Department of Computer Science and Engineering
PES University, Bangalore, India

Lecture Notes
Python for Computational Problem Solving
UE23CS151A

Lecture #94

List Comprehension

By,
Prof. Sindhu R Pai,
Anchor, PCPS - 2023
Assistant Professor
Dept. of CSE, PESU
&
Dr.Manju More E
Associate Professor
Dept. of CSE, PESU

Many Thanks to
Dr. Shylaja S S (Director, CCBD and CDSAML Research Centers, Former Chairperson, CSE, PES University)
Prof. Chitra G M, (Asst. Prof, Dept. of CSE, PCPS Anchor – 2022)

List comprehension

A **concise way of defining and creating a list within a single line of code**. It offers a **shorter syntax** when you want to create a new list based on the values of an existing list leaving the existing list unchanged. It **always creates a new list by evaluating the expression in the context of for and if clauses** which follows it. This is much faster in processing than a list using for loop.

According to the Python documentation, A list comprehension consists of **square brackets** containing an **expression followed by a for clause**, then **zero or more for or if clauses**. The result will be a new list resulting from evaluating the expression in the context of the for and if clauses which follow it.

Syntax:

[expr for <variable> in <iterable> [if condition]]

expr: This is the expression or element in the new list. If condition is applicable if you want to filter few of the elements based on the condition

Syntax is equivalent to:

for variable in iterable :

if condition:

expr

Consider an example where we want to create a list with the cube of the first 10 natural numbers.

```
cube = []
for i in range (1,11):
        cube.append(i**3)
print("result = ",cube)
```

```
result =  [0, 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

The above code is equivalent to:

```
cube = [x**3 for x in range (11)]
print ("result = ", cube)
```

Consider the tuple,

t1 = ("biscuit", "chocolate", "icecream", "cake")

Convert all of these to uppercase and store it in a list.

```
new_t1 = [x.upper() for x in t1]
print(new_t1)
```

```
['BISCUIT', 'CHOCOLATE', 'ICECREAM', 'CAKE']
```

If the item in tuple is starting with the letter 'c', convert to uppercase. Else let the item be in the same case as in the original list.

```
new_t1 = [x.upper()if x.startswith('c') else x for x in t1]
print(new_t1)
```

```
['biscuit', 'CHOCOLATE', 'icecream', 'CAKE']
```

If the item in tuple is starting with the letter 'c', convert to upper case and add that to a new list.

```
new_t1 = [x.upper() for x in t1 if x.startswith('c')]
print(new_t1)
```

```
['CHOCOLATE', 'CAKE']
```

Few points to think!

- **Does the list created by the list comprehension has a new id?**
- **Is list comprehension replacing the usage of map and filter?**

Few examples to demo list comprehension are here.

Example_code_1: Creating a list containing the squares of numbers from 1 to 5

```
li = [ x * x for x in range(1, 6)]
print(li)
```

```
[1, 4, 9, 16, 25]
```

Example_code_2: Program to create a list with the last letter of each word from the sentence entered by the user

```
sentence = input("enter the sentence")
last_letters = [ word[-1] for word in sentence.split() ]
print (last_letters)
```

```
enter the sentencei am studying in PES University
['i', 'm', 'n', 'n', 'S', 'y']
```

Example_code_3: Program to print a list with words from the given list whose length is more than 3.

```
li = ["sindhu", "r", "pai", "pesu"]
print([x for x in l1 if len(x)>3])
```

```
['sindhu', 'pesu']
```

Example_code_3: Program to create a list with only even integers from the given list of integers

```
li = [2,3,4,5]
print([x for x in li if x%2 == 0])
```

```
[2, 4]
```

If we want to square the even elements from the existing list and create a new list, how to modify the above code?

```
print([x*x for x in li if x%2 == 0])
```

But if we say, for the number which is odd, False must be there in the newly created list corresponding to that item. How do we add this in list comprehension? Think!!

```
print([x*x if x%2 == 0 else False for x in li])   #observe this carefully
```

```
[4, 16]
```

Example_code_4: Program to create a list containing the tuple of string and its length as every element of the list.

```
li = [ (x, len(x)) for x in ['pes', 'University','bangalore']]
print(li)
```

```
[('pes', 3), ('University', 10), ('bangalore', 9)]
```

Example_code_5: Find common numbers from two lists and create a new list

Without List comprehension:

```
list_a = [1, 2, 3, 4]
list_b = [2, 3, 4, 5]
common_num = []
for a in list_a:
  for b in list_b:
    if a == b:
      common_num. append(a)
print(common_num)
```

Using List comprehension:

```
list_a = [1, 2, 3, 4]
list_b = [2, 3, 4, 5]
common_num = [a for a in list_a for b in list_b if a == b]
print(common_num)
```

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

Can we use nested ifs in a list comprehension? – Yes.

Example_code_6: Create a list of numbers between 0 and 51 which are divisible by both 2 and 3.

```
num_list = [y for y in range(50) if y % 2 == 0 if y % 3 == 0]
print(num_list)
```

```
[0, 6, 12, 18, 24, 30, 36, 42, 48]
```

Nested List Comprehension

A nested list comprehension doubles down on the concept of list comprehensions. It's a **way to combine** not only one, but **multiple for loops, if statements and functions into a single line of code**. This becomes useful when you have a list of lists.

Consider the example of a 3x3 matrix which is implemented as a list of 3 lists of length 3. Think about how do we code using the nested list comprehension to get the transpose of this matrix.

```
matrix = [
   [1, 2, 3],    [5, 6, 7],    [9, 10, 11]
 ]
print([[row[i] for row in matrix] for i in range(3)])
```

```
[[1, 5, 9], [2, 6, 10], [3, 7, 11]]
```

Example_code_7: Flatten the list of lists into a single list.

Original: [[1, 2, 3], [4, 5], [6, 7, 8, 9]]

Expected output: [1, 2, 3, 4, 5, 6, 7, 8, 9]

Without comprehension

```
newone = []
for sublist in Original:
   for val in sublist:
      newone.append(val)
print(newone)
```

Using Nested list comprehension

```
newone  = [val for sublist in Original for val in sublist]
```

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

-END-