#IF THERE'S A LIST INSIDE A LIST, USE DOUBLE SQUARE

fruits_list=['Apple','Banana', ['PinaColada','Milkshake']]

print(fruits_list[2][0])
print(fruits_list[2][1])
print(fruits_list[2][-2])
print(fruits_list[2][-1])

```

```
A
D
PinaColada
Milkshake
PinaColada
```

```
Milkshake
```

In [6]:

```
#List Mutability

#Lists are mutable (changeable)
#Unlike string items in list, list items can easily be changed.
    #can easily change an existing element
    #can easily append/add new elements
    #can easily delete/remove elements

alphabets_list=['A','B','C','D']
print(alphabets_list)

#CHANGING ELEMENTS IN LIST
print(alphabets_list[2])
players_list[2] = 'Z'
print(alphabets_list[2])
#C is replaced by Z
#C is no longer in the list
print(alphabets_list)
```

```
['A', 'B', 'C', 'D']
C
Z
['A', 'B',    'Z'   , 'D']
```

In [5]:
```
#ADDING ELEMENTS IN LIST

#built in functions can be used to add elements

#add an element
#append(value) = to add one item at the end of the list

alphabets_list=['A','B','C','D']
#Syntax: list_name.append(value)
alphabets_list.append('E')
#alphabets_list.append('E', 'F') -> will give error
print(alphabets_list)

#append function is used when you want to add one value
```

```
['A', 'B', 'C', 'D', 'E']
```

In [7]:
```
#EXTENDING ELEMENTS IN LIST
#extend([values]) = to add several items to a list at once at the end

alphabets_list=['A','B','C','D']
#Syntax: list_name.extend([values])
alphabets_list.extend(['E', 'F', 'G'])
print(alphabets_list)
alphabets_list.extend(['H'])
print(alphabets_list)

#extend function is used when you want to add one or more than one values
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G']
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
```

In [6]:
```
#INSERTING ELEMENTS IN LIST AT DESIRED LOCATION
#insert(location, value) = to add one item to a list at a desired index location

alphabets_list=['A','B','C','D']
print(alphabets_list)
#Syntax: list_name.inser(location, value)
alphabets_list.insert(4,'H')
print(alphabets_list)
alphabets_list.insert(3,'Z')
print(alphabets_list)
```

```
['A', 'B', 'C', 'D']
['A', 'B', 'C', 'D', 'H']
['A', 'B', 'C', 'Z', 'D', 'H']
```

In [7]:
```
#DELETING ELEMENTS

#remove(val) = removes the first occurence of the item from the list
alphabets_list=['A','B','C', 'A', 'D']
print(alphabets_list)
#Syntax: list_name.remove(element)
alphabets_list.remove('A') #removes the first occurence only
print(alphabets_list)

```

```
['A', 'B', 'C', 'A', 'D']
['B', 'C', 'A', 'D']
```

In [8]:
```
#pop(index) = the index parameter is optional.
#Removes the last element from the list if no index provided
#else removes from the specified index.

alphabets_list=['A','B','C', 'A', 'D']
print(alphabets_list)

#Syntax: list_name.pop()
alphabets_list.pop()
#removes the last element as no index defined
print(alphabets_list)

#Syntax: list_name.pop(index)
alphabets_list.pop(2)
#removes the element from index 2
print(alphabets_list)
```

```
['A', 'B', 'C', 'A', 'D']
['A', 'B', 'C', 'A']
['A', 'B', 'A']
```

In [9]:
```
#clear() = removes all the items from the list

alphabets_list=['A','B','C', 'A', 'D']
print(alphabets_list)

alphabets_list.clear()
print(alphabets_list)
```

```
['A', 'B', 'C', 'A', 'D']
[]
```

In [10]:
```
#del- python keyword to remove specific items from the list
#or remove while list from the memory

alphabets_list=['A','B','C', 'A', 'D']
print(alphabets_list)

del alphabets_list[3] #Removes element from the specified index
print(alphabets_list)

# del alphabets_list #Removes the entire list
# print(alphabets_list)
# Output: NameError "aplphabets_list" not defined

```

```
['A', 'B', 'C', 'A', 'D']
['A', 'B', 'C', 'D']
```

In [11]:

```
#index function
#Returns the index number of the first occurrence of the specified element.

alphabets_list=['A', 'B', 'C', 'A', 'D']
#Syntax: list_name.count(element)
print(alphabets_list.index('A')) #RETURN FIRST OCCURENCE INDEX
print("=============================================")

#count
#Returns the number of times an element is in the list

numbers_list=[1, 2, 3, 3, 4, 4, 5, 5, 3]
#Syntax: list_name.count(element)
print(numbers_list.count(3))
print("=============================================")

#sort
#Reverse is optional.
#If reverse is not provided, the list will be sorted in asc order.
#If reverse is set to true, list will be sorted in desc order.
#Does not return any value; only True/False

fruits_list=['Apple','Banana','Cranberry','Dragonfruit']
#Syntax: list_name.sort()
fruits_list.sort() #reverse not provided
print(fruits_list)
#Syntax: list_name.sort(reverse=True)
fruits_list.sort(reverse=True) #reverse set to True
print(fruits_list)
print("=============================================")

#reverse
#Does not return any value. It reverses the original list.

tbbt_list=['Sheldon','Amy','Penny','Leonard']
#Syntax: list_name.reverse()
tbbt_list.reverse()
print(tbbt_list)
print("=============================================")
```

```
0
```

```
==========================================
3
==========================================
['Apple', 'Banana', 'Cranberry', 'Dragonfruit']
['Dragonfruit', 'Cranberry', 'Banana', 'Apple']
==========================================
['Leonard', 'Penny', 'Amy', 'Sheldon']
==========================================
```

In [76]:
```
# dir(list)= all functions available for the list
# alphabets_list=['A','B','C', 'A', 'D']
# dir(alphabets_list)
# dir(list)
```

In [12]:
```
#Concatenation:

list1 = [1,2,3]
list2 = [4,5,6]
print(list1 + list2)
```

```
[1, 2, 3, 4, 5, 6]
```

In [13]:
```
#Repitition:

list1 = [1,2,3]
print(list1 * 3)
```

```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

In [14]:
```
#Iteration:

list1 = [1,2,3]
for item in list1:
    print(item)
```

```
1
2
3
```

In [15]:
```
list1 = [1,2,3]
for item in list1:
    print(item, end='')
```

```
123
```

In [16]:
```
#Membership(Boolean)

list1 = [1, 2, 4]
print(4 in list1)
print(3 in list1)
```

```
True
False
```

In [18]:
```
#LIST SLICING

#list_name[start:end:step]

#start(inclusive) : specifies the starting index. Default=0
#end(exclusive) : specifies the ending index. Default=end of list (last index)
#step(optional) : specifies the increment. Default=+1

list1 = [1,2,3,4,5,6,7,8,12,13,4,3,2,1,4,5,6]
print(list1[2:6:2])
print(len(list1))
#starts from -16 as length=17
print(list1[-14: -2 :2]) #[4, 6, 8, 13, 3, 1]

print(list1[-4:-12:-2]) #[1, 3, 13, 8]

print(list1[::2])

```

```
[3, 5]
17
[4, 6, 8, 13, 3, 1]
[1, 3, 13, 8]
[1, 3, 5, 7, 12, 4, 2, 4, 6]
```

In [19]:
```
# list2 = [1,2,3,4,5,6,7,8,12,13,4,3,2,1,4,5,6]
list1 = ['A', 'B', 'C', 'D']
print(len(list1))
print(max(list1))
print(min(list1))
# print(sum(list1)) = will work only datatype=int, float
```

```
4
D
A
```

In [20]:
```
list1 = [1,2,3,4,5,6,7,8,12,13,4,3,2,1,4,5,6]
print(len(list1))
print(max(list1)) #sorts the max in case of numerical values
print(min(list1))
print(sum(list1))
```

```
17
13
1
86
```

In [22]:
```
listOfStr = ['hi', 'this', 'is', 'a', 'small', 'string', 'with', 'msg']
print(max(listOfStr))
```

```
with
```

In [23]:
```
listOfStr = ['HIASDFHJKGVBHNJM', 'I', 'AM', 'IRONMAN', 'IRONIRON']
print(max(listOfStr))
```

```
IRONMAN
```

In [24]:
```
listOfStr = ['hi', 'this', 'Is', 'A', 'Small', 'string', 'with', 'msg']
print(max(listOfStr))
```

```
with
```

```
Homework:

Exercise 1:
```

```
Reverse a list in Python

Given:
list1 = [100, 200, 300, 400, 500]

Expected output:
[500, 400, 300, 200, 100]

#=============================================================================================================

Exercise 2:
Concatenate two lists index-wise

Given:
list1 = ["M", "na", "i", "Ke"]
list2 = ["y", "me", "s", "lly"]

Expected output:
['My', 'name', 'is', 'Kelly']

#=============================================================================================================

Exercise 3:
Turn every item of a list into its square

Given:
numbers = [1, 2, 3, 4, 5, 6, 7]

Expected output:
[1, 4, 9, 16, 25, 36, 49]

#=============================================================================================================

Exercise 4:
Concatenate two lists in the following order

Given:
list1 = ["Hello ", "take "]
list2 = ["Dear", "Sir"]

Expected output:
['Hello Dear', 'Hello Sir', 'take Dear', 'take Sir']
```

```
#=========================================================================================================

Exercise 5:
Iterate both lists simultaneously

Given:
list1 = [10, 20, 30, 40]
list2 = [100, 200, 300, 400]

Expected output:
10 400
20 300
30 200
40 100

#=========================================================================================================

Exercise 6:
Remove empty strings from the list of strings

Given:
list1 = ["Mike", "", "Emma", "Kelly", "", "Brad"]

Expected output:
["Mike", "Emma", "Kelly", "Brad"]

#=========================================================================================================

Exercise 7:
Add new item to list after a specified item

Given:
list1 = [10, 20, [300, 400, [5000, 6000], 500], 30, 40]

Expected output:
[10, 20, [300, 400, [5000, 6000, 7000], 500], 30, 40]

#=========================================================================================================

Exercise 8:
Extend nested list by adding the sublist
```

```

Given:
list1 = ["a", "b", ["c", ["d", "e", ["f", "g"], "k"], "l"], "m", "n"]
#sub_list to add
sub_list = ["h", "i", "j"]

Expected output:
['a', 'b', ['c', ['d', 'e', ['f', 'g', 'h', 'i', 'j'], 'k'], 'l'], 'm', 'n']

#=========================================================================================================

Exercise 9:
Replace list's item with new value if found

Given:
list1 = [5, 10, 15, 20, 25, 50, 20]

Expected output:
[5, 10, 15, 200, 25, 50, 20]

#=========================================================================================================

Exercise 10:
Remove all occurrences of a specific item from a list.

Given:
list1 = [5, 20, 15, 20, 25, 50, 20]

Expected output:
[5, 15, 25, 50]

#=========================================================================================================
```