



# INTERMEDIATE PYTHON

LESSON 6 | List Comprehension | 30-12-18

**Video tutorial:**

**List Comprehension:** <https://www.youtube.com/watch?v=4oS9e5Rf384>

[https://www.pylenin.com/list\\_comprehensions.html](https://www.pylenin.com/list_comprehensions.html)

# List Comprehension

## ***Python List Comprehension***

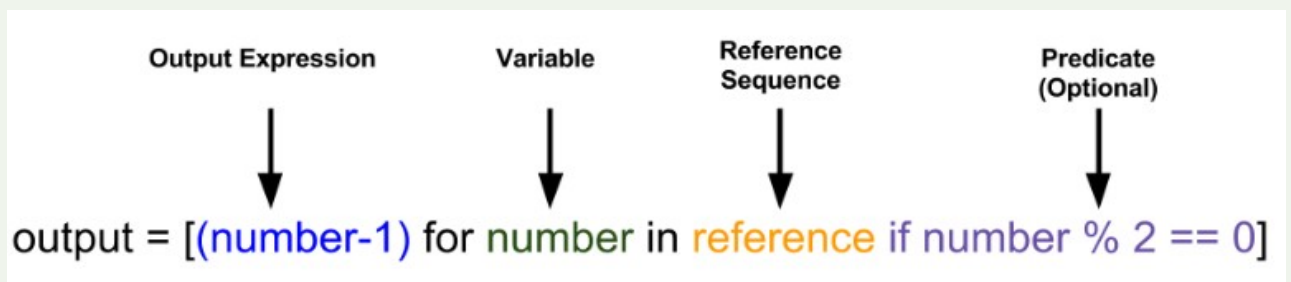
List Cpmprehension is basically used to generate a list of elements having some specific property. Moreover, Python List Comprehension make code smaller but effective,

## ***Basic Structure of Python List Comprehension***

The basic structure consist of three parameters

1. Variable
2. Expression for output
3. Reference sequence
4. Predicate (Optional)

On basis of these three parameter, python generated a new list.



### ***Conditionals in Comprehensions***

#### ***Conditionals on The Iterable***

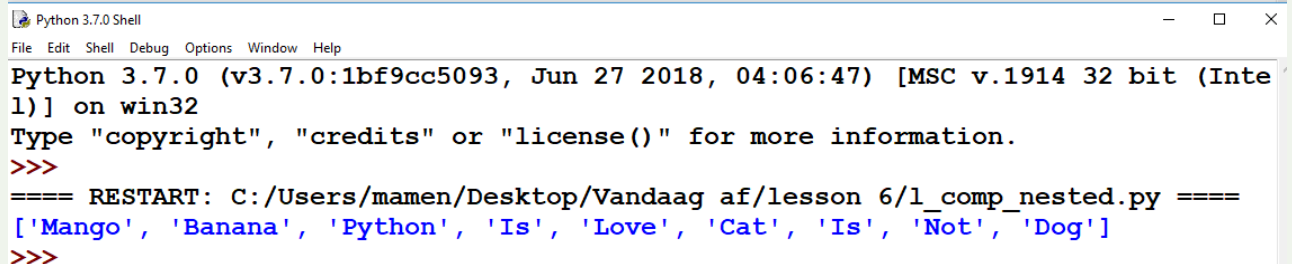
```
>>> [num**2 for num in range(10) if num%2==0]
[0, 4, 16, 36, 64]
>>>
```

## Conditionals on The Output Expression

```
>>> [num**2 if num%2 ==0 else 0 for num in range(10) ]  
[0, 0, 4, 0, 16, 0, 36, 0, 64, 0]  
>>>
```

## Nested List Comprehension

```
list_string = ['maNgo', 'BanAna', 'PytHoN iS Love', 'Cat iS not doG']  
  
correct_case = [str.upper(word[0])+str.lower(word[1:])  
                 for word in sum([sentence.split() for sentence in list_string],  
                                if len(word) > 1)]  
  
print(correct_case)
```



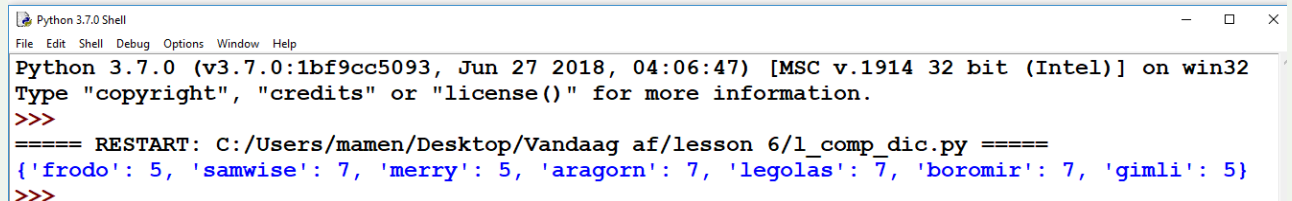
```
Python 3.7.0 Shell  
File Edit Shell Debug Options Window Help  
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>>  
==== RESTART: C:/Users/mamen/Desktop/Vandaag af/lesson 6/l_comp_nested.py ====  
['Mango', 'Banana', 'Python', 'Is', 'Love', 'Cat', 'Is', 'Not', 'Dog']  
>>>
```

But the nested list comprehension code is not easily readable in many cases. So, nested Python List Comprehension is not recommended to use in every case though it make code short.

## Dict comprehensions

*Use curly {} braces instead of brackets []*

```
fellowship = ['frodo', 'samwise', 'merry', 'aragorn', 'legolas', 'boromir', 'gimli']  
  
new_fellowship = {member : len(member) for member in fellowship}  
  
print(new_fellowship)
```



```
Python 3.7.0 Shell  
File Edit Shell Debug Options Window Help  
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>>  
==== RESTART: C:/Users/mamen/Desktop/Vandaag af/lesson 6/l_comp_dic.py ====  
{'frodo': 5, 'samwise': 7, 'merry': 5, 'aragorn': 7, 'legolas': 7, 'boromir': 7, 'gimli': 5}  
>>>
```

So that's all about Python List Comprehension. Hope, you understand the topic.

***Exercices: Use list comprehension!!!!***

1. Find all of the numbers from 1-1000 that are divisible by 7
2. Find all of the numbers from 1-1000 that have a 3 in them
3. Count the number of spaces in a string
4. Remove all of the vowels in a string
5. Find all of the words in a string that are less than 4 letters
6. Use a dictionary comprehension to count the length of each word in a sentence.
7. Use a nested list comprehension to find all of the numbers from 1-1000 that are divisible by any single digit besides 1
8. For all the numbers 1-1000, use a nested list/dictionary comprehension to find the highest single digit any of the numbers is divisible by 9

## Answers :

#Find all of the numbers from 1-1000 that are divisible by 7

```
results = [num for num in range(1000) if num % 7 == 0]
```

```
#print(results)
```

#Find all of the numbers from 1-1000 that have a 3 in them

```
results = [num for num in range(1000) if '3' in list(str(num))]
```

```
#print(results)
```

#Count the number of spaces in a string

```
teststring = 'Find all of the words in a string that are less than 4  
letters'
```

```
#print teststring.count(' ') # normally I'd just do this, but for the  
practice....
```

```
results = [character for character in teststring if character == ' ']
```

```
#print(len(results))
```

```
#Remove all of the vowels in a string [make a list of the non-  
vowels]  
  
teststring = 'Find all of the words in a string that are less than 4  
letters'  
  
vowels = ['a','e','i','o','u',' ']  
  
results = [letter for letter in teststring if letter.lower() not in  
vowels]  
  
#print(results)
```

```
#Find all of the words in a string that are less than 4 letters  
  
teststring = 'Find all of the words in a string that are less than 4  
letters'  
  
results = [word for word in teststring.split() if len(word) < 4]  
  
#print(results)
```

```
#Use a dictionary comprehension to count the length of each  
word in a sentence.  
  
sentence = 'Use a dictionary comprehension to count the length  
of each word in a sentence'
```

```
results = {word:len(word) for word in sentence.split()}
```

```
#print(results)
```

```
#Use a nested list comprehension to find all of the numbers from  
1-1000 that are divisible by any single digit besides 1 (2-9)
```

```
# comprehension testing truth for divisibility: [True for divisor in  
range(2,10) if number % divisor == 0]
```

```
results = [number for number in range(1,1001) if True in [True for  
divisor in range(2,10) if number % divisor == 0]]
```

```
#print(results)
```

```
#For all the numbers 1-1000, use a nested list/dictionary  
comprehension to
```

```
#find the highest single digit any of the numbers is divisible by.
```

```
# List comprehension for providing a list of all of the numbers a  
number is divisible by: divisor_list:
```

```
#[divisor for divisor in range(1,1001) if number % divisor == 0]
```

```
results = {number:max([divisor for divisor in range(1,10) if number  
% divisor == 0]) for number in range(1,1001)}
```

```
#print(results)
```

