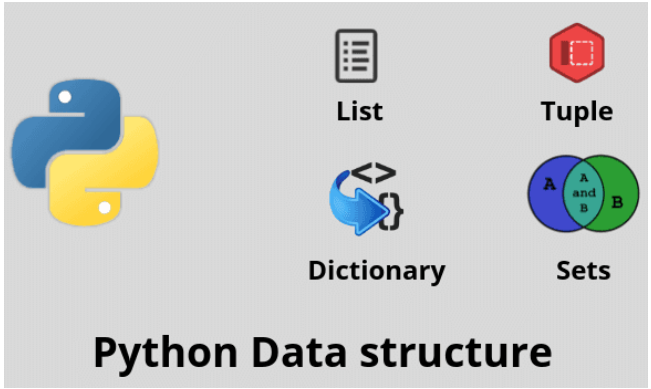
Data structures in Python

Week 7: List and Tuples

Learning objectives

- Understanding common Python data structures
- Creating Python Lists
- len(), count() functions and in operator for Lists
- how indexing, inserting, and deleting works in Python Lists
- Creating Python Tuples and more

At the conclusion of this lecutre, students will be able to understand the use of data structures and able to write programs using Lists and Tuples.



Data structures

Consider a case where your program needs to store and manipulate thousands or millions of data points.
OR
Assume you need to store 100 different values in your program then you would need 100 different named variables to handle those 100 data points.

For example : name = ”LUT”; n =2500, y = 14.502,.....Ooops!

Then you have to remember the names of variables when using in subsequent coding.....

Is there any coding techniques or built-in function that can handle this issue in efficient way?

Yes. They are called **data structures**

Data structures in Python

- Data strcuture is a collection of values.
- It allows saving mulitple values to a single variable and stored data can be accssed via same variable.
- Different kinds of data structures are meant for different kinds of applications.

Python has four basic inbuilt data structures.

- Lists (Today and Week 7)
- Tuples (Week 7)
- Dictionaries & Sets (Week 8 – self study)
- And supports more .....

Lists

- ✓ List is a dynamic sequence of zero or more consecutive items.
- ✓ Lists are used to store multiple items in a single variable.
- ✓ The contents of lists can be changed, re-arranged during the execution.

Example:

```
fruitList = ["Apple","Banana", "Cherry"]
print(fruitList)

    # A list is intilized with [ ] operator

myList  = [ ]    # create an empty list and add values later
print (myList) # returns emtpy list [ ]
```

Can List have duplicate values?

```
list1 = [ 1, 2, 3, 1, 7]
print (list1)
```

Examples

How many values are there in the list? # to find the length of the list

```
a_list = [1, 2, 3, 4, 5]
b_list = [6, 7, 8]
l = (len(a_list)          # Hei! I used len() to find length of the string
print (l)  # returns 5
```

To extract a value from a list # indexing elements

```
print (a_list [0])          # returns  a value in index 0 → 1
print ( a_list[1:3])        # returns a new list [2,3]
print (a_list[:])         # returns all values of the a_list
```

Index -->	0	1	2	3	4
a_list -->	1	2	3	4	5

#iterating through a list

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

```
n =  ["LUT", 65, 1.66, "Python"]
print (n[len(n)-1])  → would output _____?
print (n[-1])     → would output _____?
```

- Examples: combining, multiplying and more..

```
#combining lists
    a = [1, 2, 3]
    b = [6, 7, 8]
    a = a + b
    print (a)  # returns [1, 2, 3, 6, 7, 8]
```

What is the output of the following code?

```
    b = [6, 7, 8]
    c = b * 3
    print (c) # returns [6, 7, 8, 6, 7, 8, 6, 7, 8]
    print (c.count(6)) # returns 3
```

```
#Modifiying  the value of a certain index
    myList = [1, 2, 3, 4, 5, 7]
    myList[2] =  36
    print (myList)    # would print  [1, 2, 36, 4, 5, 7]
```

Examples: Adding, inserting and deleting values in the list

```
# Appending , inserting  and deleting values into the list

    x = [1, 3, 4]
    x.append(5)  # 5 will be added at the end of the list
    print (x) # would print [1, 3, 4, 5]

    x.insert (1, 2) # insert 2 at index 1
    print (x)  # would print [1, 2, 3, 4, 5]

    x.insert (9,46) → ?

    x = [1, 2, 3, 4, 5, 2]
    x.remove (2) # remove value 2 from the list not by index
    print (x)  # would print [1, 3, 4, 5, 2]

# Item in list – in operator
    a =  [1, 2, 3, 4, 5]
    print (7 in a) # returns False
    print  (3 in a) # returns True

# so to delete a value if it exists in the List
  def deleteItem (the_list, item):
      if item in the_list:  # using in operator here
         the_list.remove(item)
```

What **pop(index)** and **clear()** methods do in the Lists? Are they like **remove(item)?**

```
xample 8.py
1  def deleteItem(list1, item):
2      for i in list1:
3          if item in list1:
4              list1.remove(item)
5      print(list1)
6
7  #main program
8  listItems =[1, 2, 3, 4, 5, 2 ,2, 4, 2, 6, -2, 6, 7]
9  deleteItem(listItems, 2)

hell
ython 3.7.9 (bundled)
>> %Run 'example 8.py'

 [1, 3, 4, 5, 4, 6, -2, 6, 7]
```

- Sorting elements of list

```
1  List1 = [1,4,-23.9,89]
2  List2 = ["asd","akb@234","45"]
3
4  print(sorted(List1)) # printing sorted form but no change in the list
5  print(List1) # print as it is (line 1)
6  List2.sort() # sort the values of List2 - gets changed
7  print(List2) #
8
9  #descending order
10 print ("sorting elements in descending order")
11 print(sorted(List1,reverse=True))
12 print (List1)
13 List2.sort(reverse=True)
14 print(List2)
```

Shell

```
Python 3.7.9 (bundled)
>>> %Run examplesorty.py

  [-23.9, 1, 4, 89]
  [1, 4, -23.9, 89]
  ['45', 'akb@234', 'asd']
  sorting elements in descending order
  [89, 4, 1, -23.9]
  [1, 4, -23.9, 89]
  ['asd', 'akb@234', '45']

>>>
```

Group Task

#1 What is the output?

```
y = ["Chen", "Nuo", "Peter", "Wali", "Ali", "Ram", "Eduard"]
y.remove("Ram")
print(y)
```

```
n = [1, 2, 3, 4, 5]
s = 0
for i in range (1, len(n)):
    s = s+n[i]
print (s)
```

2 Define a procedure that remove negative numbers from the list

```
def deleteNeg_Item(the_list):
#your code here


marks = [78, 45, 0, -25, 3, -4, -8]
deleteNeg_Item(marks) # calling a procedure
print (marks)
```

hint → iterating through list (for), in operator and remove ( ) function

Group Task

#3 The program given below should get unique values from the original list and add them into new list→ uniList. However, the code has some errors. Fix the errors : Feel free to add or remove commands if necessary.

```
n = ["coffee", "tea", "soda", "tea", "burger", "coffee"]
for i in n:
    if i not in uniList:
        uniList.insert(i)
        n.remove(i)
print(uniList)
```

The fixed code should output → ['coffee', 'tea', 'soda', 'burger']