Comprehensions in Pysthon 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:

* List Comprehensions
* Dictionary Comprehensions
* Set Comprehensions
* **List Comprehensions:**
* List Comprehensions provide an elegant way to create new lists. The following is the basic structure of a list comprehension:
* *output\_list = [output\_exp for var in input\_list if (var satisfies this condition)]*
* Note that list comprehension may or may not contain an if condition. List comprehensions can contain multiple **for** (nested list comprehensions).
* **Example #1:** Suppose we want to create an output list which contains only the even numbers which are present in the input list. Let’s see how to do this using *for loops* and *list comprehension* and decide which method suits better.

|  |
| --- |
| # Constructing output list WITHOUT  # Using List comprehensions  input\_list **=** [1, 2, 3, 4, 4, 5, 6, 7, 7]    output\_list **=** []    # Using loop for constructing output list  **for** var **in** input\_list:  **if** var **%** 2 **==** 0:          output\_list.append(var)    print("Output List using for loop:", output\_list) |

* **Output:**
* Output List using for loop: [2, 4, 4, 6]

|  |
| --- |
| # Using List comprehensions  # for constructing output list    input\_list **=** [1, 2, 3, 4, 4, 5, 6, 7, 7]      list\_using\_comp **=** [var **for** var **in** input\_list **if** var **%** 2 **==** 0]    **print**("Output List using list comprehensions:",                                 list\_using\_comp) |

* **Output:**
* Output List using list comprehensions: [2, 4, 4, 6]

### Dictionary Comprehensions:

Extending the idea of list comprehensions, we can also create a dictionary using dictionary comprehensions. The basic structure of a dictionary comprehension looks like below.

*output\_dict = {key:value for (key, value) in*iterable *if (key, value satisfy this condition)}*

**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}

**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}