Comprehensions

- Comprehensions in Python provide us with a short and concise way to construct new sequences (such as lists, sets, dictionaries, etc.) using previously defined sequences.

  Python supports the following 4 types of comprehension:

  - **List Comprehensions**
  - **Dictionary Comprehensions**
  - **Set Comprehensions**
  - **Generator Comprehensions**

❖ List Comprehensions

List Comprehensions provide an elegant way to create new lists. The following is the basic structure of list comprehension:

Syntax:

output_list = [output_exp for var in input_list if (var satisfies this condition)]

Example:

```
input_list = [1, 2, 3, 4, 4, 5, 6, 7, 7]

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]
```

❖ Dictionary Comprehensions

Extending the idea of list comprehensions, we can also create a dictionary using dictionary comprehensions.

Syntax:

output_dict = {key:value for (key, value) in iterable if (key, value satisfy this condition)}

Example:

```
input_list = [1, 2, 3, 4, 5, 6, 7]


output_dict = {}


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}
```

❖ Set Comprehensions

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

Example:

```
input_list = [1, 2, 3, 4, 4, 5, 6, 6, 6, 7, 7]


output_set = 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}
```

❖ Generator Comprehensions

Generator Comprehensions are very similar to list comprehensions.

One difference between them is that generator comprehensions use circular brackets whereas list comprehensions use square brackets.

The major difference between them is that generators don't allocate memory for the whole list.

Instead, they generate each value one by one which is why they are memory efficient.

Example:

```
input_list = [1, 2, 3, 4, 4, 5, 6, 7, 7]

output_gen = (var for var in input_list if var % 2 == 0)

print("Output values using generator comprehensions:", end = ' ')

for var in output_gen:
        print(var, end = ' ')
```

Output:

```
Output values using generator comprehensions: 2  4  4  6
```