```python
# There are four types of data structures in python :
# 1. Lists => []
# 2. Dictionaries => {}
# 3. Sets => {}
# 4. Tuples => ()

# LIST


a = []              # Empty List
print(a)            # Output => []
print(type(a))      # Output => <class 'list'>
a = [10, 20, 30, "Coding Ideas!",  40, 50, 60]
print(a)            #Output => [10, 20, 30, 'Coding Ideas!', 40, 50,
60]
print(type(a))

#! append => It adds an element to the last position of the list.
a.append(100)
print(a)            #Output => [10, 20, 30, 'Coding Ideas!', 40, 50,
60, 100]

#! pop => It deletes the last element of the list.
a.pop()
print(a)            # Output => [10, 20, 30, 'Coding Ideas!', 40, 50,
60]

#! extend => It adds multiple elements starting from the last element
of the list.
a.extend([70, 80, 90, 100])
print(a)            # Output => [10, 20, 30, 'Coding Ideas!', 40, 50,
60, 70, 80, 90, 100]

#! remove => Deletes an element of the list by just typing the name of
the element.
a.remove("Coding Ideas!")
print(a)            # Output => [10, 20, 30, 40, 50, 60, 70, 80, 90,
100]

# Length
print(len(a))       # Output => 10
# Find Element
print(a[0])         # Output => 10
print(a[6])         # Output => 70

#! copy => Copies all the elements of a list to another list.
b = a.copy()
print(b)            # Output => [10, 20, 30, 40, 50, 60, 70, 80, 90,
100]
```

```python
#! clear =>  Deletes all the elements of a list.
print("****************************************")
print(b)           # Output => [10, 20, 30, 40, 50, 60, 70, 80, 90,
100]
b.clear()
print(b)           # Output => []
print(type(b))     # Output => <class 'list'>

#! del => Deletes the list from the root level.
print(a)
del a
print(a)           # Output => NameError: name 'a' is not defined.
```

```
[]
<class 'list'>
[10, 20, 30, 'Coding Ideas!', 40, 50, 60]
<class 'list'>
[10, 20, 30, 'Coding Ideas!', 40, 50, 60, [100, 20]]
[10, 20, 30, 'Coding Ideas!', 40, 50, 60]
[10, 20, 30, 'Coding Ideas!', 40, 50, 60, 70, 80, 90, 100]
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
10
10
70
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
****************************************
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
[]
<class 'list'>
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
c:\Users\PKVidyarthi\Desktop\Data Science\Notes\PythonList2.ipynb Cell
2 in <cell line: 46>()
     <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/PythonList2.ipynb#ch0000000?line=43'>44</a> print(a)
     <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/PythonList2.ipynb#ch0000000?line=44'>45</a> del a
---> <a
href='vscode-notebook-cell:/c%3A/Users/PKVidyarthi/Desktop/Data
%20Science/Notes/PythonList2.ipynb#ch0000000?line=45'>46</a> print(a)

NameError: name 'a' is not defined
```

```
# Comment Multiple Lines  => Lines will be commented which will be
written inside  ''' '''
# Shortcut Key => 1. Select Multiple Lines 2. press ctrl + /
'''
HELLO LEARNERS!
Welcome to Coding Ideas
'''
print("Hello World")
```

```
Hello World
```

```
#! sort => It will sort the elements of list in ascending order.
a = [40, 5, 4, 1, 11, 10.5, 17, 21, 25, 70]
print(a)                    # Output => [40, 5, 4, 1, 11, 10.5, 17,
21, 25, 70]
print(type(a))
a.sort()
print(a)                    # Output => [1, 4, 5, 10.5, 11, 17, 21,
25, 40, 70]   => (Sorted List)

#! sort(reverse=true) => It sorts the elements in descending order.
b = [40, 5, 4, 1, 11, 10.5, 17, 21, 25, 70]
b.sort(reverse=True)        # Use "True" not "true", otherwise it will
show an error.
print( b)                   # Output => [70, 40, 25, 21, 17, 11, 10.5,
5, 4, 1]   => (Sorted in reverse order)

#! reverse() - It reverses the list irrespective of any order.
a = [40, 5, 4, 1, 11, 10.5, 17, 21, 25, 70]
a.reverse()                 # It does reverse the list only, doesn't
sort.
print(a)                    # Output => [70, 25, 21, 17, 10.5, 11, 1,
4, 5, 40]

#sort(reverse=False)
a.sort(reverse=False)       # It does sort the list, doesn't reverse.
print(a)                    # Output => [1, 4, 5, 10.5, 11, 17, 21,
25, 40, 70]
```

```
[40, 5, 4, 1, 11, 10.5, 17, 21, 25, 70]
<class 'list'>
[1, 4, 5, 10.5, 11, 17, 21, 25, 40, 70]
[70, 40, 25, 21, 17, 11, 10.5, 5, 4, 1]
[70, 25, 21, 17, 10.5, 11, 1, 4, 5, 40]
[1, 4, 5, 10.5, 11, 17, 21, 25, 40, 70]
```

```
#! index() => It displays/prints the index position of an element of
list.
a = [1, 4, 5, 10.5, 11, 17, 21, 25, 40, 70]
```

```
print(a.index(17))          # a.index(17) will print the index of
element 17 (Which is 5). => Output => 5

#! count - Returns the count of an element of list.
a = [1, 4, 5, 10.5, 11, 17, 25, 25, 40, 70]
print(a.count(25))          # Total no. of 25 in List (i.e 2 times).
=> Output => 2
print(a.index(25))          # Prints index of 25 which comes first.
=> Output => 6

print(a[6])                 # Output => 25
print(a[7])                 # Output => 25
''' Because 25 is present in the list at index 6 and index 7. '''
a.index(25)                 # a.index(element)
a.index(25,7)               # a.index(element, indexNumber)
```

```
5
2
6
25
25

7
```

```
# Slicing & Dicing of Lists

#! SLICING => [m:n] => It cuts the elemets of list before mth index
and from nth index to last index.
''' Syntax => List[ Initial : End ]
List slicing returns a new list from the existing list. '''

a = [1, 4, 5, 8, 11, 17, 21, 25, 400, 400]
print(a[2:8])               # Output => [5, 8, 11, 17, 21, 25]
print(a[4:8])               # Output => [11, 17, 21, 25]
print(a[6:10])              # Output => [21, 25, 400, 400]

print(a[5:])                #Output => [17, 21, 25, 400, 400]

#It will cut all elements which are before the 5th index only.

print(a[0:5])               # Output => [1, 4, 5, 8, 11]
print(a[:5])                # Output => [1, 4, 5, 8, 11]
''' a[0:5] and a[:5] have same meaning.
It will cut all elements from the 5th index to last index . '''
```

```
[5, 8, 11, 17, 21, 25]
[11, 17, 21, 25]
[21, 25, 400, 400]
[17, 21, 25, 400, 400]
```

```
[1, 4, 5, 8, 11]
[1, 4, 5, 8, 11]

' a[0:5] and a[:5] have same meaning.\nIt will cut all elements from
the 5th index to last index . '


#! Reverse Indexing
print(a[-1])                 # Output => 400
print(a[-3])                 # Output => 25
print(a[-9])                 # Output => 4

400
25
4


#! Reverse Indexing Slicing
print(b[-8:-2])              # Output => [5, 8, 11, 17, 21, 25]
print(b)                     # Output => [1, 4, 5, 8, 11, 17, 21, 25,
400, 400]
print(a[-6:-3])              # Output => [11, 17, 21]
print(a[-7:])                # Output => [8, 11, 17, 21, 25, 400, 400]
print(a[:-5])                # Output => [8, 11, 17, 21, 25, 400, 400]

[25, 21, 17, 11, 10.5, 5]
[70, 40, 25, 21, 17, 11, 10.5, 5, 4, 1]
[11, 17, 21]
[8, 11, 17, 21, 25, 400, 400]
[1, 4, 5, 8, 11]


#! DICING => Slicing a list with index jumping.
# Syntax:  List[ Initial : End : IndexJump ]
print(a)
print(a[2:8:2])              # Output => [5, 11, 21]
print(a[::2])                # Output => [1, 5, 11, 21, 400]  => There
is no slicing only jumping of 2 indexes.
print(a[::9])

#! Dicing in reverse indexing slicing.
print(a[-8:-2:2])          # Output => [5, 11, 21]
# Explanation:
print(a[-8])               # Output => 5
print(a[-2])               # Output => 400
''' So, print(a[-8:-2:2]) will print the elements between 5 and 400
with jumping of 2 indexes in which 5(Initial) will be included while
400(End) will be excluded.   '''

[1, 4, 5, 8, 11, 17, 21, 25, 400, 400]
[5, 11, 21]
```

```
[1, 5, 11, 21, 400]
[1, 400]
[5, 11, 21]
5
400

#Reverse slicing, indexing and dicing
print(a)
print(a[-3:-9:-1])          # Output => [25, 21, 17, 11, 8, 5]
''' Here Slicing will be reverse b between -9th index and -3rd index
and 1 index jumping in reverse order.'''
print(a[-9:-3:-1])          # Output => []  => Empty List

print(a[-4:-10:-3])         # Output => [21, 8]

[1, 4, 5, 8, 11, 17, 21, 25, 400, 400]
[25, 21, 17, 11, 8, 5]
[]
[21, 8]

#Slicing and negative dicing
print(a)
print(a[7:1:-1])            # Output => [25, 21, 17, 11, 8, 5]
print(a[1:7:-1])            # Output => []

print(a[8:3:-2])            # Output => [400, 21, 11]

[1, 4, 5, 8, 11, 17, 21, 25, 400, 400]
[25, 21, 17, 11, 8, 5]
[]
[400, 21, 11]

#Dicing and negative slicing
print(a)
print(a[-8:-2:1])           # Output => [5, 8, 11, 17, 21, 25]
print(a[2:8:1])             # Output => [5, 8, 11, 17, 21, 25]
''' Both of above have same meaning and we can remove 1 from index
positon
because by default Slicing or Dicing takes 1 index jump.'''
print(a[-8:-2:3])           # Output => [5, 17]

print(a[10::-9])            # Output => [400, 1] => a[10::-9] and
a[10:0:-9] have same meaning.
print(a[-1:-11:-9])         # Output => [400, 1]
print(a[-1::-9])            # Output => [400, 1]
print(a[::-9])              # Output => [400, 1]

[1, 4, 5, 8, 11, 17, 21, 25, 400, 400]
[5, 8, 11, 17, 21, 25]
[5, 8, 11, 17, 21, 25]
[5, 17]
```

```
[400, 1]
[400, 1]
[400, 1]
[400, 1]

# Alternative for reverse function.
print(a)                    # Output => [1, 4, 5, 8, 11, 17, 21, 25,
400, 400]
print(a[::-1])              # Output => [400, 400, 25, 21, 17, 11, 8,
5, 4, 1]

[1, 4, 5, 8, 11, 17, 21, 25, 400, 400]
[400, 400, 25, 21, 17, 11, 8, 5, 4, 1]
```