Python Comprehensions:

- We can create new sequences using a given python sequence. This is called comprehension. It basically a way of writing a concise code block to generate a sequence which can be a list, dictionary, set or a generator by using another sequence. It may involve multiple steps of conversion between different types of sequences.
- Comprehensions in Python provide us with a short and concise way to construct new sequences (such as lists, set, dictionary etc.) using sequences which have been already defined.
- Python supports the following types of comprehensions:
  1. List Comprehensions
  2. Dictionary Comprehensions
  3. Set Comprehensions
  4. generator Comprehensions

List Comprehentions:

- List comprehensions provide a concise way to create lists.
- It consists of brackets containing an expression followed by a for clause, then zero or more for or if clauses. The expressions can be anything, meaning you can put in all kinds of objects in lists.
- The result will be a new list resulting from evaluating the expression in the context of the for and if clauses which follow it.
- The list comprehension always returns a result list.

Note: **The basic syntax uses square brackets.**

For example,

```
[ expression  for  item  in  list  if  conditional ]
```

Create a simple list:

Let’s start easy by creating a simple list.

```
>>> x = [ i  for  i  in  range(10) ]
Print(x)
# This will give the output:
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Create a list using loops and list comprehension:

For the next example, assume we want to create a list of squares. Start with an empty list.

# **You can either use loops:**

```
squares = []
for x in range(10):
    squares.append(x**2)
print(squares)
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

# Or you can use list comprehensions to get the same result:

```
squares = [ x ** 2  for  x   in range(10)]
print(squares)
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Multiplying parts of a list

---->> Multiply every part of a list by three and assign it to a new list.

```
>>> list1 = [3,4,5]
>>> multiplied = [ item * 3  for   item    in     list1]
>>> print( multiplied )      # [9,12,15]
```

Note how the item*3 multiplies each piece by 3.

Q. Show the first letter of each word

--->> We will take the first letter of each word and make a list out of it.

```
listOfWords = ["this","is","a","list","of","words"]
items = [ word[0] for word in listOfWords ]
print(items)
```

Output : [‘t’, ‘i’, ‘a’, ‘l’, ‘o’, ‘w’]

Lower/Upper case converter

Let’s show how easy you can convert lower case / upper case letters.

```
>>> [x.lower() for x in ["A","B","C"]]
['a', 'b', 'c']
>>> [x.upper() for x in ["a","b","c"]]
```

```
['A', 'B', 'C']
```

Print numbers only from a given string

This example show how to extract all the numbers from a string.

```
string = "Hello 12345 World"
numbers = [x for x in string if x.isdigit()]
print (numbers)
>> ['1', '2', '3', '4', '5']
```

Note : Change x.isdigit() to x.isalpha() if you don’t want any numbers.

Using list comprehension in functions

--->> Now, let’s see how we can use list comprehension in functions.

```
# Create a function and name it double:
def double(x):
	return x*2

# If you now just print that function with a value in it, it should look like this:
>>> print double(10)          #  20
```

--->> We can easily use list comprehension on that function.

```
>>> [double(x) for x in range(10)]
>>> print(double)
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
# You can put in conditions:
>>> [double(x) for x in range(10) if x%2==0]
[0, 4, 8, 12, 16]
# You can add more arguments:
>>> [x+y for x in [10,30,50] for y in [20,40,60]]
[30, 50, 70, 50, 70, 90, 70, 90, 110]
```

--->> See how you can put in conditions and add more arguments.

Python Dictionary Comprehension:

--->> Like List Comprehension, Python allows dictionary comprehensions.

--->> We can create dictionaries using simple expressions.

--->> A dictionary comprehension takes the form

  syntax : { key:value for (key, value) in iterable }

Let’s see a example, lets assume we have two lists named keys and value now.

```
# Python code to demonstrate dictionary  comprehension.
# Lists to represent keys and values
keys = ['a','b','c','d','e']
values = [1,2,3,4,5]
# but this line shows dict comprehension here
myDict = { k:v for (k,v) in zip(keys, values)}
# We can use below too
# myDict = dict(zip(keys, values))
print (myDict)
```

Output : {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}

---->> We can also make dictionary from a list using comprehension.

```
# Python code to demonstrate dictionary  creation using list comprehension.
myDict = {x: x**2 for x in [1,2,3,4,5]}
print (myDict)
```

Output : {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

---->> We can use Dictionary comprehensions with if and else statements and with other   expressions too.

Q). This example below maps the numbers to their cubes that are divisible by 4?

```
# Python code to demonstrate dictionary comprehension using if.

newdict = {x: x**3 for x in range(10) if x**3 % 4 == 0}
print(newdict)
```

Output : {0: 0, 2: 8, 4: 64, 6: 216, 8: 512}

Example 1:

Suppose we want to create an output dictionary which contains only the odd numbers that are present in the input list as keys and their cubes as values. Let’s see how to do this using for loops and dictionary comprehension.

```
input_list = [1, 2, 3, 4, 5, 6, 7]
output_dict = {}
# Using loop for constructing output dictionary
for var in input_list:
    if var % 2 != 0:
        output_dict[var] = var**3
print("Output Dictionary using for loop:", output_dict )
```

Output: Output Dictionary using for loop: {1: 1, 3: 27, 5: 125, 7: 343}

```
# Using Dictionary comprehensions for constructing output dictionary
input_list = [1,2,3,4,5,6,7]
dict_using_comp = {var:var ** 3 for var in input_list if var % 2 != 0}
print("Output Dictionary using dictionary comprehensions:", dict_using_comp)
```
Output: Output Dictionary using dictionary comprehensions: {1: 1, 3: 27, 5: 125, 7: 343}

Example #2: Given two lists containing the names of states and their corresponding capitals, construct a dictionary which maps the states with their respective capitals. Let’s see how to do this using for loops and dictionary comprehension.

```
state = ['Gujarat', 'Maharashtra', 'Rajasthan']
capital = ['Gandhinagar', 'Mumbai', 'Jaipur']
output_dict = {}
# Using loop for constructing output dictionary
for (key, value) in zip(state, capital):
    output_dict[key] = value
print("Output Dictionary using for loop:", output_dict)
```
Output: Output Dictionary using for loop: {'Gujarat': 'Gandhinagar', 'Maharashtra': 'Mumbai', 'Rajasthan': 'Jaipur'}

```
# Using Dictionary comprehensions for constructing output dictionary.
state = ['Gujarat', 'Maharashtra', 'Rajasthan']
capital = ['Gandhinagar', 'Mumbai', 'Jaipur']
dict_using_comp = {key:value for (key, value) in zip(state, capital)}
print("Output Dictionary using dictionary comprehensions:", dict_using_comp)
```
Output: Output Dictionary using dictionary comprehensions: {'Rajasthan': 'Jaipur', 'Maharashtra': 'Mumbai', 'Gujarat': 'Gandhinagar'}

Set Comprehensions:

--->> Set comprehensions are pretty similar to list comprehensions.
--->> The only difference between them is that set comprehensions use curly brackets { }.

Let’s look at the following example to understand set comprehensions.

Example #1 :

Suppose we want to create an output set which contains only the even numbers that are present in the input list. Note that set will discard all the duplicate values. Let’s see how we can do this using for loops and set comprehension.

```
input_list = [1, 2, 3, 4, 4, 5, 6, 6, 6, 7, 7]
output_set = set()
# Using loop for constructing output set
for var in input_list:
    if var % 2 == 0:
        output_set.add(var)
print("Output Set using for loop:", output_set)
```

Output: Output Set using for loop: {2, 4, 6}

```
# Using Set comprehensions  for constructing output set
input_list = [1, 2, 3, 4, 4, 5, 6, 6, 6, 7, 7]
set_using_comp = {var for var in input_list if var % 2 == 0}
print("Output Set using set comprehensions:",  set_using_comp)
```

Output: Output Set using set comprehensions: {2, 4, 6}