

9. Comprehension

April 21, 2023

1 Comprehension

- A comprehension is a compact way of creating a Python data structure from one or more iterators.

1.1 List Comprehension

- Write a program to create list of squares of numbers from 1 to 10

```
[1]: # program to generate list of squares of first 10 natural numbers

squares = []
for i in range(1,11):
    squares.append(i**2)
```

```
[2]: squares
```

```
[2]: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
[3]: # program to generate list of squares of first 10 natural numbers
# using list comprehension
squares_2 = [i**2 for i in range(1,11)]
```

```
[4]: squares_2
```

```
[4]: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

- Write a program to create list of squares of **even** numbers from 1 to 10

```
[5]: sq_even = []
for i in range(1,11):
    if i%2 == 0:
        sq_even.append(i**2)
```

```
[6]: sq_even
```

```
[6]: [4, 16, 36, 64, 100]
```

```
[7]: sq_even_C = [i**2 for i in range(1,11) if i%2 == 0]
```

```
[8]: sq_even_C
```

```
[8]: [4, 16, 36, 64, 100]
```

- Comprehension for nested loops

```
for i in list_1:
    if expr:
        for j in list_2:
            statements
            l.append(j)
```

```
[j for i in list_1 if expr for j in list_2 ]
```

1.2 Tuple Comprehension

```
[9]: squaresT = tuple(i**2 for i in range(1,11))
```

```
[10]: squaresT
```

```
[10]: (1, 4, 9, 16, 25, 36, 49, 64, 81, 100)
```

1.3 Dictionary Comprehension

- Generate a dictionary of squares of first 10 natural numbers, where key is the natural number and value is its square

```
[11]: squares = {x : x*x for x in range(1,11)}
```

```
[12]: squares
```

```
[12]: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

1.4 Set comprehension

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

```
[14]: # create set of even numbers from given list of numbers
set_using_comp = {var for var in input_list if var % 2 == 0}
```

```
[15]: set_using_comp
```

```
[15]: {2, 4, 6}
```

2 enumerate

- A lot of times when dealing with iterators, we also get a need to keep a count of iterations.
- Python eases the programmers' task by providing a built-in function `enumerate()` for this task.
- `enumerate()` method adds a counter to an iterable and returns it in a form of `enumerate` object.
- This `enumerate` object can then be used directly in for loops.

```
[16]: list_colors = ["red", "green", 'blue', 'pink']
```

```
[17]: for i,v in enumerate(list_colors):  
       print(i,v)  
       print("----")
```

```
0 red  
----  
1 green  
----  
2 blue  
----  
3 pink  
----
```