Kmit

SONET

Data Science

(Level-I)

Ajeet K. Jain, M. Narsimlu

(ML TEAM)- SONET, KMIT, Hyderabad

Kmit
SONET

Session - 14

DATA SCIENCE

This session deals with

List Comphrension

Tuple Data Structure

Tuple Vs Lists

Sets

Tuple Vs Sets

List Comphrensions

- 1.find cube of each element in a list
- #practice
- l1=[2,3,1]
- cubes=[n*n*n for n in l1]
- print(l1)
- print(cubes)

- 2.find square root of each element in the list
- #practice
- import math
- l1=[4,9,1,49]
- sqt=[math.sqrt(n) for n in l1]
- print(l1)
- print(sqt)

Exercise-3

Find common numbers from two list using list comprehension

```
list_a = [1, 2, 3, 4]
list_b = [2, 3, 4, 5]
```

Solution

```
common_num = [a for a in list_a for b in list_b if a == b]

print(common_num)
```

Output

```
[2, 3, 4]
```

Exercise-4

Return numbers from the list which are not equal as tuple

```
l1 = [1, 2, 3]
l2 = [2, 7]


different_num = [(x, y) for x in l1 for y in l2 if x != y]

print(different_num) # Output: [(1, 2), (1, 7), (2, 7), (3, 2), (3, 7)]


#Output: [(1, 2), (1, 7), (2, 7), (3, 2), (3, 7)]
```

Exercise-5

Convert list elements to lower case

Solution

```
list_a = ["Hello", "World", "In", "Python"]

small_list_a = [str.lower() for str in list_a]

print(small_list_a)


 # Output: ['hello', 'world', 'in', 'python']
```

Exercise-6

Write a python program to create a list of even numbers and another list of odd numbers from a given list

For Example:
listOne = [3, 6, 9, 12, 15, 18, 21]
listTwo = [4, 8, 12, 16, 20, 24, 28]
Expected Output:
Element at odd-index positions from list one
[6, 12, 18]
Element at odd-index positions from list two
[4, 12, 20, 28]
Printing Final third list
[6, 12, 18, 4, 12, 20, 28]

Sol:

```
listOne = [3, 6, 9, 12, 15, 18, 21]
listTwo = [4, 8, 12, 16, 20, 24, 28]
listThree = []
oddElements = listOne[1::2]
print("Element at odd-index positions from list one")
print(oddElements)
EvenElement = listTwo[0::2]
print("Element at odd-index positions from list two")
print(EvenElement)
print("Printing Final third list")
listThree.extend(oddElements)
listThree.extend(EvenElement)
print(listThree)
```

Exercise -7

Given an input list removes the element at index 4
and add it to the 2nd position and also, at the end of the list
Expected Output:
Origigal list [34, 54, 67, 89, 11, 43, 94]
List After removing element at index 4 [34, 54, 67, 89, 43, 94]
List after Adding element at index 2 [34, 54, 11, 67, 89, 43, 94]
List after Adding element at last [34, 54, 11, 67, 89, 43, 94, 11]

Sol:

```
sampleList = [34, 54, 67, 89, 11, 43, 94]
print("Origigal list ", sampleList)
element = sampleList.pop(4)
print("List After removing element at index 4 ", sampleList)
sampleList.insert(2, element)
print("List after Adding element at index 2 ", sampleList)
sampleList.append(element)
print("List after Adding element at last ", sampleList)
```

Exercise -8

Given a list slice it into a 3 equal chunks and reverse each list

For Example: sampleList = [11, 45, 8, 23, 14, 12, 78, 45, 89]

Expected Outcome:

Origigal list [11, 45, 8, 23, 14, 12, 78, 45, 89]

Chunk 1 [11, 45, 8]

After reversing it [8, 45, 11]

Chunk 2 [23, 14, 12]

After reversing it [12, 14, 23]

Chunk 3 [78, 45, 89]

After reversing it [89, 45, 78]

Solution:

```
sampleList = [11, 45, 8, 23, 14, 12, 78, 45, 89]
print("Origigal list ", sampleList)
length = len(sampleList)
chunkSize  = int(length/3)
start = 0
end = chunkSize
for i in range(1, 4, 1):
  indexes = slice(start, end, 1)
  #The slice() function returns a slice object.
  #slice(start, end, step)
  listChunk = sampleList[indexes]
  print("Chunk ", i , listChunk)
  print("After reversing it ", list(reversed(listChunk)))
  start = end
  if(i != 2):
    end +=chunkSize
  else:
    end += length – chunkSize
```

Tuple

Tuple is a collection of same type of data items as well as mixed type of data items.

To create a tuple in Python, place all the elements in a () parenthesis, separated by commas.

```
# tuple of strings
my_data = ("hi", "hello", "bye")
print(my_data)
# tuple of int, float, string
my_data2 = (1, 2.8, "Hello World")
print(my_data2)
# tuple of string and list
my_data3 = ("Book", [1, 2, 3])
print(my_data3)
# tuples inside another tuple
# nested tuple
my_data4 = ((2, 3, 4), (1, 2, "hi"))
print(my_data4)
```

Tuple

Note: When a tuple has only one element, we must put a comma after the element, otherwise Python will not treat it as a tuple.

```
# a tuple with single data item
my_data = (99)
Print(a)
Output:
99
My_data=(99,)
Print(my_data)
Output:
(99,)
```

Properties

1.We can access tuple elements with positive and negative indexes
2.We cannot change the elements of a tuple because elements of tuple are immutable.

```
my_data = (1, [9, 8, 7], "World")
print(my_data)
my_data[1][2] = 99
print(my_data)
```

3.we cannot delete the elements of a tuple. However deleting entire tuple is possible.

```
my_data = (1, 2, 3, 4, 5, 6)
print(my_data)
# not possible
# error
# del my_data[2]
# deleting entire tuple is possible
del my_data
```

Tuple Vs List

1. The elements of a list are mutable whereas the elements of a tuple are immutable.
2. When we do not want to change the data over time, the tuple is a preferred data type whereas when we need to change the data in future, list would be a wise option.
3. Iterating over the elements of a tuple is faster compared to iterating over a list.
4. Elements of a tuple are enclosed in parenthesis whereas the elements of list are enclosed in square bracket.

Sets

A set is an unordered collection of items.

Every element is unique (no duplicates) and must be immutable (which cannot be changed)..

We can add or remove items from it.

Sets can be used to perform mathematical set operations like union, intersection, symmetric difference etc.

A set is created by placing all the items (elements) inside curly braces {}, separated by comma or by using the built-in function set()

```
courses = set(['java', 'Data', 'science', 'java', 'Data'])
print(courses)
```

Output

{'science', 'Data', 'java'}

Sets

```
courses = set(('java', 'Data', 'science', 'java', 'Data'))
print(courses)
```

Output

{'science', 'Data', 'java'}

```
course="Data Science"
print(set(course))
```

Output

{' ', 'c', 'n', 'D', 'i', 't', 'S', 'a', 'e'}

```
courses={"java","python","python","java","Data Science"}
print(courses)
```

{'python', 'java', 'Data Science'}

Operators Vs methods

set operations in Python can be performed in two different ways: by operator or by method.

```
course1 = {'java', 'Data', 'science', 'java', 'Data',"python"}
course2={"Data","science","java","Data","c"}
print(course1|course2)
Print(course1.union(course2))
```

Output

```
{'Data', 'python', 'science', 'c', 'java'}
{'Data', 'python', 'science', 'c', 'java'}
```

Intersection

set operations in Python can be performed in two different ways: by operator or by method.

```
course1 = {'java', 'Data', 'science', 'java', 'Data',"python"}
course2={"Data","science","java","Data","c"}
print(course1&course2)
Print(course1.intersection(course2))
```

Output

```
{'java', 'Data', 'science'}
{'java', 'Data', 'science'}
```

Difference

set operations in Python can be performed in two different ways: by operator or by method.

```
course1 = {'java', 'Data', 'science', 'java', 'Data',"python"}
course2={"Data","science","java","Data","c"}
print(course1-course2)
Print(course1.difference(course2))
```

Output

```
{'python'}
{'python'}
```

Sets-Methods

Add()-	adding the elements to the list
Update()-	updating existing set
Remove()-	To remove particular element from the set
Clear()-	To clear the set
Pop()-	To remove top most element from the set

Sets-Methods


```
course1={30}
course1.add(10)
course1.add(20)
print(course1)
course1.update((45,90,125))
print(course1)
course1.remove(90)
print(course1)
course1.pop()
print(course1)
course1.clear()
print(course1)
```

Output

```
{10, 20, 30}
{10, 45, 20, 90, 125, 30}
{10, 45, 20, 125, 30}
{45, 20, 125, 30}
set()
```

Tuple Vs Sets

1. **Tuple** -
A tuple is basically an immutable list, it means you can't add, remove, or replace any elements, once you declare it ,you are done
strings, tuples are hashable.
slicing and dicing in tuples works exactly like list

Set -
A set is like a array of unique elements and its has no order . but has the advantage over a list and tuples that it is much faster, almost regardless of the size of the set. It also has some handy operations such as union,difference and intersection.
The elements of a set is not hashable.

Exercises

1. Write a python program to create a list of even numbers and another list of odd numbers from a given list
2. Write a python program to remove repeated elements from a given list without using built-in methods
3. Write a python program to find the longest word in a given sentence
4. Write a python program to find number of occurrences of given number with out using built-in methods
5. ["www.zframez.com", "www.wikipedia.org", "www.asp.net", "www.abcd.in"]
Write a python program to print website suffixes (com , org , net ,in) from this list
6. Write a python program to sort a given list of numbers with out using sort() function

Conclusion

You are aware of

List Data Structures

Tuple

We will proceed with

More on Data Structures

Kmit

SONET

DATA SCIENCE

THANK YOU